ITMB
COC252
F011321

EHR System Modernization in the Republic of Moldova

by

Calin Corcimaru

Supervisor: Dr. Georgina Cosma

Department of Computer Science
Loughborough University

May 2025

Abstract

Abstract to be added

Acknowledgements

Acknowledgements to be added

List of Figures

Contents

Chapter 1

Introduction

1.1 Background

The Republic of Moldova is a small country in Eastern Europe that borders Romania and Ukraine, with a current population of 2.4 million people (National Bureau of Statistics of the Republic of Moldova, 2024). Since its independence in 1991, Moldova has faced a number of challenges, including political instability, corruption, and economic difficulties which have left Moldova as one of the poorest countries in Europe (BBC, 2024).

Despite these challenges, Moldova has made significant progress in its digital transformation efforts, with the government launching a number of initiatives to modernize its public services and improve the quality of life for its citizens (E-Governance Agency, n.d.[a]). An example is the Citizen's Government Portal (MCabinet), which allows citizens to access personal information such as 'valid identity documents, social contributions and benefits, own properties, information about the family doctor and the health institution where the person is registered, tax payments and other information about the citizen-government relationship' (E-Governance Agency, n.d.[b]).

To continue supporting the existing transformation initiatives, Moldova's Cabinet of Ministers has recently approved the 'Digital Transformation Strategy of the Republic of Moldova for 2023-2030', which aims to transform the country into a digital society by 2030, with the ultimate goal of having 'all public services available in a digitalized format' (United Nations Development Programme, 2023).

1.2 Problem Statement

The healthcare sector in Moldova has also seen some transformations, with the introduction of a new electronic health record system (EHR) in 15 hospitals across the country in 2017, called 'Sistemul informațional automatizat „Asistența Medicală Spitalicească" (SIA AMS)' (Ciurcă, 2021). While the system has been successful in helping doctors access patient information more efficiently such as medical history, examinations, test results, and prescriptions, the system hasn't been updated since its inception in 2017 and there are still challenges that need to be addressed in 2024.

The main challenge with the current system lies in the user experience (UX) – SIA AMS feels old and isn't user-friendly, with a clunky interface that is difficult to navigate, not adhering to modern accesibility standards and only accessible via Internet Explorer or legacy version of Microsoft Edge, with no support for other browsers or devices (Ciurcă, 2021).

Another big challenge with the system is its lack of interoperability within public and private medical institutions due to a lack of a nationally-wide integrated system – each hospital and clinic have their own, siloed, information system that contains the patient information, with no communication being made between systems in different hospitals (Ciurcă, 2021).

Finally, due to the current economic situation in Moldova, the government has not allocated any funds to upgrade the current or develop new systems, and the hospitals and clinics that use the system do not have the resources to update it themselves.

1.3 The Client

The client, "Nicolae Testemiteanu" State University of Medicine and Pharmacy in Moldova (USMF), is a public university in Chisinau, Moldova, that offers a range of medical programs, including medicine, dentistry and pharmacy (USMF, 2023). Many of the faculty at USMF are also practicing doctors at hospitals and clinics across Moldova, and have first-hand experience with the current IT systems used in both public and private medical institutions. The USMF faculty members that the student will be interacting with during the project are part of an innovation team that researches potential opportunities to improve the healthcare sector in Moldova through the use of technology. As such, the client has expressed a need for a prototype that can act as a proof of concept for a modern system that could either replace or augment the current system in Moldova.

1.4 Project Objectives

This project aims to initially conduct some research on the current situation of the IT systems used in the healthcare sector in Moldova by interviewing several stakeholders from various healthcare-related institutions. Afterwards, the project will conduct a literature review on the most appropriate technologies and methodologies for developing a modernized EHR system, and an analysis of existing EHR systems to identify their existing functionality. Finally, based on the information gathered, the project will focus on designing and developing a working prototype, based on the requirements gathered and the feasibility of the chosen solution for the Moldovan healthcare sector. The student's hope is that the solution can then be used as a proof of concept to secure funding for a full-scale implementation of the new system in Moldova by the relevant authorities, such as the Ministry of Health.

As such, the objectives of the project are as follows:

1. Identify 2 to 4 stakeholders from various perspectives, such as healthcare institutions in Moldova and patients, that can provide insights into the current IT systems used in the healthcare sector in Moldova.

2. Conduct interviews with the identified stakeholders to gather information on the current IT systems used in the healthcare sector in Moldova.

3. Carry out a literature review to research the most appropriate technologies (frontend, backend and database) and project management methodologies for developing a modernized EHR system.

4. Explore at least 2 existing EHR systems and identify their strengths and weaknesses.

5. Design and develop a working web or mobile app for an EHR system based on the requirements gathered and the feasibility of the chosen solution for the Moldovan healthcare sector.

6. Offer the client the prototype to be used as a proof of concept to secure funding for a full-scale implementation of the new system in Moldova.

Chapter 2

Research and Requirements

To gain a more complete understanding of the current situation in Moldova, the existing problems and possible needs of the people involved, it is important to start with an analysis to identify the possible key stakeholders for this project. As previously mentioned in the literature review, a diverse group of stakeholders is essential to ensure that the currrent situation is reviewed from multiple perspectives.

Afterwards, the next step is to utilise the chosen stakeholders to gather as much information as possible from various perspective to ensure that the project is aligned with the needs of both patients and healthcare professionals in Moldova and solves the existing problems.

2.1 Stakeholder Analysis

The student has identified the following possible stakeholders for the project:

- Doctors and other medical staff working in hospitals
- Department head in a hospital
- IT staff members in hospitals
- Staff members at CNAM (National Health Insurance Company)
- Staff members at the Ministry of Health
- Patients

These stakeholders have been identified so that they can provide a wider picture on the needs and requirements of the project, and to ensure that the project is aligned with the expectations of workers within the healthcare industry in Moldova from multiple perspectives.

The stakeholders have also been placed in the stakeholder influence-interest grid (which will be discussed in section 3.2.2) to help the student understand the level of influence and interest that each stakeholder has in the project:

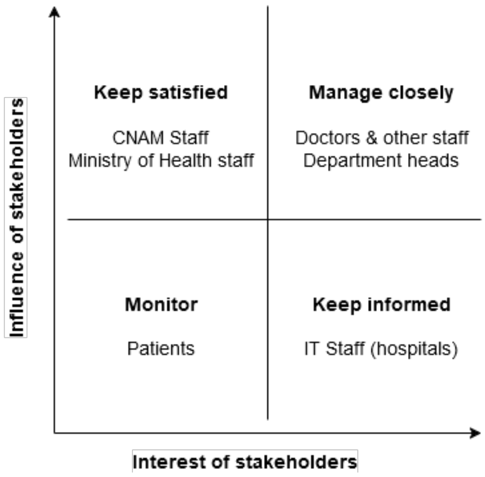


Figure 2.1: Chosen stakeholders in the stakeholder influence-interest grid

2.1.1 Current situation analysis

Following the analysis, the student has conducted several exploratory interviews with the chosen stakeholders to gather insights into current issues with the IT systems used in Moldova's healthcare sector. The student was able to reach out to every stakeholder, except for staff members at CNAM and the Ministry of Health.

After the conclusion of the interviews, three main issues and potential solutions have been identified:

1. Current EHR system is outdated, not user friendly and only accessible via Internet Explorer or legacy version of Microsoft Edge. A potential solution is to develop a new, modernized version of the existing system (and retaining the core functionality) that is accessible via modern browsers and can be augmented with additional features if necessary.

2. Lack of interoperability between medical institutions due to a lack of a nationally-wide integrated system. A potential solution is to create a new systems where patients can upload their own medical records (such as lab tests, previous medical history, etc) and

share them with any medical practitioner, regardless of the institution they work at.

3. Some systems are not digitalized at all, and still rely on paper-based records or very rudimentary data structures, such as the national transplant registry. A potential solution is a the creation of a digitalized system, as is in the case of the transplant registry, that can be accessed by any medical practitioner in Moldova.

Analysing the current issues and potential solutions, the student has determined that the solutions for issues #1 and #3 are too complex, as they require a complete overhaul and integration with existing systems. As such, the student has decided to focus on issue #2, as it is the most feasible and can be implemented within the timeframe of the project.

Consequently, the stakeholder list has been updated to reflect the changed focus of the project:

- Doctors working in hospitals and clinics
- Staff members at the Ministry of Health
- Staff members at CNAM
- Patients that are using both public and private healthcare institutions
- Other medical staff members (nurses, pharmacists, etc)
- Staff members at CNPDCP (National Center for Personal Data Protection)

At the same time, the stakeholder influence-interest grid has been updated to reflect the changes in the project focus:

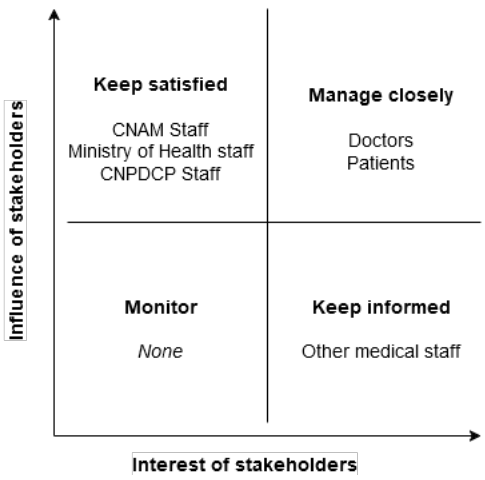


Figure 2.2: Updated stakeholders in the stakeholder influence-interest grid

2.2 Requirements

After the new stakeholders were identified, additional interviews were conducted to focus on the requirements for the chosen solution. The student was unable to reach out to the staff members at CNAM and the Ministry of Health, but was able to gather enough information from the other stakeholders to identify the main requirements for the project.

All of the gathered requirements can be found in the appendix (section A), but the most important requirements have been summarized in the table below:

ID	Category	Requirement
1	Non-functional	The system must be accessible on all modern desktop and mobile-based browsers.
2	Non-functional	The system must store the data in a secure manner, ensuring that only the patient and the doctor can access the data.
3	Document upload	The system must allow patients to upload their own medical records in a variety of formats (PDF, DOC, etc).
4	Personal cabinet	The system must display the patient's history in a chronological order in the form of a timeline.
5	Personal cabinet	The patient personal cabinet must provide an overview of the patient's history through 3 main sections: personal information, lab tests, and doctor consultations.
6	Personal Cabinet	The system must allow patients to add their own personal information, such as name, date of birth, or address.
7	Personal Cabinet	The system must allow patients to add their own allergies and vaccinations.
8	Shareable link	The system must allow the patient to generate a shareable link to provide access to their medical records.
9	Doctor view	When shared with the doctor, the system must allow the doctor to only view the patient's history, not edit it.
10	Patient medication	The system must allow patients to enter their current medication including details such as the name of the drug, dosage, frequency and start/end date.

Table 2.1: Summary of the Most Important Requirements

Chapter 3

Literature Review

This chapter will provide a review of the existing literature, which will be used guide the student in their planning and development efforts of the project.

As such, it will be covering the following areas:

- Healthcare-specific research
- Software development methodologies
- Requirement gathering
- Tech stack
- Large Language Models (LLMs)
- PHR Systems

3.1 Software development methodologies

3.1.1 Software Development Life Cycle

The Software Development Life Cycle (SDLC) is a process used to guide the development of software applications or systems (Ruparelia, 2010). The SDLC consists of multiple phases, each with its own set of activities and deliverables. Yas, Alazzawi, and Rahmatullah (2023) outline the phases of the SDLC as following:

1. Requirement gathering and analysis phase - the requirements are gathered from the project stakeholders and saved in a specific document. Based on the requirements

gathered, a development plan is created and a feasibility study is conducted.

2. Design phase - the previously gathered requirements are written in a more technical manner and system deisngs are created that will later guide the developers to develop the software.

3. Implementation phase - Actual development of the software occurs in this phase. Additionally, some smaller unit tests may be done during this phase as parts of the software are developed.

4. Testing phase - may involve multiple types of testing, such as unit testing, integration testing, and system testing. Luo (2001) describes the different types of tests as following:

   - Unit testing - done on the lowest level of the software, testing individual units or components of the software.
   - Integration testing - performed on two or more units combined together, usually focusing on the interfaces between these components.
   - System testing - focuses on the 'end-to-end quality of the entire system', testing it as a whole based on the system requirement specification.

5. Maintenance phase - involves the deployment and maintenance of the software. Additionally, this phase may include user acceptance testing, where the software is handed over to the end-users to ensure that it meets their needs (Luo, 2001).

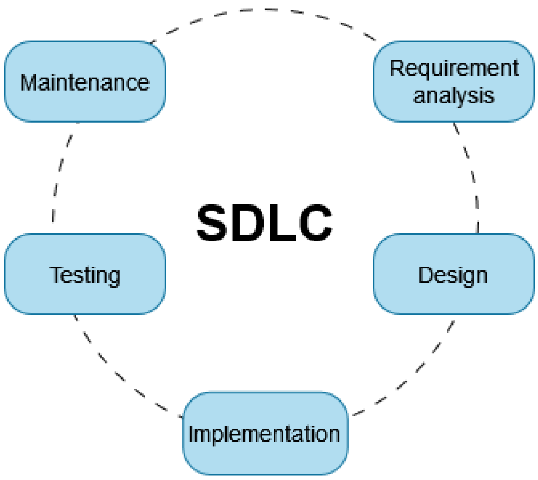


Figure 3.1: Software Development Life Cycle

3.1.2 SDLC Models

The literature describes several SDLC models that have been used in the development of software applications. Ruparelia (2010) and Yas, Alazzawi, and Rahmatullah (2023) outline the most common SDLC models: Waterfall model, V model, Spiral model, Interative model, and Agile model.

Waterfall Model

The Waterfall Model is the most well-known SDLC model. It is a linear model, where the development process is divided into distinct, sequential phases.

The Waterfall Model's strengths lie in its simplicty of use, ease of understanding and a structured approach (Alshamrani and Bahattab, 2015). An additional stregth of the Waterfall model that the authors note is its extensive documentation and planning, which is done in the early stages of a project, that is also maintened throughout the project. These two factors also help minimize the overhead that comes with planning and management of a project.

One of its main weaknesses, mentioned by Alshamrani and Bahattab (2015), is its lack of flexibility in regards to change of requirements. Once the project leaves the requirements analysis or design phase, it may be difficult to make any changes to the project deliverable. Thus, this model is not suitable for projects where the requirements are not well understood or are likely to change. Finally, the deliverable is only available at the end of the project, so the end-users can provide any feedback during its development (Alshamrani and Bahattab, 2015).

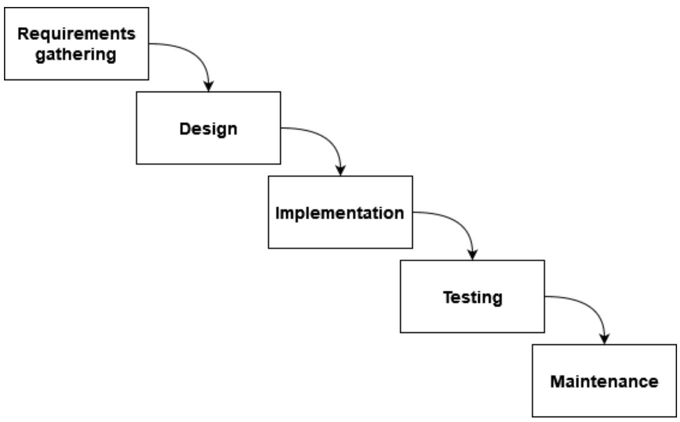


Figure 3.2: Waterfall model

Agile Model

Another well-known model is Agile. It focuses on the idea that requirements are not always well-known and accepting that change is inevitable (Sunner, 2016). Similarly, as the author mentions, the focus of this methodology is on continuous delivery of software and interaction with the customer to get feedback. The author mentions that Agile is well known for its flexibility and adaptability to change, its ability to deliver working software in short iterations and its focus on customer satisfaction.

Agile has multiple frameworks that can be used to guide the development process, such as Scrum or Kanban. Scrum is an Agile framework, where the project is divided into sprints, each lasting between 2-4 weeks and focusing on delivering value to the customer through incremental software features (Alqudah and Razali, 2018; Sunner, 2016). On the other hand, Kanban focuses on visualizing the project workflow through a visual board by using columns, cards and swimlanes. Kanban's main goal is to limit the work in progress through column limits and a pull system to maximize the flow of work through the system (Sunner, 2016).

Agile does have some drawbacks - its lack of documentation and formal planning, especially in the early stages of the project, may not be suitable for large scale projects, where extensive documentation and planning are required (Ruparelia, 2010; Sunner, 2016; Yas, Alazzawi, and Rahmatullah, 2023). Similarly, the Agile model may not be suitable for projects where the requirements are well understood or where there may be strict regulations that guide how the project should be developed.

Nowadays, the use of Scrum and Kanban together is becoming quite popular, with many teams employing both frameworks in their projects. As Alqudah and Razali (2018) mention, this can be quite beneficial as it allows teams to adopt the appropriate practices of both methods and adapt them accordingly based on their needs.

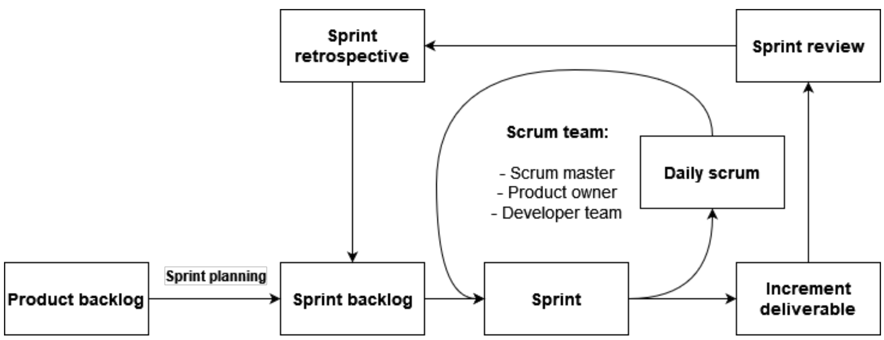


Figure 3.3: Scrum framework

3.1.3 A hybrid approach

For many years, Waterfall has been the most widely used model in software development projects, with Agile approaches gaining popularity in the recent years (Gemino, Reich, and Serrador, 2021). However, a hybrid approach has also been emerging, with various surveys reporting a combined approach being used in project work (Gemino, Reich, and Serrador, 2021; Prenner, Unger-Windeler, and Schneider, 2020). The most common combinations that form this hybrid approach are Scrum, Iterative Development, Kanban, Waterfall and DevOps, with the approach that combines Waterfall and Scrum being the most popular (Prenner, Unger-Windeler, and Schneider, 2020). In that case, only the development part is usually done in an Agile way - with the rest of the project using the traditional approach as a backbone (Prenner, Unger-Windeler, and Schneider, 2020).

Gemino, Reich, and Serrador (2021) note that projects using either Agile, traditional or hybrid approach show similar levels of success in terms of budget, time and quality. However, the authors have found that agile and hybrid approaches perform much better on the customer satisfaction metric than the traditional counterpart.

3.2 Requirements gathering

Regardless of the approaach used, the requirements gathering phase is the first step in any software development process. As described by Young (2002), a requirement is a 'necessary attribute in a system...that identifies a capability, characteristic, or quality factor of a system in order for it to have value and utility to a user'. Multiple studies mention how proper requirement gathering plays a pivotal role in the project quality and success, with a majority of project failures being attributed to poor requirements gathering (Hickey and A.M. Davis, 2003; Sharma and Pandey, 2013; Alan Davis et al., 2006a).

3.2.1 Requirement types

Requirements can be classified into 2 categories: functional and non-functional requirements. Functional requirements describe the system's behavior, such as what the system should do and how it will react to different inputs, while non-functional requirements describe the system's quality attributes, such as performance, security, reliability, etc. (Laplante, 2019, p. 6).

When writing the requirements in a document, it is important to ensure that everything is written in a clear and concise manner to avoid any ambiguity. As such, Laplante (2019, p. 112) recommends using a standard format for writing all requirements: using simple language in a consistent manner, avoiding the use of technical jargon, vague terms and not creating multiple requirements in a single statement.

Similar principles of writing requirements can be found in "ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering" (2018) which also provides a list of requirement characteristics: necessary, appropriate, unambiguous, complete, singular, feasible and verifiable. The same standard recommends using attributes to describe each requirement, such as an identification, owner, priority, risk, rationale, difficulty, and type (functional/non-functional).

Prioritizing requirements is another crucial task, especially in projects with numerous requirements. Some methods include using a low to high priority, assigning a numerical value within a specific range or MoSCoW, which classifies requirements into four categories:

- Must have - must be implemented in the software before being released
- Should have - important but not necessary for the software to be released
- Could have - desirable but not necessary for the software to be released
- Won't have - requirements that are not included in the current release

Requirements in Agile

In Agile projects, the most basic unit of requirements are usually written in the form of User Stories, which are simple descriptions of a feature desired by the customer (Laplante, 2019, p. 191). User Stories are written in a specific format, such as 'As a [user], I want to [action] so that [benefit]'. Other components of a User Story include a title, acceptance criteria (conditions that must be met for the story to be complete), priority, story points (estimated time to implement), and description. Epics are larger user stories that cannot be completed in a single sprint. Epics can be broken down into smaller user stories, which are then added to the Product or Sprint Backlog.

Both Epics and User Stories are part of the Product and Sprint backlog - the former one containing the list of requirements for the whole project and the latter containing the list of requirements for the current sprint/iteration.

3.2.2 Stakeholders

Stakeholders represent the 'set of individuals who have some interest (a stake) in the success (or failure) of the system in question' (Laplante, 2019, p. 34). The author stresses the importance of accurately identifying all possible stakeholders in the early stages of the project, as leaving out key stakeholder could lead to missing out on important requirements or constraints later on the project.

After identifying the stakeholders, it is important to do a stakeholder analysis by understanding the stakeholders' needs, expectations, and influence on the project. One way of doing it is by using a stakeholder matrix, such as the Influence/Interest grid (see figure 3.4), which classifies stakeholders based on their influence and interest in the project (Morphy, 2020; reddi, 2023).

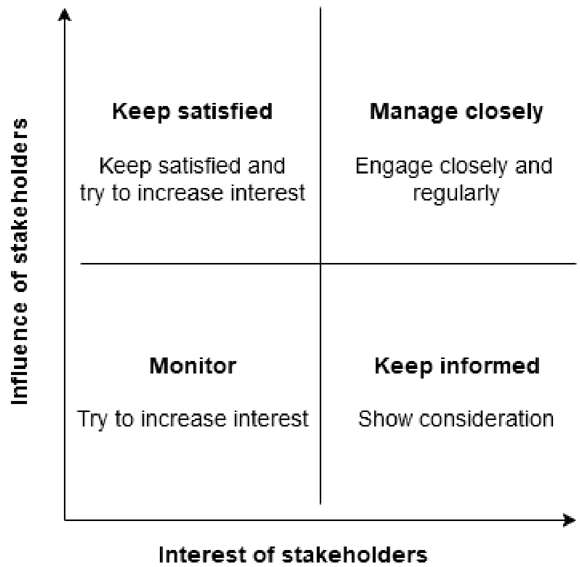


Figure 3.4: Stakeholder Influence/Interest matrix

3.2.3 Requirement gathering techniques

There are multiple requirement gathering techniques, the most popular ones being interviews, workshops, prototyping, modelling, brainstorming, storyboards and observing users (Hickey and A.M. Davis, 2003; Young, 2002; Sharma and Pandey, 2013; Tiwari, Rathore, and Gupta, 2012). One of the studies interviewed several individuals with multiple years of experience in the field of requirement gathering and analysis. As a result, the authors found that the most used requirement gathering techniques were collaborative meetings, interviews, ethnography and modeling (Hickey and A.M. Davis, 2003).

3.2.4 Interview technique considerations

Multiple research papers point to interviews being recognised as the most commonly used technique for requirement gathering (Alan Davis et al., 2006b; Donati et al., 2017; Bano et al., 2019). Additionally, some studies have looked at best practices and common mistakes when conducting interviews for requirement gathering - the results of which are summarized in the tables below.

Study	**Recommended Practices**
Mohedas et al. (2022) and Loweth et al. (2021)	• Encourage deep thinking through situational thinking or rationales. • Avoid ambiguity by asking clarifying questions. • Be flexible by probing into relevant topics. • Verify alignment with the customer's vision. • Use projective techniques like scenarios or role-playing. • Allow stakeholders to teach analysts complex topics. • State goals at the beginning and allow customer input at the end.

Table 3.1: Recommended Practices for Requirements Gathering Interviews

Study	Common Mistakes
Donati et al. (2017)	1. Wrong opening: failing to understand the context before discussing the problem. 2. Not leveraging ambiguity to reveal knowledge gaps. 3. Implicit goals: failing to ask or clarify stakeholder goals. 4. Implicit stakeholders: overlooking third-party stakeholders. 5. Ignoring non-functional requirements. 6. Wrong closing: skipping interview summaries or feedback.
Bano et al. (2019)	1. Poor question formulation: vague, technical, irrelevant, or too long. 2. Question omission: not asking about business processes or doing follow-up questions. 3. Incorrect order: incorrect opening (no introductions, no description of the current situation) or ending (no summary). 4. Weak communication: too much technical jargon usage or not listening to the customer. 5. Poor customer interaction: failing to build rapport. 6. Lack of planning: unstructured sequence of questions.

Table 3.2: Common Mistakes in Requirements Gathering Interviews

3.3 Tech stack

3.3.1 Database

There are several types of databases, such as: relational (SQL), NoSQL databases, graph databases or object-oriented databases (Oracle, 2020). The choice of database depends on the project or organisation requirements, such as the amount of data, the complexity of the data, the need for scalability, etc.

Relational databases

Relational databases store structured data in tables, linked through keys to create relationships between entries (Anderson and Nicholson, 2022). They use SQL (Structured Query Language) to create queries and schemas to help organise and manage data efficiently. Anderson and Nicholson (2022) highlight that relational databases are used, thanks to their high data integrity, for industries like finance and healthcare. Relational databases are widely used and have a large community, making it easier to find support and resources. However, the rigid schema limits adaptability to rapid data changes, and scaling horizontally in distributed systems can be challenging. Examples include MySQL, PostgreSQL, Oracle, and Microsoft SQL Server.

NoSQL databases

NoSQL databases manage unstructured or semi-structured data without rigid schemas or relationships (Anderson and Nicholson, 2022). As the authors describe, NoSQL databases, such as key-value, document, column-family, and graph databases, excel in flexibility and scalability. NoSQL databases often prioritize performance over strict consistency, making them suitable for large, varied datasets of unstructured data but less ideal for complex transactions. Although growing in popularity, NoSQL systems lack SQL's mature standardization and support. Examples include MongoDB, Cassandra, Couchbase, and Redis.

3.3.2 Backend framework

Choosing the right backend framework is crucial for the development of the project deliverable. The backend framework is responsible for handling the business logic of the application, such as processing requests, interacting with the database, and returning responses to the client. A good framework can also come with the added benefit of included features such as security and authentication, database support and a vast user base that can help through documentation or additional tooling.

Based on recent a recent survey by Vailshery (2024), the most popular backend frameworks are Express, Flask, Spring Boot, Django and Laravel. This data is also supported by the Stack Overflow Developer Survey 2024, which lists the most popular programming languages as JavaScript (Express), Python (Flask, Django), Java (Spring Boot) and PHP (Laravel) (Stack Overflow, 2024).

Due to their high popularity among developers, the decision was made to compare the above frameworks, as the student will be able to much easier find support and online resources if necessary. Utilising the information from Broadcom (2024), Laravel Holdings Inc. (2024), OpenJS Foundation (n.d.), and Django (2024), the table below will compare the most popular backend frameworks based on the following criteria: programming language used, learning curve, community support, security features, database features, and project size suitability.

Framework	Django	Laravel	Spring Boot	Express
Language	Python	PHP	Java	JavaScript
Learning curve	Medium	Low	High	Low
Community Support	High	High	High	High
Security features	High	High	High	Medium
Database features	Medium	Medium	High	Medium
Project size suitability	Small to medium	Small to medium	Medium to large	Small to medium

Table 3.3: Comparison of backend frameworks

3.3.3 Frontend framework

Similar to the backend frameworks, choosing a suitable frontend framework is equally important. The frontend framework is responsible for the user interface of the application, such as displaying data, handling user interactions, and making requests to the backend.

Based on the same survey by Vailshery (2024), the most popular frontend frameworks are React, Angular, Vue.js, and Svelte. This data is also supported by the Stack Overflow Developer Survey 2024, where the above frameworks rank among the highest for desirability and admirability among developers (Stack Overflow, 2024).

As such, the table below will utilise the information from Meta Platforms (n.d.), Google (2024b), You (2024), and Svelte (n.d.) to compare the most popular frontend frameworks based on the following criteria: learning curve, community and documentation, ecosystem and tooling support, performance, state management, and project size suitability.

Framework	React	Angular	Vue.js	Svelte
Learning curve	Low	High	Medium	Low
Community and documentation	High	High	High	Medium
Ecosystem and tooling support	High	High	Medium	Low
Performance	High	Medium	Medium	High
State management	High	High	Medium	Low
Project size suitability	Small to large	Medium to large	Small to medium	Small to medium

Table 3.4: Comparison of frontend frameworks

3.4 Large Language Models (LLMs)

Large language models (LLMs) are artificial intelligence systems that are used for natural language processing (NLP) tasks such as human-like text generation, translation, summarization question answering and many more (Naveed et al., 2023; Nazi and Peng, 2024).

Additionally, LLMs have been found to have emergent capabilities, like reasoning, planning, decision-making and in-contenxt learning (Naveed et al., 2023). These extraordinary capabilities are achieved through extensive training on large corpus of text data, high parameter count (in the billions) and usage of techniques such as fine-tuning or prompt engineering to improve their performance (Naveed et al., 2023; Nazi and Peng, 2024).

LLMs are built on the transformer architecture, which allows them to process and understand text data by learning and remembering the relationships between words, sentences, and paragraphs (Vaswani et al., 2017). These models are first pre-trained on large amounts of unlabeled data using self-supervised learning methods, allowing them to excel in a wide variety of tasks (Zhou et al., 2023; Naveed et al., 2023). These pre-trained models, known as foundation models such as the GPT or Llama families, can then be fine-tuned for specific tasks, improving their performance and accuracy even further (Brown, 2020; Dubey et al., 2024; Naveed et al., 2023).

Examples of popular LLMs include GPT-4, Mistral, Llama, Claude and Gemini (OpenAI et al., 2024; Jiang et al., 2023; Dubey et al., 2024; Anthropic, 2024; Team et al., 2024).

3.4.1 Multimodal LLMs

One advancement in the field of LLMs has been the addition of multimodal ability to these models, allowing them to process, understand and generate not only text but also images, audio, and videos (Yin et al., 2024; D. Zhang et al., 2024). These new multimodal LLMs (MLLMs) utilise the existing capabilities of LLMs as a reasoning engine, which is connected to an encoder that processes images, audio or videos and a generator that helps with generating multimodal outputs (Yin et al., 2024). This integration of new modalities allows MLLMs to become much more versatile tools, expanding their possible use cases in a variety of industries and bridging the gap between human and machine communication (Nazi and Peng, 2024).

Example of popular multimodal LLMs are such as GPT-4, Gemini, Claude, and Llama (OpenAI et al., 2024; Team et al., 2024; Dubey et al., 2024; Anthropic, 2024).

3.4.2 LLMs in healthcare

One such application is in healthcare, where the growing volume and complexity of healthcare data creates the need for more advanced tools to process and analyze it. LLMs, with their ability to process and understand large amounts of data, have found use in various healthcare applications, either by using existing models or by developing new, specialized medical models such as Med-PaLm2, BioMistral or Med-Gemini (Labrak et al., 2024; Google, 2024a; Singhal et al., 2023). Some of these applications include:

- **Improving medical diagnosis:** By combining patient records, existing symptoms, and medical history, LLMs can use their deep understanding, reasoning capabilities and memory to assist in diagnosing or preventing health conditions (Nazi and Peng, 2024; Niu et al., 2024; Xiao et al., 2024).
- **Medical Imaging and Multimodal Capabilities:** In diagnostic imaging, multimodal models can assess both text and images (such as X-rays and MRIs) to offer comprehensive analysis. Through interactive chat interfaces, clinicians can input medical images and contextual information to comprehensive diagnostic intepretations, making LLMs valuable assistants in the real-time diagnostic processes (Niu et al., 2024).
- **Virtual Health Assistants:** LLMs can also be deployed as virtual assistants, helping patients with personalised care, symptom tracking, medication reminders, and general health inquiries (Nazi and Peng, 2024; Niu et al., 2024). Patients in areas with limited healthcare access can benefit from virtual assistants, receiving guidance, reminders, and medical information, which also supports healthcare providers by lightening their workloads.
- **EHR Generation and Administrative Support:** LLMs can assist in generating Electronic Health Records (EHRs) by using templates and auto-filling relevant information, allowing healthcare providers to focus more on patient interaction (Xiao et al., 2024). Additionally, they can also help translate complex medical terms into more simple language for better patient understanding, assist in administrative tasks, and more.

3.4.3 Prompt Engineering

The success of LLMs depends not only on the model itself - but also on how it's effectively used by the users, using techniques like prompt engineering, which involves designing and refining prompts to guide the output of LLMs (Meskó, 2023). Prompts represent instructions given to the model to guide its output, such as providing context, examples, or constraints to the model (Giray, 2023). Prompt engineering provides several ways to improve prompt writing: being as specific as possible, providing examples (one/few shot), providing context, using roles, using constraints, and many more. It is an interative process, so it is important to experiment with different prompts to see which one works best for the task at hand.

3.4.4 Challenges and concerns of using LLMs

While LLMs bring a myriad of benefits when applied to the healthcare domain, it is important to note that their use does come with several challenges:

- **Data Privacy and Compliance:** Patient data is highly sensitive and protected under regulations like GDPR and HIPAA. Ensuring compliance with these standards (or other local ones) is essential, requiring data anonymization and secure handling practices to ensure patient data safety (Nazi and Peng, 2024; Harrer, 2023; Xiao et al., 2024).
- **Transparency and Explainability:** LLMs are often described as 'black boxes,' making it difficult to explain their decision-making processes. In healthcare, where model outputs can influence critical clinical decisions, transparency is crucial (Nazi and Peng, 2024; Harrer, 2023; Xiao et al., 2024). Unclear model architecture poses risks and raises ethical concerns about relying on such systems in high-stakes scenarios.
- **Bias and Fairness:** LLMs trained on vast datasets can inherit biases present in the data, leading to skewed or unfair outcomes (Harrer, 2023). In healthcare, where biased recommendations could negatively impact patient outcomes, these biases must be carefully managed to avoid harm or discrimination.
- **Hallucinations:** LLMs sometimes generate information not directly derived from their inputs, which can lead to "hallucinations" or fabricated outputs. In healthcare, this poses significant risks, as incorrect or misleading information could jeopardize patient safety and trust in the technology (Xiao et al., 2024; Nazi and Peng, 2024).
- **Accountability:** Accountability is another big issue, where responsibility must be clearly communicated and understood by all parties involved in the development and use of the model (Harrer, 2023). The author recommends the usage of clear guidelines, policies and code of conducts to ensure that all parties are aware of their responsibilities and obligations.
- **High Costs and Infrastructure Needs:** Training and operating large LLMs requires extensive computational resources, which can be a limiting factor for healthcare institutions (Xiao et al., 2024). Implementing LLMs in healthcare settings may necessitate significant investment in hardware, software, and staff training.

3.5 PHR Systems

A Personal Health Record (PHR) is an electronic resource or tool used by patients to manage and share their own health information, which can be either self-generated or extracted from healthcare institutions (Hosseini et al., 2023; Lee et al., 2021). PHRs are different from Electronic Health Records (EHRs) and Electronic Medical Records (EMRs) which are inter-organisational or internal systems to organise patient health records (Heart, Ben-Assuli, and Shabtai, 2017; Lee et al., 2021). Three different types of PHRs are described by Hosseini et al. (2023): stand-alone, which do not connect with other systems and require manual entry to populate or update the records; instituion-specific, which are connected to a specific healthcare provider or institution; and integrated, which can connect to multiple healthcare providers and systems to aggregate data from multiple sources.

Usage of PHRs can bring many benefits to patients, such as: empowering patients to manage their self-health, improving patient outcomes, decreasing the cost of healthcare and improving the taking of medication (Hosseini et al., 2023).

However, since PHRs contain health information, which is highly sensitive, it is important to ensure that the data is secure and private. In their research, which surveyed the opinion of health information management and medical informatics experts, Hosseini et al. (2023) outline 7 dimensions that need to be addressed when developing a PHR system:

1. Confidentiality
2. Availability
3. Integrity
4. Authentication
5. Authorization
6. Non-repudiation
7. Access rights

The authors go further and recommend mechanisms to ensure adherence to the above-mentioned dimensions, such as encrypting the data in the database, using a backup, defining users and their access to the system data, allowing the option to set or revoke access rights, among others.

3.5.1 Existing Solutions

PHR systems have been implemented nation-wide in many developed countries, such as the NHS App in the UK (Lee et al., 2021). Additionally, there are many privately-owned solutions that offer similar features to the one proposed in this project.

Medvalet

A mobile app developed in Romania that allows patients to upload their medical history as PDFs or scanned documents (Medvalet, 2024). See Table 3.5 for a summary of its features and limitations and figure C.1 for a screenshot of the app.

Key Features/Benefits	**Limitations/Drawbacks**
• Upload medical history as PDFs or scanned documents. • Categorize documents by type (e.g., prescriptions, lab results). • Graphically track vitals like blood pressure and weight over time. • Patients can input personal details such as name, age, and weight. • Doctors can access patient history directly via the app.	• Requires doctors to create accounts, which may deter use. • Doctors can access patient history without explicit consent, raising privacy concerns. • App acts as document storage, which can be cumbersome to access for lengthy histories. • Lacks data extraction or summarization features from uploaded documents. • Only available as a mobile app, limiting accessibility for desktop-only users.

Table 3.5: Medvalet Features and Limitations

Andaman7

A mobile app developed by a Belgian-American eHealth company with the goal to improve doctor-patient communication, compliant with GDPR and HIPAA (A7 Software, 2022). See Table 3.6 for a summary of its features and limitations and figure C.2 for a screenshot of the app.

Key Features/Benefits	**Limitations/Drawbacks**
• Offers sections for personal information, medical history, allergies, vaccinations, medications, etc. • Automatically collects health data from over 300 hospitals and clinics in the US and Europe. • Supports input from diverse sources like hospitals, labs, smart devices or even manual input. • Stores data locally on patients' devices, ensuring privacy. • Data sharing with QR codes and revokable access. • AI tools for summarization, translation, and simplifying medical jargon.	• Requires patients and doctors to both create accounts. • Does not extract data or values from uploaded documents like lab results. • Limited to mobile platforms, which may limit usability for desktop-only users.

Table 3.6: Andaman7 Features and Limitations

Fasten Health

An open-source, self-hosted electronic medical record aggregator with optional paid desktop versions for Windows and Mac (Fasten Health, 2022). See Table 3.7 for a summary of its features and limitations and figure C.3 for a screenshot of the app.

Key Features/Benefits	**Limitations/Drawbacks**
• Automatically aggregates records from multiple providers, like hospitals and labs. • Supports self-hosting for complete control over data, stored locally. • Compatible with protocols such as DICOM, FHIR, and OAuth2. • Allows manual entry for allergies, vaccinations, and medications. • Offers multiple dashboards with graphs to visualize health data. • Supports multi-user functionality for families.	• Paid desktop versions may deter users. • Manual data entry limited to new or existing encounters, complicating usage. • Does not support OCR or automatic data extraction from documents. • Lacks data-sharing capabilities with doctors. • Requires technical expertise for self-hosting. • Restricted to healthcare providers in the United States.

Table 3.7: Fasten Health Features and Limitations

Chapter 4

Project Planning and Design

4.1 Project Management Methodology and Tooling

Based on the research above, the student has decided that he will be using a hybrid approach, with Waterfall as the main methodology for planning managing the project. The development part of the project will be done using ScrumBan, so that the student will be able to utilise elements from both frameworks. There are several reasons for this choice:

1. The nature of the project - the student is working on a project that has a limited timeframe (about 6-7 months) and is of a smaller scale.
2. Documentation requirements - the student is required to document the progress during the project in this report, including the requirements gathered, design considerations and implementation decisions and outcomes.
3. Regulatory requirements - the student is required to adhere to the regulations and standards of the healthcare industry, which may require extensive documentation and planning.
4. Customer involvement - the student will be working closely with the project stakeholder, who will be providing feedback and guidance throughout the project.
5. Familiarity with both Agile and Waterfall - the student has experience with both Agile (specifically Scrum and Kanban) and Waterfall methodologies, and has worked on projects that have used both approaches.

The student will use Jira Software as their project management tool, which is one of the most popular project management tool for software development projects that supports working with Agile frameworks such as Scrum and Kanban (Springer, 2023). The student has experience with Jira Software, having used it in previous projects, and is familiar with its features and capabilities.

4.2 Product Backlog

4.3 System Design

4.3.1 UML Diagrams

UML, or Unified Modeling Language, is a standardized modeling language that consists of a set of diagrams used for modeling business processes and documenting software systems (Paradigm, 2024). UML employs graphical notations to represent a system, helping better communicating potential designs and architectural decisions.

The most common UML diagrams include:

- **Use Case Diagram** - Illustrates the system's intended functionality in terms of actors, use cases, and their relationships, showing how the system delivers value to users.
- **Class Diagram** - Depicts the structure of the system by showing classes, attributes, operations, and static relationships between classes.
- **Sequence Diagram** - Demonstrates how objects interact in a particular, timed sequence scenario, focusing on the messages passed between objects.
- **Activity Diagram** - Represents the workflow of a target use case or business process through a series of activities, emphasizing steps, choices, iterations, and concurrency.

The student has used UML diagrams to present the stakeholders with a visual representation of the system's design and architecture. The diagrams can be found below:

- Use Case Diagram - see Figure B.1
- Class Diagram
- Sequence Diagram
- Activity Diagram

4.3.2 Database Design

4.4 Project Tech Stack

4.4.1 Frontend

4.4.2 Backend

4.4.3 Database

References

A7 Software (2022). *Who are we ?* URL: `https://www.andaman7.com/en/who-are-we`. (Accessed: 11 Nov 2024).

Alqudah, Mashal and Rozilawati Razali (2018). "An empirical study of Scrumban formation based on the selection of scrum and Kanban practices". In: *Int. J. Adv. Sci. Eng. Inf. Technol* 8.6, pp. 2315–2322.

Alshamrani, Adel and Abdullah Bahattab (2015). "A comparison between three SDLC models waterfall model, spiral model, and Incremental/Iterative model". In: *International Journal of Computer Science Issues (IJCSI)* 12.1, p. 106.

Anderson, Benjamin and Brad Nicholson (2022). *SQL vs. NoSQL Databases: What's the difference?* URL: `https://www.ibm.com/think/topics/sql-vs-nosql`. (Accessed: 28 Oct 2024).

Anthropic (2024). *Claude 3.5 Sonnet.* URL: `https://www.anthropic.com/news/claude-3-5-sonnet`. Accessed: 17 Nov 2024.

Bano, Muneera et al. (2019). "Teaching requirements elicitation interviews: an empirical study of learning from mistakes". In: *Requirements Engineering* 24, pp. 259–289.

BBC (2024). *Moldova country profile.* URL: `https://www.bbc.co.uk/news/world-europe-17601580`. (Accessed: 22 Oct 2024).

Broadcom (2024). *Why Spring?* URL: `https://spring.io/why-spring`. (Accessed: 14 Nov 2024).

Brown, Tom B (2020). "Language models are few-shot learners". In: *arXiv preprint arXiv:2005.14165*.

Ciurcă, Aliona (2021). *Sistemul informațional automatizat din spitale: cum funcționează în R. Moldova și ce cred medicii că ar trebui ajustat.* URL: `https://www.zdg.md/stiri/stiri-sociale/sistemul-informational-automatizat-din-spitale-cum-functioneaza-in-r-moldova-si-ce-cred-medicii-ca-ar-trebui-ajustat/`. (Accessed: 22 Oct 2024).

Davis, Alan et al. (2006a). “Effectiveness of Requirements Elicitation Techniques: Empirical Results Derived from a Systematic Review”. In: *14th IEEE International Requirements Engineering Conference (RE’06)*, pp. 179–188. DOI: 10.1109/RE.2006.17.

— (2006b). “Effectiveness of Requirements Elicitation Techniques: Empirical Results Derived from a Systematic Review”. In: *14th IEEE International Requirements Engineering Conference (RE’06)*, pp. 179–188. DOI: 10.1109/RE.2006.17.

Django (2024). *Why Django?* URL: https://www.djangoproject.com/start/overview/. (Accessed: 14 Nov 2024).

Donati, Beatrice et al. (2017). “Common mistakes of student analysts in requirements elicitation interviews”. In: *Requirements Engineering: Foundation for Software Quality: 23rd International Working Conference, REFSQ 2017, Essen, Germany, February 27–March 2, 2017, Proceedings 23*. Springer, pp. 148–164.

Dubey, Abhimanyu et al. (2024). *The Llama 3 Herd of Models*. arXiv: 2407.21783 [cs.AI]. URL: https://arxiv.org/abs/2407.21783.

E-Governance Agency (n.d.[a]). *About EGA*. URL: https://egov.md/en/about-ega. (Accessed: 22 Oct 2024).

— (n.d.[b]). *Citizen’s Government Portal*. URL: https://egov.md/en/content/citizens-government-portal. (Accessed: 22 Oct 2024).

Fasten Health (2022). *What is Fasten?* URL: https://docs.fastenhealth.com/. (Accessed: 12 Nov 2024).

Gemino, Andrew, Blaize Horner Reich, and Pedro M. Serrador (2021). “Agile, Traditional, and Hybrid Approaches to Project Success: Is Hybrid a Poor Second Choice?” In: *Project Management Journal* 52.2, pp. 161–175. DOI: 10.1177/8756972820973082. eprint: https://doi.org/10.1177/8756972820973082. URL: https://doi.org/10.1177/8756972820973082.

Giray, Louie (2023). “Prompt engineering with ChatGPT: a guide for academic writers”. In: *Annals of biomedical engineering* 51.12, pp. 2629–2633.

Google (2024a). *Advancing medical AI with Med-Gemini*. URL: https://research.google/blog/advancing-medical-ai-with-med-gemini/. Accessed: 17 Nov 2024.

— (2024b). *What is Angular?* URL: https://angular.dev/overview. (Accessed: 14 Nov 2024).

Harrer, Stefan (2023). “Attention is not all you need: the complicated case of ethically using large language models in healthcare and medicine”. In: *eBioMedicine* 90.104512. DOI: https://doi.org/10.1016/j.ebiom.2023.104512.

Heart, Tsipi, Ofir Ben-Assuli, and Itamar Shabtai (2017). “A review of PHR, EMR and EHR integration: A more personalized healthcare and public health policy”. In: *Health Policy and Technology* 6.1, pp. 20–25. ISSN: 2211-8837. DOI: https://doi.org/10.1016/j.

hlpt.2016.08.002. URL: https://www.sciencedirect.com/science/article/pii/S2211883716300624.

Hickey, A.M. and A.M. Davis (2003). "Elicitation technique selection: how do experts do it?" In: *Proceedings. 11th IEEE International Requirements Engineering Conference, 2003.* Pp. 169–178. DOI: 10.1109/ICRE.2003.1232748.

Hosseini, Azamossadat et al. (2023). "Integrated personal health record (PHR) security: requirements and mechanisms". In: *BMC Medical Informatics and Decision Making* 23.1, p. 116.

"ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering" (2018). In: *ISO/IEC/IEEE 29148:2018(E)*, pp. 1–104. DOI: 10.1109/IEEESTD.2018.8559686.

Jiang, Albert Q. et al. (2023). *Mistral 7B.* arXiv: 2310.06825 [cs.CL]. URL: https://arxiv.org/abs/2310.06825.

Labrak, Yanis et al. (2024). *BioMistral: A Collection of Open-Source Pretrained Large Language Models for Medical Domains.* arXiv: 2402.10373 [cs.CL]. URL: https://arxiv.org/abs/2402.10373.

Laplante, Phillip A (2019). *Requirements engineering for software and systems.* 3rd. Auerbach Publications.

Laravel Holdings Inc. (2024). *The PHP Framework for Web Artisans.* URL: https://laravel.com/. (Accessed: 14 Nov 2024).

Lee, Jisan et al. (2021). "Review of national-level personal health records in advanced countries". In: *Healthcare Informatics Research* 27.2, pp. 102–109.

Loweth, Robert P. et al. (Mar. 2021). "A Comparative Analysis of Information Gathering Meetings Conducted by Novice Design Teams Across Multiple Design Project Stages". In: *Journal of Mechanical Design* 143.9, p. 092301. ISSN: 1050-0472. DOI: 10.1115/1.4049970. eprint: https://asmedigitalcollection.asme.org/mechanicaldesign/article-pdf/143/9/092301/6667332/md_143_9_092301.pdf. URL: https://doi.org/10.1115/1.4049970.

Luo, Lu (2001). "Software testing techniques". In: *Institute for software research international Carnegie mellon university Pittsburgh, PA* 15232.1-19, p. 19.

Medvalet (2024). *Despre noi.* URL: https://medvalet.ro/. (Accessed: 11 Nov 2024).

Meskó, Bertalan (2023). "Prompt engineering as an important emerging skill for medical professionals: tutorial". In: *Journal of medical Internet research* 25, e50638.

Meta Platforms (n.d.). *React. The library for web and native user interfaces.* URL: https://react.dev/. (Accessed: 14 Nov 2024).

Mohedas, Ibrahim et al. (2022). “The use of recommended interviewing practices by novice engineering designers to elicit information during requirements development”. In: *Design Science* 8, e16. DOI: 10.1017/dsj.2022.4.

Morphy, Tamzin (2020). *Stakeholder Analysis | Definition and best method*. URL: https://www.stakeholdermap.com/stakeholder-analysis.html. (Accessed: 31 Oct 2024).

National Bureau of Statistics of the Republic of Moldova (2024). *Population*. URL: https://statistica.gov.md/en/statistic_indicator_details/25. (Accessed: 22 Oct 2024).

Naveed, Humza et al. (2023). “A comprehensive overview of large language models”. In: *arXiv preprint arXiv:2307.06435*.

Nazi, Zabir Al and Wei Peng (2024). “Large Language Models in Healthcare and Medical Domain: A Review”. In: *Informatics* 11.3. ISSN: 2227-9709. DOI: 10.3390/informatics11030057. URL: https://www.mdpi.com/2227-9709/11/3/57.

Niu, Qian et al. (2024). *From Text to Multimodality: Exploring the Evolution and Impact of Large Language Models in Medical Practice*. arXiv: 2410.01812 [cs.CY]. URL: https://arxiv.org/abs/2410.01812.

OpenAI et al. (2024). *GPT-4 Technical Report*. arXiv: 2303.08774 [cs.CL]. URL: https://arxiv.org/abs/2303.08774.

OpenJS Foundation (n.d.). *Fast, unopinionated, minimalist web framework for Node.js*. URL: https://expressjs.com/. (Accessed: 14 Nov 2024).

Oracle (2020). *What is a Database?* URL: https://www.oracle.com/uk/database/what-is-database/. (Accessed: 28 Oct 2024).

Paradigm, Visual (2024). *What is Unified Modeling Language (UML)?* URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/#sequence-diagram. (Accessed: 07 Nov 2024).

Prenner, Nils, Carolin Unger-Windeler, and Kurt Schneider (2020). “How are hybrid development approaches organized? A systematic literature review”. In: *Proceedings of the International Conference on Software and System Processes*, pp. 145–154.

reddi, Latha Thamma (2023). *Stakeholder Analysis using the Power Interest Grid*. URL: https://www.projectmanagement.com/wikis/368897/Stakeholder-Analysis--using-the-Power-Interest-Grid. (Accessed: 16 Dec 2024).

Ruparelia, Nayan B. (May 2010). “Software development lifecycle models”. In: *SIGSOFT Softw. Eng. Notes* 35.3, pp. 8–13. ISSN: 0163-5948. DOI: 10.1145/1764810.1764814. URL: https://doi.org/10.1145/1764810.1764814.

Sharma, Shreta and SK Pandey (2013). “Revisiting requirements elicitation techniques”. In: *International Journal of Computer Applications* 75.12.

Singhal, Karan et al. (2023). *Towards Expert-Level Medical Question Answering with Large Language Models*. arXiv: 2305.09617 [cs.CL]. URL: https://arxiv.org/abs/2305.09617.

Springer, Andreas (2023). *What is Jira Software, and why use it?* URL: https://community.atlassian.com/t5/Jira-articles/What-is-Jira-Software-and-why-use-it/ba-p/2323812. (Accessed: 26 Oct 2024).

Stack Overflow (2024). *Technology*. URL: https://survey.stackoverflow.co/2024/technology#most-popular-technologies-language. (Accessed: 14 Nov 2024).

Sunner, Daminderjit (2016). "Agile: Adapting to need of the hour: Understanding Agile methodology and Agile techniques". In: pp. 130–135. DOI: 10.1109/ICATCCT.2016.7911978.

Svelte (n.d.). *Overview*. URL: https://svelte.dev/docs/svelte/overview. (Accessed: 14 Nov 2024).

Team, Gemini et al. (2024). *Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context*. arXiv: 2403.05530 [cs.CL]. URL: https://arxiv.org/abs/2403.05530.

Tiwari, Saurabh, Santosh Singh Rathore, and Atul Gupta (2012). "Selecting requirement elicitation techniques for software projects". In: *2012 CSI Sixth International Conference on Software Engineering (CONSEG)*, pp. 1–10. DOI: 10.1109/CONSEG.2012.6349486.

United Nations Development Programme (2023). *A 100% digital state: The strategy for digital transformation of the Republic of Moldova for 2023-2030, approved by the Executive*. URL: https://www.undp.org/moldova/press-releases/100-digital-state-strategy-digital-transformation-republic-moldova-2023-2030-approved-executive. (Accessed: 22 Oct 2024).

USMF (2023). *Brief history*. URL: https://usmf.md/en/brief-history-usmf. (Accessed: 22 Oct 2024).

Vailshery, Lionel Sujay (2024). *Most used web frameworks among developers worldwide, as of 2024*. URL: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/. (Accessed: 14 Nov 2024).

Vaswani, Ashish et al. (2017). "Attention is all you need". In: *Advances in neural information processing systems*.

Xiao, Hanguang et al. (2024). *A Comprehensive Survey of Large Language Models and Multimodal Large Language Models in Medicine*. arXiv: 2405.08603 [cs.CL]. URL: https://arxiv.org/abs/2405.08603.

Yas, Qahtan, Abdulbasit Alazzawi, and Bahbibi Rahmatullah (Nov. 2023). "A Comprehensive Review of Software Development Life Cycle methodologies: Pros, Cons, and Future Directions". In: *Iraqi Journal For Computer Science and Mathematics* 4.4, pp. 173–190.

DOI: 10.52866/ijcsm.2023.04.04.014. URL: https://journal.esj.edu.iq/index.php/IJCM/article/view/664.

Yin, Shukang et al. (2024). "A survey on multimodal large language models". In: *National Science Review*, nwae403.

You, Evan (2024). *What is Vue?* URL: https://vuejs.org/guide/introduction.html. (Accessed: 14 Nov 2024).

Young, Ralph R (2002). "Recommended requirements gathering practices". In: *CrossTalk* 15.4, pp. 9–12.

Zhang, Duzhen et al. (2024). "Mm-llms: Recent advances in multimodal large language models". In: *arXiv preprint arXiv:2401.13601*.

Zhou, Ce et al. (2023). "A comprehensive survey on pretrained foundation models: A history from bert to chatgpt". In: *arXiv preprint arXiv:2302.09419*.

Appendix A

Requirements

ID	Requirement	Priority
1.1	The system must be accessible on all modern desktop and mobile-based browsers.	Must Have
1.2	The system must be accessible from any location by using an internet connection.	Must Have
1.3	The system must store the data in a secure manner, ensuring that only the patient and the doctor can access the data.	Must Have
1.4	When shared with the doctor via a link, the system must load within 3 to 5 seconds when accessed via a desktop browser.	Should Have
1.5	When shared with the doctor via a link, the system should secure the data with a unique token or PIN that expires after the specified time frame.	Could Have
1.6	The system could be accessible on all modern mobile devices via a mobile application.	Could Have

Table A.1: Non-functional Requirements

ID	Requirement	Priority
2.1	The system must provide a secure login mechanism for patients via Multi Factor Authentication.	Must Have
2.2	If used on mobile, the system should allow the patient to use biometric authentication for logging in.	Should Have

Table A.2: Login Requirements

ID	Requirement	Priority
3.1	The system must allow patients to upload their own medical records in a variety of formats (PDF, DOC, etc).	Must Have
3.2	The system must allow the patient to specify and categorise the type of document they are uploading (lab test, doctor consultation, etc).	Must Have
3.3	If used on mobile, the system should allow the patient to take a picture of the document and upload it.	Could Have

Table A.3: Document Upload Requirements

ID	Requirement	Priority
4.1	The system must allow the patient to generate a shareable link to provide access to their medical records.	Must Have
4.2	When creating the shareable link, the system must allow the patient to set an expiration date for the link.	Must Have
4.3	When creating the shareable link, the system should allow the patient to set an access password for the link.	Should Have
4.4	When creating the shareable link, the system should allow the patient to select which records to share with the doctor.	Should Have

Table A.4: Patient Shareable Link Requirements

ID	Requirement	Priority
5.1	The patient personal cabinet must provide an overview of the patient's history through 3 main sections: personal information, lab tests, and doctor consultations.	Must Have
5.2	The system must display the patient's history in a chronological order in the form of a timeline.	Must Have
5.3	The system must allow patients to add their own personal information, such as name, date of birth, or address.	Must Have
5.4	The system must allow the patient to add their own allergies.	Must Have
5.5	The system must allow the patient to add their own vaccinations.	Must Have
5.6	When viewing doctor consultations, the system should divide them into categories based on the domain of the doctor (cardiology, neurology, etc).	Should Have
5.7	The system should allow the patient to enter vitals information, such as height, weight, blood pressure, etc.	Should Have
5.8	When multiple vital entries are made, the system could display a historical graph of the patient's vitals.	Could Have
5.9	The system could allow the patient to switch between viewing the lab tests in the document format or in a tabular, numerical format.	Could Have

Table A.5: Patient Personal Cabinet Requirements

ID	Requirement	Priority
6.1	The system must provide an overview of the patient history through 3 main sections: personal information, lab tests, and doctor consultations.	Must Have
6.2	When shared with the doctor, the system must allow the doctor to only view the patient's history, not edit it.	Must Have
6.3	The system must allow the doctor to view blood tests in a graphical format.	Must Have
6.4	The system must allow the doctor to view blood tests in a numerical, tabular format.	Must Have
6.5	The system must allow the doctor to view the patient's history in a chronological order.	Must Have
6.6	The system must display the doctor consultation and every lab test, except for blood tests, in a free text or document format.	Must Have
6.7	When viewing blood test results, the system should show the source document of the blood test value.	Should Have
6.8	For blood test results, the system should display the normal range values for each test.	Should Have

Table A.6: Shared Patient Information Requirements (Doctor View)

ID	Requirement	Priority
7.1	The system must allow patients to enter their current medication including details such as the name of the drug, dosage, frequency and start/end date.	Must Have
7.2	The system must allow patients to add new medication to their list.	Must Have
7.3	When adding medication, the system should have 2 options: add a simplified version of the medication or add a detailed version of the medication.	Should Have
7.4	When choosing the simplified version, the system should allow the patient to just add the name, dosage and duration of the medication.	Should Have
7.5	When choosing the detailed version, the system should allow the patient to add the name, dosage, frequency, start/end date, and the reason for taking the medication.	Should Have
7.6	The system should allow patients to add their past medication	Should Have
7.7	The system should allow patients to set medication reminders.	Should Have
7.8	After entering the medication, the system could allow the patient to track the medication intake.	Could Have

Table A.7: Patient Medication Requirements

Appendix B

UML Diagrams

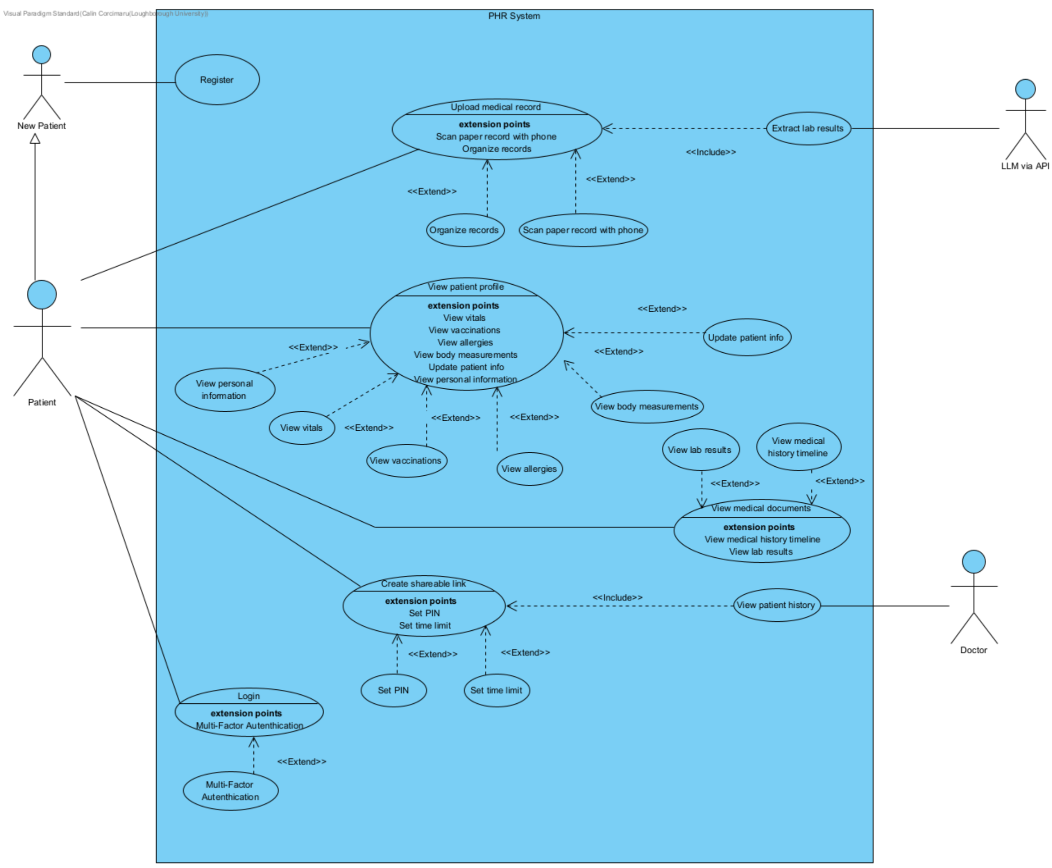
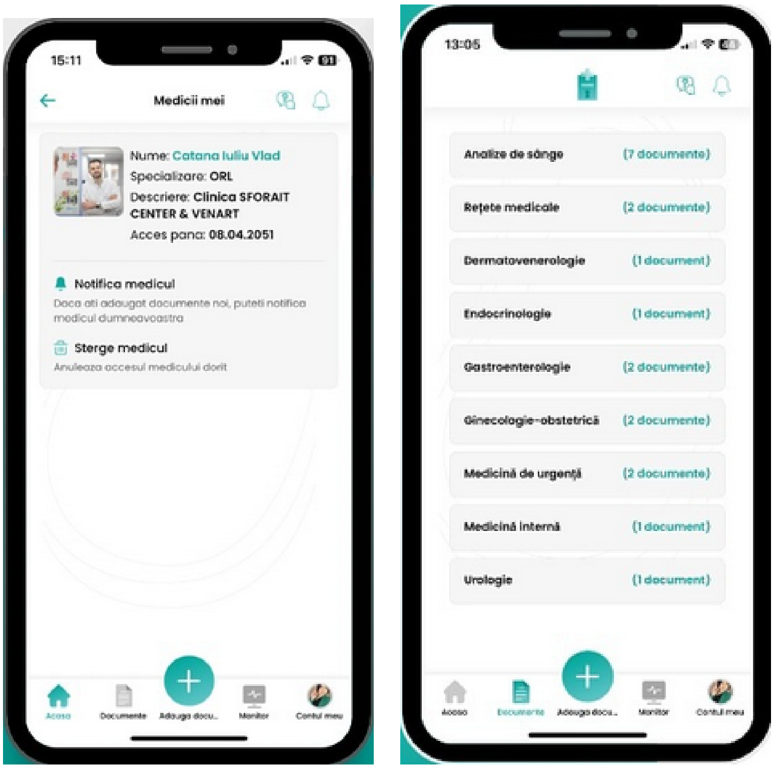


Figure B.1: UML Use Case Diagram

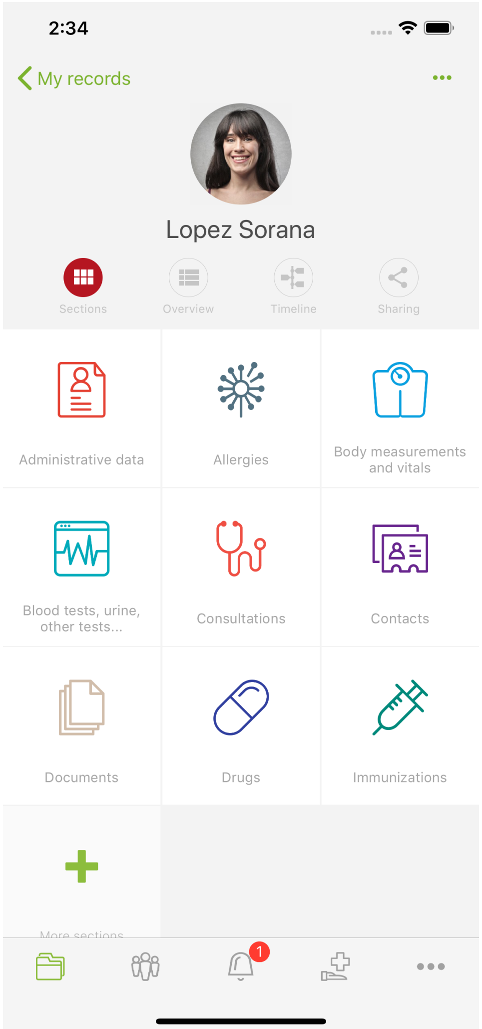
Appendix C

Existing Applications Screenshots

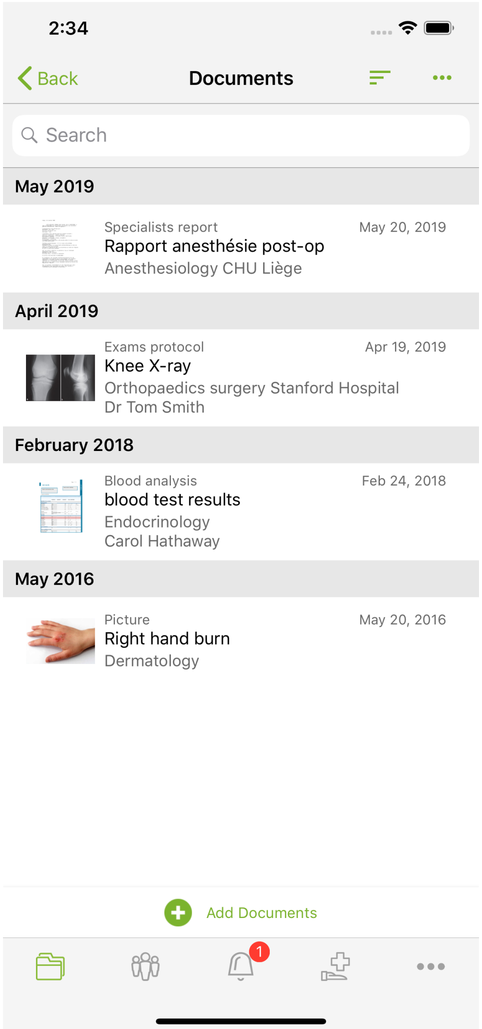


(a) My doctors screen (b) Uploaded documents screen

Figure C.1: Medvalet screenshots

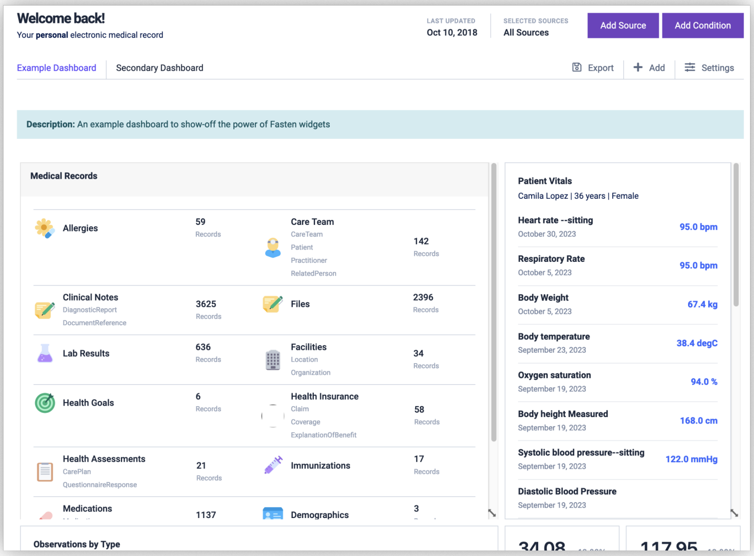


(a) PHR Sections screen

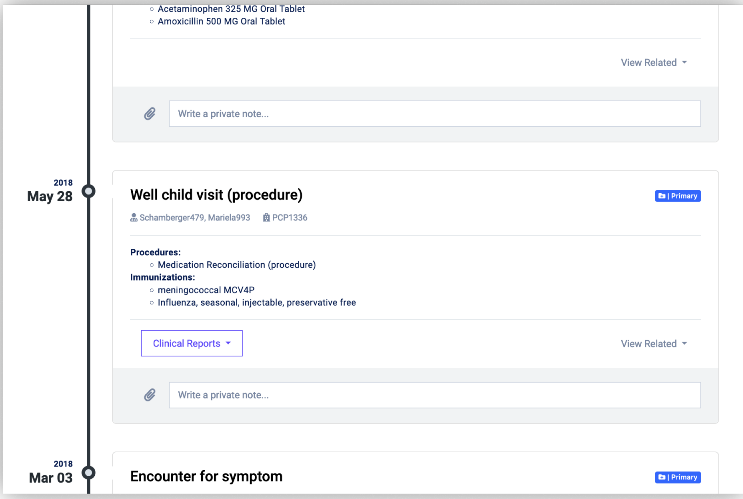


(b) Documents section screen

Figure C.2: Andaman7 screenshots



(a) Dashboard screen



(b) Visit history screen

Figure C.3: Fasten Health screenshots