Design and development of a secure and patient-controlled system to share healthcare data for research

Anton Gladyr

School of Computer Science

McGill University, Montreal

November, 2021

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of

Master of Computer Science

Abstract

Nowadays, almost all hospitals and clinics in developed countries store patients' personal medical data in digital format taking into account appropriate security measures and in compliance with applicable legal requirements. Due to the rapid development of so-called Big Data research tools, such as artificial intelligence and machine learning, these data, if accessible, have the potential to benefit medical research in the search for new medications and improved treatment approaches. However, in Quebec as is the case in many jurisdictions, access to patient data is difficult for two main reasons. First, patient consent is required, and second, patients typically have their medical data spread across multiple source systems in multiple institutions, which makes it difficult to piece together their complete medical history. Moreover, in most contexts patients do not know who has access to their data and they cannot control access rights.

Opal (opalmedapps.com) is a patient portal developed at the Research Institute of the McGill University Health Centre (RI-MUHC) that provides patients with access to some of their medical data at the MUHC. Opal's long-term roadmap calls for carefully developed infrastructure to link multiple hospitals simultaneously so that patients will be able to access their medical records that are stored in different institutions. However, the originally-designed infrastructure does not allow patients to control access to their data and contribute them for research. Therefore, in this thesis project, we explored the design and development of a new infrastructure for secure and user-controlled personal medical data sharing. Initially, we studied the architecture and workflow of the existing Opal platform. Then, we analyzed various modern decentralized tools and technologies for storing and controlling personal data. Based on the analysis and knowledge acquired, we designed and implemented a novel prototype system for controlling and sharing personal medical data with researchers using a blockchain-based infrastructure.

The blockchain is a tamper-proof mechanism for storing data in an immutable way by using cryptographic and network technologies. As described in this thesis, our novel data-sharing infrastructure is designed to record (1) the permissions that each patient gives to research study personnel to access their data, (2) the data-access privileges that a "public trust" committee provides to researchers to access shared data, and (3) the data access logs of researchers who access the shared data.

Thus, patients can contribute their data to a specific research study by providing electronic consent in a patient portal such as Opal and by specifying which of their data records they wish to share. The system is designed to allow patients to withdraw their consents at any time and stop further sharing of their data if they change their minds. Also, as all data access requests are automatically recorded on the blockchain, each patient has the ability to know who accessed their data and when.

Abrégé

De nos jours, presque tous les hôpitaux et cliniques des pays développés stockent les données médicales personnelles des patients enformat numérique tout en tenant compte des mesures de sécurité appropriées et en conformité avec les exigences légales. En raison du développement rapide des outils de recherche utilisant les mégadonnées, tels que l'intelligence artificielle et l'apprentissage automatique, ces données ont le potentiel de profiter à la recherche médicale dans la recherche de nouveaux médicaments et pour améliorer les approches thérapeutiques. Cependant, au Québec comme c'est le cas dans plusieurs juridictions, l'accès aux données des patients est difficile pour deux raisons principales. Premièrement, le consentement du patient est requis, et deuxièmement, les données médicales des patients sont généralement réparties sur plusieurs systèmes informatiques localisés dans plusieurs établissements, ce qui rend difficile la reconstitution des antécédents médicaux complets. De plus, dans la plupart des situations, les patients ne savent pas qui peut accéder à leurs données et ne peuvent contrôler les droits d'accès.

Opal (opalmedapps.com) est un portail patient développé à l'Institut de recherche du Centre universitaire de santé McGill (IR-CUSM) qui permet aux patients d'accéder à certaines de leurs données médicales au CUSM. La feuille de route à long terme d'Opal nécessite une infrastructure soigneusement développée afin derelier plusieurs hôpitaux simultanément et pourque les patients puissent accéder à leurs dossiers médicaux stockés dans différentes institutions. Cependant, l'infrastructure conçue à l'origine ne permet pas aux patients de contrôler l'accès à leurs données et de les partager pour la recherche. Par conséquent, dans ce projet de thèse, nous avons exploré la conception et le développement d'une nouvelle infrastructure pour le partage de données médicales personnelles sécurisées et contrôlées par l'utilisateur. Dans un premier temps, nous avons étudié l'architecture et le déroulement des opérations de la plate-forme Opal existante. Ensuite, nous avons analysé divers outils et technologies modernes et décentralisés pour le stockage et le contrôle des données personnelles. Sur la base de l'analyse et des connaissances acquises, nous avons conçu et mis en œuvre un nouveau prototype pour contrôler et partager des données médicales personnelles avec des chercheurs à l'aide d'une infrastructure basée sur la chaîne de blocs (blockchain).

La blockchain est un mécanisme inviolable permettant de stocker des données de manière immuable en utilisant des technologies cryptographiques en réseau. Comme décrit dans cette thèse, notre nouvelle infrastructure de partage de données pour Opal est conçue pour enregistrer (1) les autorisations données par le patient au personnel des études de recherche pour accéder à leur données, (2) les privilèges d'accès aux données qu'une "fiducie publique" fournit aux chercheurs avant d'accéder aux données partagées, et (3) les relevés d'accès aux données de tous les chercheurs qui accèdent aux données partagées.

Ainsi, les patients peuvent partager leurs données à une étude de recherche spécifique en fournissant un consentement électronique dans Opal et en spécifiant quelles données ils souhaitent partager. Le système est conçu pour permettre aux patients de retirer leur consentement à tout moment et de cesser le partage de leurs données s'ils changent d'avis ultérieurement. De plus, toutes les demandes de données sont automatiquement enregistrées sur la blockchain, chaque patient a la possibilité de voir qui a accédé à ses données et quand.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor, Dr. John Kildea, for the continuous support throughout my M.Sc study and research. I am extremely grateful for the time and effort he spent with me on this work and for the invaluable help and guidance he provided. It was a great privilege and honor to work and study under his supervision.

Besides my supervisor, I would like to acknowledge the members of the PARTAGE team: Luc Galarneau, Andrea Laizner, Kayla O'Sullivan-Steben, Susie Judd, Tristan Williams, Briana Cabral, and Brandon Woolfson. I am thankful for the fruitful weekly meetings and conversations that were inspiring and motivating me. I appreciate all your kind feedback and advice that helped me to improve this project.

I thank the Opal development team, especially Yick Mo, Dominic Bourdua, Victor Matassa, Yuan Chen, and Stacey Beard, for providing technical support and helping me to resolve any Opal-related issues. I am grateful for all the knowledge and insights you have imparted to me.

Of course, my appreciation extends to the NICE-ROCKS group. I am thankful for their comprehensive and thoughtful feedback on my project during our weekly meetings.

Finally, and most importantly, I am exceedingly grateful to my parents and sister for their endless love and support throughout my two years of Master's study. I give my heartfelt appreciation and gratitude to each of them for their conscientiousness and encouragement throughout this process.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

We are living in an era where data-based techniques (e.g., machine learning, statistical methods, data science, etc.) help humans perform intellectual tasks with enhanced speed, precision, and effectiveness. These techniques have been around for almost 50 years. However, they have started being actively used only in recent years thanks to modern computer processing power and a sufficient quantity of data. Healthcare is one of many fields where artificial intelligence and data-based techniques are employed, helping researchers find new treatment methods and medications. In this thesis project, we designed and developed a working prototype version of a secure and user-controlled medical data-sharing platform in order to allow patients to share their medical data for research purposes.

1.1.1 Real-world Data and Evidence

In clinical research and healthcare delivery, researchers and clinicians use scientific methods to explore and invent new medical treatments, surgical techniques, drugs, biomedical devices, and medications. These methods include a variety of experimental techniques to prove or refute hypotheses. One of the tried and trusted techniques is the clinical trial—a research study of human subjects that is designed to determine if new biomedical or behavioral interventions are safe and effective [1, 2]. When conducting a clinical trial, researchers are eager to collect as much data as possible in order to strengthen the analysis results and optimize the risk-benefit ratio of the treatment being studied [3, 4]. However, clinical trials require many resources, such as personnel (and hence funding) and time [5, 6]. Resource availability affects the quantity of data collected (i.e., the number of patients recruited) and, consequently, the applicability of the trial's findings.

In recent years, technological advances have broadened the possibilities available to clinical researchers. For example, in 2018, the United States Food and Drug Administration (FDA) introduced a new framework to accelerate medical product development and to bring about new innovations and advances faster using real-world data (RWD) and real-world evidence (RWE) [7]. RWD is a term that describes observational data obtained outside regulated clinical trials and generated during routine clinical practice. RWD may come from a number of sources including electronic health records (EHRs), insurance billing and claims, patient-reported outcomes (PROs), and biometric monitoring devices (such as smartphones and smartwatches). Using these unconventional sources of data, new treatment methods can be evaluated under real-world conditions in larger populations and at much lower cost than is possible with typical clinical trials. RWD are closely related to RWE—the clinical evidence regarding the usage and potential benefits or risks of a medical product derived from the analysis of RWD. RWE can be generated by different study designs and analyses, including but not limited to, randomized trials, including large simple trials, pragmatic trials, and observational studies (prospective and/or retrospective). Thus, the FDA's new framework along with available RWD and associated RWE help medical product developers to expand on the clinical trial process. Also, it helps to monitor and evaluate the safety of previously-approved therapies and satisfy post-marketing study requirements.

1.1.2 Comparative Effectiveness Studies

In 2009, the United States Institute of Medicine published a report on comparative effectiveness research (CER) intending to assist consumers, clinicians, purchasers, and policymakers to make informed decisions to improve health care at both the individual and population levels [8]. The purpose of CER is to determine which existing healthcare techniques and treatments work best for which patients and which pose the greatest benefits and harms. CER generally focuses on broader, more heterogeneous populations (i.e., real-world populations) than traditional clinical trials with the goal to provide evidence about the effectiveness of a new medical approach under consideration. Although clinical trials, in particular randomized control clinical trials, do provide evidence about effectiveness, they are typically based on a small number of patients that meet a rigorous set of eligibility criteria. CER allows researchers to address the question of which of the available treatments is most effective for a specific patient rather than the question of whether a treatment works or not. Thus, many CER studies heavily rely on RWD [9]. However, it is not trivial to get access to or collect such RWD.

1.1.3 Data Science and Big Data in Healthcare

Data science is a multidisciplinary field that combines scientific techniques and theories from many fields, such as mathematics, statistics, data mining, artificial intelligence, Big Data, etc., in order to extract knowledge and insights from data. Data science can trace its origins back in 1962 when John Tukey called for a reformation of academic statistics [10]. Tukey pointed to the existence of an as-yet unrecognized science, whose subject of interest was learning from data, or "data analysis". The term "data science" was proposed by Peter Naur in 1974 and became more widely used at the beginning of the 2000s due to an expansion of statistics beyond theory into practice [11, 12, 13]. Modern data science has a close relationship with so-called Big Data—a field that deals with very large, diverse data sets that include structured, semi-structured, and unstructured data, from multiple sources, and in different sizes ranging from terabytes to zettabytes. Big Data challenges include capturing data, data storage, data analysis, searching, sharing, transfering, visualizing, querying, updating, information privacy, and data source handling. Big Data has become an important tool for research and business, where data scientists are responsible for breaking down big data sets into usable information and creating software and algorithms that help companies and organizations determine optimal operations

[14]. The world's biggest technology companies are constantly analyzing their data to produce more efficiency and develop new products. In 2010, the global data-driven industry was worth more than $100 billion and was growing at almost 10% a year: about twice as fast as the software industry as a whole [15].

Data science techniques and Big Data have had a huge impact on healthcare research. With the rapid integration of so-called mHealth (mobile health), eHealth (electronic or digital health) and wearable technologies in healthcare systems, the volume of healthcare data is constantly increasing. This includes EHR data, imaging data, patient-generated data, sensor data, and other forms of difficult-to-process data. These data help data analysts and data scientists to improve healthcare by providing personalized medicine and prescriptive analytics, clinical risk intervention analytics, and predictive analytics, as well as automated external and internal reporting of patient data, and many other data-driven applications [16, 17]. One of the first applications of data science and Big Data in healthcare was made by Google in 2008. The company developed a web service that provided estimates of influenza ("the flu") activity by aggregating Google Search queries [18, 19]. Although the predicting system was sometimes very inaccurate and was eventually discontinued in 2015, it demonstrated the potential of data science in healthcare [20, 21]. In 2015, Google used Big Data and data science to identify breast cancer tumors that metastasize to nearby lymph nodes. Their deep learning–based approach called Lymph Node Assistant (LYNA) was able to correctly distinguish a pathology slide with metastatic cancer from a slide without cancer 99% of the time [22]. As of summer 2021, the most recent applications of data science and Big Data were demonstrated during the COVID-19 pandemic. Various technologies have been used to minimize the impact of the disease, including minimizing the spread of the virus, case identification, and development of medical treatments [23, 24, 25, 26]. However, despite the benefits of Big Data and data science, the use of these technologies in healthcare has raised significant ethical challenges ranging from risks for individual rights, privacy and autonomy, to transparency and trust [27, 28].

A typical data science project has a certain life cycle pattern that can involve different phases. According to Prof. Ruths at McGill University, the data science pipeline includes six phases that

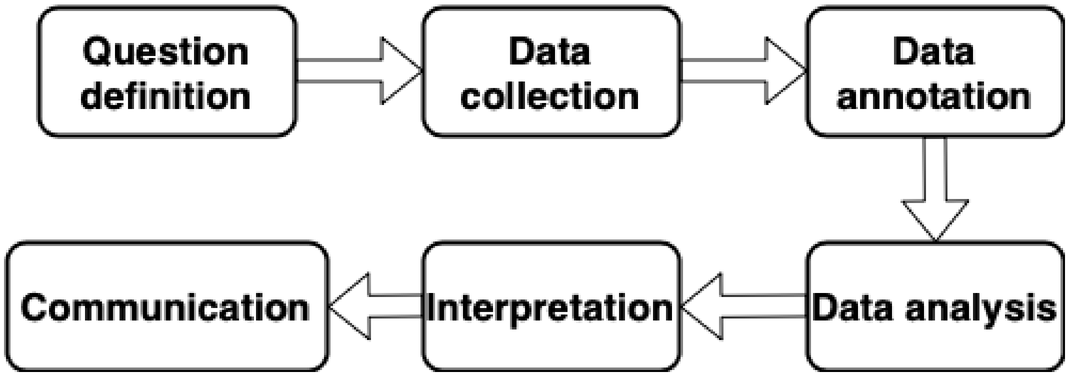


Figure 1.1: Data science project six-phase pipeline [29].

are shown in Figure 1.1 [29]. The first phase of the pipeline is the **question definition**. It is the most crucial and important step that defines the problem that the data scientists are trying to solve. A poorly formulated question can lead to issues in the following phases and poor results for the whole project. Each word of the question must be "clear" and understandable without any vagueness or ambiguity. At the same time, the question must be defined in a way that can be measurable and possible to interpret in numbers. The second phase of the pipeline is **data collection**. This step includes obtaining and arranging the data so they can be worked with. Namely, in accordance to the question definition, data scientists must decide what data to collect, obtain the raw data, identify anomalies in the data, correct issues in the data, and standardize the structure of the data. The third phase is **data annotation**, which involves inferring features that will be used for analysis. This step can include creating new features. There are three primary ways of generating features: (1) human annotation (e.g., crowd-sourced annotation or by involving experts), (2) programmatic (e.g., machine learning), and (3) mixed human and programmatic annotation. The data collection and data annotation phases are very much related. Therefore, a good data science project spends about 60%-80% of the project time in these two phases [30]. The fourth phase is the **data analysis**, where the goal is to generate the results or findings. Data scientists create models and visualizations using existing statistical methods and machine learning techniques. The fifth phase is the **interpretation** phase and it puts the analysis results into the context of the objective(s) of the project. The goal of this step is to explore how the analysis findings relate to the original question. The last phase of the pipeline is **communication**, where data scientists present and explain the insights of the project in the form of a report, publication, or presentation. A good data science project is an iterative cyclic process, where the results at each phase are continuously assessed. If an implemented phase produces bad results or does not relate to the underlying problem, then the approaches of the previous phases must be analysed and re-implemented [31].

1.1.4 Problems: Data are Spread Out and Difficult to Access Legally and Technically

As discussed above, in the last few decades, clinical research and healthcare have been enhanced with new techniques and methodologies thanks to digitization and technological advances. However, it is difficult to fully exploit these techniques and methodologies due to numerous technical and legal challenges that are discussed in this section.

Despite the popularization of eHealth and electronic health records (EHRs), it is hard to aggregate medical data (the data collection phase) into an electronic data set or database. First of all, healthcare data are often fragmented among different sources. Since a patient can visit different healthcare providers over the course of their treatment, their medical data are spread out among different organizations. This makes it difficult to automate the data collection process due to technical issues (e.g., organizations' firewalls and security measures) and leads to another challenge. Different healthcare providers may store their patients' records in different formats, which makes data aggregation from different sources even more complicated, even within a single institution. Although the use of digitized and structured data simplify this process, each healthcare provider may use different metadata naming conventions (e.g., Microsoft Excel table headers, database table names or database fields, etc.) that require merging solutions. Fast Healthcare Interoperability Resources (FHIR) is a modern standard created by the Health Level Seven International (HL7) healthcare standards organization that helps healthcare providers to store data in a single format. The standard describes data formats and elements (known as "resources") as well as interfaces for exchanging EHRs [32]. The practice of storing medical data in a standardized format has been popularized only in recent years, therefore many (if not most) healthcare providers and software vendors currently store data in their internal formats. Another issue relates to the constantly changing data. For example, patients and physicians move, change their names and professions, retire and die. Thus, healthcare data are not static, and most elements will require relatively frequent updates in order to remain clean, complete and current. The last, but not least, technical challenge is to ensure that aggregated datasets are accurate, correct, consistent, relevant, without duplications, and not corrupted in any way.

Besides the technical challenges, the data collection process often faces various legal hurdles. Since medical records contain sensitive information about individuals, these data are considered personal data or personally-identifiable information. Each legal jurisdiction has its own regulations for storing, processing, and accessing personal data. Thus, not anyone is allowed to access such data. For example, in Canada, data protection laws and regulations are encapsulated in a complex set of federal and provincial statutes. These laws and regulations include statutes of general application, as well as sector-specific statutes, such as health privacy regulations, and related regulations such as anti-spam and consumer protection regulations [33]. Therefore, there are varying definitions of "personal health information" under provincial health privacy laws, which generally relate to identifying information about an individual related to their physical and mental health. In Quebec, the medical records of patients are considered confidential, so only patients and their treating clinicians can access them. Patients must, in most circumstances, give permission for other people to see their medical records [34].

An example solution that partially solves the aforementioned technical issues is the Opal application developed at the Research Institute of the McGill University Health Centre (RI-MUHC). In 2014, McGill Computer Science professor Laurie Hendren, as a cancer patient, encountered a lack of effective personal medical data management [35]. Prof. Hendren's medical data were spread out among different healthcare providers, so she had to organize all of her medical records by herself. Also, it showed to her that patients do not have the power of controlling their medical records. Thus, along with Dr. John Kildea (medical physicist) and Dr. Tarek Hijal (radiation oncologist), Prof. Hendren founded the Opal Health Informatics Group, which has been developing the Opal patient portal platform (opalmedapps.com) for Quebec patients [36]. The Opal platform aggregates various fragmented data that it has access to, which in turn enables patients to store, organize and share their health data with their caregivers.

1.2 Thesis Goals

Due to the popularity and increased exposure of Opal, the number of Opal users is constantly increasing. As of Summer 2021, the Opal app has about 1,700 users. Because of this, the amount of medical data processed by the platform is growing. As discussed earlier, data such as these are a crucial component of modern healthcare research methods and techniques. For example, these data could be used for clinical trials, CER, or for a data science project. Thus, researchers have a high interest in data that are aggregated and accessed by a platform such as Opal.

Although Opal aggregates medical data and partially solves the aforementioned technical issues, these data cannot be used for research without patients' consent. Researchers have to get patients' consent to use their personal medical data in each specific research study. However, patient consent does not guarantee the safety or confidentiality of personal data. For example, a healthcare provider's or a researcher's system can be compromised by a malicious actor to get access to patients' data without consent.

The founders of Opal have outlined its core principles as a patient-centric app, where the data belong to a patient who controls them. Therefore, following Opal's core principles, the goals of this thesis are as follows:

- Participate in a process of stakeholder co-design among patients, clinicians, and researchers to co-design the requirements of a secure and patient-controlled data-sharing system called the "Research Portal".
- Design, develop, and deploy the Research Portal as a prototype solution for providing researchers with access to Opal users' data for research purposes. The system must store (1) patients' electronic consent (e-consent), (2) researchers' data-access privileges, and (3)

researchers' data-access history in a tamper-proof manner. The system's storage mechanism must be immutable (append-only) to ensure the integrity of the record of consents, privileges and data accesses. At the same time, a tamper-proof solution must guarantee that Opal users' data, as stored in the host healthcare institution cannot be used or modified even by the institution's IT staff or by other users without authorized access. However, using the Research Portal a system administrator should be able to set the privileges for accessing the shared data for each researcher participating in a study. The system should allow patients to withdraw their consents and stop further sharing of their data if they change their minds.

- Analyze modern decentralized technologies and their possibilities for storing data in an immutable (append-only), tamper-proof manner so that Opal users may confidently control access to their data for research use.
- Provide recommendations for further improvements to the Research Portal based on the knowledge gained from the stakeholder co-design and development processes. Also, provide guidance on deploying a production-ready solution.

1.3 Structure of the Thesis

This thesis is organized as follows:

In ***Chapter 2***: we introduce the Opal patient portal app and platform as well as its features, infrastructure, and workflow. Also, we discuss the concepts and existing solutions for data-sharing and data-access control. Moreover, we present the co-design process for designing and building a prototype of the Research Portal—our data-sharing service in the ecosystem of the existing Opal patient portal but extendable to any similar ecosystem.

In ***Chapter 3***: we provide an overview of decentralized technologies. First, we present one of the earliest and best-known decentralized systems, the World Wide Web, and discuss its transformation into a centralized system. Also, we discuss the problems of the centralized Web and issues related to personal data. Then, we present modern decentralized platforms, such as Solid and blockchain. We analyze in depth these two decentralized technologies and evaluate the complexity of using them to support our Research Portal needs.

In ***Chapter 4***: we present the design process of a patient-controlled data-sharing service—the Research Portal. First, we outline the desired solution and list its high-level requirements. Then, we discuss modern software development methodologies and define the functional requirements of the system as text use cases. The text use cases are visualized in the form of use case diagrams and a sequence diagram. Also, we present the technologies used for developing the Research Portal for Opal-sourced data. In detail, we explain the Hyperledger Fabric platform for running a private permissioned blockchain network. At the end of the chapter, we present the design of our prototype blockchain network and the Research Portal.

In ***Chapter 5***: we present the results of this thesis project. The chapter describes the developed prototype Research Portal—a secure and patient-controlled system for data sharing. We explain the existing features and its workflow. Also, we discuss the results of the co-design process and its impact on the technology development. At the end of the chapter, we discuss the feasibility of the blockchain solution.

In ***Chapter 6***: we summarize the major points of this thesis project and discuss the remaining challenges to be addressed within the scope of the overall Opal patient portal platform project.

Chapter 2

Background

2.1 Opal App Infrastructure and Workflow

Opal is a mobile phone application that acts as a patient portal. It was initially developed for radiation oncology patients at the McGill University Health Centre (MUHC) but its use is being expanded to encompass all types of patients. The app allows patients to access their personal health information, such as appointment schedules, lab results, personalized educational material, radiotherapy treatment planning views, and more. Figure 2.1 illustrates some of the Opal app's screens.

The app has five information categories arranged as tabs: "Home", "My Chart", "General Information", "Educational Materials", and "Account Settings". The "Home" screen provides notifications and up-to-date pertinent information personalized to the individual patient, such as next appointment, status of treatment/treatment planning, and waiting room management. The "My Chart" screen contains personal health information specific to the individual patient, including appointment schedule, doctors' notes, radiotherapy treatment plan views, lab test results, and secure messages from the treating team. The "General Information" page includes information regarding the hospital that is not personal to an individual patient. The "Educational Materials" screen has relevant, just-in-time educational material that is specific to the individual patient's diagnosis and phase of treatment. The fifth tab is the "Account Settings"

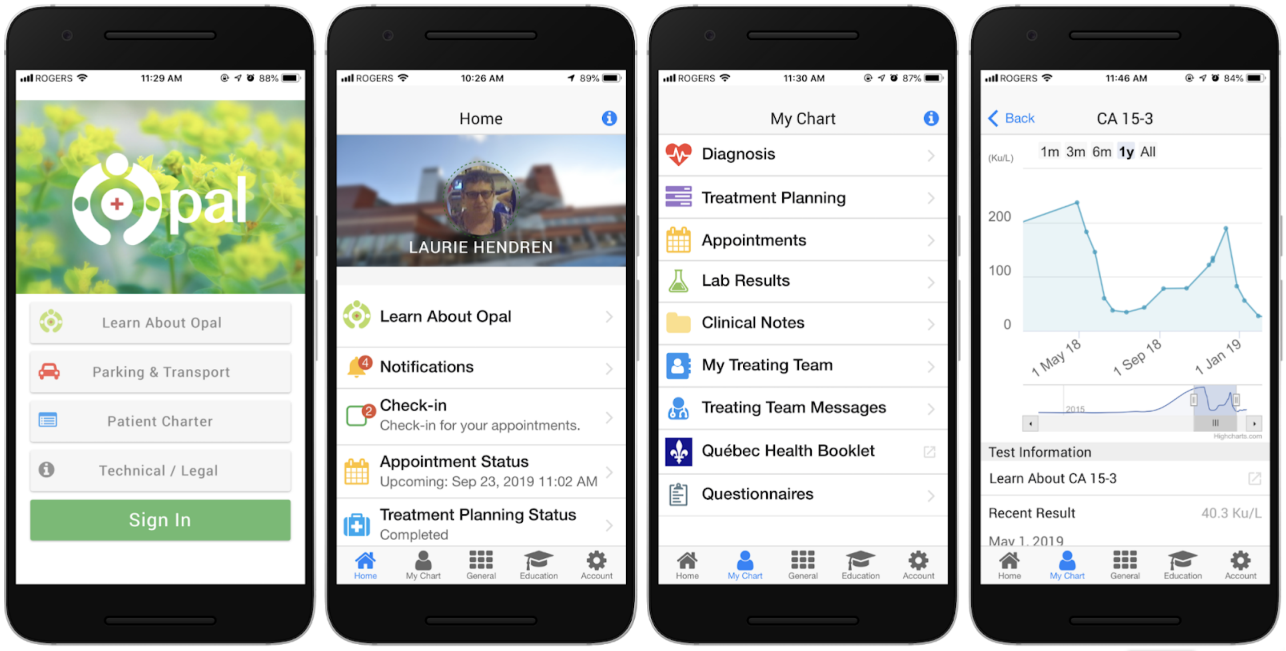


Figure 2.1: Illustration of the Opal app screens. The first screen shows the application's landing page. The second screen illustrates the "Home" page, which is shown to users after signing in. The third screen shows "My Chart" tab that contains patients' information, and the fourth screen is an example of visualized lab results data.

screen that provides tools to allow the patient to change preferences such as language, font size and password. Table 2.1 lists Opal's features and functionality grouped by categories.

Also, Opal has features that are under development and not available to its users yet. One of these new features, called "all-in-one", will allow Opal patients to access their data from multiple healthcare institutions simultaneously in a single app. The all-in-one solution will initially connect to two hospitals in Montreal: the MUHC and Ste-Justine hospital. Using Opal's multi-institution data communication infrastructure, it is intended that patient data, linked and standardized in Opal, may be securely copied to a research database and accumulated there for research studies. Figure 2.2 shows the concept of the 'All-in-one' solution.

Table 2.1: Categories of information and features/functionality provided to patients via Opal.

Category (Menu/tab)	**Features & Functionality**
Home Screen/Overview	Next appointment Notifications (e.g., new document, new message, etc) Posts (messages from treating team and general hospital announcements) Status of treatment/treatment planning Waiting room management (check-in, call-in, waiting time estimate)
My Chart	Diagnosis information Notification archive Appointment schedule with appointment location maps Appointment change requests Treatment/Treatment planning information Access to (selected) doctors' notes and nursing notes Lab test results Messages from treating team Secure two-way messaging with clinicians Patient-reported outcome questionnaires
General Information	Facility to update contact information (personalised on login) General hospital announcements Patient charter Parking information General hospital maps Way-finding Leave feedback regarding app/portal Facility to report bugs in the app/portal
Educational Material	Videos Booklets Pamphlets/fliers
Account Settings	Language preference Font size Change password

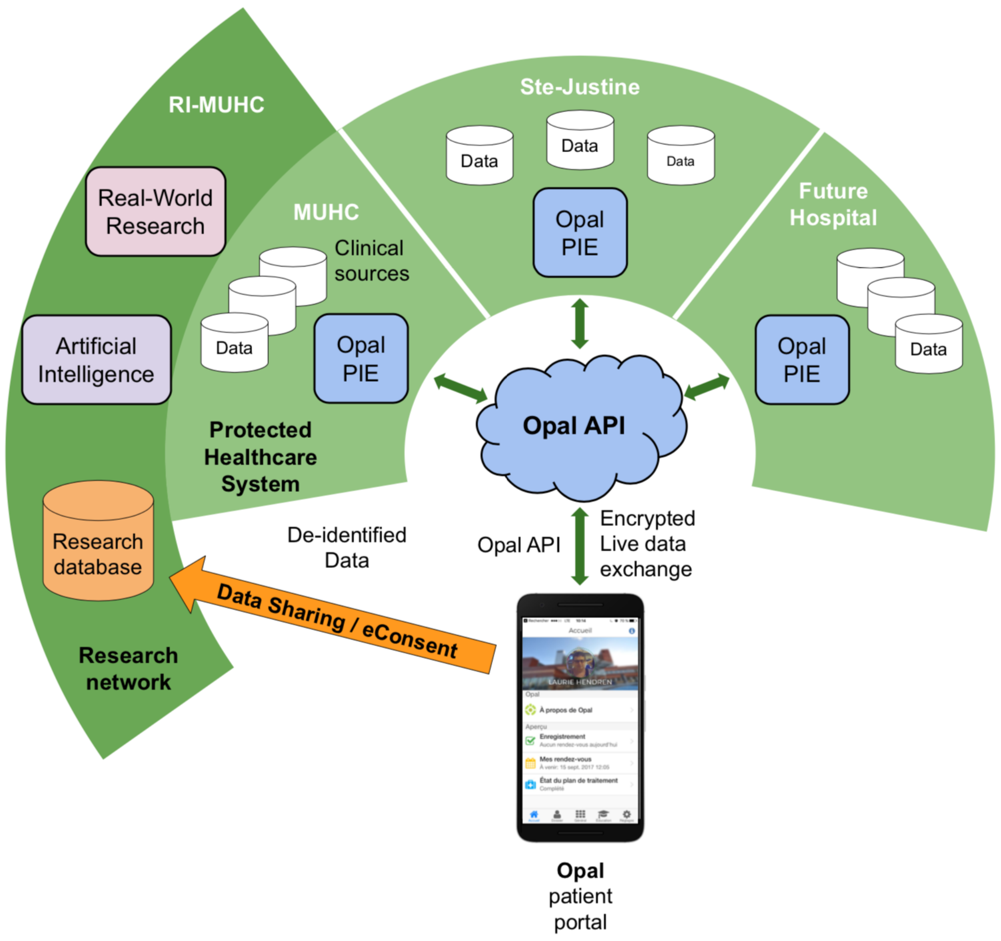


Figure 2.2: High-level architecture of the multi-institutional version of Opal ("All-in-One"). The Opal-PIE or patient information exchange is the backend Opal software that must be installed in each hospital to connect the patient app with the hospital-sourced personal health information for each patient user. The vision of the Opal Health Informatics Group is a solution that provides patients with direct data-sharing capabilities via e-consent for research data use.

Figure 2.3 presents a schematic of Opal's communication architecture. To securely serve data to the Opal app and isolate access to the EMR source databases (e.g., Aria, ORMS, etc.), the architecture contains a custom-developed database, known as OpalDB. The database is run in the MySQL database management system and hosted on a Linux server with automatic mirroring to an independent fail-over server and nightly backups to a backup server. The database and server are internal to the hospital's firewall.

Selected data are transferred from the EMRs to OpalDB using the "Auto Update" scheduler (essentially customizable Linux cron jobs) that copies across approved appointments, tasks, documents, lab test results, etc. All of the data that reside in OpalDB are accessible to the patients that own them. Therefore, it is important to strictly control their insertion into OpalDB. The OpalAdmin Publish Manager provides this control. It is a secure web application that allows the clinical team to (a) control the personal health information that will be shared with patients (b) create content (educational material, posts, questionnaires, etc) and apply personalization rules, and (c) create tagged data (such as maps and appointment explanations) that will be used to supplement the personal and personalised data.

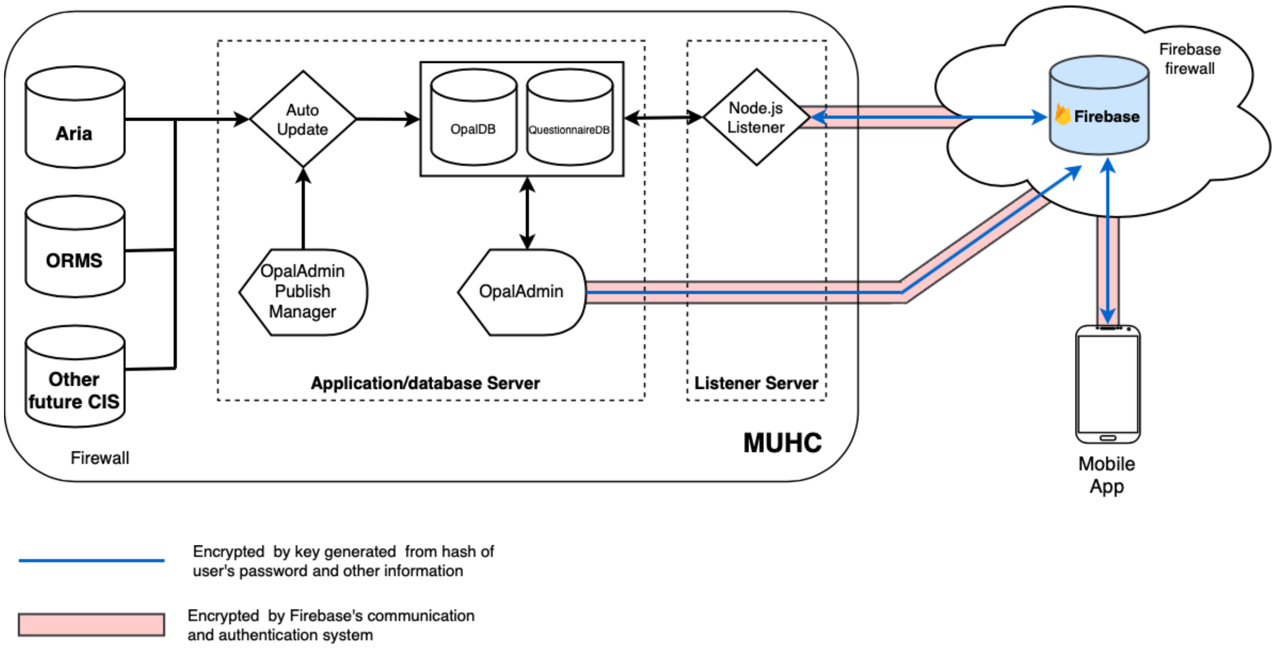

Aria
ORMS
Other future CIS
Firewall
Auto Update
OpalDB
QuestionnaireDB
OpalAdmin Publish Manager
OpalAdmin
Application/database Server
Node.js Listener
Listener Server
MUHC
Firebase firewall
Firebase
Mobile App
Encrypted by key generated from hash of user's password and other information
Encrypted by Firebase's communication and authentication system

Figure 2.3: Opal's communication architecture. The dashed line outlines Opal's back end software known as the Opal PIE (Patient Information Exchange).

Secure serving of data through a hospital's firewall to Opal is facilitated by Firebase, a real-time backend-as-a-service cloud database operated by Google Inc. [37]. Firebase is designed such that all applications that are connected to it are served data in real time. As such, if any data on Firebase are changed, all connected applications immediately see the change, and vice-versa. Also, Firebase provides authentication tools and Secure Sockets Layer (SSL) encryption.

Therefore, Opal users' security and confidentiality are achieved at several levels. First, all data sent to/from the hospital and to/from Opal are encrypted by Opal's infrastructure (Listener or patient app) using the Advanced Encryption Standard (AES) algorithm. A hash representation of the user's password, generated using the Secure Hash Algorithm SHA256 and other information, is used as the key to encrypt all data. The second layer of security is provided by Firebase itself, which communicates on an SSL layer. Encrypting the patient's data, before sending them through Firebase, ensures that all data within Firebase are encrypted and cannot be read by Firebase employees. Furthermore, in the event of a compromise (e.g., password retrieval from an Opal user through a phishing attack), only the patient's own data are compromised.

The Listener is a *node.js* script that monitors and communicates with Firebase. When the user-authenticated app requests data, it places a token on Firebase that is immediately seen by the Listener. The Listener then fetches the appropriate data from OpalDB, encrypts them and copies them to Firebase, where they are immediately propagated to the Opal app. As soon as delivery is complete, Opal deletes the data from Firebase. In the event of a disruption in the connection to the patient's device, the Listener deletes the data from Firebase after five minutes. This ensures that stale patient data (albeit encrypted) are never sitting on Firebase. In general, the Listener pushes data outward from the hospital. However, certain data types, such as patient-reported outcomes (PROs) questionnaire responses and appointment check-ins are accepted inward.

OpalAdmin is a roles-based web interface that is internal to each hospital where the Opal platform is installed. It is used by the clinical teams to set the rules to publish data to patients and to see which data patients have accessed. Also, it is used by superusers and the hospital security team to access Opal's activity logs. All patient logins, data access requests and data changes are logged in OpalDB using MySQL's Triggers function. The activity log storage mechanism was one of the sources of inspiration for this thesis project. Since logs stored in regular database storage can be modified by compromised superusers or people who get unauthorized access, such an approach does not guarantee the immutability of previously-added log records. Thus, the log may not be a reliable source of historical system events (e.g., as evidence to be used in court). This potential limitation was a motivation to explore storing activity logs in an immutable, tamper-proof manner.

2.2 Data Sharing

Nowadays, every person generates massive amounts of personal data. The vast majority of these data are generated from digital technologies. This includes wearable devices, online activity (e.g., social media posts), banking, e-commerce, EHRs and many others. As discussed in Chapter 1, these data can be significant for academic research, particularly in healthcare. Historically, it was difficult to access such data due to several reasons: (1) the absence of established frameworks for transferring personal data to researchers, (2) the absence of choice for individuals to decide with whom to share their data, (3) technical and legal issues. A solution to these problems is user-controlled data sharing (a.k.a. data donation).

Data sharing is an act of an individual who contributes their personal data for research purposes by providing consent. People voluntarily allow transfer of their data that were generated for a different purpose to a collective research dataset. This allows researchers to use otherwise private data for the benefit of society. The “donation” concept comes from medical fields and science, where organs and blood are often donated. Therefore, the term “data donation” can be used to increase awareness on the need to make personal data available for scientific research.

In 2019, the Opal team conducted a survey among Opal users about their interest in new features [36]. The results showed that 177 patients out of 260 were positively interested in having an option to anonymously share their medical data for research. According to other research studies, many people are willing to share their data and biospecimens for research that could benefit the wider general public [38, 40, 39]. Anya Skatova’s research shows that social duty is the biggest reason to share personal data [41]. At the same time, understanding how personal data will be used plays a significant role in the decision of the participants to share their data. This includes legal and technical aspects to ensure data are not passed on to unauthorized persons. Also, giving participants the power to decide who has access to their data, when, and for how long, will increase their trust and willingness to participate. Technologies like APIs (Application Programming Interfaces) allow for direct computational access to data sources and make it possible that the data sharing may involve providing access to an ongoing data stream. In these cases, it is important to inform participants about how long this authorization will remain active and what options the participant has to revoke access in the future [42].

2.2.1 Related Work

The data-sharing concept is not new and becomes more popular every year. The US website PatientsLikeMe (patientslikeme.com) was one of the first platforms for patient data sharing/donation. It has been in operation since 2005 and counts over 830,000 data donors with over 2,900 conditions. Members are able to share personal stories and information about their health, symptoms, and treatments. Also, users have the opportunity to learn from the aggregated data of others with the same disease and see how they are doing in comparison with others. Thanks to the collected data, PatientsLikeMe's research team has published more than 100 peer-reviewed scientific articles in leading journals such as the BMJ, Nature Biotechnology, and Neurology [43].

Another popular data-sharing project is Open Humans (openhumans.org) that was founded in 2008. The platform allows users to upload, connect, and privately store their personal data, such as genetic, activity, or social media data. Once users connected their data sources, they can join projects that help to explore and analyse their data. Also, participants can choose to share their data with studies and other projects run by third-party researchers. As of fall 2021, the platform has 12,495 members and 54 tools.

In 2013, an open-source project called Tidepool (tidepool.org) was founded. The platform helps patients with diabetes to retrieve data from over 50 supported devices (e.g., insulin pumps, continuous glucose monitor) and analyse them via Tidepool Web and Tidepool Mobile. The Tidepool Big Data Donation Project allows users to share these data in anonymized form with researchers. Participants can opt in or opt out of the Tidepool Big Data Donation Project at any time. Also, users have the power to decide who has access to their account, and how much access they can have. The platform charges researchers a fee for accessing shared data, and then donates 10% of the proceeds to diabetes organizations. The Tidepool Big Data Donation Project also make some datasets publicly available at no cost to researchers.

The Swiss group Healthbank (healthbank.coop) has been in operation since 2015 and it allows participants to upload their data to its site and from there share them with clinicians and researchers. The platform allows users to provide health data from any source and in any format. Also, participants may stop sharing their data at any time. The main feature of the platform is that users can monetize their personal data. Researchers have to pay users to have access to their data. Users can review the details of the offer and then decide if they are willing to participate in the research or not.

In 2019, Apple released a research app (Apple Research app) that allows users to participate in health studies from their Apple devices. Apple teamed up with different researchers and health organizations that run different health research studies. Thus, users can opt to participate in three health studies: (1) a women's health study, (2) a hearing study, and (3) a heart and movement study. The app puts data sharing in users' control, so they can withdraw at any time. Also, the app encrypts users' data for security reasons, informs users how their data will support the research, and guarantees that the data will not be sold to third parties.

2.3 Data Access Control

In recent years, identity and access management (IAM) have become more prevalent and critical in data management systems as regulatory compliance requirements have become increasingly more rigorous and complex. IAM addresses security concerns that the right subjects (users) access the right objects (resources) at the right times for the right reasons [44]. IAM identifies, authenticates, and controls access for subjects who will be utilizing IT resources, including hardware and applications' employees. Thus, the access control process is a crucial part of the IAM concept.

Access control is generally considered in three successive steps: (a) identification, (b) authentication, and (c) authorization [45]. Identification is an assertion of who someone is or what something is. This can be a person, process, system, or other subject. Identification is only a claim of identity and does not imply that this claim is correct. The username is the most common form of identification. Authentication is the act of verifying a claim of identity. There are three different types of information that can be used for authentication: (1) something that the subject knows (e.g., password, personal identification number (PIN), pass-phrase, etc.), (2) something that the subject has (e.g., magnetic swipe card, USB flash drive, etc.), (3) the subjects's biometrics (e.g., palm prints, fingerprints, voice prints, and retina (eye) scans, etc.). Strong authentication requires providing more than one piece of authentication information (e.g., two-factor authentication). Authorization defines what operations a subject can perform in the context of a specific object (e.g., application). After a subject has successfully been identified and authenticated, a system must determine what informational resources they are permitted to access and what actions they are allowed to perform. The determination is based on administrative policies and access control mechanisms (models). For example, one subject might be authorized to enter a sales order, while a different subject is authorized to approve the credit request for that order.

Access-control models tend to fall into one of two classes: based on capabilities and based on access control lists (ACLs). Both classes have four basic actions: (1) allowing access, (2) denying access, (3) limiting access, and (4) revoking access. However, the way these actions are performed will differ based on the implementation involved. In a capability-based model, access is granted or conveyed to a subject by transmitting such a capability over a secure channel. Subjects have an unforgeable token (capability) that provides access to the object. For example, we can think of a token capability as being analogous to the access card we might use to open the door of a building. Many people can have a token to open the door, but each person might have a different level of access (e.g., access hours). Also, the token can be transferred to a different person. Thus, the right to access an object is based entirely on possession of the token, and not on who possesses it. In an ACL-based model, a subject's access to an object depends on whether its identity appears on a list associated with the object. ACL systems have a variety of different conventions regarding who or what is responsible for editing the list and how it is edited. ACLs are typically built specifically to a certain object. They contain the identifiers of the subjects allowed to access the object and permissions that describe what each subject is allowed to do with the object. For example, ACLs are commonly used in file systems, where subjects have different access permissions: read, write, and execute.

There are many different access control models. The most common models in real-world environments are the following:

- Discretionary Access Control (DAC)—the data owner determines who can access specific objects. For example, a system administrator may create a hierarchy of files to be accessed based on certain permissions.

- Mandatory Access Control (MAC)—the owner of the object does not get to decide who gets to access it, but instead access is decided by a group or individual who has the authority to set access on objects. Usually, MAC is implemented in governmental organizations, where access to a given object is largely dictated by the sensitivity label applied to it (e.g., secret, top secret, etc.).

- Role-Based Access Control (RBAC)—the model is similar to MAC, where functions on access controls set by an authority responsible for doing so, rather than by the owner of the object. The difference between RBAC and MAC is that access control in RBAC is based on the role the individual being granted access is performing. For example, a human resources specialist should not have permissions to create network accounts; this should be a role reserved for network administrators.

- Attribute-Based Access Control (ABAC)—access rights are granted to subjects through the use of policies which evaluate attributes (e.g., subject attributes, object attributes and environment conditions). Completely Automated Public Turing Test to Tell Humans and Computers Apart (CAPTCHA) is an example of ABAC, where a subject has to prove they are a human [46].

- Multilevel Access Control—combination of the simpler access control models (e.g., DAC and MAC). Such access controls are used extensively by military and government organizations, or those that often handle data of a very sensitive nature (e.g., nuclear secrets, health information). In this thesis project, we used this access control model where both DAC and MAC models were used (i.e., patients share their data by signing e-consents and administrators set privileges for accessing the shared data).

2.3.1 Related Work

Various solutions have been proposed to address the security and access control concerns of healthcare [47, 48, 49, 50, 51, 52, 53]. However, in recent years, there has been a proliferation of solutions for the use of blockchain to manage healthcare records both for clinical care and for research. As far as we are aware, all of the solutions involve storing the data themselves on the blockchain.

In 2016, Huiju Wang et. al. proposed an architecture for storing and sharing healthcare data using blockchain technology [54]. They were one of the first researchers who incorporated blockchain technology into the design of a healthcare data system. Their proposed Healthcare Data Gateway (HDG) app's architecture utilizes blockchain technology that enables patients to own, control, manage, and share their own data easily and securely without compromising their privacy. The architecture consists of three layers: (1) the storage layer—a private blockchain network that stores healthcare data, (2) the data management layer—a set of individual's HDGs that are independent but connected to each other, and (3) the data usage layer—entities that use patient healthcare data (e.g., electronic medical record systems and data analytics algorithms). Also, they proposed one purpose-centric access control model, where data requests are expressed using a single unified interface.

Another blockchain-based solution for efficient medical data sharing and data access control was proposed by Kai Fan et. al. [55]. Their solution allows patients to provide doctors with access to their EMRs when they visit different hospitals. For the blockchain network, they developed their own hybrid consensus mechanism which can effectively avoid network congestion.

Also, to assure the security and the privacy of medical data, they developed an encryption solution to store the data on the blockchain. In this model, a requester must have the corresponding decryption key in order to read data from the blockchain.

In 2017, Daisuke Ichikawa et. al. proposed a tamper-resistant mobile health solution using the Hyperledger Fabric blockchain platform [56]. Specifically, they developed an mHealth system for cognitive behavioral therapy for insomnia using a smartphone app. In their solution, medical data were collected via the mobile app in JavaScript Object Notation (JSON) format and sent to their private Hyperledger Fabric blockchain network. Although they ensured that the data on the blockchain were resistant to tampering and revision, the data were not encrypted.

Another healthcare blockchain system was proposed by Griggs et. al. in 2018 [57]. Using a private blockchain Ethereum protocol, they created a proof-of-concept system where sensors can communicate with smart devices (e.g., smart watches, fitness trackers, etc.) that call smart contracts and write the records of all events on the blockchain. The system utilizes contracts to facilitate automatic analysis of the collected data from Wireless Body Area Network (WBAN) devices with custom threshold values for each patient. These thresholds allow the system to trigger alerts for unusual activity. Raw sensor data are aggregated by the master device (a mobile phone) and then sent to nodes in the blockchain for processing by the smart contracts. All data records are stored on the blockchain network, however only authorized entities can access the blockchain for inspection and block verification.

In 2020, SudeepTanwar et. al. proposed (1) a system architecture and algorithm to provide an access control policy for participants to achieve privacy and security, and (2) implementation of a EHR sharing system, based on the blockchain network [58]. Their system utilizes a Hyperledger Fabric blockchain, where its participants have different roles: (1) Patient, (2) Clinician, (3) Lab, and (4) System administrator. Participants can only access records that they have been granted access to by patients. The patients' data are stored on the blockchain and patients can add records using a client application. The client application invokes a chaincode for committing a transaction to the network. Records are updated and visible to every user in the blockchain network.

Thus, many modern data access control mechanisms are based on the blockchain technology. All the blockchain-based solutions described in the literature involve putting the healthcare data themselves on the blockchain. Such an approach allows healthcare providers to store data in an immutable, tamper-proof manner. However, the data can be easily accessed by the blockchain network members.

2.4 User-centered Design: Building a Prototype of the Research Portal

This thesis project was a part of a larger research project called PARTAGE—Patients and Researchers Team Up and Generate Evidence, which was the winning project of the 2019 Trottier-Webster Award for Innovation at the RI-MUHC. The project explores the ethico-legal issues and possible solutions for data sharing. Also, the PARTAGE project aims to develop a secure data-sharing infrastructure, where (1) patients can securely share de-identified data, (2) researchers can request access to previously-consented patient data available via the Research Portal, and (3) the system can permanently document by whom and when the data were accessed.

Following the original development principle of Opal, the PARTAGE project has a patient-centered focus [36]. Therefore, the project utilizes a stakeholder co-design approach, where patients, clinicians, researchers, and software developers are included in the design and development process at all times. This allows the PARTAGE team collaboratively discuss new features and get feedback from all stakeholders, in particular from patients. The team includes four working groups:

- **Stakeholder co-design:** The first working group established the group of key stakeholder participants, including all the co-applicants, patients, Opal software developers, and other researchers at the MUHC. Also, the first working group formulated the project conception and estimated what resources were needed for the remainder of the research project.

- **ELSI:** The second working group has been focusing on the ethical, legal and social issues (ELSI) posed by data sharing. This includes the legality and meaning of electronic consent in Quebec, as well as whether consent can have a broad purpose for all of a patient's data that Opal has access to. Also, the group explores the legal issues surrounding data storage, provenance of data originating in multiple institutions and the right of patients to revoke consent.

- **Technology development:** The third working group has been developing the data-sharing portal based on the second working group's analysis and co-design collaboration. The group includes software developers from the Opal development team who help with the integration of data sharing functionality into the existing patient-facing app. Also, Opal's developers participate in the stakeholder co-design to ensure alignment between the design and development. The design and development phases of this thesis project were a part of the stakeholder co-design and technology development working groups.

- **Pilot:** The goal of the fourth working group is to analyse the hypothesis that patients will be willing to share their data through the deployment and testing of a prototype solution. The group will form a group of test users, who will be asked to provide feedback on the developed solution. Also, test users will be invited to join a focus group to share their experiences with the research team.

Figure 2.4 represents the PARTAGE's participatory co-design flow.

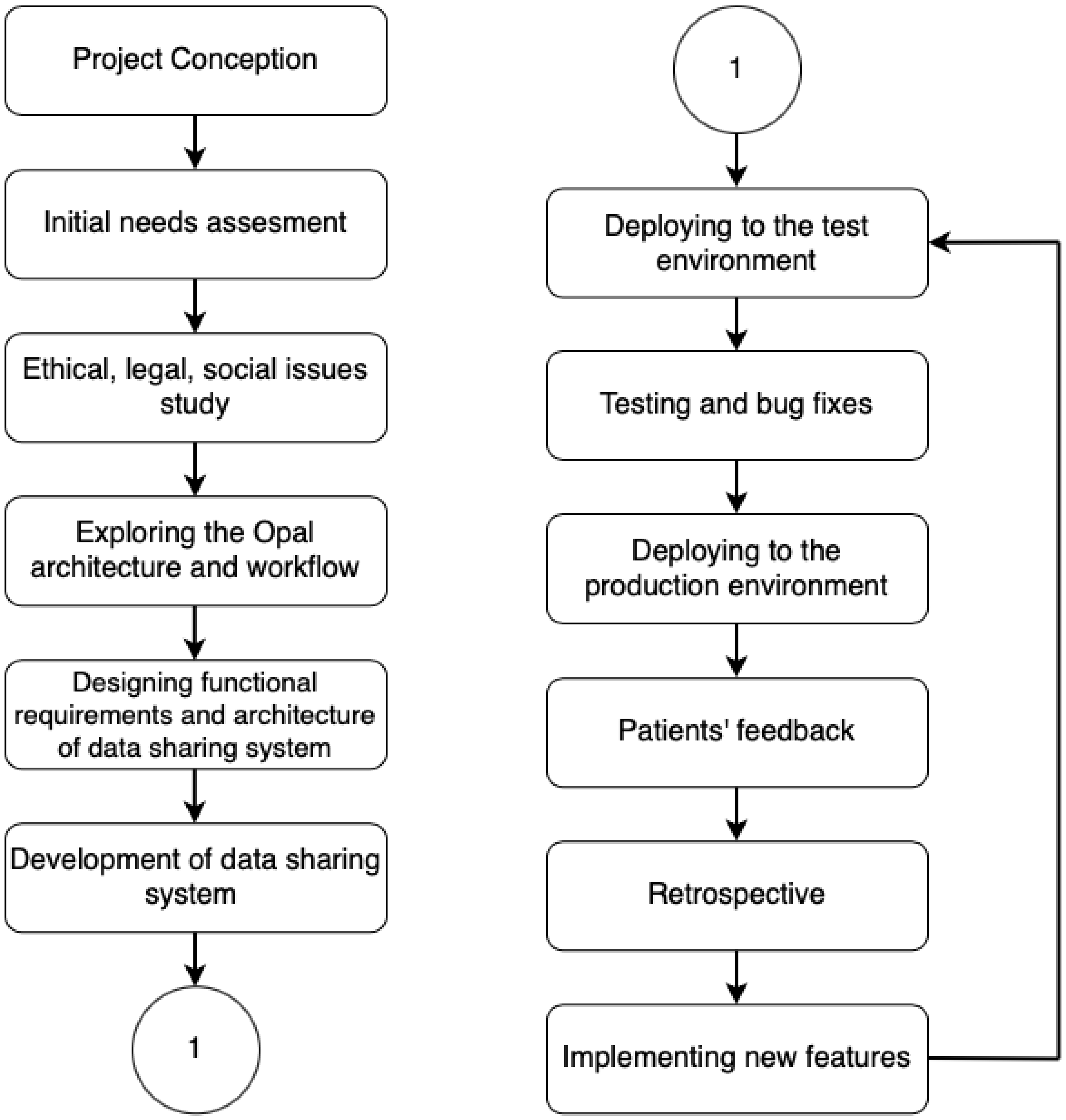

Project Conception
Initial needs assesment
Ethical, legal, social issues study
Exploring the Opal architecture and workflow
Designing functional requirements and architecture of data sharing system
Development of data sharing system
1
1
Deploying to the test environment
Testing and bug fixes
Deploying to the production environment
Patients' feedback
Retrospective
Implementing new features

Figure 2.4: Schematic overview of the participatory co-design process of the PARTAGE research project. The project follows a patient-centered approach, so patients work with the development team to offer new features and explain their concerns. Therefore, the co-design group includes patients, clinicians, researchers, and Opal's software developers to ensure all the parties involved in the project can provide their feedback.

Chapter 3

Overview of Decentralized Technologies

3.1 Background

In 1989, Tim Berners-Lee invented the World Wide Web (the Web)—an information system where web resources are identified by Uniform Resource Locators (URLs) and accessible over the Internet [59]. Berners-Lee envisioned the Web as a decentralized system that is controlled by many participants, such that no one individual or entity can own it, control it, or switch it off for everyone else. In Web 1.0 (the first version of the World Wide Web), any two machines connected to the Internet could send packets to each other without firewalls and other security measures. The web pages thus accessed were static with read-only access. Thus, Web 1.0 was essentially a virtual library, where most of the users were consuming the content, and only qualified users could generate new content. However, Web 1.0 did not meet users' demands, such as instant user interactions, content creation, and e-commerce, and these demands led to Web 2.0—the current Web [60, 61]. Web 2.0 allows users to interact with each other and easily publish user-generated content through centralised services provided by big companies, such as Google, Facebook, Microsoft and Amazon. Figure 3.1 depicts the client-server architecture that is widely used in the current Web.

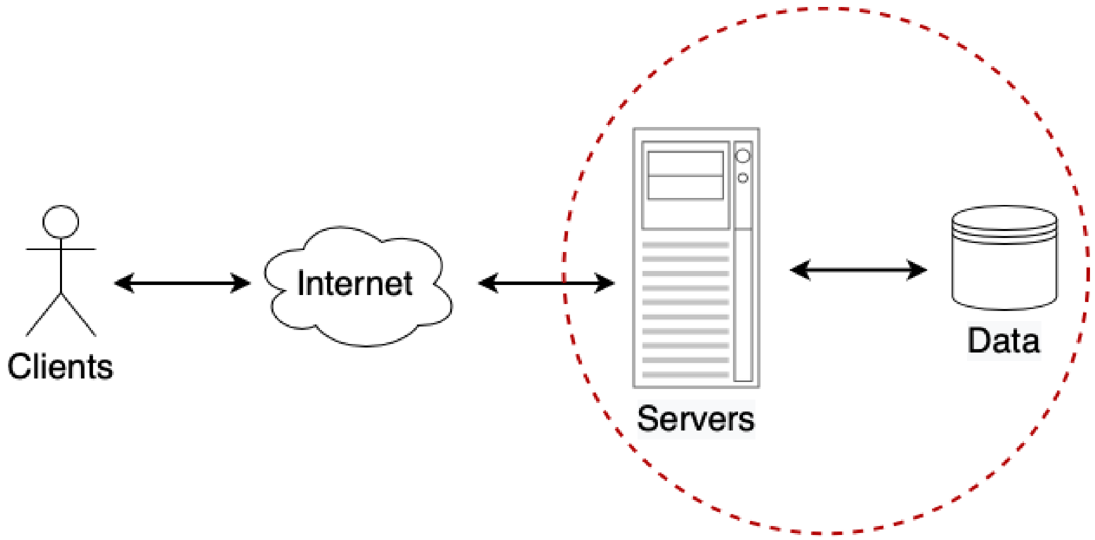


Figure 3.1: The client-server model of Web 2.0 that separates workloads between clients and servers. Clients send requests to servers via the Internet network in order to access services (e.g., connect with other users, online shopping, etc.). Servers are programmed based on business logic that handles clients' requests and provides concrete actions in response. All the clients' service-related data are stored on the servers and controlled by the service providers (red dashed line), so clients do not have full control over them.

Very quickly, Web 2.0 evolved into an effectively centralized ecosystem, in which corporations control, store, and monetize user-generated data and personal information. Business companies develop web platforms and mobile applications (apps) that attract users who create profiles by uploading their personal data. Often, those platforms and apps allow the businesses behind them to collect from users' devices other data and generate statistics on the users over time, such as user activity, metadata, etc. The personal user information is sent to the servers and stored in the databases of business companies. Table 3.1 demonstrates the differences between decentralized, centralized, and distributed networks.

Collected data become an asset of the business and are used for their internal marketing purposes or sold on to third parties. In some cases, this may be done without user consent. As an example, in 2018, it was disclosed that British consulting firm Cambridge Analytica collected personal data of Facebook users without their consent [62, 63]. The data were used for analytical assistance to the 2016 presidential campaigns of Ted Cruz and Donald Trump. Cases of personal data misuse may also take place in healthcare systems. In July 2021, Financial Times published an article in which they showed that more than 40 companies have accumulated years of detailed medical records from hospitals in the UK [64]. These and many other cases have shown the imperfection of the Web 2.0 model: users' data can be easily misused by businesses for their profits, while users do not have control over their data. Therefore, in 2017, Tim Berners-Lee highlighted the main three challenges for the current Web: (1) taking back control of our personal data, (2) preventing the spread of misinformation, (3) realizing transparency for political advertising [65].

Since Opal is a mobile web application, it is based on the client-server model structure where user's data are stored centrally on a server, so users do not have full control over them. Therefore, in this chapter, we examine and discuss two modern decentralized technologies and the possibility of integrating them into Opal's infrastructure so that Opal users may regain certain control over their health data. Also, we analyze the possibility of using decentralized technology to store patients' data-access consents, researcher privileges, and access logs in a tamper-proof manner.

Table 3.1: Comparison of centralized, distributed, and decentralized systems [66].

	Centralized	**Distributed**	**Decentralized**
Network/hardware resources	Maintained & controlled by a single entity in a centralized location	Spread across multiple data centers & geographies; owned by network provider	Resources are owned & shared by network members; difficult to maintain since no one owns it
Solution components	Maintained & controlled by central entity	Maintained & controlled by solution provider	Each member has exact same copy of distributed ledger
Data	Maintained & controlled by central entity	Typically owned & managed by customer	Only added through group consensus
Control	Controlled by central entity	Typically, a shared responsibility between network provider, solution provider & customer	No one owns the data & everyone owns the data
Single Point of Failure	Yes	No	No
Fault tolerance	Low	High	Extremely high
Security	Maintained & controlled by central entity	Typically, a shared responsibility between network provider, solution provider & customer	Increases as # of network members increase
Performance	Maintained & controlled by central entity	Increases as network/hardware resources scale up and out	Decreases as # of network members increase
Example	Enterprise resource planning (ERP) system	Cloud computing	Blockchain

3.2 Inrupt Solid

In 2009, Berners-Lee proposed an architecture of socially-aware cloud storage for decentralization of the current Web [67]. The proposed architecture implies separation of web applications from data storage, allowing the user to control access to their data regardless of the applications that use them. The applications act as services, which can access users' remote data by using the users' credentials under the control of the user. Berners-Lee implemented the high-level idea in the Inrupt Solid Project [68, 69]. Solid (Social Linked Data) is a web decentralization specification that aims to radically change the way web applications work today, resulting in true data ownership as well as improved privacy [70]. In 2018, Berners-Lee launched a start-up called Inrupt. Inrupt is an open-source platform that helps developers create new applications and services that are built on the Solid specification [71].

The novel aspect of Solid is "Personal Online Data Stores" (Pods) – user-owned data stores for keeping personal data. Users create one or more Pods and grant permission to applications, web services, businesses, etc., as needed, to access their personal data stored within them. The specification empowers users with control of what information is allowed to be accessed by others and allows users to cut off access to anyone whenever they wish. Thus, user data never becomes an asset to anyone except the user. Figure 3.2 illustrates the architecture of the Solid platform.

Pods are hosted on a Solid server that follows the Solid protocol [72]. One Solid server can host many Pods that are fully isolated from each other. Each Pod has its own set of data and access rules and is fully controlled by a user. To host a Pod, users can use a Pod provider (e.g., Amazon, DigitalOcean, etc.) or set up their own personal Solid server. Also, users can have more than one Pod hosted on different servers and move data between their Pods. All of a user's data are linked through their WebID–a unique identifier associated with a specific user. A WebID is an Internationalised Resource Identifier (IRI) that can be dereferenced as a "friend of a friend" (FOAF) profile document serialized in the Resource Description Framework (RDF) [73]. FOAF is a machine-readable ontology that allows groups of people to describe social networks (their

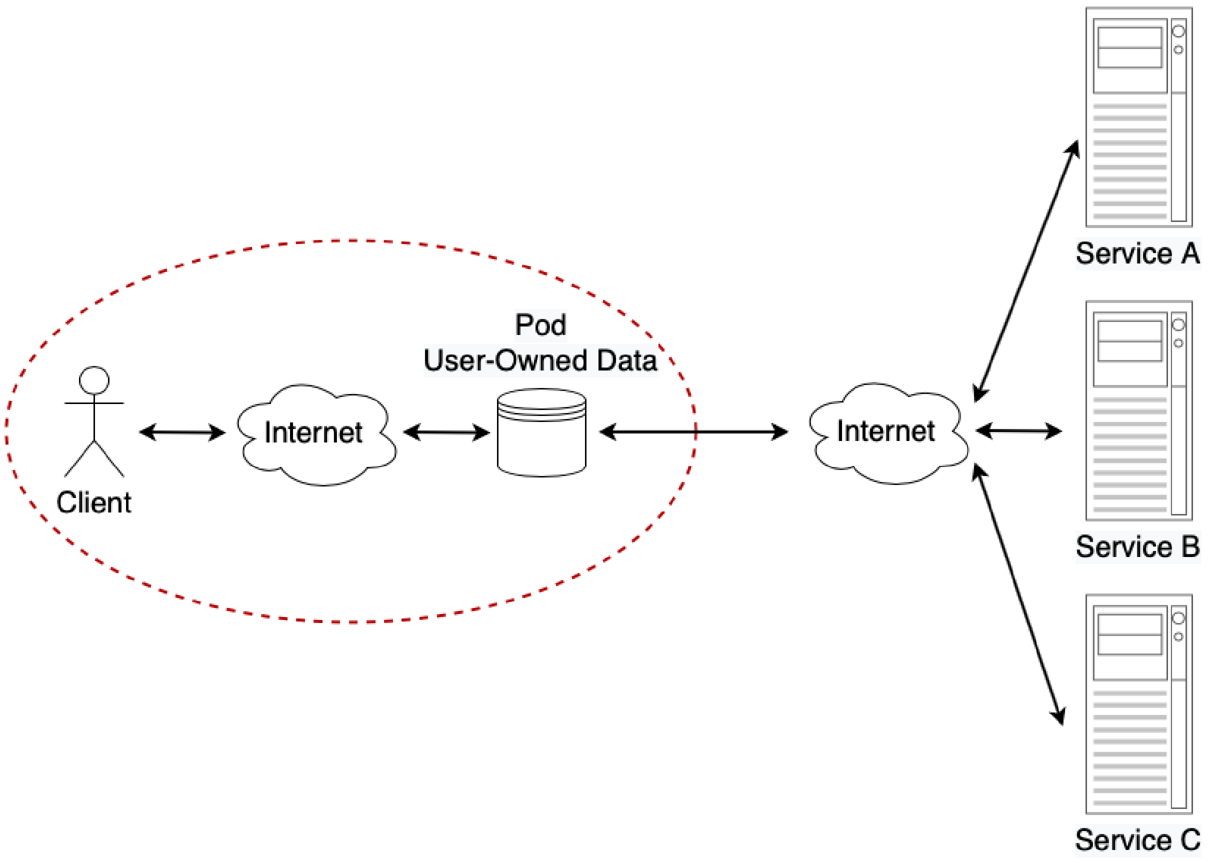


Figure 3.2: Architecture of the Solid platform. Users control their Pods and everything in them (red dashed line), whereas servers provide only services (business logic) without storing any data. Users decide who can access their Pods' data and can restrict access whenever they wish.

activities and relations to other people and objects) without the need for a centralised database [74]. RDF is used as a standard model for data interchange on the Web using a variety of syntax notations and data serialization formats [75]. Thus, using both FOAF and RDF, the WebID helps applications and services get access to data stored in different places. Solid's authentication and authorization systems help users to set different access rules to their personal data.

Although the Solid specification provides a high level of personal data control and strong security, we decided not to use it for this thesis project for several reasons as shown in the Table 3.2. First, the Opal app and its existing infrastructure would have to have been rewritten using the Solid platform, which would require more time and resources than available. Second, the specification does not provide a mechanism for storing data in an immutable way.

Last but not least, as of Summer 2021, the specification was still in the testing stage, and many features were still under development. However, as described in the Conclusions and Future Work chapter, the Solid technology must be seriously considered as an option to enhance future patient-centred improvements of the Opal app.

Table 3.2: Comparison of the Opal's current implementation and a possible Solid-based solution. The third column shows the modifications and changes of the current implementation that would be required to implement a Solid-based solution.

Opal	**Current solution**	**Solid**	**Required Modifications**
Architecture	Centralized	Decentralized	Designing new architecture, system workflow and requirements.
Data Storage	Single central database	Pod	Every user must have a personal Pod. This would require a solution for migrating the existing users' data to their Pods.
Data model (logical structure)	Relational (MySQL)	Vocabularies (ontologies)	To migrate to a Pod, the data stored in the Opal database(s) would need to be described with vocabularies. This would require a deep exploration of vocabulary concepts to build them. mCode might be one of the possible solutions to describe the data for cancer patients [76]. Also, the existing data would need to be migrated to the created vocabularies.
Registration & Authorization	User account	WebID	The app would need to guide users on how to register their WebID and personal Pod. The authorization process must be based on the WebID.
Front end	AngularJS	AngularJS	The front end views would not require any changes. However, the communication logic of the app would need to be modified based on the new architecture (e.g., registration/authorization, accessing personal data from a Pod, etc.). Firebase communication might need to be eliminated.
Back end	Server logic & Firebase	Server logic	The back end, including Listener, Opal Admin, Auto Update, and Publish Manager would need to be modified or completely rewritten based on the new architecture. There would be no need for the Opal database(s) since all the data would be stored in users' Pods. Firebase and the Listener might need to be eliminated. However, the new back end would still require a secure solution for connecting to different Pods through the hospital's firewall.

3.3 Blockchain

The second decentralized technology that we considered, and ultimately adopted for use in this thesis project, is blockchain, which originated in the world of online digital currencies. In this section, the principles and the underlying technology of the blockchain are described.

In digital currencies there is a potential situation called the double-spending problem that occurs when malicious users attempt to spend the same digital token (e.g., virtual money) more than once [77, 78]. Physical money is not affected by this problem as it can only be spent once. However, a digital token can be duplicated or falsified and potentially reused. In centralized systems (e.g., Web 2.0), the double-spending problem is solved by using a central trusted third party (e.g., a bank) that can verify whether a token has been spent already or not. In decentralized systems, it is significantly harder to prevent double spending, since all servers in the network must be synchronized and make the same decisions based on mutual agreement. For example, two broadcasted transactions to the distributed network that attempt to spend the same token can cause desynchronization of the servers. Since these two transactions will arrive at each server at different times, each server will consider the first transaction it sees as valid, and the second as invalid. This will lead to a disagreement between the servers and, consequently, it will not be possible to determine true balances. A solution to this problem is a consensus algorithm. By 2008, researchers proposed different conceptual solutions, but none of them were fully implemented [79, 80].

The first implemented solution to the double-spending problem in a decentralized system using a peer-to-peer network was proposed by an anonymous inventor going by the alias Satoshi Nakamoto in 2008 [81]. Nakamoto's design was the core of the first decentralized digital currency called Bitcoin. Bitcoin was able to solve the double-spending problem without the need for a trusted authority or a central server. The solution is based on an ongoing cryptographically secured chain of timestamped transaction blocks known as a blockchain. By design, a blockchain is resistant to modification of the data stored in the chain of blocks. Each block in the chain contains a cryptographic hash of the previous block, a timestamp, and a batch of transactions. Transactions are hashed and encoded into a Merkle tree [82]. A blockchain can store different types of information, but the most common is a ledger of transactions.

Blocks are chained to each other through hash pointers and together they form the blockchain. In any blockchain-based protocol, the first block is known as the genesis block. It is the basis on which additional blocks are added to form a chain of blocks. This block is sometimes referred to Block 0. It is a special case as it does not references a previous block. Thus, the genesis block is almost always hardcoded. Figure 3.3 shows a structure of the growing chain of blocks (i.e., the blockchain) that is used in most blockchain networks.

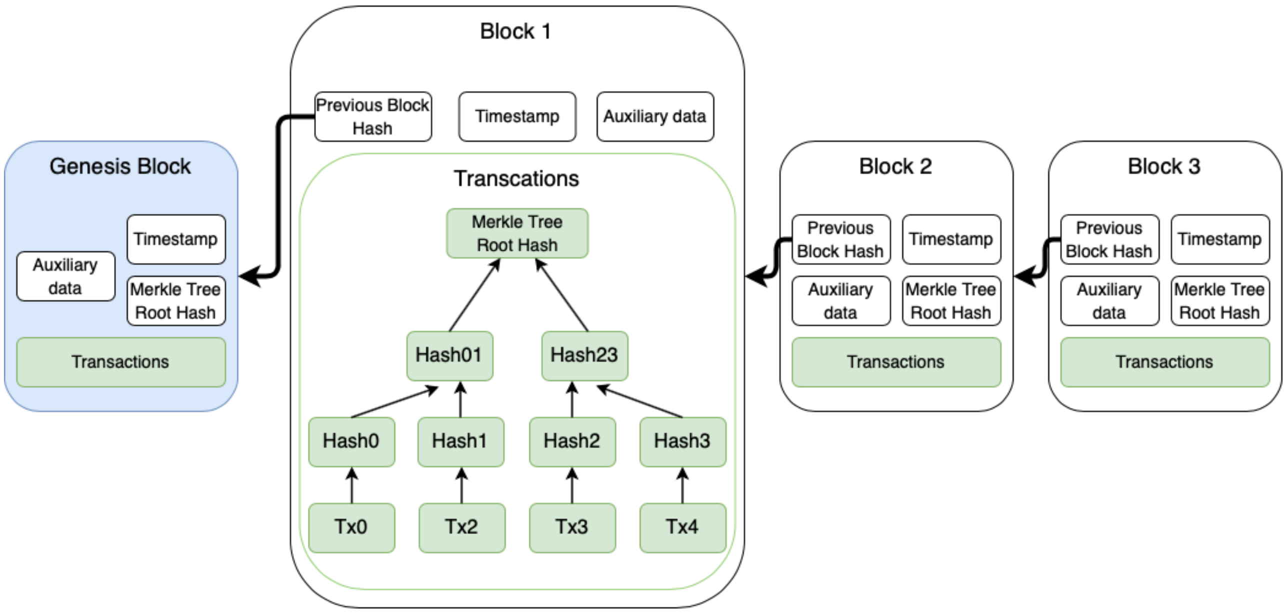


Figure 3.3: Structure of a digital ledger (blockchain) consisting of records called blocks. Each block stores a timestamp, a hash of the previous block, and transactions. Transactions are hashed in a Merkle tree. A block can store any auxiliary data based on its specific implementation. The chain of blocks replicates a linked list data structure, where each block is linked to the previous one through hashes. The first block is a genesis block, which does not have a previous block.

The blockchain is maintained by a decentralized network, in which the ledger is fully replicated on all network nodes. Each node is controlled by a different party, so no single person or group has control over the ledger and the network. All nodes collectively maintain the ledger and work to agree on the order of transactions based on the network rules and ordering algorithms. Transactions are permanently recorded and can be seen by any member of the network. Network nodes group new transactions into a block with a fixed size (e.g., the original Bitcoin's block size is 1 mebibyte). Network participants validate new transactions to make sure that: (1) transactions on the new block do not conflict with each other and (2) transactions on the new block do not conflict with the previous block's transactions. Therefore, network nodes need to agree on the next block to be added to the blockchain. However, sometimes the network may end up in a disagreement between the nodes. Thus, separate blocks can be produced concurrently, creating a fork. Based on the blockchain implementation, forks can be either temporary or permanent. Temporary forks occur due to the difficulty of reaching fast consensus in a distributed system (e.g., Bitcoin). In these cases, the fork is resolved when subsequent blocks are added and one of the chains becomes longer than the alternatives. The longest series of blocks starting from the genesis block becomes the main chain, and the network abandons the blocks that are not in it. Permanent forks occur due to different versions of the validating software on the nodes. In case of the new rules in the network, all the nodes must upgrade their software. If one group of nodes continues to use the old software while the other nodes use the new software, a permanent fork can occur. Figure 3.4 illustrates an example of a blockchain with forks.

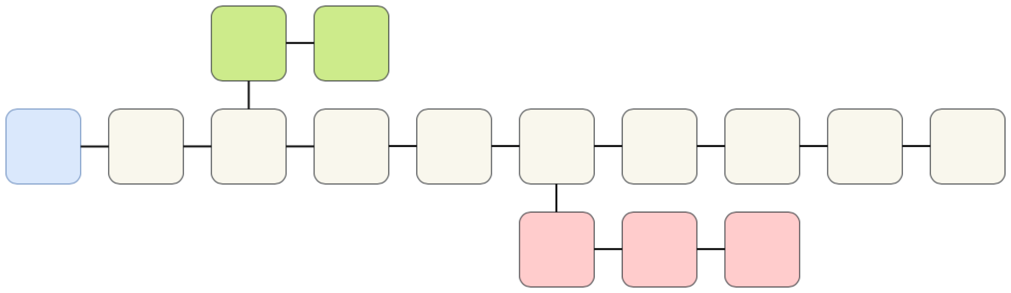


Figure 3.4: Schematic structure of the growing list of blocks that form a blockchain with forks. The blue block is the genesis block, which is the first block in the blockchain. The beige blocks that connect to the genesis block form the main chain. The green and red blocks are orphan blocks that form two separate forks. Forks occur when not all the network participants agree on the newly generated block. Thus, a blockchain diverges into two potential paths forward.

Since blocks are connected through hash pointers, any tampering with the content of any block can easily be detected by the network. If one network participant tampers with a record of transactions, all other nodes would cross-reference each other and easily pinpoint the node with the incorrect information. Thus, the distributed ledger is immutable, where the data entered are irreversible. No participant can change or tamper with a transaction after it has been recorded to the shared ledger. Each additional block strengthens the verification of the previous block and hence the entire blockchain. Figure 3.5 demonstrates the peer-to-peer blockchain network.

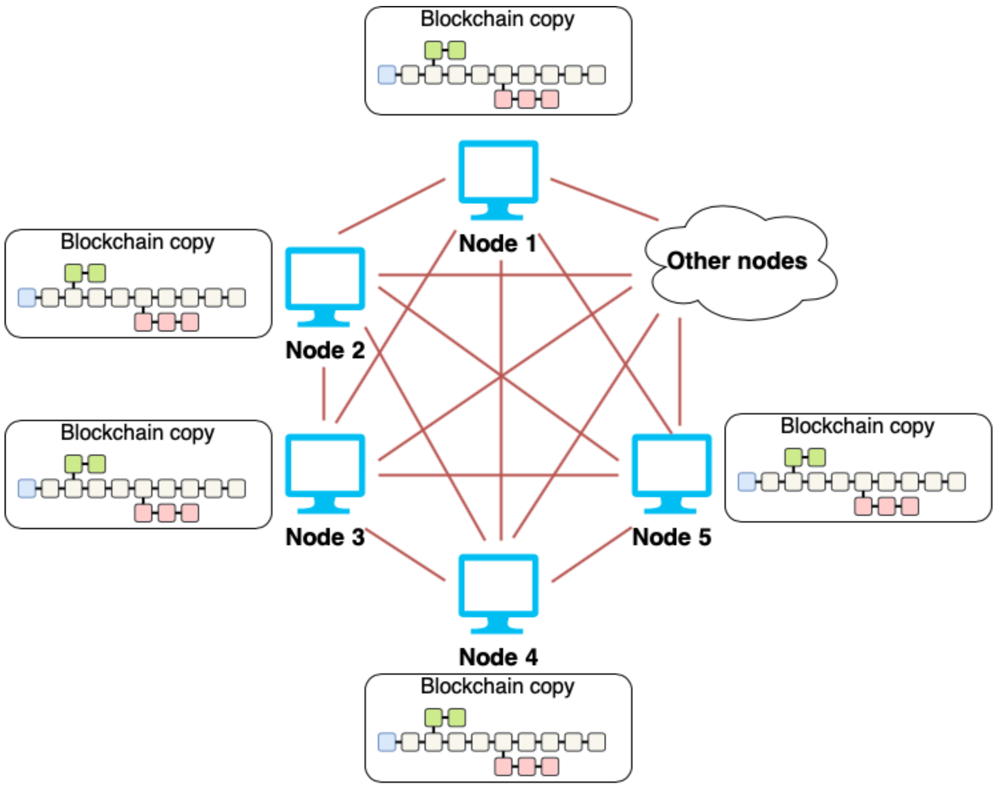


Figure 3.5: Schematic of a peer-to-peer network that maintains a blockchain. Each node keeps a copy of the blockchain ledger. The network works to agree on the order of all transactions based on the network rules and ordering algorithms. If a node decides to falsify any block, the network will detect it and reject the changes.

Therefore, blockchain technology can be described as an immutable (tamper-proof) transaction ledger. The records, called blocks, are linked together using cryptography and distributed in a peer-to-peer network. Although the primary use of blockchain is for cryptocurrencies in networks with no central authority, Nakamoto's solution has been adapted for different uses. New solutions are based on different implementations, namely, consensus algorithms and network configurations. Thus, nowadays, blockchain technology is being used in fields such as supply chain management, energy trading, anti-counterfeiting, video games, healthcare, etc. [86, 83, 85, 84, 87, 88]. Below, we discuss cryptographic hash functions, digital signatures, blockchain types, consensus algorithms, and blockchain platforms.

3.3.1 Hashing and Digital Signatures

Blockchain technology is based on cryptographic hash functions (CHFs). A CHF is a function that converts data of arbitrary length ("input" or "message") into a bit string of a fixed size ("output" or "hash"). The conversion process utilized by the hash function is called hashing. The source data are called the input array. The result of the conversion (i.e., the output) is called the hash. There are many hashing algorithms with different properties, such as bit depth, computational complexity, cryptographic strength, etc. For example, Bitcoin is based on the SHA256 algorithm.

An "ideal" hash-function must satisfy three conditions: (1) fast calculation, (2) determinism (a given input must always generate the same hash), and (3) collision free (no two x, y can exist, such that $Hash(x) = Hash(y)$). A CHF is a one-way function, which is practically infeasible to invert. Hashes help to preserve the integrity of the blockchain. By continuously utilizing previous hashes of each block we are able to ensure the blocks in the blockchain are kept in the right order and are dependent on each other. If a malicious party were to come in and try to manipulate any of the data, the hashes would change quickly and the chain would "break", so everyone in the network would know not trust the malicious chain. Also, hashes help to save space, since it more efficient to store a single hashed string, instead of copying all the actual data in the preceding blocks.

Blockchain security methods include the use of public-key (asymmetric) cryptography—a cryptographic system that uses a pair of keys: (1) public key that may be known to others for verification purposes, and (2) private key that is used to create the owner's signature and known only to the owner [89]. In such a system, it is mathematically impossible for a user to forge another user's private key from their public key [90]. Blockchain systems use asymmetric cryptography to secure transactions between users and move assets from one identity to another. For example, in the Bitcoin network, owners digitally sign their coins to transfer them to other recipients. Figure 3.6 shows the simplified chain of ownership originally described in Nakamoto's white paper.

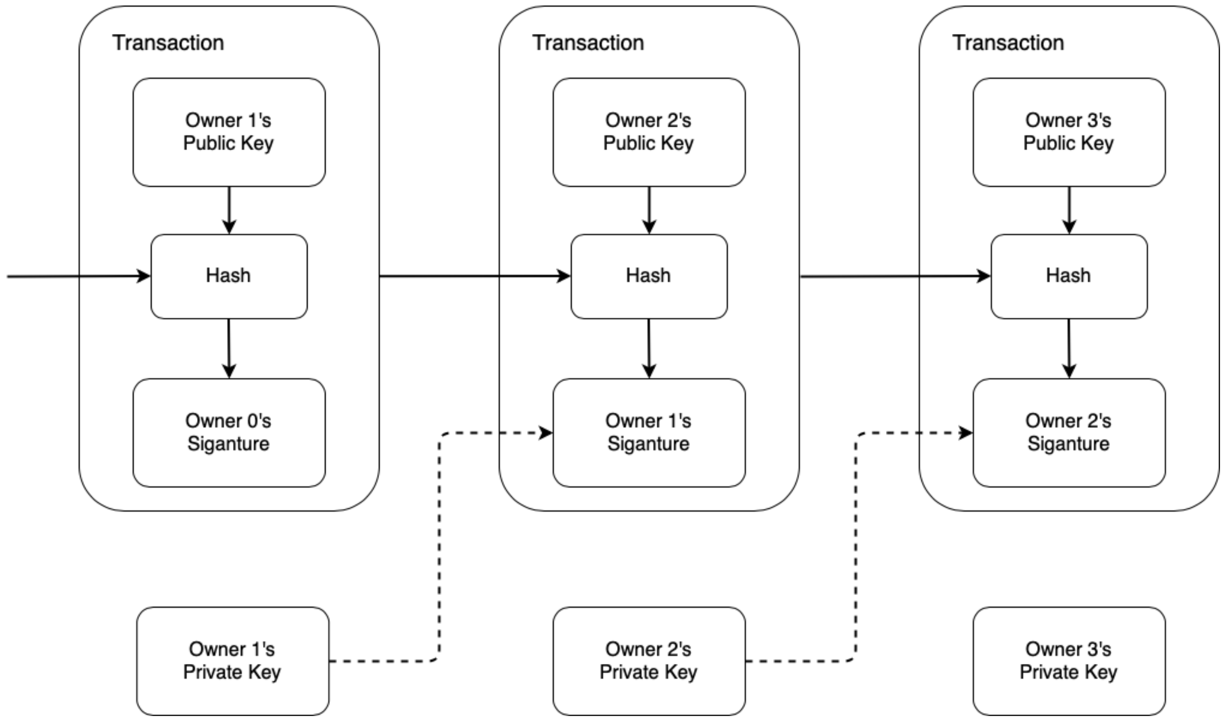


Figure 3.6: Simplified structure of sequential transactions with one input and one output for transferring assets between different identities in Blockchain. To transfer an asset (e.g., bitcoin), the sender has to have the receiver's public key. The current owner creates a new transaction that includes information about a transferring asset (e.g., amount of bitcoins) and a hash of the previous transaction digitally signed by the current owner. The previous transaction, by which bitcoins were received, becomes the input of the new transaction. Also, the new transaction includes the public key of the receiver (the output). Transactions can have many inputs and outputs. The transaction is broadcast to the network through open channels without encryption. All the nodes verify the digital signature of the received transaction before processing it.

3.3.2 Blockchain Types

There are four different blockchain types with their pros and cons. These types are based on different consensus mechanisms, and thus, operate differently.

3.3.2.1 Public Permissionless Blockchains

A public permissionless blockchain is decentralized and does not have any single node that controls the network. Anyone can join the blockchain network and participate within it. Participants can read and write transaction data. Also, any participant can validate transactions and generate new blocks. A public permissionless blockchain consumes more energy than a private blockchain as it requires a significant amount of electrical resources to function and achieve network consensus (e.g., Bitcoin). All transactions that take place on public blockchains are fully transparent, so anyone can examine the transaction details. Hence public permissionless blockchains provide no privacy for transactions. Also, public permissionless blockchains don't scale well, which leads to slow performance of the network.

3.3.2.2 Private Blockchains

A private blockchain works in a closed network, where the participants' and validators' accesses are restricted. A participant can join a private blockchain network only through an authentic and verified invitation from the controlling organization. The controlling organization can set up permission levels, security, authorizations, and accessibility. Thus, some transactions can have restricted access based on the access rules. Since the number of authorized participants is typically less than in a public blockchain, it can process hundreds or even thousands of transactions per second. However, private blockchains are more centralized than public blockchains.

3.3.2.3 Hybrid Blockchains

A hybrid (public permissioned) blockchain has a combination of centralized and decentralized features (public and private blockchains). Such an approach allows organizations to set up a permission-based system alongside a public permissionless system. The controlling organization of the hybrid blockchain can appoint privileged parties. These parties are able to run a node with abilities that are unavailable to the general public. For example, on the Ethereum blockchain network, a participant can define a smart contract (automatic transaction) that can be performed only by the contract's owner and not by others.

3.3.2.4 Consortium Blockchains

A consortium blockchain is similar to a hybrid blockchain in that it has private and public blockchain features. However, multiple organizations share the responsibilities of maintaining the blockchain. The consensus process is controlled by a pre-selected set of organizations that determine who may submit transactions or access the data. A consortium blockchain is ideal for business when all participants need to be permissioned and have a shared responsibility for the blockchain.

3.3.3 Consensus Mechanism

Since a blockchain is a decentralized peer-to-peer system with no central authority, it creates a major problem of achieving overall system reliability in the presence of a number of faulty processes. Thus, all nodes in the blockchain network should come to a consensus using consensus mechanisms. The consensus problem (e.g., double-spending problem in decentralized systems) requires agreement among a number of nodes for a single data value [91]. Some of the nodes may fail or be unreliable in other ways, so consensus protocols must be fault tolerant or resilient. Also, the consensus algorithm defines the blockchain's network type (e.g., public, private, etc.). There are different kinds of consensus algorithms. Each algorithm is based on different principles and each has pros and cons. Below we describe in detail four basic consensus algorithms: (1) the proof-of-work algorithm, (2) the proof-of-stake algorithm, (3) the delegated proof-of-stake algorithm, (4) the practical Byzantine fault tolerance algorithm, and (5) the Raft algorithm.

3.3.3.1 Proof of Work (PoW)

PoW is the original consensus algorithm in a blockchain network as proposed by Nakamoto. This algorithm is used to confirm the order of transactions and add new blocks to the chain. It is a type of "zero-knowledge proof", where so-called "miners" compete against each other by solving a complex mathematical puzzle—a puzzle friendly hash [92]. All the nodes in the network "work" to find a hash with a specific pattern (mining) in order to add a new block to the blockchain. A hash function H is puzzle friendly if for a given random input (e.g., blockchain block) and a target set Y (e.g., desired hash), it is hard to find a random value (called nonce) such that $H(input||nonce) = Y$. In one-way hash functions, it is practically impossible to reverse the function to find the input—given $H(x)$ it is infeasible to find x. Therefore, all the blockchain nodes keep trying random nonce values until one of them finds a solution. The more computing power a participant has, the faster they can solve the puzzle. The target hash has a specific pattern that is hard to find (e.g., 13 hexadecimal leading zeros in the SHA256 hash).

Thus, the PoW algorithm requires a huge amount of energy and computing power. The cost of doing the work disincentivizes bad actors from participating. When a miner successfully finds the correct hash, they present their block, including the mined hash, to the network for verification. Verifying whether the block belongs to the chain or not is a relatively simple process. If the majority of the participants approve the new block, all the network accepts the solution. The successful miner gets a reward for finding the hash.

The drawbacks of PoW algorithm are:

- It is an extremely inefficient process because of the energy consumption involved.
- Some nodes have more chances of finding the correct hash than others because of differences in computing power.
- It is vulnerable to a 51% attack—if a participant or organization owns more than 51% of the nodes in the network, it can corrupt the blockchain by gaining the majority of the network.

- Transactions are not instantaneous, since it takes some time to mine the transaction and add it to the blockchain. Thus, transaction confirmation takes from 10 to 60 minutes.

3.3.3.2 Proof of Stake (PoS)

The main purpose of PoS is to solve the problem of high energy consumption associated with PoW. This method requires that participants (nodes) have a legitimate stake in the blockchain (e.g., a certain number of bitcoins). PoS algorithms replace the hash calculation with a simple digital signature. Instead of investing in computational powers for mining, blockchain participants invest in the coins of the system by locking up some of their coins as stake. By investing personal coins, users become validators—network node operators that validate data, similarly to PoW systems. However, there is no energy-intensive computational process to earn the right to validate. For example, in Ethereum blockchain, a user has to stake 32 ETH (ethers, or virtual currency tokens) to become a validator. When it comes time to validate a new transaction block, the network randomly selects a validator to approve the block based on their proportional stake in the network. The network itself runs a lottery to decide which node will forge a new block, and system participants are exclusively and automatically entered into that lottery in direct proportion to their total stake in the network. The higher the number of virtual currency tokens that quantify a participant's stake, the higher their probability to be chosen as the winner who gets to forge the new block. When the block is confirmed, the validator is rewarded with network transaction fees. A user's stake is also used as a way to incentivise good validator behavior. For example, if validators act maliciously or fail to validate (e.g., node goes offline), they can lose a portion of their stake or their entire stake for deliberate collusion.

However, such an approach is biased towards the "rich" participants, who can stake more virtual currency tokens than others. So the PoS protocol must take into account that the lottery cannot be completely random to ensure the network is truly decentralized and secure.

3.3.3.3 Delegated Proof of Stake (DPoS)

DPoS works in the same way as the PoS system, except that users choose an entity that will represent their portion of stake in the system. Each participant votes for delegates by pooling their digital coins into a staking pool and linking them to a particular delegate. The delegates will represent individual stakes in the system, and the delegate elected by the system will validate a new block. The reward is shared with users who pooled their coins in the successful delegate's pool. The rewards are shared based on each user's stake—the more a user stake, the higher a share of the block reward they receive. This allows individuals with smaller stakes to team up to magnify their representation, thereby creating a mechanism to help balance out the power of large stake holders. However, this comes at the cost of greater network centralization.

3.3.3.4 Practical Byzantine Fault Tolerance (PBFT)

This consensus algorithm was proposed by Miguel Castro and Barbara Liskov in 1999 [93]. PBFT was designed to work efficiently in asynchronous systems and solve many problems associated with already available Byzantine Fault Tolerance solutions. Byzantine Fault Tolerance (BFT) is a condition of distributed computing systems, where some of the nodes in the network may fail to respond or respond with incorrect information. The term is derived from the Byzantine Generals Problem [94]. PBFT method of establishing consensus requires less effort than other methods. The algorithm tries to provide a practical Byzantine state machine replication that can work even when malicious nodes are operating in the system. Nodes in a PBFT-enabled distributed system are sequentially ordered with one node being the primary (or the leader node) and others referred to as the secondary (or the backup nodes). Any eligible node in the system can become the primary by transitioning from secondary to primary (typically, in the case of a primary node failure). The goal is that all honest nodes help in reaching a consensus regarding the state of the system using the majority rule. A PBFT system can function on the condition that the maximum number of malicious nodes must not be greater than or equal to one-third of all the nodes in the system. As the number of nodes increases, the system becomes more secure. However, PBFT does not scale well because of its communication overhead, where every node multi casts their responses to every other node. Thus, the PBFT works efficiently only when the number of nodes in the network is small.

A necessary prerequisite for the PBFT is that all participants should be known in advance. Therefore, this algorithm is mainly used in permissioned blockchains. PBFT consensus rounds are broken into the following phases:

- The client sends a list of transactions to the primary (leader) node.
- The leader orders the transaction candidates that should be included in a block, and broadcasts this list of ordered transactions to all the secondary (backup) nodes.
- The nodes (primary and secondaries) perform the ordered transactions one by one and then send back a reply to the client. The reply includes the calculated hash code for the newly created block.
- Each peer broadcasts its answer (the resulting hash code) to other peers in the network, and starts counting the responses from them. If the node sees that 2/3 of all validation peers have the same hash code, it will commit the new block to its local copy of the ledger.
- The round is completed successfully when the client receives $m+1$ replies from different nodes in the network with the same result, where m is the maximum number of faulty nodes allowed.

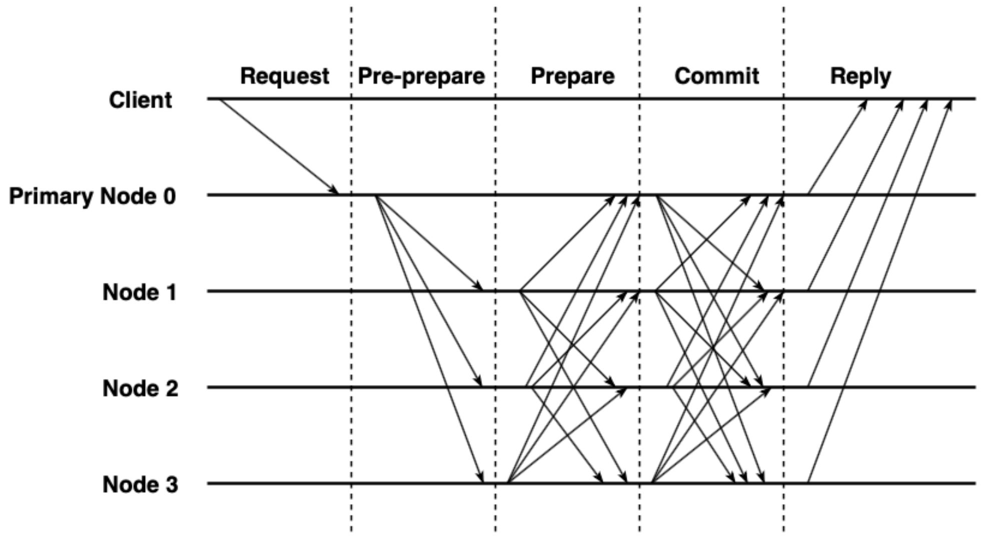


Figure 3.7: Normal case operation of the PBFT algorithm proposed by Liskov and Castro. When the primary node receives a client request, it starts a three-phase protocol to atomically multicast the request to the nodes. The pre-prepare and prepare phases are used to totally order requests sent in the same view even when the primary, which proposes the ordering of requests, is faulty. The prepare and commit phases are used to ensure that requests that commit are totally ordered across views [95].

If needed, a majority of the honest nodes can vote on the legitimacy of the current leading node and replace it with the next leading node in line. This algorithm is used in private blockchains. Figure 3.7 illustrates the operation of the algorithm in the normal case with no faults.

3.3.3.5 Raft

Raft is a consensus algorithm that is used in private blockchains. The algorithm was designed in a way that is easy to understand compared to other algorithms. Also, it provides additional features, such as (1) a strong form of a leader node (e.g., log entries only flow from the leader to other servers), (2) leader election (i.e., randomized timers to elect leaders), and (3) membership changes (i.e., allows a cluster of servers to continue operating normally during configuration changes) [96, 97]. Raft nodes are always in one of three states:

- Leader—the node that is responsible for processing new log entries, replicating them to follower ordering nodes, and managing when an entry is considered committed. The network elects a single node to be the leader.

- Followers—nodes that receive the logs from the leader and replicate them deterministically, ensuring that logs remain consistent. Followers also receive "heartbeat" messages from the leader. If the leader stops sending those message for a configurable amount of time, the followers will initiate a leader election.

- Candidates—nodes that do not receive communication messages from the leader over a period called the "election timeout".

Initially, all nodes start out as a follower. In this state, they can accept log entries from a leader (if one has been elected), or cast votes for leader. If no log entries or heartbeats are received for a certain amount of time (e.g., five seconds), nodes self-promote to the candidate state. In the candidate state, nodes request votes from other nodes. If a candidate receives a quorum of votes, the majority of nodes, then it is promoted to be the leader. The leader must accept new log entries and replicate them to the followers. Raft uses a randomized election timeout to ensure that split vote problems are resolved quickly. If a quorum of nodes is unavailable for any reason, the ordering service cluster becomes unavailable for both read and write operations, and no new logs can be committed.

3.3.4 Blockchain Frameworks and Platforms

Since it requires a significant amount of time and resources to implement a solid blockchain codebase from scratch (e.g., networking, agreement protocol, blocks, transactions, smart contracts, etc.), we decided to explore existing ready-to-use blockchain platforms and frameworks. In our analysis, we took into account the following factors:

- Blockchain type.

- What consensus protocols are supported by the platform.

- What programming languages are supported by platform's SDKs (client application).
- Support of smart contracts functionality.
- Platform's popularity—reputation of a blockchain platform on websites (e.g., GitHub).
- Scalability—a blockchain network should be able to scale to adapt to the growth.

We analyzed nine the best known modern blockchain platforms and their main features that are presented in Table 3.3. Hyperledger Sawtooth and Hyperledger Fabric platforms were fitting our requirements the best. Both platforms are private permissioned blockchains that have modular architectures. The platforms allow to use different programming languages, both for smart contracts and platform's SDKs, and set custom network configuration (e.g., number of nodes, block size, block time, etc.). Also, the platforms show high transaction rate and can be simulated in the Docker platform. However, Hyperledger Fabric had better official documentation and more materials online.

Table 3.3: Comparison table of the best known modern blockchain platforms.

Blockchain Platform	**Ledger**	**Network**	**Consensus Protocol**	**Smart Contracts**	**Smart Contract Execution**	**Block Time**	**Average Transactions per Second (TPS)**	**Industry Focus**
Bitcoin	Public	Permissionless	PoW	No	N/A	600s	3.5 [98]	Financial Services
Ethereum	Public	Permissionless	PoW, PoS (Casper)	Solidity, Vyper, Yul	Ethereum Virtual Machine (EVM)	~ 15s	11.5 [99]	Cross-Industry
R3 Corda	Public	Permissioned	Pluggable (Validity, Uniqueness)	Kotlin, Java	Java Virtual Machine (JVM)	N/A	170 [100]	Cross-Industry
Ripple	Public	Permissioned	XRP Ledger	No	N/A	N/A	1500 [101]	Financial Services
Quorum	Private	Permissioned	Pluggable (Raft, Istanbul BFT, Clique)	Solidity	Docker	50ms - 20000ms [102]	~ 50 - 400 [103, 104]	Cross-Industry
EOSIO	Public	Permissioned	DPoS + asynchronous BFT [105]	EOSIO.CDT	EOS VM	0.5s	1000 - 8000 [106]	Cross-Industry
Stellar	Public	Permissioned	Stellar Consensus Protocol (SCP)	JavaScript, Golang, Java, Ruby, Python, C#	Docker	5s [107]	1000-5000 [108, 109]	Financial Services
Hyperledger Sawtooth	Private	Permissioned	Pluggable (Proof of Elapsed Time, PBFT, Raft)	Python, JavaScript, Go, C++, Java, Rust	Docker	Based on the configuration [110]	~ 1000 - 2300 [111]	Cross-Industry
Hyperledger Fabric	Private	Permissioned	Raft	Go, Node.js, Java	Docker	Based on the configuration [112]	Up to 20000 [113]	Cross-Industry

3.4 Hyperledger Fabric

Since the Hyperledger Fabric platform (1) is compatible with the technology stack of the existing Opal infrastructure, (2) stores, by its nature, data in an immutable, tamper-proof way, and (3) has regular major releases and detailed documentation with examples, we decided to use it to run our prototype blockchain network. The Hyperledger Fabric platform is a private, permissioned blockchain initially developed by IBM and Digital Asset [114, 115]. One of the key features and advantages of the platform is the support of modern programming languages, such as Go, JavaScript, and Java, for writing smart contracts. Also, the platform has a highly modular architecture that allows developers to configure and modify the components of the network, such as endorsement policies, key-value databases for storing a world state (e.g., CouchDB, LevelDB), ordering service, membership service provider, etc. In this section we discuss the main components of the Hyperledger Fabric blockchain network.

3.4.1 Organizations and Consortia

Participants who form a Hyperledger Fabric network are known as organizations or "members". Each organization can have different goals, business interests, and subsequently different roles. However, all members are interested in transparent cooperation between each other and fair functioning of the network. A consortium defines the set of organizations in the network who share a need to transact with one another (i.e., common goal). A Hyperledger Fabric network can have multiple consortia. In most cases, multiple organizations come together as an initial consortium to form the network. Permissions and access rights for each organization are defined by a set of policies, which are agreed by the consortium at the moment of creating the network. However, the network policies can change over time—subject to the agreement of the organizations in the consortium (e.g., adding new administrators, adding another consortium). A consortium is a group of organizations that own peers and chaincodes (discussed below) with no ordering service nodes. For this research project, we deployed one consortium with three organizations within it. Figure 3.8 displays the high-level architecture of our proposed blockchain network.

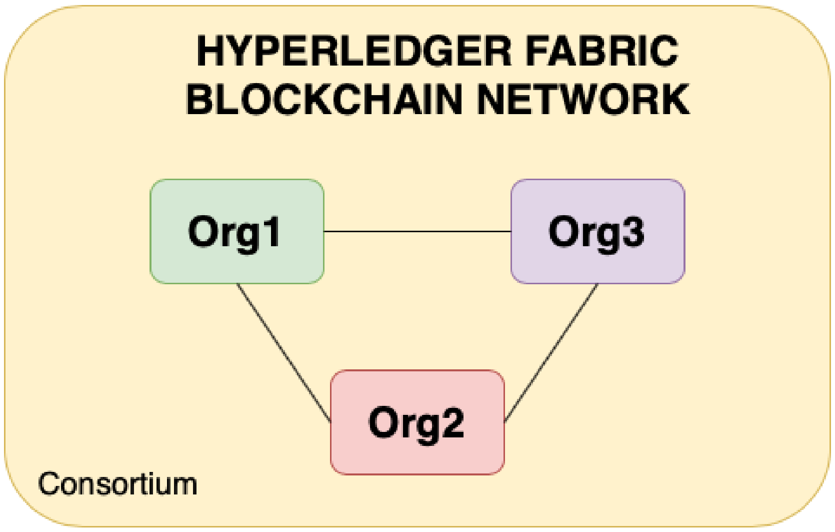


Figure 3.8: High-level architecture of our Hyperledger Fabric blockchain network with three organizations in one consortium.

3.4.2 Peers

Any blockchain network consists of the fundamental elements called peer nodes. In Hyperledger Fabric, each organization may have one or more peers based on the network policies and requirements. Each peer keeps a local copy of the append-only blockchain ledger and a snapshot of the world state (current state). A peer receives an ordered set of transactions in the form of blocks to form a ledger and world state. A world state is a key-value store that keeps the current values of all ledger states. It allows chaincodes to get the current value of a ledger instead of traversing the whole transaction log. Also, a peer can optionally store a chaincode – a smart contract that allows client applications to communicate with the Hyperledger Fabric blockchain network and make changes in the world state.

Hyperledger Fabric distinguishes four different types of peers:

- Endorsing peer – a peer that has installed smart contracts to endorse transactions sent by a client. The peer executes the received transactions on the local copy of the ledger and digitally signs the results. It submits the digitally signed transaction response and generated read/write sets back to the client. However, the transactions are not committed to the blockchain ledger.

- Committing peer – a peer that receives from the ordering service ordered transactions packaged in a block. The peer validates all the transactions, marks them as valid or invalid, and commits the block on the local copy of the blockchain.
- Anchor peer – a special peer that is authorized for cross-organization communication.
- Leading peer – a peer that communicates with the ordering service and distributes received packaged transactions within the organization.

Figure 3.9 shows a Hyperledger Fabric network of two organizations, where each organization has three peer nodes.

3.4.3 Channels

A channel is a communication line between peers that allows a group of participants to keep their separate blockchain and ledger of transactions. All peers in the channel maintain a channel-specific ledger, and transacting parties must be authenticated to a channel to interact with it. Thus, only the members of the channel can see the transactions and data transmitted in the channel. A peer can simultaneously be connected to different channels and maintain multiple ledgers. Also, channels support atomic delivery (total-order broadcast) of all messages, which guarantees delivery to the connected peers the same set of messages in the same order. Figure 3.10 depicts a sample Hyperledger Fabric network of three organizations with two channels.

3.4.4 Chaincode

In a blockchain network, a smart contract is a program that allows client applications to interact with the world state database objects based on the rules defined in this program. Hyperledger Fabric's smart contracts are packaged into the chaincodes and installed on peers. Mul-

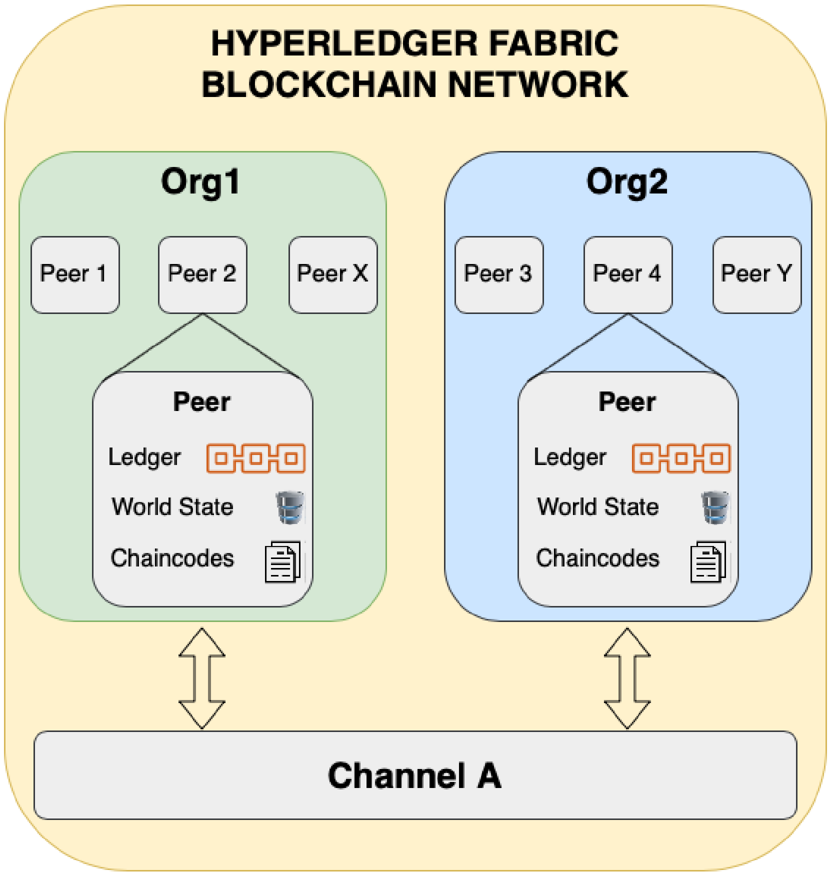


Figure 3.9: A Hyperledger Fabric blockchain network of two organizations both connected to channel A. Each organization consists of three peer nodes. Each peer locally manages their copy of the blockchain ledger and the world state. Also, all the peer nodes have installed the chaincodes for executing proposed transactions by a client.

tiple smart contracts can be defined within the same chaincode. Chaincodes can be deployed on different peers and channels, however not every peer has to run a chaincode to get access to the world state database objects. One of the main advantages of the chaincodes, compared to the standard smart contracts of different blockchain platforms, is the possibility to manage client application's access to the network by setting restricted access to the invoking functions of the chaincode. Also, Hyperledger Fabric supports the Go, JavaScript, and Java languages for writing chaincodes. This allows developers to use modern standard programming languages instead of using a closed smart contract language. For example, blockchain platforms, such as Bitcoin and Ethereum, have their own smart contract languages [116, 117].

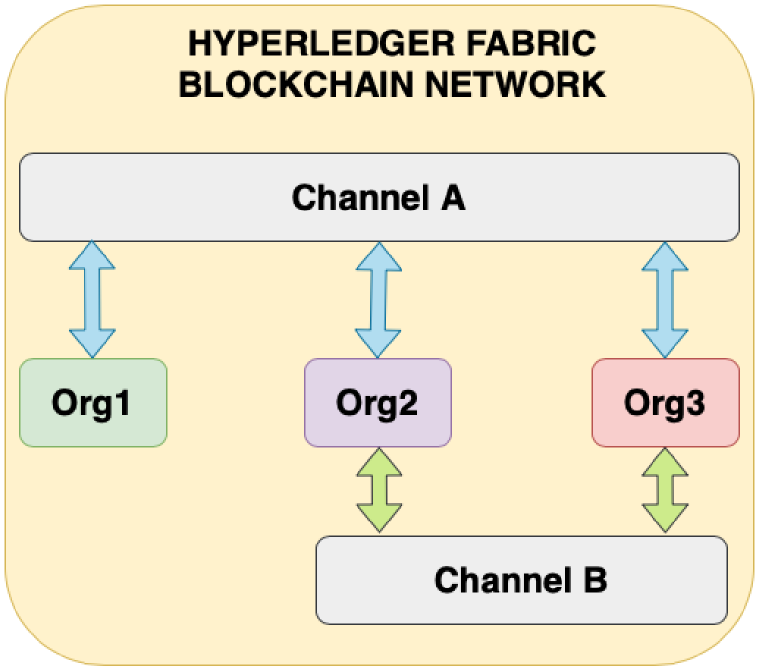


Figure 3.10: A Hyperledger Fabric blockchain network of three organizations and two channels. All organizations are connected to channel A, while at the same time Org2 and Org3 are simultaneously connected to channel B. Thus, both Org2 and Org3 maintain two blockchains, whereas Org1 has only one.

3.4.5 Orderers

In any blockchain network, all transactions are ordered and recorded to the shared blockchain ledger in the form of blocks. The order of the transactions is based on the consensus of the network. In permissionless blockchain networks, such as Bitcoin and Ethereum, any node can participate in the consensus process. These systems rely on probabilistic consensus algorithms (e.g., mining), where the algorithm defines the ordering node for each round of ordering transactions and generating a new block. Probabilistic consensus algorithms very often lead to divergent ledgers (forks), where the participants have a different view of the accepted order of transactions, and thus, different blockchain ledgers. For example, there are currently 105 forks in the Bitcoin network and eight forks in the Ethereum network [118]. Hyperledger Fabric is based on deterministic consensus algorithms that prevent a network from developing situations where nodes have ledger forks. The network has one or more special nodes called orderers that together form an ordering service. This service establishes consensus on the order of transactions and distributes a new generated block to connected peers for validation and commit. Deterministic consensus algorithms guarantee that validated blocks by peers are final and correct. The implementation of the crash fault tolerant ordering service is based on the Raft protocol. Figure 3.11 illustrates an example of a Hyperledger Fabric network with three ordering nodes that form an ordering service.

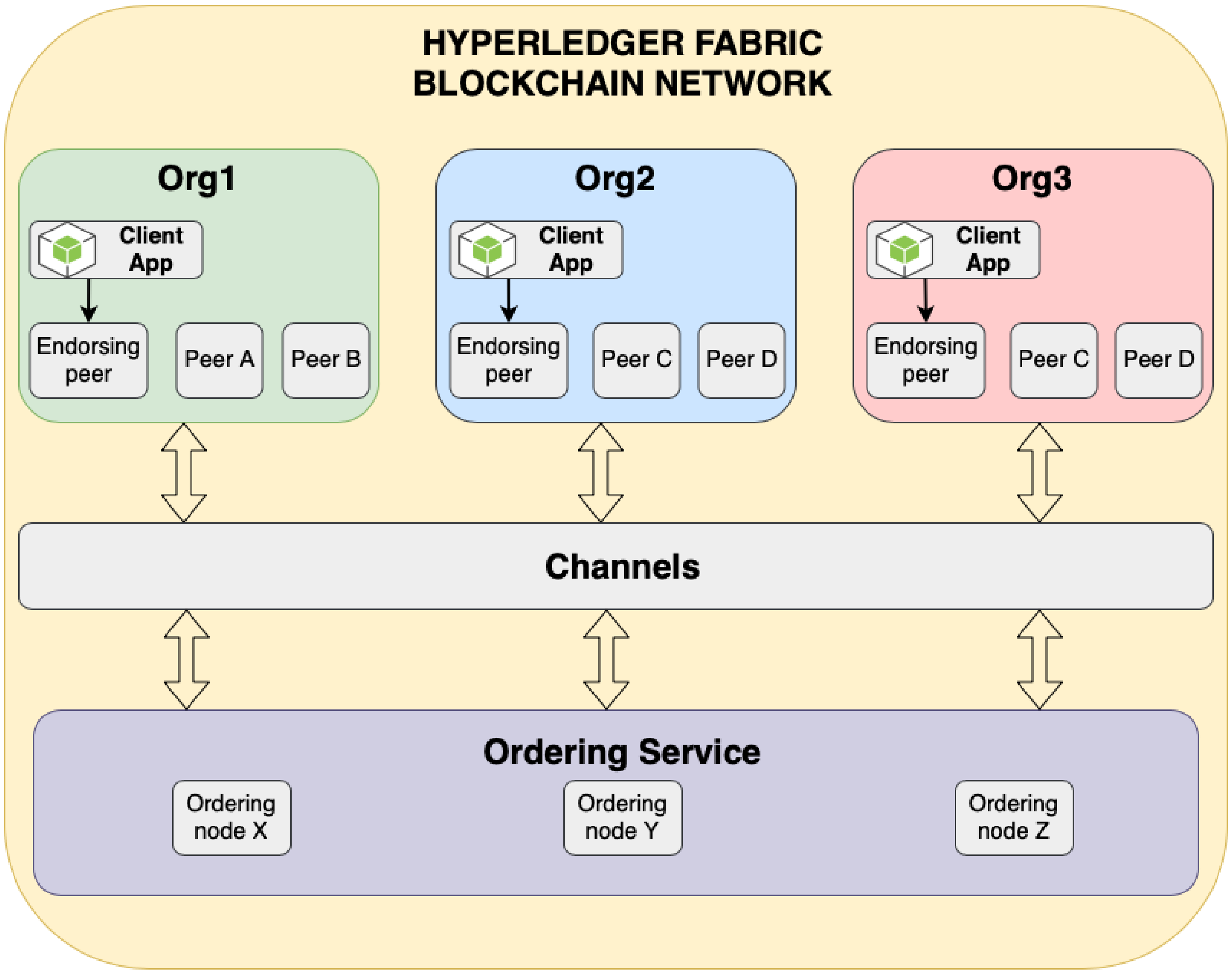


Figure 3.11: A Hyperledger Fabric blockchain network of three organizations that communicate through the channels. The network has three ordering nodes that group into an ordering service for ordering transactions in a strict order and bundling them into blocks.

The whole Hyperledger Fabric's transaction flow is depicted as a sequence diagram in Figure 3.12. At first, a client (SDK) generates a transaction proposal that is distributed across all the endorsing peers. The endorsing peers verify that the client's digital signature is valid and execute the transaction against the current state database to produce transaction results. The blockchain ledger is not updated at this point. At the next step, the client receives proposal responses back and compares them between each other to make sure the responses are identical. If the original transaction had only read operations, the client application inspects the responses, and the transaction flow finishes at this point. If the original transaction involves write operations, the client application sends the transaction proposal and response to the ordering service to update the blockchain ledger. The ordering service receives transaction proposals from different channels in the network and simply orders them by channel. So at this stage, the ordering service orders the transactions, generates a new block, and distributes it to all peers on the channel. At the last stage, the peers verify that the endorsement policy is fulfilled and update the blockchain ledger and the world state database.

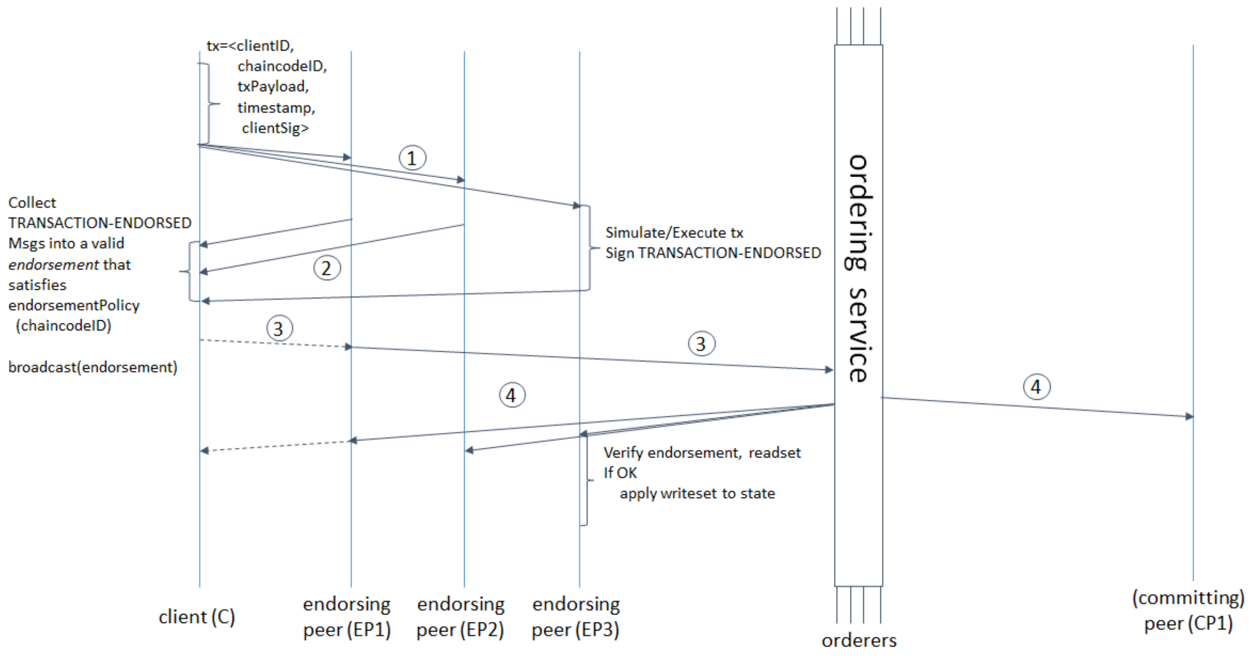


Figure 3.12: Sequence diagram of the transaction flow in a Hyperledger Fabric network [119].

3.4.6 Hyperledger Fabric SDK

To interact with a Fabric blockchain network, the distributed platform provides a software development kit (SDK) [120]. It serves as a high level API for client applications to submit transactions to a ledger or query the contents of a ledger. Currently, Hyperledger Fabric supports SDKs for Node.js and the Java runtime environment. Therefore, for communicating with the blockchain network, we built a RESTful (Representational State Transfer) API on top of the Node.js SDK. The SDK consists of four modules:

- Fabric-network – includes the API's to connect to a Fabric network and interact with chaincodes (smart contracts).
- Fabric-ca-client – allows applications to interact with the optional Certificate Authority component, fabric-ca, for establishing trusted identities on the blockchain network.
- Fabric-common – provides APIs to interact with the main components of a Hyperledger Fabric network: peers, orderers, and event streams.
- Fabric-protos - includes the protocol buffers for communication over gRPC.

Figure 3.13 shows the structure of a client application for interacting with the blockchain network using Node.js SDK.

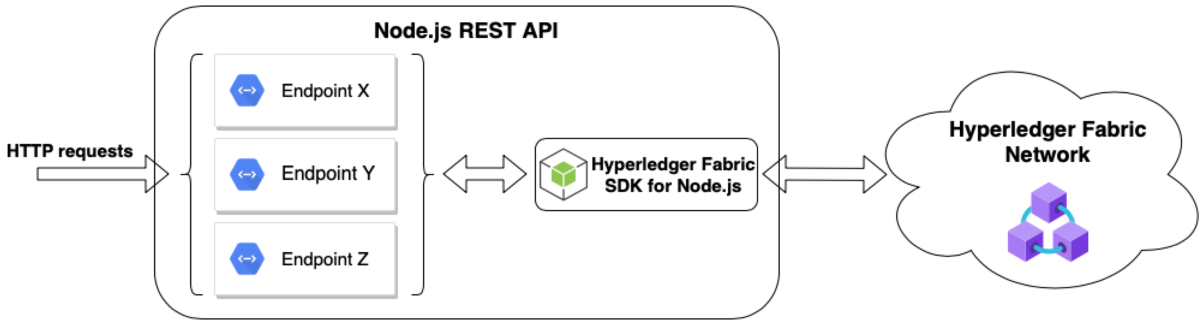

Node.js REST API
HTTP requests
Endpoint X
Endpoint Y
Endpoint Z
Hyperledger Fabric SDK for Node.js
Hyperledger Fabric Network

Figure 3.13: Client application request flow for interacting with the Hyperledger Fabric.

Chapter 4

Design and Implementation of a Patient-Controlled Data Sharing Service for Healthcare

4.1 Desired Solution

Based on our analysis described in Chapter 2, we identified the blockchain as a possible solution to manage the sharing of healthcare data in a tamper-proof manner. To the best of our knowledge, all the previously-discussed methods (see Section 2.3.1) imply storing sensitive healthcare data in the blockchain network itself. In this research project, we propose a novel approach for controlling personal medical data for use in research in a tamper-proof manner by storing just the data-sharing consents (from patients) and data-access privileges (of researchers) in a blockchain network. In our approach, the medical data themselves remain in the data providers' infrastructure until accessed by the researchers and all access attempts are logged in the blockchain. One of the main advantages of this method is an easy integration to existing infrastructure as there is no need to export medical records to the blockchain network. Also, in this way the blockchain network has to store and handle less data, which improves its performance and requires less configuration. And last but not least, it is harder to compromise the data access mechanism, since the data and the access privileges are stored in two separate systems, allowing the data to be stored in encrypted format on a dedicated server that is not public-facing. Figure 4.1 shows the proposed concept of controlled data access based on the access rights and privileges stored in a blockchain network.

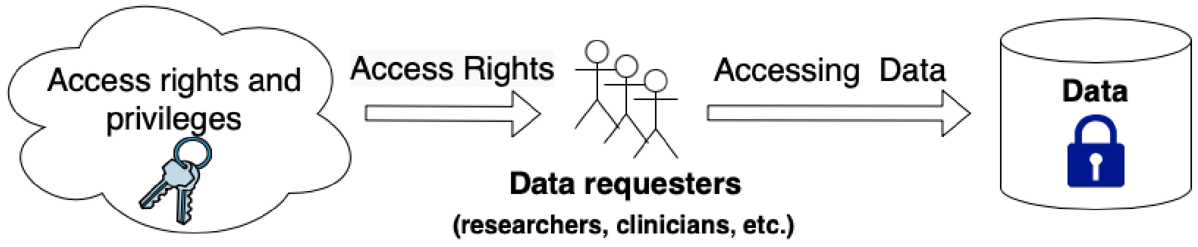


Figure 4.1: Schematic design of the proposed concept. Before accessing the data, requesters have to retrieve their personalized access rights stored in the blockchain. If there are no privileges, a requester cannot access the data.

To test the proposed concept in practice, we built a prototype Research Portal in the framework of the PARTAGE project, where:

- Opal users can provide consent to share their medical data with a specific research study by signing an electronic consent form (e-consent). The consent can be revoked by the patient at any time, at which point the data will not be available to researchers anymore.
- Researchers access the shared data based on the match between their privileges and patients' consents. The system automatically logs all data access requests.
- A public-trust committee (with administrator rights) controls the granting of researcher privileges. However, the public trust committee itself cannot access the data and cannot grant access to data that patients have not consented to make available. In practice, the public trust committee would likely be a single or multi-institution research ethics board. However, for the purpose of this project, we are not defining who exactly will participate in this committee or specifying how they will operate.

In our prototype system, researchers and public-trust administrators operate via an on-premises web portal (Research Portal), and we emulate patient-provided consents using a dedicated demonstration module with the Research Portal. The tamper-proof blockchain system is required to generate and record:

- Patients' consents specifying the data they wish to share and the studies they wish to share them with;
- Researchers' privileges specifying which studies containing patient-shared data they are permitted to access;
- Data access logs for regulator and patient audits.

Figure 4.2 visualizes the high-level architecture of the designed system. Figure 4.3 represents a diagrammatic representation of the system's flow.

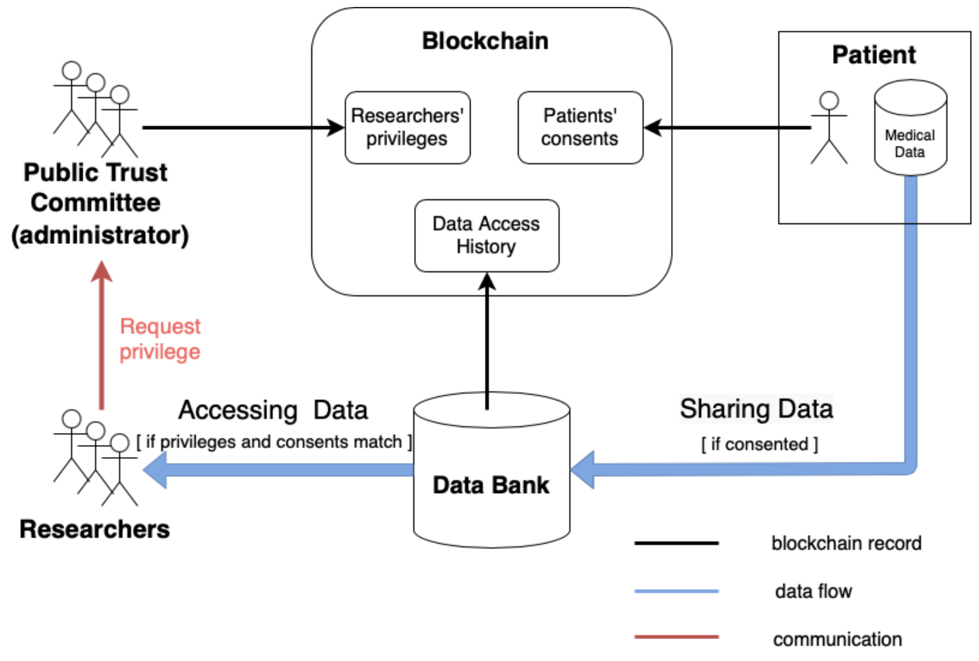


Figure 4.2: The high-level architecture of the designed data access system. Patients share their medical data by signing an e-consent form in the Opal app, while the Public Trust Committee (administrator) controls who can access the data by assigning researcher privileges upon request from researchers. The blockchain network stores patient's consents, researchers' privileges, and data access history, which guarantees that this information is stored in a tamper-proof and immutable way. Researchers can only access data that they have appropriate privileges to access and to which patients have consented for sharing.

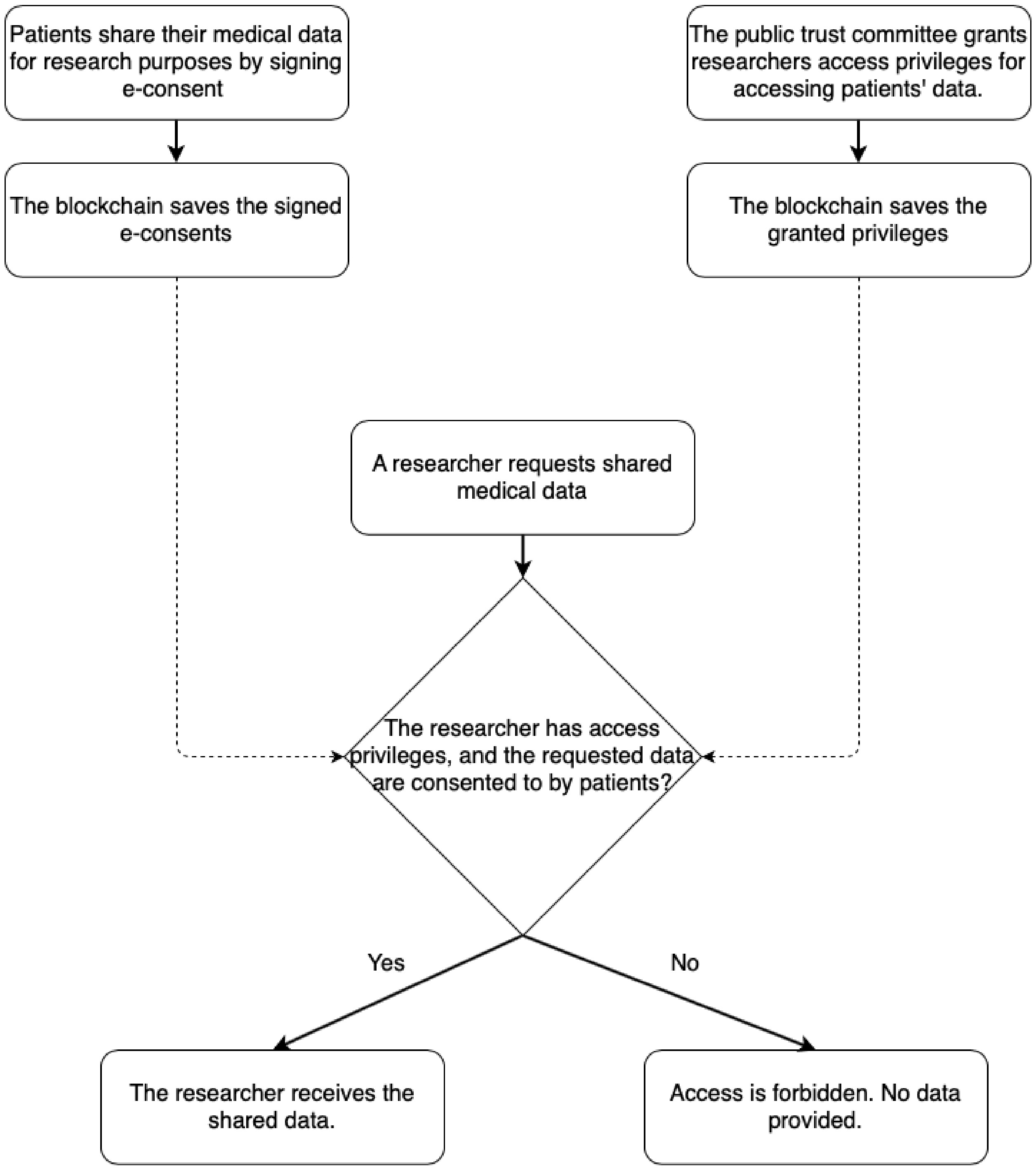

Patients share their medical data for research purposes by signing e-consent
The public trust committee grants researchers access privileges for accessing patients' data.
The blockchain saves the signed e-consents
The blockchain saves the granted privileges
A researcher requests shared medical data
The researcher has access privileges, and the requested data are consented to by patients?
Yes
No
The researcher receives the shared data.
Access is forbidden. No data provided.

Figure 4.3: Flowchart of the proposed Research Portal. Both the patient and public trust committee flows are independent processes and can occur asynchronously. A researcher can access medical data only if they have both patient-provided consent and the public trust committee-provided access privilege. The public trust cannot provide access to data that patients have not provided their consents to be accessed.

4.2 Functional requirements

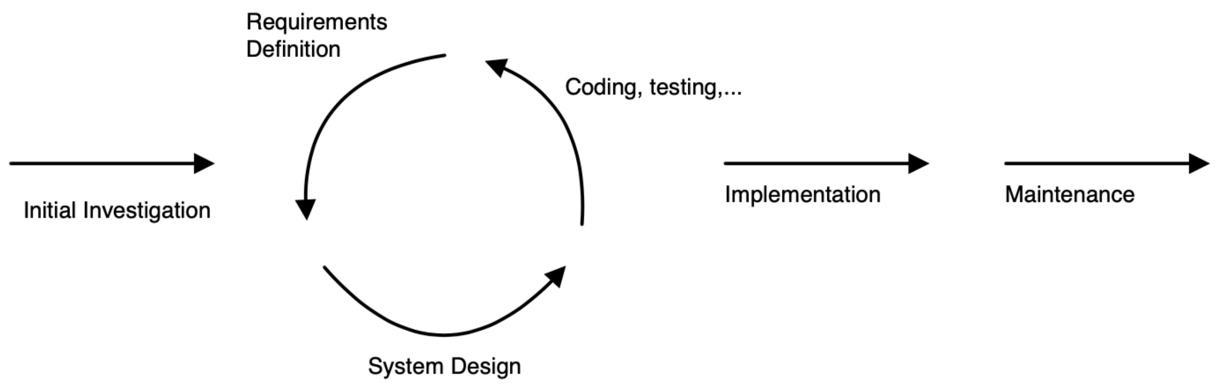


Figure 4.4: The flow of the prototyping process. At the first stage, the development team investigates a real-world problem and how to solve it. The team works with potential users of the system and ascertains their needs. The second stage is a cyclic process, in which the developers define the requirements and the system design, and implement them. This step repeats until the prototype meets the users' requirements. The last stage involves implementing a production version of the system and supporting it [121].

Modern software development processes propose a wide variety of models and methodologies to organize and structure the development to achieve the final goals. They help developers to improve the design, architecture, and quality of the final software product. These methodologies include Agile, Waterfall, Prototyping, Iterative and Incremental development, Spiral development, Rapid application development, and Extreme programming [121, 122, 123, 124]. However, the initial phase for all development methodologies involves a requirements analysis – the process of defining users' expectations and vision on how the system should operate. The result of the analysis is a set of documented requirements in formal technical language – i.e., the functional requirements.

Since the desired solution of this thesis project was a proof-of-concept, we followed the Prototyping methodology. An important principle of this approach is to break the project into smaller segments and develop small-scale mock-ups of the system, which allows the development team to provide a quick implementation of an incomplete, but functional application.

The development of a prototype is a cyclic process, where on each iteration the developers reflect on the system objectives, design and architecture, user interface, missing functionality, etc., until the prototype meets the users' requirements. Therefore, prototyping helps to identify the requirements for a production application and appropriate system design. It also allows the developers to explore and gain knowledge on new technologies. Figure 4.4 represents a schematic diagram of the prototyping methodology.

4.2.1 Defining Use Case Scenarios

Functional requirements are a specification that describes the system behavior and includes technical details on calculations, input data processing and output results presentation [125]. Functional requirements can be written in several different ways, such as a specification document, user stories, or use cases. For this project, the functional requirements were written in the form of use cases. A use case is a usage scenario for a piece of software, which contains a list of actions or event steps defining the interactions between actors (users) and a system to achieve a specific goal [126, 127]. There are different templates for writing use cases in text. The most common templates are use case brief, casual, outline, fully dressed, Fowler style, etc.

In this section we introduce written use cases based on the Fowler style, a simplified variant of the Cockburn template. According to Fowler, there is no standard way to write the content of a use case, and different formats work well in different cases [128]. The Fowler template includes:

- Title – the goal that the use case is trying to satisfy.
- Actor – a user who calls on the system to deliver a service.
- Main success scenario – a sequence of numbered steps that describe the interaction between an actor and the system. Each step is a simple statement that clearly shows who is carrying it out.

- Extensions – conditions that result in different scenarios other than the main one. A use case can have many extensions. Thus each extension has its own number that relates to the step it is extending.

Prototyping iterations helped us to capture the core functional requirements of the proposed Research Portal, which are presented in Tables 4.1-4.7 as text use cases.

Table 4.1: Researcher authentication and authorization use case

Field	**Specification**
Use case	Authenticate and authorize as a researcher
Actors	Researcher (registered user)
Goal	A researcher-user wants to access their profile (authorize in the system)
Success Scenario	Steps: 1. An unauthorized researcher goes to the login page; The system requests login and password; 2. The researcher enters login and password; 3. The system verifies the user's credentials; 4. The system verifies if a request was made by a real user (reCAPTCHA); 5. The system redirects the user to the researcher's page.
Result	The researcher-user is successfully authorized in the system and has access to their profile
Extensions	
*	No access to the database. The system returns an error message.
2a	The user chooses "I forgot my password". The system starts the "I forgot my password" scenario.
2b	The user chooses "Register a new researcher". The system starts the "Register a new researcher" scenario.
3a	The user with the specified credentials does not exist. Result: denial of access. The system returns an error message. Return to the second step.
3b	The researcher reaches the maximum number of unsuccessful attempts to enter credentials. Result: denial of access. The system returns an error message. The user is blocked for some period of time. This period is set in the configuration of the system.

Table 4.2: Administrator authentication and authorization use case

Field	**Specification**
Use case	Authenticate and authorize as an administrator
Actors	Administrator
Goal	An administrator wants to access the admin page (authorize in the system)
Success Scenario	Steps: 1. An authorized administrator goes to the login page. The system requests login and password; 2. The administrator enters login and password; 3. The system verifies the user's credentials; 4. The system verifies if a request was made by a real user (reCAPTCHA); 5. The system redirects the user to the admin page.
Result	The admin-user is successfully authorized in the system and has privileges to review and approve proposed studies by researchers.
Extensions	
*	No access to the database. The system returns an error message.
2a	The user chooses "I forgot my password". The system invokes the "I forgot my password" scenario.
3a	The user with the specified credentials does not exist. Result: denial of access. The system returns an error message. Return to the second step.
3b	The administrator reaches the maximum number of unsuccessful attempts to enter credentials. Result: denial of access. The system returns an error message. The user is blocked for some period of time. This period is set in the configuration of the system.

Table 4.3: Creating new studies use case

Field	**Specification**
Use case	Create a new research study proposal
Actors	Researcher, administrator
Goal	An authorized researcher wants to create a research study proposal and make the research study available to patients for participation.
Success Scenario	Steps: 1. An authorized researcher opens the “New research study” page. 2. The researcher fills in all the required fields to create a research study proposal. 3. The researcher submits the proposal to the administrator for review.
Result	The system saves the research proposal. The administrator receives a request to review the research proposal.
Extensions	
*	No access to the database. The system returns an error message.
3a	The researcher chooses “Save draft”. The system saves a draft of the research proposal.

Table 4.4: Research proposal review use case

Field	**Specification**
Use case	Review a research study proposal
Actors	Administrator
Goal	An administrator wishes to approve a research study proposal and make the study available to patients.
Success Scenario	Steps: 1. The administrator opens a review request attached to a research study proposal. 2. The administrator sets the "Study Status" as "Approved". 3. The administrator sets the privileges for accessing the shared data for each researcher participating in the study.
Result	The research study proposal is approved, and patients can participate in the study. The system saves researchers' privileges on the blockchain.
Extensions	
*	No access to the database. The system returns an error message.
2a	The administrator sets the "Study Status" as "Rejected". Result: The research proposal is rejected, patients cannot participate in the study and researchers cannot get privileges to access data in the study.

Table 4.5: Signing a consent form use case

Field	Specification
Use case	Sign a consent form
Actors	Patients
Goal	A patient wishes to participate in a research study and share their medical data with the study by signing an electronic consent form.
Success Scenario	Steps: 1. Patient goes to the "Research Menu" in the Opal app. 2. Patient selects the study in which they wish to participate. 3. Patient signs the e-consent form for participation in the study. 4. The system saves the consent in the blockchain.
Result	The patient's medical data that have been shared by the patient become accessible to the appropriate study-user[1].
Extensions	
*	No access to the database. The system returns an error message.

[1]study-user concept is explained in Chapter 5.

Table 4.6: Pulling (harvesting) shared data to the Research Portal server use case

Field	**Specification**
Use case	Pull the shared data to be available to the researcher for analysis
Actors	Research Portal
Goal	The Research Portal service wishes to update its shared data database.
Success Scenario	Steps: 1. The Research Portal service, acting as a “study-user“, logins into each of the individual Opal listeners to pull newly-shared data records once per X hours (e.g., once per 24 hours). 2. Each Opal listener provides to the study-user the newly-shared data records that patients consented to sharing in the last X hours, and sends the data to the Research Portal service. 3. The Research Portal service updates its shared data database. 4. The Research Portal service removes the previously-shared data records that are no longer consented to (based on recently-withdrawn consents).
Result	The Research Portal shared data database is updated based on the patients’ consents.
Extensions	
*	No access to the database. The system returns an error message.

Table 4.7: Accessing shared data use case

Field	**Specification**
Use case	Access shared data
Actors	Researcher
Goal	An authorized researcher wants to download the shared data for their research study.
Success Scenario	Steps: 1. The researcher selects their approved study and opens the “Shared Data“ page. 2. The system verifies that the researcher has access rights to the “Shared Data“ of the approved research study. 3. The system selects the consented records and returns them to the researcher in CSV format. 4. The system creates a new logging record in the blockchain with the details of who (the researcher) downloaded what data, when and from which IP address.
Result	The researcher downloads the shared data in CSV format.
Extensions	
*	No access to the database. The system returns an error message.
2a	If the study is not approved or the researcher does not have the appropriate access rights, the system returns a “Forbidden Access” page.
3a	If there are no consented records, the system returns an empty dataset.

4.2.2 Use Case Diagram

A use case diagram is a graphical representation of interactions between actors (users) and systems. A use case diagram consists of actors, use cases within a system, associations between actors and use cases, relations between use cases, and relations between actors.

Use case diagrams are text use cases that are presented in a visual form. The main purpose of use case diagrams is to provide a visual representation of a system's functionality and behavior that allows customers, users, and developers to jointly discuss the designed or existing system.

Figure 4.5 shows a use case diagram of the proposed Research Portal—a data sharing system that is based on the use cases discussed in Section 4.2.1.

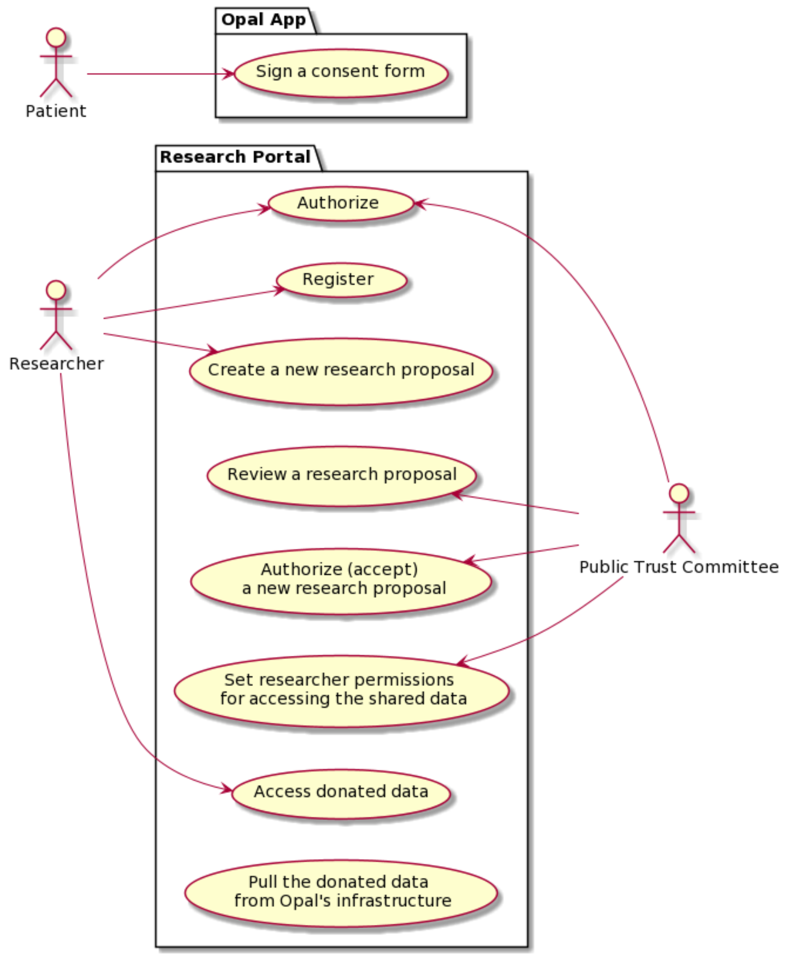


Figure 4.5: Use case diagram of the proposed data-sharing approach. The diagram contains two systems and three actors. The Patient interacts with the Opal App system, whereas the Researcher and the Public Trust Committee interact with the Research Portal.

4.2.3 Sequence Diagram

A sequence diagram is a visual representation of object interactions in a time sequence based on text use cases and use case diagrams. It depicts a sequence of actions in use case scenarios (e.g., creating, processing, deleting) and objects' life cycles. The main elements of a sequence diagram are:

- Actors and objects that interact with each other.
- Parallel vertical dotted lines (lifelines) that reflect time going from top to bottom.
- Activation boxes – rectangles drawn on top of lifelines that reflect the execution of a process.

The interactions between actors and objects are shown by horizontal arrows called messages. Messages transfer control from the sender (from whom the arrow goes) to the recipient (the one to whom the arrow is directed). The arrows show the course of the scenario and the events that occur during it. There are three basic arrows for interactions:

- Synchronous message – the sending actor passes the control to the receiving actor or object that needs to perform some action. The sending actor loses the ability to perform any actions until the action performed by the receiving actor is completed. The arrow is depicted with a solid head.
- Asynchronous message – the sending actor passes the control to the receiving actor or object that needs to perform some action. The main difference from the synchronous message is that the sending actor does not lose the ability to perform other actions while waiting the action to be completed. The arrow is depicted with an open head.
- Reply message – contains a response for the sending actor. The message returns control to the sending actor. The arrow is depicted with a dashed line.

Figure 4.6 illustrates the flow of interactions between actors and the proposed system.

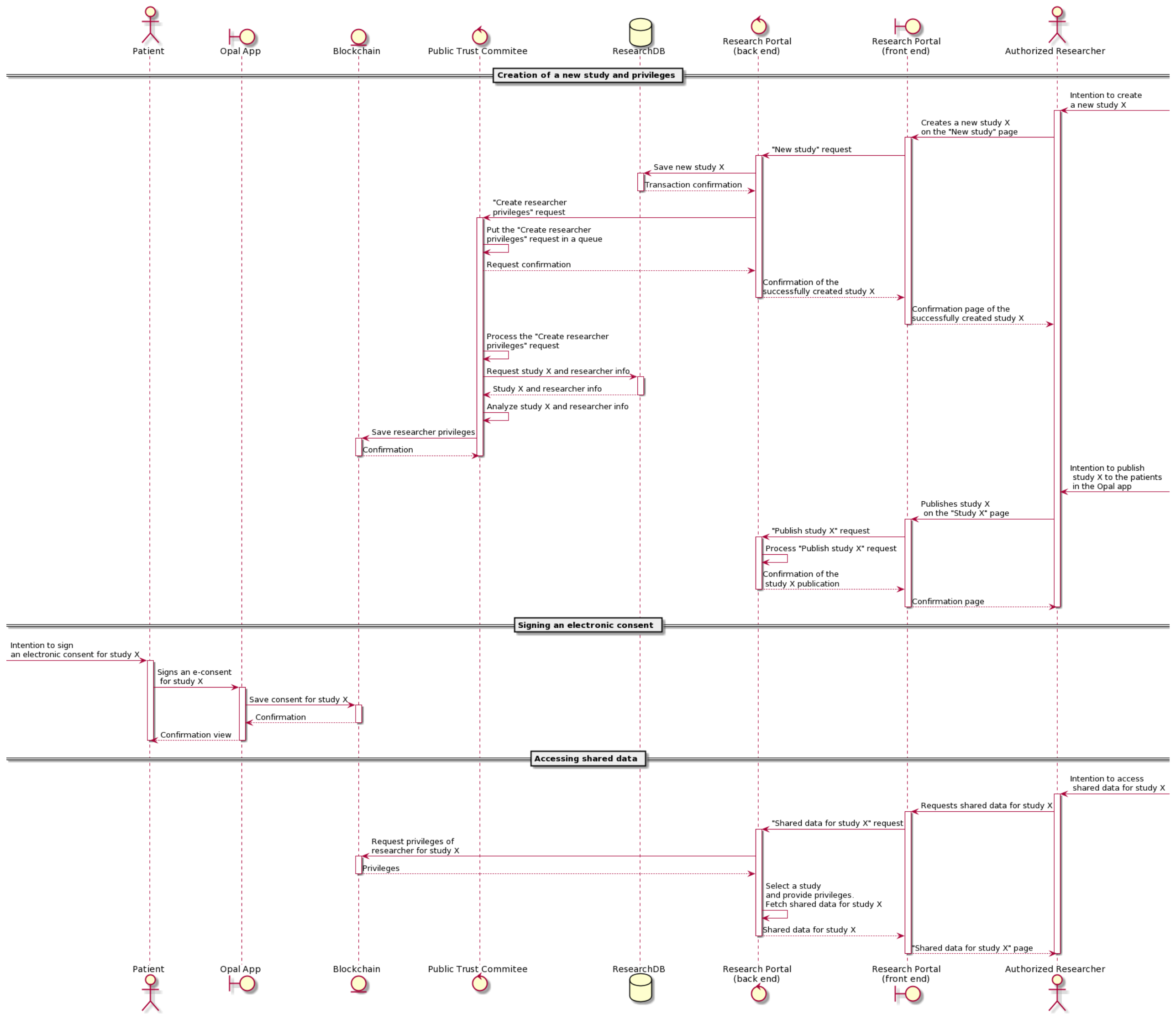

Patient
Opal App
Blockchain
Public Trust Commitee
ResearchDB
Research Portal (back end)
Research Portal (front end)
Authorized Researcher
Creation of a new study and privileges
Intention to create a new study X
Creates a new study X on the "New study" page
"New study" request
Save new study X
Transaction confirmation
"Create researcher privileges" request
Put the "Create researcher privileges" request in a queue
Request confirmation
Confirmation of the successfully created study X
Confirmation page of the successfully created study X
Process the "Create researcher privileges" request
Request study X and researcher info
Study X and researcher info
Analyze study X and researcher info
Save researcher privileges
Confirmation
Intention to publish study X to the patients in the Opal app
Publishes study X on the "Study X" page
"Publish study X" request
Process "Publish study X" request
Confirmation of the study X publication
Confirmation page
Signing an electronic consent
Intention to sign an electronic consent for study X
Signs an e-consent for study X
Save consent for study X
Confirmation
Confirmation view
Accessing shared data
Intention to access shared data for study X
Requests shared data for study X
"Shared data for study X" request
Request privileges of researcher for study X
Privileges
Select a study and provide privileges. Fetch shared data for study X
Shared data for study X
"Shared data for study X" page

Figure 4.6: Sequence diagram of the proposed Research Portal. The diagram does not include the process of data gathering from the OpalDB (harvesting).

4.3 Technology stack

In this section we present the technology stack used to implement a patient-controlled data sharing service.

4.3.1 Laravel Framework

Since the Opal backend has several services written in the PHP language, we decided to be consistent with the existing code base and use PHP for developing our prototype of the Research Portal. Doing so will help prevent the Opal project from having too many different technologies and languages. Also, it will allow Opal developers to quickly understand the prototype code and start developing the production version.

In order to write clean PHP code, adhering to best practices, we decided to use a PHP framework. We analyzed the most popular PHP frameworks, such as CakePHP, Symfony, Yii, and Laravel. According to Google Trends search statistics, Laravel is the most popular framework. Figure 4.7 demonstrates the popularity of these frameworks from 2004 till the present time.

Due to the popularity and variety of features, we decided to use Laravel [130, 131]. The framework follows the model-view-controller (MVC) architectural pattern and based on the Symfony framework. It provides developers with a rich set of features and robust tools that allow them to boost the speed of web development and build scalable applications. The main features of the framework are the following:

- Built-in packages that have frequently used modules and libraries. Packages are provided through Composer and Packagist.
- Artisan – a command-line utility that automates frequent actions performed by developers during development.
- Configuration management that allows developers to keep different configurations for different environments in an efficient way.

- Schema builder that helps to create a database schema by using PHP code. Also, it supports migrations to track the changes of the schema.
- Query builder for querying databases without using SQL. Laravel provides PHP classes and methods for making queries programmatically.
- Eloquent - an Object Relational Mapper (ORM) for presenting database tables as classes and each single row in the table as an object instance (active record pattern).
- Native authentication which includes features such as registration, authorization, password recovery.

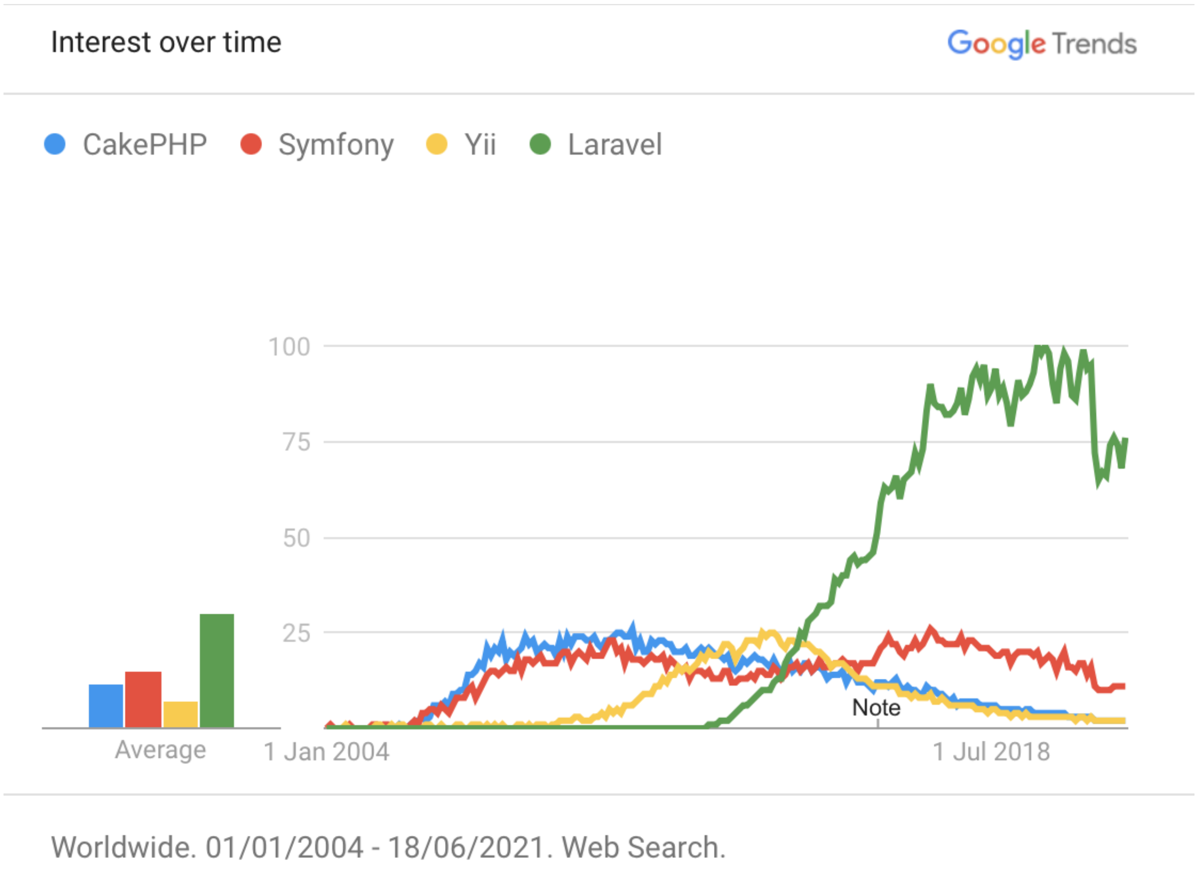


Figure 4.7: Search per keyword in the world since 2004. Numbers represent search interest relative to the highest point on the chart for the given region and time. A value of 100 is the peak popularity for the term. A value of 50 means that the term is half as popular. A score of 0 means that there was not enough data for this term [129].

4.3.2 Bootstrap & AdminLTE

For implementing the front-end part of the Research Portal, we used the Bootstrap framework and specifically, the AdminLTE template. Bootstrap is an open-source framework that simplifies the development of responsive websites [132]. The framework provides ready-to-use components with a variety of CSS styles. One of the most important features of the framework is a grid system that allows web pages to adjust their size dynamically on different screen sizes. AdminLTE is an open source admin dashboard and control panel theme that provided us with a range of responsive and reusable templates [133].

4.3.3 Docker

For developing and testing purposes, we deployed a Hyperledger Fabric Network by using Docker [134]. Docker is an OS-level virtualization platform for running applications in isolated environments called containers. Containers contain all the needed libraries and configuration files to run an application and multiple containers can be deployed on the same host simultaneously. Therefore, Docker can simulate a blockchain network on the local computer by running several containers. Container orchestration tools, such as Docker Swarm, Kubernetes, and OpenShift, allow coordination, management, and deployment of containers across different hosts.

Chapter 5

Results

5.1 Stakeholder Co-design

For the participatory stakeholder co-design process, the PARTAGE team formed a group of stakeholders that met on a weekly basis to discuss the design and development of the data-sharing solution. Table 5.1 lists the stakeholders who participated in these weekly meetings and the expertise and perspectives that they brought to the team.

Table 5.1: List of the stakeholders who participated in the participatory stakeholder co-design process. Participants met weekly to discuss the data-sharing solution and to ensure that the results of the project meet their needs and are usable. Some participants played more than one role.

Perspective	**Professional role of Individuals**	**Number of Individuals**
Patients	Cancer patients	2
Researchers	Informatics researcher/medical physicist, bioinformatician	2
Clinicians	Registered nurse	1
Ethico-legal	Lawyer specializing on personal health information	1
Provateurs	Students (medical physics and computer science)	2
Opal development team	Opal chief architect and DevSecOps manager	2

In the period between June 2020 and September 2021, the PARTAGE team held a total of 60 weekly meetings. Each meeting was one hour and 15 minutes long on average. All meetings were conducted online via the communication platform Microsoft Teams. Through the active participation of all the team members, the following aspects of the data-sharing solution were investigated:

- **Privacy impact assessment (PIA)**: The lawyer on the team investigated all the factors that can have a positive or negative impact on the privacy of the individuals in framework of the Opal app. These factors are: (1) the compliance of the project with regard to privacy and data protection legislation, (2) the identification of privacy risks generated by the project and the assessment of their respective impacts, (3) strategies to avoid or effectively reduce these risks. A report was prepared that includes the PIA analysis as well as recommendations for better personal data protection for the privacy of individuals whose data pass through Opal.

- **Patients' preferences for the consent process**: One of the central discussions in the co-design meetings was to determine if the data-sharing consent process should be global (i.e., consent once to share data to all studies) or per-study (i.e., provide consent to share data for each individual study). Another aspect of data-sharing that was studied was the question of whether or not the sharing of individual types of data should be controlled by the patient and how such a sharing feature should be displayed in the Opal app (e.g., if patients want the ability to only share specific data types). While the PARTAGE team had its own vision and preferences, gaining further perspectives from the wider patient population was required and emerged as a main discussion topic for the focus group described below.

- **De-identification instead of anonymization**: Two data protection strategies were investigated. One strategy was de-identification of data (a.k.a. pseudonymization)—a procedure by which personally identifiable information is removed or replaced with a fictitious value (i.e., pseudonym). In this way, researchers will get the shared data without the personal information of participants. However, personal information can still be linked back to the patient in the Opal system. Another strategy was data anonymization. This approach removes all personally identifiable information and permanently severs the link to the original data source. This prevents the data being connected back to the patient, even in Opal. Both strategies were explored extensively in the discussions and in the focus group to determine patients' understanding and preferences.

- **System requirements**: Based on the PIA analysis and the co-design discussions, we identified the system requirements of the Research Portal. First, we formulated the requirements in the text use cases form as discussed in Section 4.2.1. Then, we created use case and sequence diagrams for a visual representation of interactions between users and systems. The designed requirements and graphical visualizations were presented in Chapter 4. Also, the PARTAGE team determined that the Research Portal must be a cloud solution and deployed outside of the RI-MUHC/MUHC network. This is due to the hospital's firewall and security measures, which prohibit external connections to the network. A cloud-based solution will allow researchers to use the Research Portal even if they do not have access to the hospital's network. Last but not least, we identified that (1) patients' consents, (2) researchers' data-access privileges, and (3) researchers' data-access history must be stored in a decentralized environment, to ensure the data are recorded in a tamper-proof manner. As explained previously, we decided to use blockchain technology.

- **User interface design of the Research Portal**: One of the first phases of implementing the Research Portal was designing web page mockups of it. The mockups were discussed collaboratively by the co-design participants and then modified according to the feedback received. After several iterations, the team agreed on the final user interface design.

- **User interface design of the Opal Research Menu**: A medical physics student worked on a user-friendly, intuitive interface for a Research Menu in Opal—a sub-menu in the Opal app where patients can share their medical data for research purposes. Different mockup interfaces were created in order to explore different possible options.

- **Focus group**: The PARTAGE team held a focus group to get patients' feedback on their preferences for (1) the consent and data-sharing process, (2) the data protection strategy (de-identification instead of anonymization and other considerations), and (3) Research Menu interfaces in the Opal app. Participants were invited to the focus group via Opal and 42 patients expressed interest to participate. 10 invitations were randomly sent to the interested patients, however only four patients ultimately attended the focus group. The focus group session was split into two parts of 45 minutes each. During the first part, PARTAGE team members gave a presentation on the PARTAGE project's concepts and its goals. In the second section, patients provided feedback on their preferences for the data-sharing solution. Patients indicated that they themselves would choose the global data-sharing option, (i.e., consent once to share data to all studies) but at the same time the per-study consent option should be included for flexibility. Also, patients explored different data protection strategies. Due to limitations for researchers and patients being unable to revoke access to their data if they are anonymized and the inability to link anonymized data from different sources, patients agreed that team should move forward with the de-identification approach. Last but not least, the focus group participants provided feedback on the user interface design of the Opal Research Menu. To make the data-sharing process more rewarding, patients identified they would like to see a report or some feedback regarding the research studies that their data are used in.

Based on the co-design findings, we built an architecture for the data-sharing mechanism shown in Figure 5.1. The solution is based on the blockchain technology. On the left side of the architecture, we have the Opal App which communicates with the Opal Infrastructure (clinical world). When a patient decides to share their data, first, the app by using a blockchain client on the Opal Infrastructure sends consents to the blockchain network. After that, the specified medical data are de-identified and sent to the research database. The actual data-sending mechanism is beyond the scope of this project but would like involve a harvester mechanism and proxy "research study user" access to the patient's data. On the right side of the architecture, we have the Research Portal for the Public Trust Committee and the researchers (research world). The portal has its own back end and database, which keeps researchers' profiles. The portal's back end checks (1) if the researcher's study was approved by the Public Trust Committee, and (2) if a researcher has privileges to access the shared data within it. If both conditions are met, a researcher can access the patient-shared data. All the requests to the Research Database are logged on the blockchain.

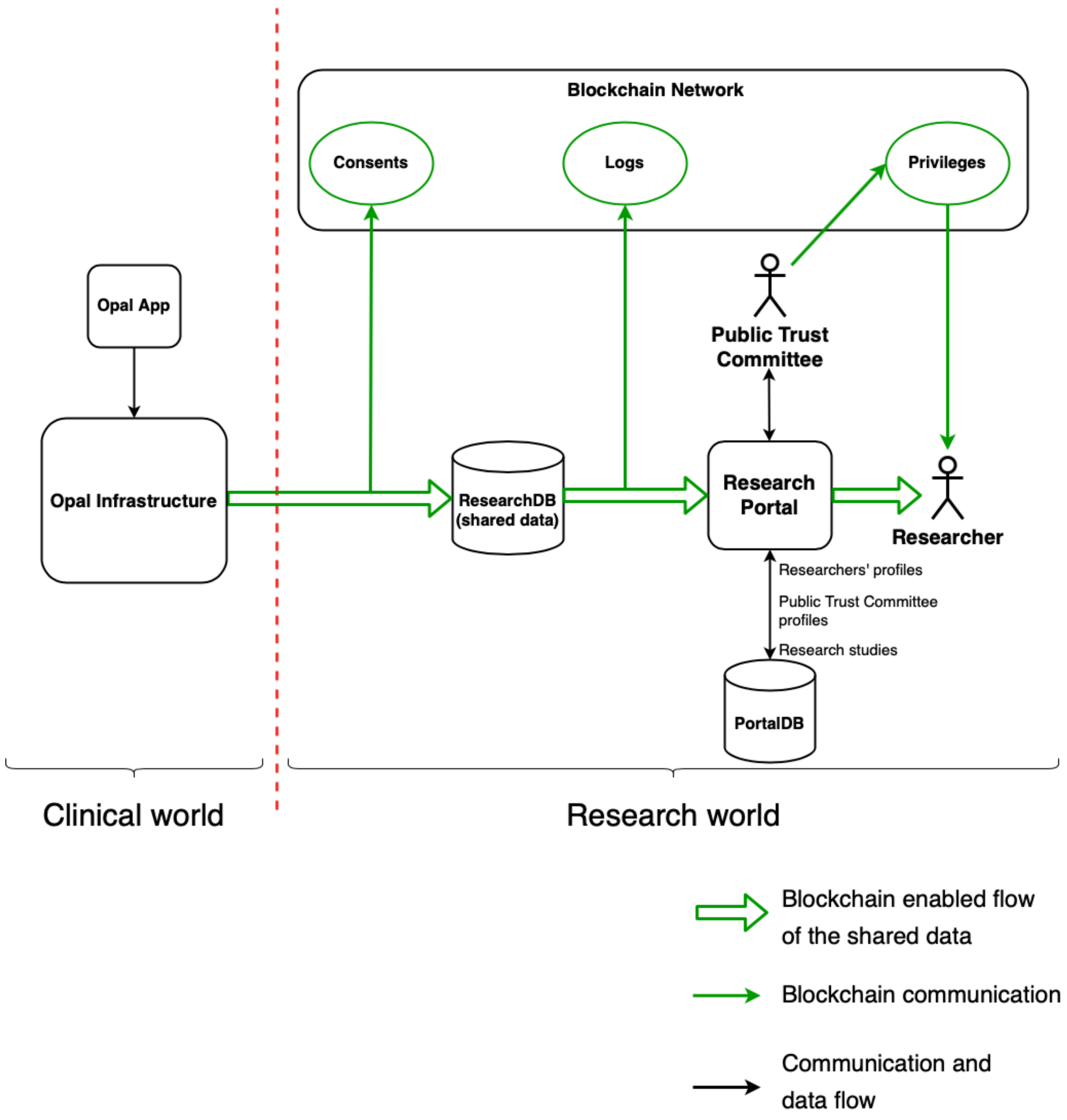


Figure 5.1: Designed architecture of the Research Portal based on the results of the stakeholder co-design process. The architecture consists of two parts that are separated by the red dashed line: the clinical world (hospitals) and the research world. The clinical world includes the Opal app and its back end infrastructure, whereas the research world includes the Research Portal and a permissioned blockchain network. Patients share their medical data through the Opal app by signing an e-consent form. The patient consents are stored in the blockchain network. Shared de-identified data end up (technical details of the transfer beyond the scope of this thesis) in a Research Database (i.e., ResearchDB) that is securely hosted outside the hospital's network. The shared data are transferred on a regular basis to the ResearchDB based on the patient consents (blockchain enabled flow). Researchers can access the shared data via the Research Portal if there is a match between the patient consents and their data access privileges as provided to them by the Public Trust Committee. All data accesses are logged on the blockchain.

5.2 Description of the Implemented Solution

In this section, we explain the implemented system design as well as the design of the deployed blockchain network. Also, we present the Research Portal and its features.

5.2.1 System Design

Using (1) the system requirements, (2) the designed architecture, and (3) the knowledge gained during the co-design process, we designed a system for the Research Portal as shown in Figure 5.2. The Research Portal is a web application, where the front end (i.e., website) communicates with the back end (i.e., server) via Hypertext Transfer Protocol Secure (HTTPS). Both researchers and the Public Trust Committee use the same website. They have different user permissions and web pages based on their roles and as authenticated by their credentials (i.e., email address and password). The Research Portal is publicly available, so, in principle, anybody can register for a researcher account. The Public Trust Committee administrators can be added or deleted only by the system owners (Opal development team initially). The front end is based on the AdminLTE template and Bootstrap framework. The server side includes four main components: (1) back end logic, (2) the Research Portal database, (3) a blockchain client application, and (4) the shared data databases.

The back end logic utilizes the PHP Laravel framework for handling front end requests and managing access to the portal's resources. The Portal Database (PortalDB) stores all the data related to the Research Portal, such as users, research studies, etc. However, the PortalDB does not store shared data. The PortaDB is implemented using Eloquent ORM and MariaDB—a fork of the MySQL relational database management system (RDBMS). Each PortalDB table has a corresponding Eloquent ORM "model" in the back end that is used to interact with that table. Eloquent models allow the back end to retrieve, insert, update, and delete records from the database tables.

The back end communicates with the blockchain network via the blockchain client application. The blockchain client provides RESTful API endpoints that are implemented using

JavaScript and run in the Node.js runtime environment. The endpoints utilize the Hyperledger Fabric SDK library that (1) allows connections to a peer within the blockchain network, (2) enables access to any of the blockchain channels for which that peer is a member, and (3) provides access to the chaincodes running within that blockchain network.

When a researcher requests the shared data within a study, the back end checks if the researcher has access privileges for that study stored on the blockchain. If (1) the study was approved by the Public Trust Committee, (2) and the researcher has privileges, the shared data are transferred to the researcher. To pull the data from the "Shared data" databases, the back end utilizes raw SQL (Structured Query Language) queries. For simplicity in this thesis project, the "Shared data" databases have the OpalDB and QuestionnairesDB schemas of the Opal Infrastructure (see Figure 2.3). Fake medical records were hardcoded into these databases for testing purposes.

The following components of the design were beyond the scope of this thesis project, and thus they were not implemented:

- Study-user: An Opal user account that aggregates the shared data within a study. When a Public Trust Committee member approves a new research study, the Research Portal's back end sends a request to the Opal's infrastructure to create a new Opal account—a study-user. Patient's consented data will be automatically shared with a study-user account as part of the consent process. Thus, the Harvester service will know which data can be pulled to the "Shared data" databases. Also, it will allow the Harvester service to distinguish what shared records belong to what studies.

- Blockchain client application for hospitals: RESTful API endpoints that allow Opal app users to sign or revoke e-consents. The endpoints will have the same implementation as the implemented blockchain client for the fake consents in the Research Portal (for the purpose of this thesis project, just fake consents were implemented). However, the client must be configured and deployed according to the host hospitals' servers' characteristics and security regulations. Also, the Opal App and the Listener implementations must be updated as appropriate.

- Harvester: A cron-job (or equivalent) service that updates the “Shared data” databases once per X hours. The service logs in to each study-user account and copies the shared records to the “Shared data” databases that the study-user has access to. Also, the service should remove records from the databases for which e-consents were revoked, so researcher users get only consented records.

The design of the Research Portal follows the Multilevel Access Control model—a combination of the DAC and MAC models. First, the DAC model is applied for the signing e-consents process (i.e., patients determines who can access their data). Then, the Public Trust Committee represents the MAC model (i.e., the administrators decide who can access the shared data).

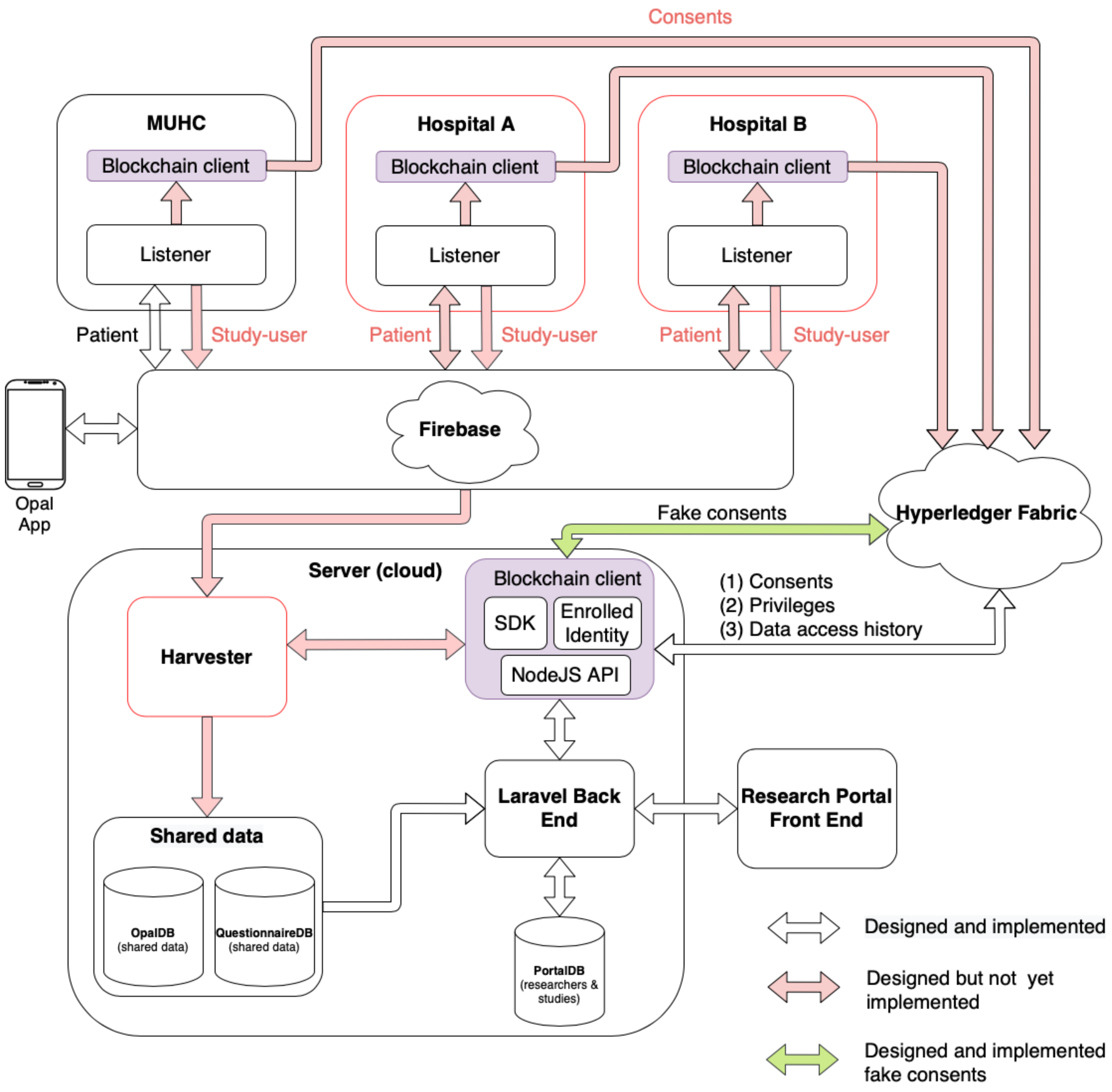


Figure 5.2: The design of the Research Portal. The portal is a web application that uses the Laravel and Bootstrap frameworks for its back end and front end respectively. For communication with the blockchain network, the server runs a Node.js API (blockchain client application). The Harvester is envisaged as a cron-job service that logs in to each study-user, collects the shared data and puts them into the Shared Data storage. The storage consists of two databases that have the OpalDB and QuestionnaireDB schemas to facilitate de-identified replication of data from Opal. Details and terms are explained in the main text. Components that have been designed but that have not been implemented as part of this thesis project are shown in pink. For the purpose of this project, fake patient consents were used.

The implemented prototype Research Portal solution was deployed on a server at the RI-MUHC. Tables 5.2-5.3 list the server's technical characteristics and the software used to run its back end and front end. To test the designed system and its business logic, we implemented and deployed additional endpoints and chaincodes that allow Public Trust Committee members to create fake e-consents on behalf of fake patients. This functionality was implemented only for testing purposes and will be eliminated in future versions of the portal.

Table 5.2: The technical characteristics and components of the server used for deployment of the Research Portal. The server is a virtual machine that is physically located at the RI-MUHC.

	Name	**Details**
Server Type	Virtual Machine (VM)	Virtualization type: full
		Hypervisor: Kernel-based Virtual Machine (KVM)
		Physical location: RI-MUHC
Central Processing Unit (CPU)	Intel Core Processor (Broadwell)	Model: 61
		Architecture: x86_64
		CPU(s): 4
		Thread(s) per core: 1
		Core(s) per socket: 1
		Socket(s): 4
		CPU MHz: 2394.454
Storage	Hard Disk Drive (HDD)	Write speed: 626 MB/s
		Real read speed: 949 MB/s
		Read speed from buffer: 4.8 GB/s
Random Access Memory (RAM)	Dual in-line memory module (DIMM)	Size: 4GB
Operating System (OS)	Linux	Distribution: CentOS
		Release: 7.9.2009

Table 5.3: List of the software and frameworks used to run the Research Portal on the RI-MUHC's server.

	Software/Framework	**Version**
Back end	Web server nginx	1.16.1
	PHP	7.4.19
	Laravel Framework	8.29.0
	Composer—Dependency Manager for PHP	1.10.20
	MariaDB	10.4.8
Front end	AdminLTE Template	3.0.4
	Bootstrap Framework	4.4.1
	jQuery—JavaScript library	3.4.1
	npm—Dependency Manager for JavaScript	7.12.1

5.2.2 Hyperledger Fabric Network for the Prototype Research Portal

For implementing the developed use cases and the system design, both of which require storing data in a blockchain, we deployed a Hyperledger Fabric network. The architecture of the deployed blockchain network is shown in Figure 5.3. The network consists of three organizations (i.e., Org1, Org2, Org3), three channels (i.e., patients' consents channel, researchers' privileges channel, and data access logs channel), and an ordering service. Each organization consists of one peer that connected to the networks channels. The ordering service contains only one ordering peer that utilizes the Raft algorithm. To run the network locally, we used the Docker platform, where each peer was simulated in an isolated container. The chaincode binaries were installed on the Org1 peer. The list of software used to run the blockchain network, including the client application, are shown in Table 5.4. Tables 5.5-5.9 describe the implemented client application endpoints and installed chaincodes.

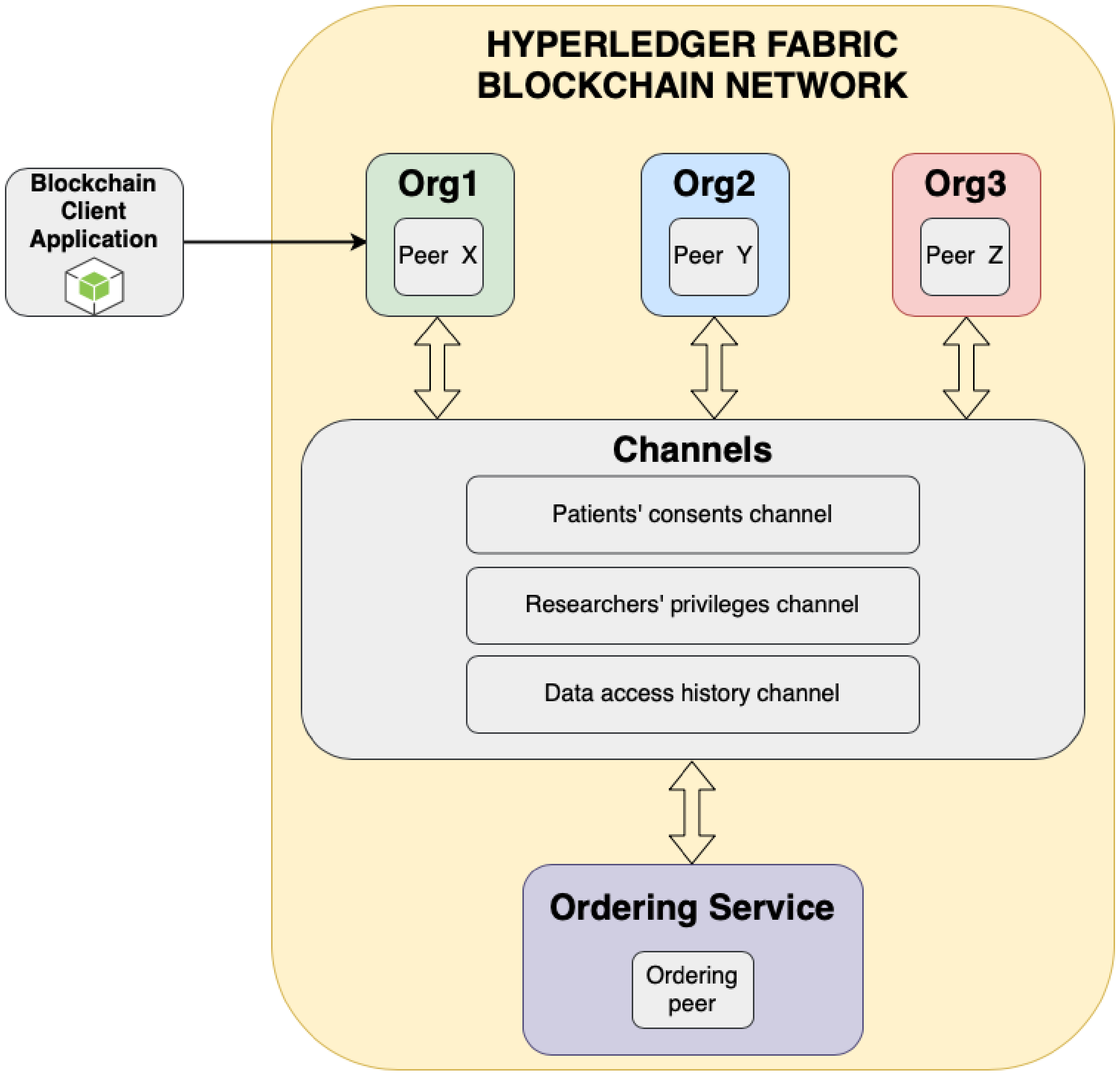

HYPERLEDGER FABRIC BLOCKCHAIN NETWORK
Blockchain Client Application
Org1
Peer X
Org2
Peer Y
Org3
Peer Z
Channels
Patients' consents channel
Researchers' privileges channel
Data access history channel
Ordering Service
Ordering peer

Figure 5.3: The design of the Hyperledger Fabric blockchain network used in the prototype Research Portal. The network consists of three organizations, where each organization has one peer. Peers Y and Z are committing peers, whereas peer X is both an endorsing and a committing peer. All three organizations are connected to three channels, where each channel is intended for a specific type of data: (1) patients' consents, (2) researchers' privileges, (3) data access logs. The ordering service contains only one ordering peer that utilizes the Raft algorithm. Peer X has chaincodes installed on it that are invoked by the client application using the Node.js SDK library. The network was deployed to the RI-MUHC's server using Docker, where each of the peers were simulated in separate containers.

Table 5.4: List of the software used to run the Hyperledger Fabric blockchain network and a client application.

Name	Version
Hyperledger Fabric	2.3
Node.js—back end JavaScript runtime environment.	14.16.1
Go language—used for developing and compiling the chaincodes.	1.16.4
Docker—used for simulating the blockchain network.	20.10.5

Table 5.5: List of the API endpoints provided by the blockchain client application for managing researchers' privileges.

Blockchain client API (endpoint)	Description
/api/studies/create-study/	Allows the Public Trust Committee to create a new study record on the blockchain that contains (1) an ID of the study that links to the study information stored in the PortalDB, (2) a binary flag of the study status (i.e., approved or not approved), and (3) list of the researchers who can access the study. For each individual researcher, the committee sets privileges (e.g., no access, read only, full access).
/api/studies/update-study/{study index}	Allows the Public Trust Committee to update a study record with the researchers' privileges stored on the blockchain.
/api/studies/query/{study index}	Allows retrieval of a study record stored on the blockchain. The endpoint is used by the Public Trust Committee to see the current privileges of the researchers.

Table 5.6: List of the API endpoints provided by the blockchain client application for managing patients' consents.

Blockchain client API (endpoint)	**Description**
/api/consents/update-recipient	Allows patients to create/update personal e-consents. An e-consent contains the patient's ID, metadata regarding the shared data, and date/time information regarding when the consent was created and updated. Every e-consent is assigned to a recipient—a user who will get access to the shared data. In the current version of the Research Portal an e-consent can be assigned only to a study-user. However this functionality can be extended in future versions, so e-consents could potentially be used to allow patients to share their data with family members, caregivers, other patients, etc. For testing purposes, this endpoint is made available for Public Trust Committee users, who can create "fake e-consents".
/api/consents/get-all-consents	Allows Public Trust Committee users to retrieve all patients' e-consents. This endpoint is implemented for testing purposes to ensure that all the e-consent operations work correctly. The endpoint may be eliminated in future versions of the Research Portal.

Table 5.7: List of the API endpoints provided by the blockchain client application for checking if a researcher is allowed to access the shared data.

Blockchain client API (endpoint)	**Description**
/api/privileges/check-shared-data-access	Checks if a researcher can access the study's "Shared Data" page based on the (1) study's approval and (2) researcher's privileges.
/api/privileges/permitted-data-for-download	Checks what shared data can be downloaded based on the (1) study approval, (2) researcher's privileges, and (3) patients' consents.

Table 5.8: List of the API endpoints provided by the blockchain client application for adding and retrieving data access history records to/from the blockchain.

Blockchain client API (endpoint)	**Description**
/api/log/add-log-record/	Creates a new log record on the blockchain. A record is created when the researcher accesses shared data. The record includes the researcher's ID (requester), metadata regarding the data accessed, and date/time information about when the data were accessed.
/api/log/get-all-log-records	Allows retrieval of all the log records stored on the blockchain.

Table 5.9: List of the designed and implemented chaincodes. The chaincodes deployed to three different channels: (1) researchers' privileges channel, (2) patients' consents channel, and (3) data access history channel.

Chaincode	**Description**
Researchers' privileges channel	
CreateStudy	Creates a new study record that contains its status (e.g., approved, not approved) and researchers' privileges. The chaincode is invoked by the "/api/studies/create-study/" API endpoint. The details of a study record are provided in Table 5.5.
UpdateStudy	Updates a study record and its researchers' privileges. The chaincode is invoked by the "/api/studies/update-study/{study index}" API endpoint.
ReadStudy	Retrieves a study with its researchers' privileges. The chaincode is used by the (1) "/api/studies/query/{study index}", (2) "/api/privileges/check-shared-data-access", and (3) "/api/privileges/permitted-data-for-download" API endpoints.
Patients' consents channel	
UpdateRecipientUser	Creates/updates an e-consent assigned to a recipient (e.g., study-user). The chaincode is invoked by the "/api/consents/update-recipient" API endpoint. The purpose of the chaincode is described in Table 5.6.
ReadRecipientUser	Retrieves a recipient (e.g., study-user) with its all assigned consents. The chaincode is invoked by the "/api/privileges/permitted-data-for-download" API endpoint.
GetAllRecipientUsers	Returns a list of the recipient users and their assigned consents. The chaincode is invoked by the "/api/consents/get-all-consents" API endpoint. It is used for testing purposes and may be eliminated in the future versions of the Research Portal.
Data access history (logs) channel	
CreateLogRecord	Creates a new log recrod. The chaincode is invoked by the "/api/log/add-log-record/" API endpoint.
GetAllLogRecords	Retrieves a list of all log records. The chaincode is invoked by the "/api/log/get-all-log-records" API endpoint.

5.2.3 The Research Portal

This section presents the implemented client web application of the Research Portal (front end). As stated above, the Research Portal is intended for two user types: researchers and the Public Trust Committee members (administrators). Both user types use the same web page for authorization, but access different web pages and rights once authorized. Anybody can register for a new researcher account via the "Registration" page. At this point, new Public Trust Committee user cannot be registered via the portal and can be created only by the Opal development team. Figure 5.4 shows the implemented forms for registering new researchers and authenticating in the portal.

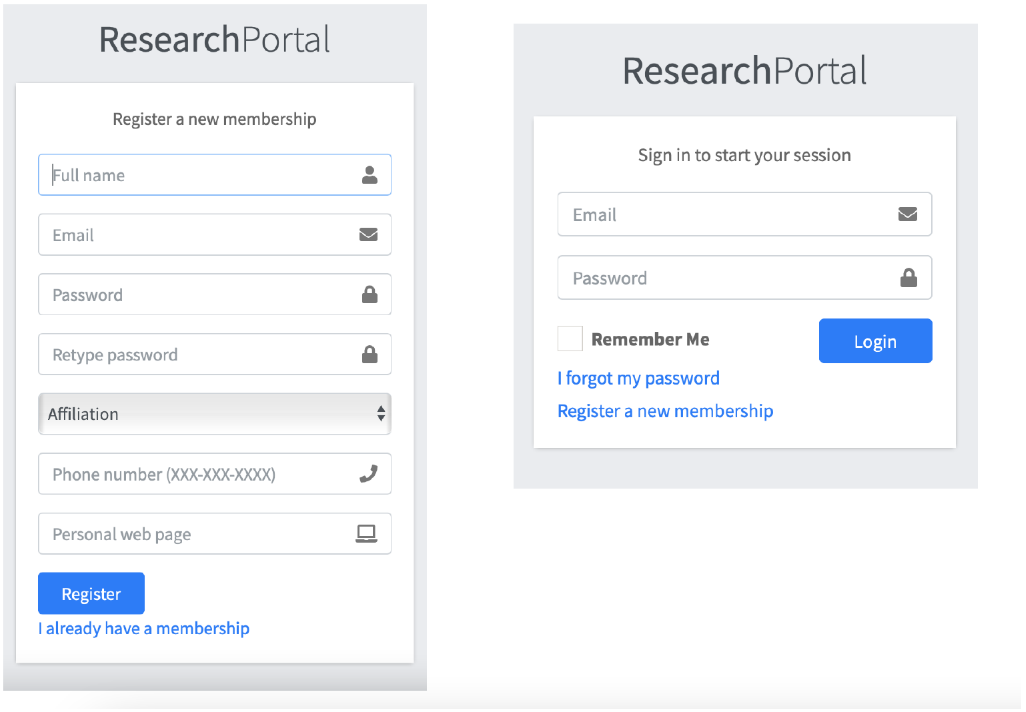


Figure 5.4: The Research Portal's web forms for registration and authentication. The left form is intended for registering a new researcher. The right form is intended for authentication.

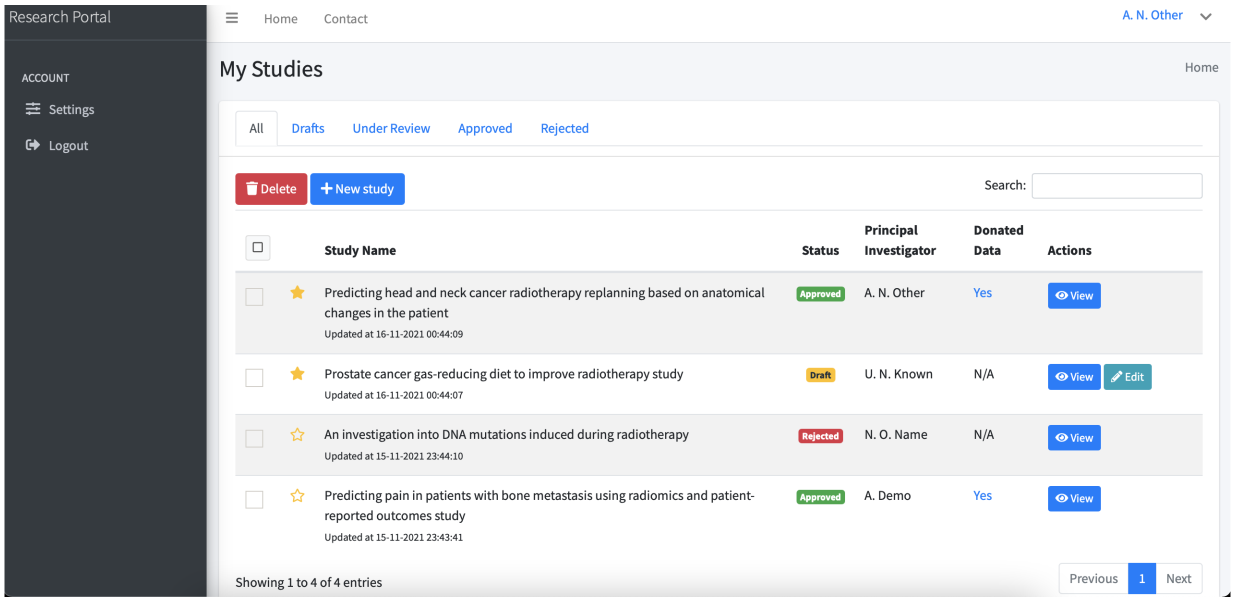
The following subsections present the web pages of an authorized researcher and an authorized Public Trust Committee user (administrator).

5.2.3.1 Researchers

Once authenticated as a researcher, the user is redirected to the Research Portal “Home” page. This page allows the researcher to create new research studies and to manage existing ones. When a researcher creates a new research study, the study must be approved by the Public Trust Committee (administrator) before it can be used. Only then, the study becomes available for patients so they can share their data with it. Thus, there are four research study status types:

- **Draft**: A new research study that is created and saved in the Research Portal but not yet submitted to the Public Trust Committee for review. This study is not available for study participants (patients).
- **Under review**: A new research study that is saved in the Research Portal and submitted to the Public Trust Committee for review. This study is not available for study participants.
- **Approved**: A study that is approved by the Public Trust Committee and available for study participants to share their data with it. Researchers with appropriate privileges can access the data shared with this study.
- **Rejected**: A study that has been rejected by the Public Trust Committee and not available for study participants.

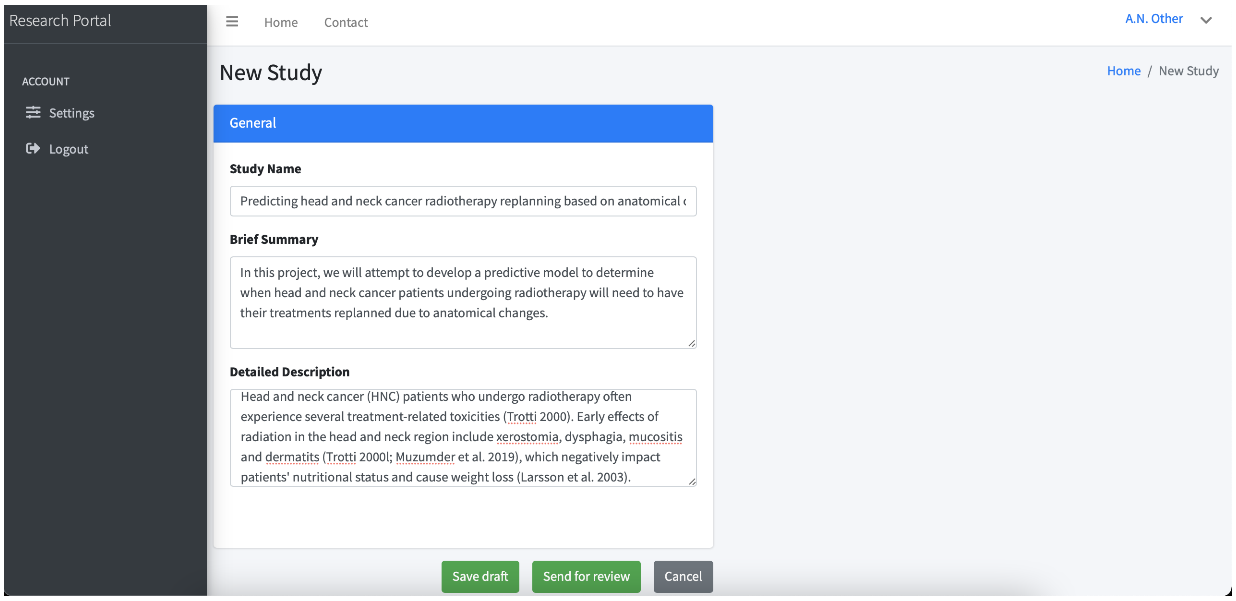
To create a new study, a researcher must provide (1) the study name, (2) a brief summary of the study, and (3) a detailed description of the study. Once the committee approves a new study and sets the privileges for it, the researcher can access the data shared by patients with the study. The data are grouped by categories and provided in comma-separated (CSV) files. In the current version of the Research Portal, there are four categories of patient data: (1) questionnaires, (2) lab results, (3) diagnosis, (4) appointments. Figures 5.5-5.7 illustrate the implemented “Home”, “New Study”, “Shared Data” pages available for authorized researchers.



		Study Name	Status	Principal Investigator	Donated Data	Actions
	★	Predicting head and neck cancer radiotherapy replanning based on anatomical changes in the patient Updated at 16-11-2021 00:44:09	Approved	A. N. Other	Yes	View
	★	Prostate cancer gas-reducing diet to improve radiotherapy study Updated at 16-11-2021 00:44:07	Draft	U. N. Known	N/A	View Edit
	☆	An investigation into DNA mutations induced during radiotherapy Updated at 15-11-2021 23:44:10	Rejected	N. O. Name	N/A	View
	☆	Predicting pain in patients with bone metastasis using radiomics and patient-reported outcomes study Updated at 15-11-2021 23:43:41	Approved	A. Demo	Yes	View

Showing 1 to 4 of 4 entries

Figure 5.5: Researcher's "Home" page that lists all the research studies linked to the authorized researcher. The page allows a researcher to create, read, update, and delete studies. Also, from this page, a researcher can access a study's shared data if (1) the study was approved by Public Trust Committee and (2) the researcher has privileges to access the data as set by the committee.



Study Name

Predicting head and neck cancer radiotherapy replanning based on anatomical c

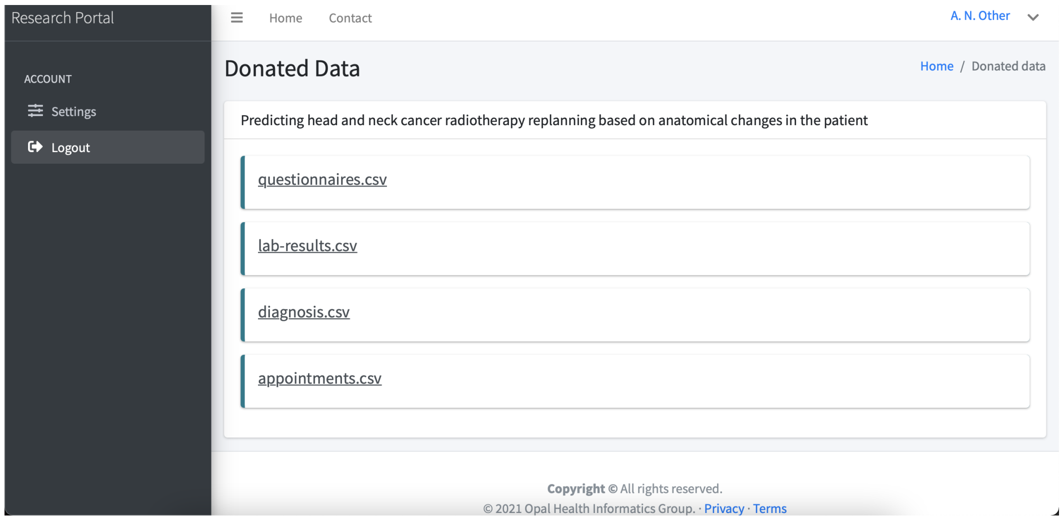
Brief Summary

In this project, we will attempt to develop a predictive model to determine when head and neck cancer patients undergoing radiotherapy will need to have their treatments replanned due to anatomical changes.

Detailed Description

Head and neck cancer (HNC) patients who undergo radiotherapy often experience several treatment-related toxicities (Trotti 2000). Early effects of radiation in the head and neck region include xerostomia, dysphagia, mucositis and dermatits (Trotti 2000l; Muzumder et al. 2019), which negatively impact patients' nutritional status and cause weight loss (Larsson et al. 2003).

Figure 5.6: The "New Study" page that allows an authorized researcher to create a new study. A research study must include three components: (1) study name, (2) study summary, and (3) study detailed description. A researcher can save a study as a draft or submit it for review to the Public Trust Committee.



Donated Data

Predicting head and neck cancer radiotherapy replanning based on anatomical changes in the patient

questionnaires.csv

lab-results.csv

diagnosis.csv

appointments.csv

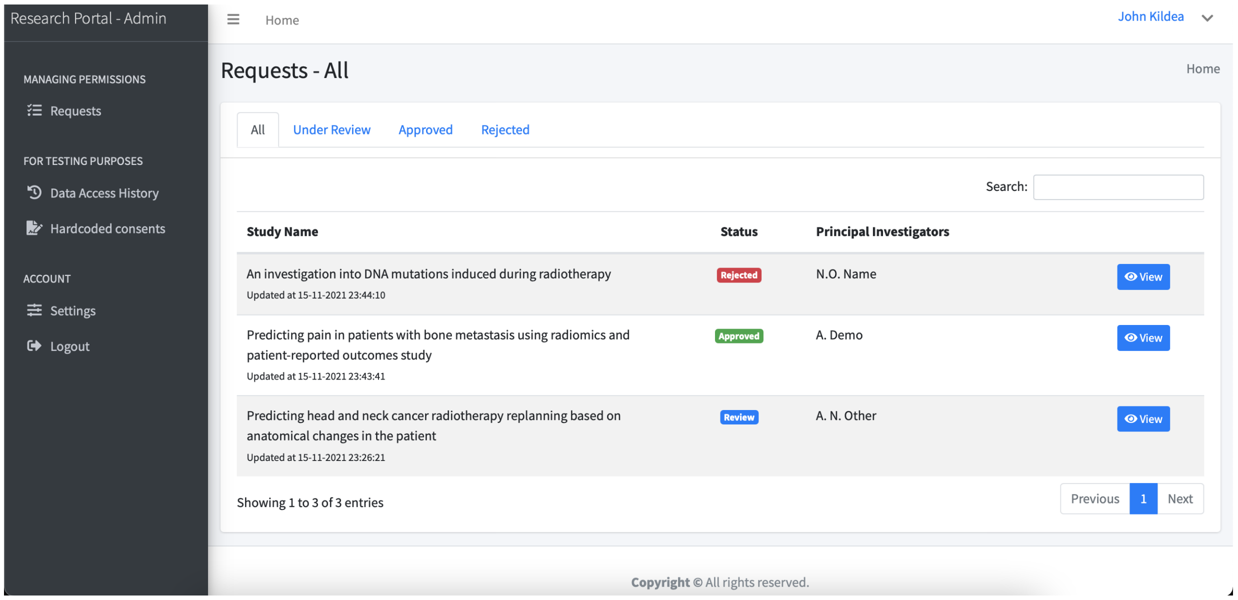
Copyright © All rights reserved.
© 2021 Opal Health Informatics Group. · Privacy · Terms

Figure 5.7: The "Shared Data" page allows an authorized, privileged researcher to download the shared data of a particular study. The data are grouped in categories and provided in a comma-separated value (CSV) files. The implemented prototype version of the Research Portal supports four data categories: (1) questionnaires, (2) lab results, (3) diagnosis information, and (4) appointments. Each research study has different data sets based on the patients' consents. The page is not available to researchers who do not have appropriate privileges.

5.2.3.2 Public Trust Committee

Once authorized as a member of the Public Trust Committee, a Research Portal user is redirected to the "Home" page of the Public Trust Committee. The page lists all the studies that have been submitted for review. During the review process, the administrator evaluates a proposed research study and decides if it can be published in Opal for participation by setting the study status as "approved" or "rejected". Also, the administrator sets the researchers' privileges, so only privileged researchers can access the shared data. To test the designed solution and the blockchain's functionality , we implemented various pages for testing purposes. On the "New Consent" page, the administrator can create or update fake e-consents on behalf of fake patients. On the "Data Access History" page, the administrator can see the log records of the data requests by researchers. Pages for e-consents and data access history were implemented only for this prototype of the Research Portal and will be eliminated in future versions. Figures

5.8-5.12 illustrate the implemented "Home", "Review Study", "Consents", "New Consent", "Data Access Hisotry" pages that are available for authorized Public Trust Committee users.



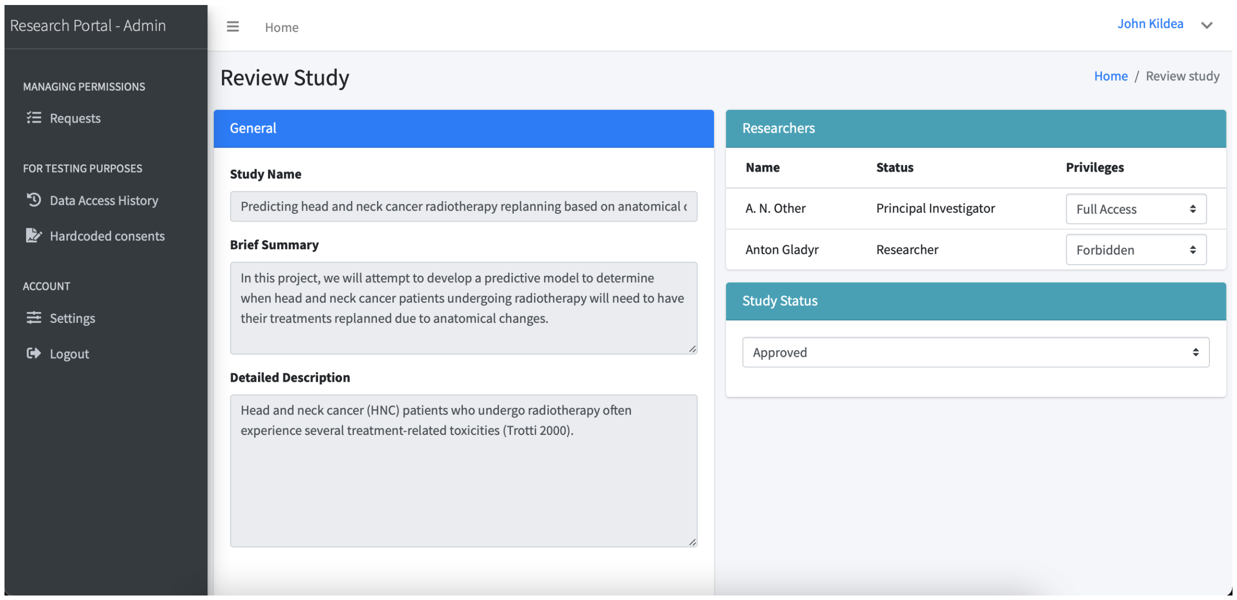
Study Name	Status	Principal Investigators	
An investigation into DNA mutations induced during radiotherapy Updated at 15-11-2021 23:44:10	Rejected	N.O. Name	View
Predicting pain in patients with bone metastasis using radiomics and patient-reported outcomes study Updated at 15-11-2021 23:43:41	Approved	A. Demo	View
Predicting head and neck cancer radiotherapy replanning based on anatomical changes in the patient Updated at 15-11-2021 23:26:21	Review	A. N. Other	View

Showing 1 to 3 of 3 entries

Figure 5.8: Public Trust Committee's "Home" page lists all the research studies submitted by researchers for review. The page allows an authorized administrator to choose a study and open it for review.

5.3 Feasibility Analysis of the Blockchain Solution

Based on our experience developing the Research Portal and the blockchain network, we identified that a production-ready blockchain-based solution would require a team of IT specialists with appropriate expertise. First, the team should include a solution & application architect, who will build the system design and its workflow based on the collected functional requirements. Also, the team should include two blockchain developers and one blockchain network administrator. One blockchain developer will be responsible for developing blockchain smart contracts, whereas the second developer will work on the blockchain client application. The blockchain network administrator will be responsible for (1) deploying the network, (2) deploying the smart contracts, and (3) maintaining the operation of the network. Table 5.10 shows



Review Study

General

Study Name

Predicting head and neck cancer radiotherapy replanning based on anatomical c

Brief Summary

In this project, we will attempt to develop a predictive model to determine when head and neck cancer patients undergoing radiotherapy will need to have their treatments replanned due to anatomical changes.

Detailed Description

Head and neck cancer (HNC) patients who undergo radiotherapy often experience several treatment-related toxicities (Trotti 2000).

Researchers

Name	Status	Privileges
A. N. Other	Principal Investigator	Full Access
Anton Gladyr	Researcher	Forbidden

Study Status

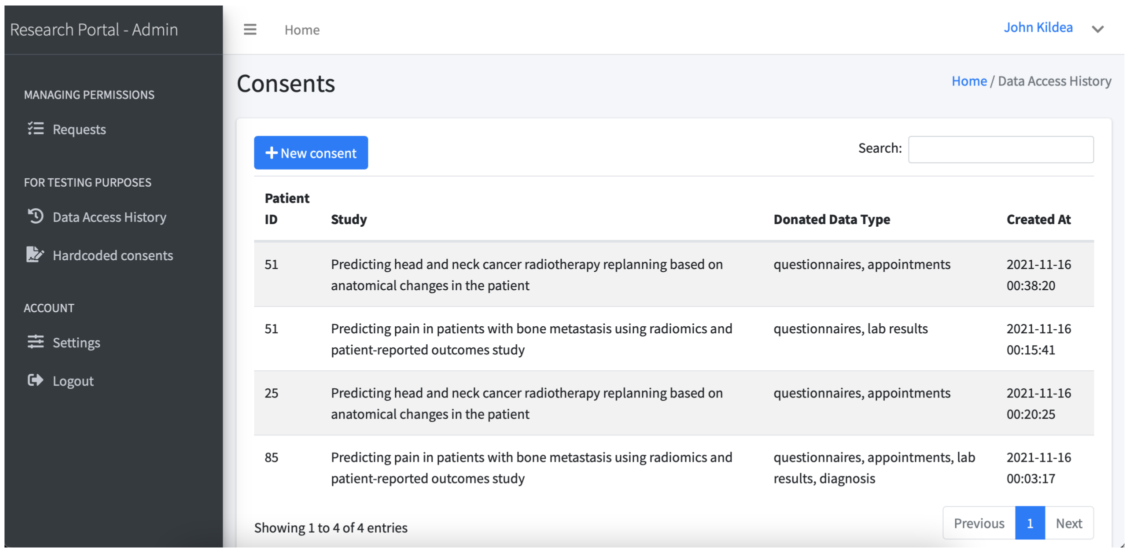
Approved

Figure 5.9: The “Review Study” page allows an authorized Public Trust Committee user to review a newly-proposed study. The page contains information such as study owner (e.g., principle investigator), study name, study summary, and the study’s detailed description. To complete the review process, an administrator has to set the researcher’s privileges (e.g., full access, read only, forbidden) and the study status (e.g., approved, rejected).

a list of IT specialists that we identified. To develop the Research Portal web application, an additional full-stack software developer would be needed.

To support the design and development of an enterprise-grade blockchain network, we estimate that the project owner would need to budget for $78,018 CAD. Also, the owner would need to spend an additional $20,924 CAD ($34.49 CAD per hour) on a four-month work contract for the full-stack developer [135] to complete the Research Portal.

For the operational costs (e.g., maintenance and support of the blockchain network) over a six month period after go-live, the owner would need to budget for $23,932 CAD to support a part-time blockchain network administrator/operator. Thus, in total, for a six months development period and six months of operation, a project’s owner would need to spend $122,873 CAD. This number assumes that the Opal patient portal platform is installed and supported separately. It also does not include overhead costs, hardware costs, software licensing costs,



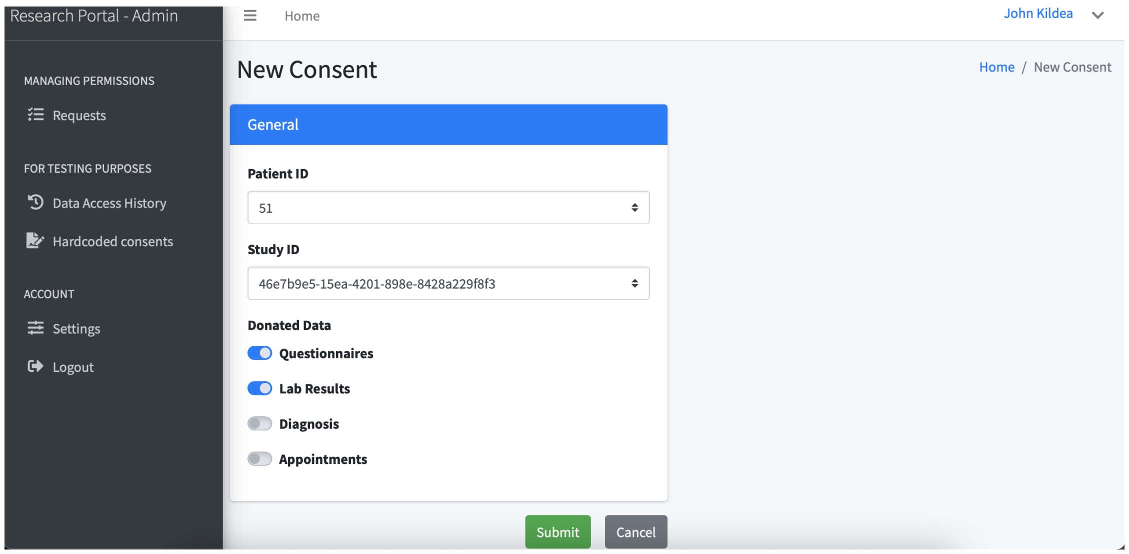
Patient ID	Study	Donated Data Type	Created At
51	Predicting head and neck cancer radiotherapy replanning based on anatomical changes in the patient	questionnaires, appointments	2021-11-16 00:38:20
51	Predicting pain in patients with bone metastasis using radiomics and patient-reported outcomes study	questionnaires, lab results	2021-11-16 00:15:41
25	Predicting head and neck cancer radiotherapy replanning based on anatomical changes in the patient	questionnaires, appointments	2021-11-16 00:20:25
85	Predicting pain in patients with bone metastasis using radiomics and patient-reported outcomes study	questionnaires, appointments, lab results, diagnosis	2021-11-16 00:03:17

Showing 1 to 4 of 4 entries

Figure 5.10: The "Consents" page lists all the patients' e-consents stored on the blockchain. The page is available only to the authorized Public Trust Committee users, who can see existing consents and create new ones on behalf of fake patients. This functionality was implemented only for testing purposes and will be eliminated in future versions of the Research Portal.

management costs, and employee benefits. All included, a budget of $200,000 CAD could be justified for a one-year project.

This feasibility analysis was done for one institution, assuming that the institution implements a production-ready product (i.e., Research Portal and blockchain network) and shares the implemented solution with its collaborating institutions. Thus, the expenses will need to be spent only once. However, collaborating institutions will need to cover their own operational costs (e.g., blockchain network administrator/operator) and the cost of deploying Opal if that is the patient portal technology they wish to use.



New Consent

Patient ID

51

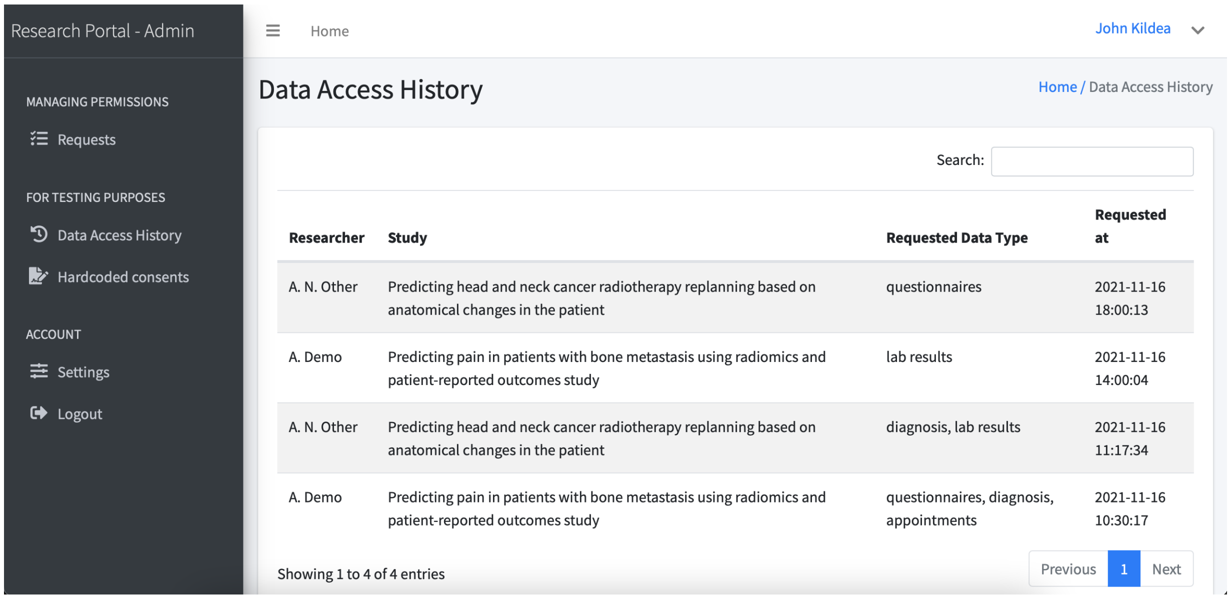
Study ID

46e7b9e5-15ea-4201-898e-8428a229f8f3

Donated Data

- Questionnaires ✓
- Lab Results ✓
- Diagnosis
- Appointments

Figure 5.11: The "New consent" page allows Public Trust Committee users to create a new e-consent on behalf of a fake patient. An e-consent includes the patient's ID (Opal user's ID who shares the data), the recipient's ID (a study-user to which the data are being shared), and toggles that allow the patient to specify which data they are sharing. Created e-consents are stored on the blockchain and can be updated using this page. The page was created only for testing purposes and will be eliminated in future versions of the Research Portal.



Data Access History

Home / Data Access History

Search:

Researcher	Study	Requested Data Type	Requested at
A. N. Other	Predicting head and neck cancer radiotherapy replanning based on anatomical changes in the patient	questionnaires	2021-11-16 18:00:13
A. Demo	Predicting pain in patients with bone metastasis using radiomics and patient-reported outcomes study	lab results	2021-11-16 14:00:04
A. N. Other	Predicting head and neck cancer radiotherapy replanning based on anatomical changes in the patient	diagnosis, lab results	2021-11-16 11:17:34
A. Demo	Predicting pain in patients with bone metastasis using radiomics and patient-reported outcomes study	questionnaires, diagnosis, appointments	2021-11-16 10:30:17

Showing 1 to 4 of 4 entries

Previous 1 Next

Figure 5.12: The “Data Access History” page lists all the log records of the shared data requests stored on the blockchain. The page is available only for the authorized Public Trust Committee users. Each log record includes the researcher’s ID (requester), user type, requested data category, and when the data were requested. The page was created only for testing purposes and will be eliminated in future versions of the Research Portal.

Table 5.10: List of specialists that would need to be hired to implement a blockchain-based data-sharing solution as per the design described in this thesis. The list does not include other developers who will work on the components not related to the blockchain development.

Job Title	**Description**	**Knowledge required**	**Estimated time**	**Salary**
Solution & application architect	Responsible for formulating the functional requirements. Designs the blockchain network and Research Portal.	1. Blockchain concepts 2. Hyperledger Fabric	3.5 months (490 hours)	$61.54 CAD per hour [136] as of November, 2021. Total: $30,154.6 CAD
Smart contract (chaincode) developer	Responsible for developing chaincodes based on the functional requirements and system design. Collaborates with the blockchain client application developer to agree on the contract functions and their parameters (i.e., interfaces). Also collaborates with the network administrator to deploy the developed contracts.	1. Blockchain concepts 2. Hyperledger Fabric 3. Chaincodes development 4. One of the following languages: Node.js, Java, Go 5. Fabric Contract APIs	2 months (280 hours)	$56.98 CAD per hour [137] as of November, 2021. Total: $15,954.4 CAD
Blockchain client application developer	Responsible for developing a blockchain client application that will invoke installed chaincodes utilizing an SDK library. Also, develops an API that will allow external applications to call the chaincodes. Collaborates with the smart contract developer and network administrator.	1. Blockchain concepts 2. Hyperledger Fabric 3. One of the following languages: Node.js, Java, Go 4. Hyperledger Fabric SDK library 5. RESTful API (e.g., Express.js, Spring)	2 months (280 hours)	$56.98 CAD per hour [137] as of November, 2021. Total: $15,954.4 CAD
Blockchain network administrator/operator	Responsible for deploying and configuring the blockchain network based on the system's design and formed consortium. Enrolls and removes network participants, and manages their access rights. Also, deploys the chaincodes and organizes the network's security.	1. Blockchain concepts 2. Hyperledger Fabric 3. Hyperledger Fabirc network configuration, deployment, and administration 4. Installing chaincodes 5. Docker	2 months (280 hours) plus part-time network support	$56.98 CAD per hour [137] as of November, 2021. Total: $15,954.4 CAD
		Totals	6 months	$78,017.8 CAD

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis project, we designed and developed a prototype of a secure and patient-controlled system that allows patients to share their health and healthcare data for research purposes.

First, we reviewed modern research methods and frameworks such as real-world data, real-world evidence, and comparative effectiveness research that have become more popular among health researchers. As we saw, these methods and frameworks require researchers to collect as much data as possible in order to strengthen their findings. Also, we explored the data science and big data fields as well as their application in healthcare research. Moreover, we investigated the technical and legal challenges that often arise during the data collection phase of health research.

Then, we reviewed the Opal app's infrastructure and its various features. Here we studied the app's workflow and how the communication between the patient-facing app and its back end is organized. Also, we explored the high-level architecture of the multi-institutional version of Opal ("All-in-One") as well as its back end components. Furthermore, we examined data sharing and data access control concepts and their application in real-world systems. Since this work was a part of a larger research project known as PARTAGE, we also presented the PARTAGE team's goals, its working groups, and its participatory co-design approach.

As the goal of this project was to design and develop a data-sharing system where the data owners (patients) may control access to their data for research, we analyzed modern decentralized technologies and their possibilities for storing data in an immutable, tamper-proof way. We analyzed the World Wide Web technology—a pioneer of decentralized systems—its evolution and the drawbacks it presents. Also, we explored the Solid specification, the Inrupt Solid platform, and the possibility of using the technology with Opal. Then we studied blockchain technology, its design and functioning principles. We provided an overview of different blockchain types, consensus protocols, and modern blockchain frameworks and platforms. As we decided to design and implement our data-sharing system based on the Hyperledger Fabric platform, we investigated the platform's components, such as organizations, consortia, peers, channels, chaincodes, ordering service, and the Hyperledger Fabric SDK.

Based on our analysis and on the PARTAGE co-design process, we identified the desired solution, namely we (1) designed a high-level architecture of the data-sharing system, (2) determined that certain data need to be stored on the blockchain, and (3) distinguished the actors of the system. Also, we designed the system's flowchart and defined its functional requirements in the form of use cases. Moreover, we visualized the use cases in the form of use a case diagram and a sequence diagram. These diagrams help describe the interactions between the various actors and the system, as well as the object interactions in a time sequence. Furthermore, we identified the technology stack needed for implementation of our solution.

As a result, we designed and built a prototype system for data sharing called the Research Portal. The system was designed for providing researchers with access to patients' data originating in the Opal patient portal platform. However, the developed solution can be extended for use by other platforms and services as well. In order to store (1) patients' e-consents, (2) researchers' data-access privileges, and (3) researchers' data-access logs in an immutable, tamper-proof manner, we designed a Hyperledger Fabric blockchain network. Also, we developed appropriate Hyperledger Fabric chaincodes and a blockchain client application for interaction with the blockchain network. The implemented Research Portal and blockchain client were deployed to a virtual server inside the RI-MUHC's network, and the blockchain network was simulated using Docker on the same server.

6.2 Future Work

We envision the following future work:

Study-user accounts management: Currently, the Research Portal utilizes test shared data that were hardcoded into the "Shared data" database. In a production-ready implementation, the data should be pulled to the "Shared data" databases using the study-user accounts. Thus, new study-user types (accounts) must be implemented and integrated to Opal code base (or equivalently in any other patient portal infrastructure that may wish to employ our technology). The implementation would involve modifications to the OpalDB database and the Listener software. Also, functional requirements in the form of use cases need to be designed for the following operations: (1) creating a new study-user account, (2) generating a study-user's credentials, (3) transferring study-user credentials to a Harvester service.

Harvester implementation: The process of transferring shared data from study-user accounts to the "Shared data" databases should be executed by a Harvester service. Also, this service should remove previously-shared records from the "Shared data" databases that are no longer consented to (e.g., consents were revoked or expired). The implementation will require (1) modifications of the Listener, and (2) an additional client blockchain application to facilitate the Harvester's communication with the blockchain network.

Supoort for e-consents: Based on the designed interfaces for a Research Menu in Opal, an update to Opal's code base and its back end is needed to allow patient-controlled saving of signed e-consents on the blockchain. This includes modifications to the Listener and installing the client blockchain APIs that invoke appropriate blockchain chaincodes.

Blockchain network implementation: Define which organizations (e.g., hospitals, research institutes, non-profit organizations, etc.) will form the blockchain consotrium. Based on the network participants and their nodes, an actual blockchain network should be deployed. The network administrators (i.e., the participants who will control the network) should be defined as well.

Encryption of data at rest: To maximize the security of the system, patients' data should be stored in an encrypted format. Thus, an encryption mechanism should be developed that would encrypt existing data at rest in each hospitals' databases and in the "Shared data" storage of the Research Portal. It should only be possible to decrypt the data on the receiver side (e.g., patient, researcher) by applying a decryption key. Such an approach will add an extra layer of security (i.e., data-centric security) in addition to the existing networks, servers and application security. The encryption will prevent unauthorized data access in cases when attackers (1) gain physical access to the data storage (e.g., access to the data center), and (2) gain read access to the raw database.

Opal Research Menu improvements: Since some patients might not want to share their medical data due to reasons that are not related to privacy concerns (e.g., a patient does not have the incentive to participate in a study), possible incentive solutions should be explored in depth (e.g., gamification strategy to reward data donors in order to encourage them to participate). The ethico-legal aspects should be investigated as well. Another possible improvement of the Research Menu is a recommender system. If a patient decides to choose manually the data records that can be shared, there may be too much information that needs to be filtered. Using artificial intelligence technologies, a recommender system could help patients to filter the data automatically. Also, the recommender system could utilize the patient's sharing preferences, and ensure that the patient does not share any personal information that they do not wish to share.

Besides the suggestions made above, a more in-depth analysis of the Solid specification and the Inrupt Solid framework should be carried out. Namely, the specification should be studied in more detail, existing implementations should be explored, and a Solid-based prototype of the Opal app should be developed for demonstration purposes with the minimum viable functionalities.

Bibliography

[1] Stephanie Green et al. *Clinical Trials in Oncology*. Second edition. Chapman and Hall/CRC, 2002. DOI: https://doi.org/10.1201/9781420035308.

[2] Shayne Cox Gad. *Clinical Trials Handbook*. WILEY, 2009. ISBN: 978-0-471-21388-8.

[3] Marcus Woo. "An AI boost for clinical trials". In: *Nature* 573.7775 (2019), pp. 100–102. DOI: 10.1038/d41586-019-02871-. URL: https://ideas.repec.org/a/nat/nature/v573y2019i7775d10.1038_d41586-019-02871-3.html.

[4] MIT Technology Review Insights. *Clinical trials are better, faster, cheaper with big data.* 2021. URL: https://web.archive.org/web/20210611134751/https://wp.technologyreview.com/wp-content/uploads/2021/06/MITTR_Medidata.v4.pdf.

[5] Rob Sanson-Fisher et al. "Limitations of the Randomized Controlled Trial in Evaluating Population-Based Health Interventions". In: *American journal of preventive medicine* 33 (Sept. 2007), pp. 155–61. DOI: 10.1016/j.amepre.2007.04.007.

[6] Oded Yitschaky, Michael Yitschaky, and Yehuda Zadik. "Case report on trial: Do you, Doctor, swear to tell the truth, the whole truth and nothing but the truth?" In: *Journal of medical case reports* 5 (May 2011), p. 179. DOI: 10.1186/1752-1947-5-179.

[7] The United States Food and Drug Administration (FDA). *Framework for FDA's Real-World Evidence Program.* 2018. URL: https://web.archive.org/web/20210803193040/https://www.fda.gov/media/120060/download.

[8] Institute of Medicine (IOM). *Initial national priorities for comparative effectiveness research.* Nov. 2009, pp. 1–227. DOI: 10.17226/12648.

[9] K. Armstrong. “Methods in comparative effectiveness research.” In: *Journal of clinical oncology : official journal of the American Society of Clinical Oncology* 30 34 (2012), pp. 4208–14.

[10] David Donoho. “50 Years of Data Science”. In: *Journal of Computational and Graphical Statistics* 26 (Oct. 2017), pp. 745–766. DOI: 10.1080/10618600.2017.1384734.

[11] Longbing Cao. “Data Science: A Comprehensive Overview”. In: *ACM Comput. Surv.* 50.3 (June 2017). ISSN: 0360-0300. DOI: 10.1145/3076253. URL: https://doi.org/10.1145/3076253.

[12] Forbes (Gil Press). *A Very Short History Of Data Science.* 2013. URL: https://www.forbes.com/sites/gilpress/2013/05/28/a-very-short-history-of-data-science.

[13] William Cleveland. “Data Science: An Action Plan for Expanding the Technical Areas of the Field of Statistics”. In: *International Statistical Review / Revue Internationale de Statistique* 69 (Mar. 2001). DOI: 10.1111/j.1751-5823.2001.tb00477.x.

[14] Forbes (Peter Pham). *The Impacts Of Big Data That You May Not Have Heard Of.* 2015. URL: https://www.forbes.com/sites/peterpham/2015/08/28/the-impacts-of-big-data-that-you-may-not-have-heard-of.

[15] The Economist. *Data, data everywhere.* 2010. URL: https://web.archive.org/web/20210704102929if_/https://www.economist.com/special-report/2010/02/27/data-data-everywhere.

[16] Wullianallur Raghupathi and Viju Raghupathi. “Big data analytics in healthcare: Promise and potential”. In: *Health Information Science and Systems* 2 (Feb. 2014), p. 3. DOI: 10.1186/2047-2501-2-3.

[17] Marco Viceconti, Peter Hunter, and Rod Hose. “Big Data, Big Knowledge: Big Data for Personalized Healthcare”. In: *IEEE Journal of Biomedical and Health Informatics* 19.4 (2015), pp. 1209–1215. DOI: `10.1109/JBHI.2015.2406883`.

[18] Google. *Google Flu Trends.* 2008. URL: `https://web.archive.org/web/20121022154915/http://www.google.org/flutrends/about/how.html`.

[19] Jeremy Ginsberg et al. “Detecting Influenza Epidemics Using Search Engine Query Data”. In: *Nature* 457 (Dec. 2008), pp. 1012–4. DOI: `10.1038/nature07634`.

[20] David Lazer et al. “The Parable of Google Flu: Traps in Big Data Analysis”. In: *Science (New York, N.Y.)* 343 (Mar. 2014), pp. 1203–5. DOI: `10.1126/science.1248506`.

[21] Declan Butler. “When Google Got Flu Wrong”. In: *Nature* 494 (Feb. 2013), pp. 155–6. DOI: `10.1038/494155a`.

[22] Stumpe Martin and Mermel Craig. *Applying Deep Learning to Metastatic Breast Cancer Detection.* 2018. URL: `https://web.archive.org/web/20210617095257/https://ai.googleblog.com/2018/10/applying-deep-learning-to-metastatic.html`.

[23] Abid Haleem et al. “Significant Applications of Big Data in COVID-19 Pandemic”. In: *Indian journal of orthopaedics* 54 (May 2020), pp. 1–3. DOI: `10.1007/s43465-020-00129-z`.

[24] Katherine Unger Baillie, Michele W. Berger, and Erica K. Brockmeier. *The role of data in a world reshaped by COVID-19.* 2020. URL: `https://web.archive.org/web/20210301035653/https://penntoday.upenn.edu/news/role-data-world-reshaped-covid-19`.

[25] Munther Baara et al. *Blockchain opportunities for patient data donation & clinical research.* 2020. URL: `https://www2.deloitte.com/content/dam/Deloitte/us/Documents/process-and-operations/us-cons-blockchain-opportunities-patient-data-donation-clinical-research.pdf`.

[26] Jennifer Rainey Marquez. *The COVID-19 Data Plan: 3 Innovative Ways Johnson & Johnson Is Using Data Science to Fight the Pandemic.* 2021. URL: https://web.archive.org/web/20210623012429/https://www.jnj.com/innovation/how-johnson-johnson-uses-data-science-to-fight-covid-19-pandemic.

[27] Effy Vayena, Anna Mastroianni, and Jeffrey Kahn. "Ethical Issues in Health Research With Novel Online Sources". In: *American journal of public health* 102 (Oct. 2012). DOI: 10.2105/AJPH.2012.300813.

[28] Effy Vayena et al. "Ethical Challenges of Big Data in Public Health". In: *PLoS computational biology* 11 (Feb. 2015), e1003904. DOI: 10.1371/journal.pcbi.1003904.

[29] Derek Ruths. *What is Data Science?* COMP-598-001 Introduction to Data Science. Class lecture #03. School of Computer Science at McGill University, 2020.

[30] Derek Ruths. *Data collection and annotation.* COMP-598-001 Introduction to Data Science. Class lecture #05. School of Computer Science at McGill University, 2020.

[31] Derek Ruths. *Good Data Science.* COMP-598-001 Introduction to Data Science. Class lecture #09. School of Computer Science at McGill University, 2020.

[32] Health Level Seven International (HL7). *Welcome to FHIR.* 2019. URL: https://hl7.org/FHIR/.

[33] Alex Cameron and Daanish Samadmoten. *Canada - Data Protection Overview.* 2021. URL: https://www.dataguidance.com/notes/canada-data-protection-overview.

[34] Éducaloi. *The Right to Access Medical Records.* 2021. URL: https://web.archive.org/web/20210121135719/https://educaloi.qc.ca/en/capsules/the-right-to-access-medical-records/.

[35] Laurie Hendren. *Patient-controlled data.* 2018. URL: http://web.archive.org/web/20210823044422/http://www.oncologyex.com/pdf/vol17_no1/comment_hendren-patient-controlled.pdf.

[36] John Kildea et al. “Design and Development of a Person-Centered Patient Portal Using Participatory Stakeholder Co-Design”. In: *J Med Internet Res* (2019). DOI: 10.2196/11371. URL: https://www.jmir.org/2019/2/e11371.

[37] Google. *Firebase*. 2021. URL: https://firebase.google.com.

[38] Jihoon Kim et al. “Patient Perspectives About Decisions to Share Medical Data and Biospecimens for Research”. In: *JAMA Network Open* 2 (Aug. 2019), e199550. DOI: 10.1001/jamanetworkopen.2019.9550.

[39] Nora Tophof Maximilian Tischer. *Data donation: better health and quality of life for all.* 2020. URL: https://web.archive.org/web/20211022153459/https://www.data4life.care/en/library/journal/data-donation-in-medicine/.

[40] David Shaw, Juliane Gross, and Thomas Erren. “Data donation after death: A proposal to prevent the waste of medical research data”. In: *EMBO reports* 17 (Dec. 2015). DOI: 10.15252/embr.201541802.

[41] Anya Skatova and James Goulding. “Psychology of personal data donation”. In: *PLOS ONE* 14 (Nov. 2019), e0224240. DOI: 10.1371/journal.pone.0224240.

[42] Matthew Bietz, Kevin Patrick, and Cinnamon Bloss. “Data Donation as a Model for Citizen Science Health Research”. In: *Citizen Science: Theory and Practice* 4 (Mar. 2019). DOI: 10.5334/cstp.178.

[43] PatientsLikeMe. *Research manuscripts bibliography. The complete collection of PatientsLikeMe research publications.* 2019. URL: https://web.archive.org/web/20211007094124/https://patientslikeme-bibliography.s3.amazonaws.com/PLM%5C%20Research%5C%20Manuscripts%5C%20Bibliography.pdf.

[44] Gartner. *Identity and Access Management (IAM)*. 2021. URL: https://web.archive.org/web/20210628052052/https://www.gartner.com/en/information-technology/glossary/identity-and-access-management-iam.

[45] Jason Andress. *The Basics of Information Security: Understanding the Fundamentals of InfoSec in Theory and Practice.* Syngress, 2011, p. 34. ISBN: 9780128008126.

[46] Lazy Ai et al. “Telling Humans and Computers Apart (Automatically)”. In: *Communications of the ACM* 47 (May 2002). DOI: 10.1145/966389.966390.

[47] HL7 Security Technical Committee. *HL7 Role-Based Access Control (RBAC) Role Engineering Process.* 2007. URL: https://web.archive.org/web/20211123163127/https://csrc.nist.gov/csrc/media/projects/role-based-access-control/documents/hl7_role-based_access_control_%5C%28rbac%5C%29.pdf.

[48] Science Applications International Corporation (SAIC). *Role-Based Access Control (RBAC) Role Engineering Process.* 2004. URL: https://web.archive.org/web/20211123163525/https://csrc.nist.gov/csrc/media/projects/role-based-access-control/documents/healthcarerbactfroleengineeringprocessv3_0.pdf.

[49] Subhojeet Mukherjee et al. “Attribute Based Access Control for Healthcare Resources”. In: ABAC ’17. Scottsdale, Arizona, USA: Association for Computing Machinery, 2017, pp. 29–40. ISBN: 9781450349109. DOI: 10.1145/3041048.3041055. URL: https://doi.org/10.1145/3041048.3041055.

[50] Vincent Hu et al. “Guide to attribute based access control (ABAC) definition and considerations”. In: *National Institute of Standards and Technology Special Publication* (Jan. 2014), pp. 162–800.

[51] Lillian Røstad and Ole Edsberg. “A Study of Access Control Requirements for Healthcare Systems Based on Audit Trails from Access Logs”. In: Dec. 2006, pp. 175–186. DOI: 10.1109/ACSAC.2006.8.

[52] Bandar Alhaqbani and Colin Fidge. “Access Control Requirements for Processing Electronic Health Records”. In: vol. 4928. Sept. 2007. ISBN: 978-3-540-78237-7. DOI: 10.1007/978-3-540-78238-4_38.

[53] Karim Abouelmehdi, Abderrahim Beni-Hessane, and Hayat Khaloufi. "Big healthcare data: preserving security and privacy". In: *Journal of Big Data* 5 (Jan. 2018). DOI: 10.1186/s40537-017-0110-7.

[54] Xiao Yue et al. "Healthcare Data Gateways: Found Healthcare Intelligence on Blockchain with Novel Privacy Risk Control". In: *Journal of medical systems* 40 (Aug. 2016), p. 218. DOI: 10.1007/s10916-016-0574-6.

[55] Bingqing Shen, Jingzhi Guo, and Yilong Yang. "MedChain: Efficient Healthcare Data Sharing via Blockchain". In: *Applied Sciences* 9.6 (2019). ISSN: 2076-3417. DOI: 10.3390/app9061207. URL: https://www.mdpi.com/2076-3417/9/6/1207.

[56] Daisuke Ichikawa, Makiko Kashiyama, and Taro Ueno. "Tamper-Resistant Mobile Health Using Blockchain Technology". In: *JMIR mHealth and uHealth* 5 (July 2017), e111. DOI: 10.2196/mhealth.7938.

[57] Kristen Griggs et al. "Healthcare Blockchain System Using Smart Contracts for Secure Automated Remote Patient Monitoring". In: *Journal of Medical Systems* 42 (June 2018). DOI: 10.1007/s10916-018-0982-x.

[58] Sudeep Tanwar, Karan Parekh, and Richard Evans. "Blockchain-based electronic healthcare record system for healthcare 4.0 applications". In: *Journal of Information Security and Applications* 50 (Feb. 2020), p. 102407. DOI: 10.1016/j.jisa.2019.102407.

[59] Tim Berners-Lee. "Information Management: A Proposal". In: *CERN* (1989). URL: https://web.archive.org/web/20210729152622/http://www.w3.org/History/1989/proposal.html.

[60] Darcy DiNucci. "Fragmented future". In: *Print* 53.4 (1999), p. 32.

[61] Tim O'Reilly. "What Is Web 2.0: Design Patterns And Business Models For The Next Generation Of Software". In: *University Library of Munich, Germany, MPRA Paper* 65 (Jan. 2007).

[62] The New York Times. *Cambridge Analytica and Facebook: The Scandal and the Fallout So Far*. 2018. URL: https://web.archive.org/web/20210731050044/https://www.nytimes.com/2018/04/04/us/politics/cambridge-analytica-scandal-fallout.html.

[63] Jim Isaak and Mina J. Hanna. "User Data Privacy: Facebook, Cambridge Analytica, and Privacy Protection". In: *Computer* 51.8 (2018), pp. 56–59. DOI: 10.1109/MC.2018.3191268.

[64] Financial Times. *NHS shares English hospital data with dozens of companies*. 2021. URL: https://www.ft.com/content/6f9f6f1f-e2d1-4646-b5ec-7d704e45149e.

[65] Tim Berners-Lee. "Three challenges for the Web, according to its inventor". In: *Web Foundation* (2017). URL: https://web.archive.org/web/20210712155512/https://webfoundation.org/2017/03/web-turns-28-letter/.

[66] Amazon. *What is Decentralization in Blockchain?* 2021. URL: http://web.archive.org/web/20210811223748/https://aws.amazon.com/blockchain/decentralization-in-blockchain/.

[67] Tim Berners-Lee. *Socially Aware Cloud Storage*. 2009. URL: https://web.archive.org/web/20210801165553/https://www.w3.org/DesignIssues/CloudStorage.html.

[68] Solid. *About Solid*. 2021. URL: https://web.archive.org/web/20210729153859/https://solidproject.org/about.

[69] Solid. *Solid Technical Reports*. 2021. URL: https://web.archive.org/web/20210509085403/https://solid.github.io/specification/.

[70] Solid. *What is Solid?* 2017. URL: https://web.archive.org/web/20210801165454/https://solid.mit.edu/.

[71] Inrupt. *About Inrupt*. 2021. URL: https://web.archive.org/web/20210704031810if_/https://inrupt.com/about/.

[72] Sarven Capadisli et al. *Solid Protocol*. 2021. URL: https://web.archive.org/web/20210704033732/https://solidproject.org/TR/protocol.

[73] Andrei Sambra, Henry Story, and Tim Berners-Lee. *WebID 1.0*. 2014. URL: https://web.archive.org/web/20201111014624/https://dvcs.w3.org/hg/WebID/raw-file/tip/spec/identity-respec.html.

[74] Michael Fitzgerald. "XML Hacks". In: O'Reilly Media, Inc., 2004. Chap. 4. XML Vocabularies. ISBN: 9780596007119. DOI: https://web.archive.org/web/20210831233005/https://www.oreilly.com/library/view/xml-hacks/0596007116/ch04s07.html.

[75] World Wide Web Consortium (W3C). RDF Working Group. *Resource Description Framework (RDF)*. 2014. URL: https://web.archive.org/web/20210818022336/https://www.w3.org/RDF/.

[76] Travis J. Osterman, May Terry, and Robert S. Miller. "Improving Cancer Data Interoperability: The Promise of the Minimal Common Oncology Data Elements (mCODE) Initiative". In: *JCO Clinical Cancer Informatics* 4 (2020). PMID: 33136433, pp. 993–1001. DOI: 10.1200/CCI.20.00059. eprint: https://doi.org/10.1200/CCI.20.00059. URL: https://doi.org/10.1200/CCI.20.00059.

[77] David Chaum, Amos Fiat, and Moni Naor. "Untraceable Electronic Cash". In: *Proceedings of the 8th Annual International Cryptology Conference on Advances in Cryptology*. CRYPTO '88. Berlin, Heidelberg: Springer-Verlag, 1988, pp. 319–327. ISBN: 3540971963.

[78] N. Asokan et al. "The state of the art in electronic payment systems". In: *Computer* 30 (Oct. 1997), pp. 28–35. DOI: 10.1109/2.612244.

[79] Jaap-Henk Hoepman. "Distributed Double Spending Prevention". In: *CoRR* abs/0802.0832 (2008). arXiv: 0802.0832. URL: http://arxiv.org/abs/0802.0832.

[80] Ivan Osipkov et al. "Combating Double-Spending Using Cooperative P2P Systems". In: *27th International Conference on Distributed Computing Systems (ICDCS '07)*. 2007, pp. 41–41. DOI: 10.1109/ICDCS.2007.91.

[81] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System". In: *Cryptography Mailing list at https://metzdowd.com* (Mar. 2009).

[82] Ralph C. Merkle. "A Digital Signature Based on a Conventional Encryption Function". In: *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings.* Vol. 293. Lecture Notes in Computer Science. Springer, 1987, pp. 369–378. DOI: 10.1007/3-540-48184-2_32.

[83] Merlinda Andoni et al. "Blockchain technology in the energy sector: A systematic review of challenges and opportunities". In: *Renewable and Sustainable Energy Reviews* 100.C (2019), pp. 143–174. DOI: 10.1016/j.rser.2018.10.01. URL: https://web.archive.org/web/20201109155157/https://www.sciencedirect.com/science/article/pii/S1364032118307184.

[84] Dapper Labs Inc. *CryptoKitties: Collectible and Breedable Cats Empowered by Blockchain Technology.* 2017. URL: https://drive.google.com/file/d/1soo-eAaJHzhw_XhFGMJp3VNcQoM43byS/view.

[85] Naif Alzahrani and Nirupama Bulusu. "Block-Supply Chain: A New Anti-Counterfeiting Supply Chain Using NFC and Blockchain". In: June 2018, pp. 30–35. DOI: 10.1145/3211933.3211939.

[86] Sara Saberi et al. "Blockchain technology and its relationships to sustainable supply chain management". In: *International Journal of Production Research* 57 (2019), pp. 2117–2135.

[87] Tsung-Ting Kuo, Hugo Zavaleta Rojas, and L. Ohno-Machado. "Comparison of blockchain platforms: a systematic review and healthcare examples". In: *Journal of the American Medical Informatics Association : JAMIA* 26 (2019), pp. 462–478.

[88] Thomas McGhin et al. "Blockchain in healthcare applications: Research challenges and opportunities". In: *Journal of Network and Computer Applications* 135 (Feb. 2019). DOI: 10.1016/j.jnca.2019.02.027.

[89] William Stallings. *Cryptography and Network Security: Principles and Practice.* Seventh edition. Pearson Education Limited, 2017, pp. 283–308. ISBN: 9780134444284.

[90] Archana Joshi, Meng Han, and Yan Wang. “A survey on security and privacy issues of blockchain technology”. In: *Mathematical Foundations of Computing* 1 (Jan. 2018), pp. 121–147. DOI: 10.3934/mfc.2018007.

[91] Michael Barborak, Anton Dahbura, and Miroslaw Malek. “The Consensus Problem in Fault-Tolerant Computing”. In: *ACM Comput. Surv.* 25.2 (June 1993), pp. 171–220. ISSN: 0360-0300. DOI: 10.1145/152610.152612. URL: https://doi.org/10.1145/152610.152612.

[92] Jean-Jacques Quisquater et al. “How to Explain Zero-Knowledge Protocols to Your Children”. In: Aug. 1989, pp. 628–631. ISBN: 978-0-387-97317-3. DOI: 10.1007/0-387-34805-0_60.

[93] Miguel Castro and Barbara Liskov. “Practical Byzantine Fault Tolerance and Proactive Recovery”. In: *ACM Trans. Comput. Syst.* 20.4 (Nov. 2002), pp. 398–461. ISSN: 0734-2071. DOI: 10.1145/571637.571640. URL: https://doi.org/10.1145/571637.571640.

[94] Leslie Lamport, Robert Shostak, and Marshall Pease. “The Byzantine Generals Problem”. In: *ACM Trans. Program. Lang. Syst.* 4.3 (July 1982), pp. 382–401. ISSN: 0164-0925. DOI: 10.1145/357172.357176. URL: https://doi.org/10.1145/357172.357176.

[95] Miguel Castro and Barbara Liskov. “Practical Byzantine Fault Tolerance”. In: *Proceedings of the Third Symposium on Operating Systems Design and Implementation.* OSDI ’99. New Orleans, Louisiana, USA: USENIX Association, 1999, pp. 173–186. ISBN: 1880446391.

[96] Diego Ongaro and John Ousterhout. “In Search of an Understandable Consensus Algorithm”. In: *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference.* USENIX ATC’14. Philadelphia, PA: USENIX Association, 2014, pp. 305–320. ISBN: 9781931971102.

[97] Raft. *The Raft Consensus Algorithm.* 2021. URL: https://raft.github.io.

[98] BitInfoCharts. *Bitcoin Transactions historical chart.* 2021. URL: https://web.archive.org/web/20210920210900/https://bitinfocharts.com/comparison/bitcoin-transactions.html#3y.

[99] BitInfoCharts. *Ethereum Transactions historical chart.* 2021. URL: https://web.archive.org/web/20210921155431/https://bitinfocharts.com/comparison/ethereum-transactions.html#3y.

[100] Corda. *Transactions Per Second (TPS).* 2018. URL: https://web.archive.org/web/20210922172724/https://www.corda.net/blog/transactions-per-second-tps/.

[101] Ripple. *XRP: The Best Digital Asset for Global Payments.* 2021. URL: https://web.archive.org/web/20210905053503/https://ripple.com/xrp/.

[102] Kaleido Knowledge Center. *Quorum.* 2021. URL: https://docs.kaleido.io/kaleido-platform/protocol/ethereum/quorum/.

[103] Marco Mazzoni, Antonio Corradi, and Vincenzo Di Nicola. "Performance evaluation of permissioned blockchains for financial applications: The ConsenSys Quorum case study". In: *Blockchain: Research and Applications* (2021), p. 100026. ISSN: 2096-7209. DOI: https://doi.org/10.1016/j.bcra.2021.100026. URL: https://www.sciencedirect.com/science/article/pii/S209672092100021X.

[104] 101 Blockchains. *Quorum Blockchain Ultimate Guide.* 2019. URL: http://web.archive.org/web/20210922193822/https://101blockchains.com/quorum-blockchain-tutorial/.

[105] EOSIO. *Consensus Protocol.* 2021. URL: https://web.archive.org/web/20210922195914/https://developers.eos.io/welcome/latest/protocol/consensus_protocol.

[106] Daniel Larimer. *EOSIO Dawn 3.0 Now Available.* 2018. URL: https://web.archive.org/web/20210923065817/https://medium.com/eosio/eosio-dawn-3-0-now-available-49a3b99242d7.

[107] Stellar.org. *Stellar.org Dashboard.* 2021. URL: https://dashboard.stellar.org.

[108] CRYPTOEQ. *CORE Report: Stellar.* 2021. URL: https://www.cryptoeq.io/corereports/stellar-abridged.

[109] Kyle McCollom. *How Many Transactions Per Second Can Stellar Process?* 2018. URL: https://web.archive.org/web/20201109034749/https://www.lumenauts.com/blog/how-many-transactions-per-second-can-stellar-process.

[110] Hyperledger Sawtooth. *Appendix: Sawtooth Settings.* 2018. URL: https://web.archive.org/web/20210923070941/https://sawtooth.hyperledger.org/faq/settings/.

[111] Benjamin Ampel, Mark Patton, and Hsinchun Chen. "Performance Modeling of Hyperledger Sawtooth Blockchain". In: *2019 IEEE International Conference on Intelligence and Security Informatics (ISI).* 2019, pp. 59–61. DOI: 10.1109/ISI.2019.8823238.

[112] Hyperledger Fabric. *Updating a channel configuration.* 2021. URL: https://web.archive.org/web/20210923010442/https://hyperledger-fabric.readthedocs.io/en/release-2.2/config_update.html.

[113] Christian Gorenflo et al. *FastFabric: Scaling Hyperledger Fabric to 20,000 Transactions per Second.* 2019. arXiv: 1901.00910 [cs.DC].

[114] Elli Androulaki et al. "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains". In: *Proceedings of the Thirteenth EuroSys Conference.* EuroSys '18. Porto, Portugal: Association for Computing Machinery, 2018. ISBN: 9781450355841. DOI: 10.1145/3190508.3190538. URL: https://doi.org/10.1145/3190508.3190538.

[115] Hyperledger. *A Blockchain Platform for the Enterprise*. 2020. URL: https://web.archive.org/web/20210710225436/https://hyperledger-fabric.readthedocs.io/en/release-2.2/index.html.

[116] Ethereum. *Solidity*. 2021. URL: https://web.archive.org/web/20210728222151/https://docs.soliditylang.org/en/latest/.

[117] Nadav Ivgi. *A Miniscript-based scripting language for Bitcoin contracts*. 2020. URL: https://web.archive.org/web/20210428164154/https://min.sc/.

[118] forkdrop.io. *How Many Bitcoin Forks Are There?* 2019. URL: https://web.archive.org/web/20210601220754/https://forkdrop.io/how-many-bitcoin-forks-are-there.

[119] Hyperledger. *Transaction Flow*. 2020. URL: https://web.archive.org/web/20210517094701/https://hyperledger-fabric.readthedocs.io/en/release-2.2/txflow.html.

[120] Hyperledger. *Module: fabric-network*. 2020. URL: https://web.archive.org/web/20201028103758/https://hyperledger.github.io/fabric-sdk-node/release-2.2/module-fabric-network.html.

[121] Centers for Medicare & Medicaid Services (CMS) Office of Information Service. Department of Health and Human Services (HHS). *Selecting a development approach*. 2005. URL: https://www.academia.edu/13239574/SELECTING_A_DEVELOPMENT_APPROACH.

[122] Iqbal Sarker et al. "A Survey of Software Development Process Models in Software Engineering". In: *International Journal of Software Engineering and its Applications* 9 (Nov. 2015), pp. 55–70. DOI: 10.14257/ijseia.2015.9.11.05.

[123] S. Shylesh. "A Study of Software Development Life Cycle Process Models". In: *Social Science Research Network* (2017).

[124] Gerry Coleman and Rory O'Connor. "An investigation into software development process formation in software start-ups". In: *J. Enterprise Inf. Management* 21 (Oct. 2008), pp. 633–648. DOI: 10.1108/17410390810911221.

[125] Kevin MacG. Adams. *Non-functional Requirements in Systems Analysis and Design.* Springer, 2015, pp. 45–50. ISBN: 9783319183435. DOI: https://doi.org/10.1007/978-3-319-18344-2_3.

[126] Dr. Ivar Jacobson, Ian Spence, and Kurt Bittner. *Use-Case 2.0 ebook.* 2011. DOI: https://web.archive.org/web/20210416030851/https://www.ivarjacobson.com/publications/white-papers/use-case-ebook.

[127] Alistair Cockburn. *Use cases, ten years later.* 2002. URL: https://web.archive.org/web/20080915162727/http://alistair.cockburn.us/index.php/Use_cases%5C%2C_ten_years_later#Structuring_Use_Cases_with_Goals.

[128] Martin Fowler. *UML Distilled.* Third edition. Addison-Wesley, 2004, p. 80.

[129] Google. *Google Trends: Interest over time.* URL: https://www.google.com/trends.

[130] Matt Stauffer. *Laravel: Up and Running. A Framework for Building Modern PHP Apps.* Second edition. O'REILLY, 2019.

[131] Laravel. *Meet Laravel.* 2020. URL: https://web.archive.org/web/20210802013224/https://laravel.com/docs/8.x.

[132] Mark Otto and Jacob Thornton. *Bootstrap.* 2011. URL: https://web.archive.org/web/20210801213632/https://github.com/twbs/bootstrap.

[133] Colorlib. *AdminLTE.* 2014. URL: https://web.archive.org/web/20210407123910/https://github.com/colorlibhq/AdminLTE.

[134] Docker. *Docker overview.* 2021. URL: https://web.archive.org/web/20210724071910/https://docs.docker.com/get-started/overview/.

[135] Glassdoor. *Full Stack Developer*. 2021. URL: https://www.glassdoor.ca/Salaries/full-stack-web-developer-salary-SRCH_KO0,24.htm.

[136] Glassdoor. *Solutions Architect Salaries*. 2021. URL: https://www.glassdoor.ca/Salaries/montreal-solutions-architect-salary-SRCH_IL.0,8_IM990_KO9,28.htm.

[137] Glassdoor. *Blockchain Developer Salaries*. 2021. URL: https://www.glassdoor.ca/Salaries/blockchain-developer-salary-SRCH_KO0,20.htm.

[138] Stuart Haber and W. Scott Stornetta. “How to Time-stamp a Digital Document”. In: *Journal of Cryptology* 3 (1991), pp. 99–111.

[139] Dave Bayer, Stuart Haber, and W. Stornetta. “Improving the Efficiency and Reliability of Digital Time-Stamping”. In: (Sept. 1999). DOI: 10.1007/978-1-4613-9323-8_24.

[140] M. Pease, R. Shostak, and L. Lamport. “Reaching Agreement in the Presence of Faults”. In: *J. ACM* 27.2 (Apr. 1980), pp. 228–234. ISSN: 0004-5411. DOI: 10.1145/322186.322188. URL: https://doi.org/10.1145/322186.322188.