# HealthLink

Team 13

Mackenzie Pascual 22518529

CS353 Project Report

20/12/2024

Department of Computer Science Maynooth

## Table of Contents

# Chapter 0

Introduction to Agile Development and SCRUM

## What is Agile Development?

Agile development is a general term used in approaches in developing software. One of the main focuses with Agile development (compared to other methodologies) is that it focuses on the people doing the work and how they collaborate to find a solution.

Before Agile Methodology was created, Software Developers used more heavyweight development methods such as Waterfall. While these methods brought much needed discipline and structure to software development through well thought-out specification documents it came with a few downsides. These methodologies were often rigid and slow, making implement changes difficult and expensive to do so. They also did not allow for improvement and iterative feedback during the development cycle leading to problems only being discovered post deployment. Software developed with these methods was typically complex and monolithic but ensured the final deliverable matched the specification set out at the beginning of development.

However, with the rise of the Internet in the early 2000s, many Software Developers started developing Internet applications. With the rapid growth of the Internet, there was increased pressure to bring these applications to market faster. There were also increased financial and competitive pressures emerging as smaller organizations of Software Developers (later known as Start-Ups) started developing applications for the Internet often competing against each other and often on a very lean budget. These organizations also consisted of developers who did not have traditional backgrounds in Computer Science and thus were not completely familiar with heavyweight development methods.

Taking all these factors into account, especially the long deployment cycles of applications, these smaller organizations realized that the heavyweight development methodologies they have always used were impractical, inefficient, and costly. This led these organizations to look for alternative ways to make their development cycles quicker and more adaptable to the ever-expanding world of the Internet.

These shortcomings then led to seventeen Software Developers gathering in 2001 to combine their experience and discuss a new way to approach Software Development and thus Agile Methodology was born.

The Agile Methodology was created as a new lightweight development method compared to its heavyweight development predecessors such as Waterfall. With the Waterfall methodology, development was divided into distinct sequential phases where each phase had to be fully completed before continuing to the next one, detailed planning and documentation had to be created before any development could take place which made changes difficult to implement once a phase was completed, and as discussed, the final product was only available at the end of development. Agile methodology on the other hand takes an iterative approach which divides the development process into shorter phases called sprints, which typically lasts 2-4 weeks, allowing an incremental development process. Agile allows for flexibility and adaptability in the sense that changes and adjustments can be implemented easier even if that phase of development has concluded. This approach allows for continuous delivery of the product and gradually releasing new features and improvements once they are ready to be released (i.e. there is no definite end-product as development can go on throughout the product's lifecycle).

For the Agile Methodology to work effectively and efficiently, the seventeen Software Developers behind the Agile Methodology created the Agile Manifesto, a document that outlines the core values and principles that Software Developers should follow to guide them through the development process to achieve flexibility, collaboration and customer satisfaction. In the next section we discuss these core values and principles in more depth.

## The 4 core values and main principles of Agile Methodology

The 4 core values are as follows:

1.) **Individuals and interactions over processes and tools:**

In short, this value entails that effective communication and collaboration amongst members within the development team is key to making good software rather than the processes and tools that a development team follow or use.

2.) **Working software over comprehensive documentation:**

This means that while having a comprehensive specification of software and how it would theoretically work is good to have, it effectively has less value than an actual working software since at the end of the day is what we are after in the development cycle. Rather have working software and create documentation off that than creating software based on an overly detailed specification as working software can be used as the primary measure of progress.

3.) **<u>Customer collaboration</u> over contract negotiation:**

Software Developers should encourage customers to give feedback on their product so they are able to find ways to improve their product. This brings a sense of active participation to consumers who feel more involved in the development of the software they use whilst also giving valuable information to Software Developers. Examples of this comes in beta programs for software which also furthers the consumers' anticipation for the finalized release.

4.) **<u>Responding to change</u> over following the plan:**

With every development cycle, no matter how well planned it is, there will be unforeseen circumstances that may affect the development of the product which may require the development team to change its requirements. This value, in short, says that teams should be able to embrace flexibility and adaptability in their development cycle, no matter the phase of development they are in and not solely follow a plan that might look better on paper rather than in practice.

While Agile methodology values the unlined parts of the core values more, it does not mean the non-underlined parts are not important but should have less priority.

The 12 principles of the Agile Manifesto are ways to help and support the core values and to help Software Developers to implement the core values in their development cycles. The principles are as follows:

1.) **To satisfy the customer through early and continuous delivery of valuable software**

The customer should be valued as important as someone on the development team because without the customer you'll have no one using your product. This principle ensures that customer satisfaction is achieved through transparent and consistent communication or information about the development cycle with them.

2.) **Welcome changing requirements, even late in development**

As discussed in the 4th core value, being able to adapt to change is key to a successful development cycle. By being open to change and being able to adapt to it, developers can implement features for the customers evolving needs.

3.) **Delivering working software frequently**

By regularly releasing functional software increments in short spans of time, customers can provide crucial feedback faster to developers which enables them to adapt and change requirements easier and implement them faster.

4.) **Businesspeople and developers working together daily**

Communication between the developers and people on the business side of a product should always communicate with each other daily to give feedback and updates on what they're working and if there are any delays so neither side assume incorrectly what they are doing and ensure everyone is on the same page.

5.) **Build projects around motivated individuals**

By having passionate and motivated people working on your product it ensures that the work that they produce will be of a higher standard and quality as they will take more pride in it. This requires a level of trust from the stakeholders that the development team will complete their job well and should offer them support when needed.

6.) **Most effective and efficient communication is face-to-face**

Having face-to-face interactions ensures there are minimal misunderstandings and can provide instant acknowledgment that your communication was heard and received so there is no possibility that communications are overlooked or lost.

7.) **Working software is the primary measure of progress**

Relating back to the 2nd core value, working software is a good measure of progress as it ensures deliverables are functional and working as intended rather than releasing half completed features. It also ensures that customers can receive continuous delivery of valuable software as stated in the 1st Agile principle.

8.) **Working at a constant pace to promote sustainable development**

To avoid burnout, development teams should work at a consistent pace. This guarantees long-term success, and that development of features can be released on a continuous basis.

9.) **Good design practices and attention to details**

This principle is included to maintain a high standard of deliverables coming from the development team. It also future proofs the product by ensuring easy maintenance and adaptability of the features.

10.) **Simplicity**

The Agile Manifesto states the following: “the art of maximizing the amount of work not done – is essential”. This means that developers should prioritize their most valuable features first and avoid overcomplicating their products with complex or unnecessary features. This can also be found out via customer feedback and can determine what to prioritize more.

11.) **Allowing teams to self-organize**

Development teams understand how to operate amongst themselves the best. Teams should be able to self-organize to properly utilize their strengths which can optimize efficiency.

12.) **Regular team reflections**

By reflecting and looking back on past development cycles, teams can spot inefficiencies which they can take account of in future development. This can be effective and improve the team’s performance in the future.

## What is SCRUM and how does it work?

As stated before, Agile is a general term used for lightweight software development. One of these development strategies is the Scrum framework. Inspired by the rugby term, Scrum strives to get a team together to move a product to completion. Scrum focuses on teamwork, incremental delivery of features, and self-organizing teams' aspects of Agile.

Scrum consists of 3 key roles, 3 artifacts, and 4 ceremonies (meetings). The 3 roles are described as:

1.) **Scrum Master:**

Scrum Masters is responsible for ensuring the team operates as efficiently as possible. They keep track, plan and lead daily standup and sprint meetings (discussed later). They also serve as the communication channel between stakeholders and the development team. They keep track of team members and offer support to them when needed.

2.) **Product Owner:**

The responsibility of the Product Owner is to ensure that the development team remains on the path of the overall product goals. They manage product backlogs by prioritization, set a product vision for the team to achieve and make sure the team is focused on product needs that are communicated from customers and shareholders.

3.) **Development team:**

Responsible for completing the hands-on tasks needed for the product such as feature development. They help in planning upcoming sprints, setting deliverable goals and ensuring that the goals are completed after each sprint. In short, they are the technical team that creates the product from the ground up.

The 3 Scrum Artifacts are:

1.) **Product Backlog**

This is all the desired work and requirements that needs to be completed for the project. Each item in the product backlog should hold value to the customers of the product. The items are prioritized by the Product Owner at the start of development and re-prioritized after each sprint. Each item is given an estimated time of completion and adjusted accordingly.

2.) **Sprint Backlog**

The items, taken from the product backlog, that will be worked on during the next sprint. This is done by selecting a task/item to be completed in the product backlog and breaking it down into smaller pieces. Work is never assigned but chosen by the development team members. The estimated work time is adjusted daily, and any team members may add, modify or delete items in the backlog.

3.) **Burndown Charts**

Burndown charts can be used for both product and sprint backlogs. It is a way to visualize and track the progress towards the sprint or product goal which allows team members to determine if they are on track to complete their desired goals on-time. Using a burndown chart helps teams better estimate the time allocated to a task in the future as they can study their previous trends. It also helps to determine whether development is sustainable or not and can adjust accordingly.

The 4 Scrum Ceremonies are:

1.) **Sprint Planning**

The meeting where the sprint backlog is created. This is done collaboratively and not done alone by the Scrum Master. The team selects items from the Product Backlog which they can commit to completing after the incoming sprint is completed. This is also when time estimation takes place and allocated to each task to be completed. The Product Owner should also be involved to help the team to prioritize what tasks should be worked on and keep them on track of the product goal.

2.) **Sprint review**

This is an informal meeting that gives the development team the opportunity to demo/display the task they've been working on and/or completed during the sprint. It should be an informal gathering and should have all members present (Product Owner, Scrum Master, Development team). This can generate feedback that might help the development team to reach their sprint goal faster or to keep them on track of the goal at hand.

3.) **Sprint Retrospective**

This refers to one of the 12 principles of Agile mentioned previously. It is a moment where the whole team (Product Owner, Scrum Master, Development team) can come together and reflect on the sprint that has just concluded. It is good way to evaluate what is and is not working during the sprint and a place where people can address their concerns. It should follow what the team should start doing, what they should stop doing, what they should continue doing, and what they could do better.

4.) **Daily Scrum**

A meeting at the start of each day that last about 15-20 minutes. It's the chance for each team member to 'stand-up' and talk about the task they are currently working on. If they need any help or have something preventing them continuing their work this is the moment they can express it and the Scrum Master, who leads these meetings, will find solutions by outsourcing or offering help after the meeting. It is a good way to keep track of everyone's progress during the ongoing sprint.

## How we implemented SCRUM with our project

We implemented Scrum by assigning Randon Reddy as the Product Owner to keep us on track of and prioritize the product goals and organize meetings. All team members were a part of the development team and unlike traditional Scrum where one person is chosen as Scrum master, each team member had the opportunity to be Scrum Master and to lead the Daily Scrum meetings. This was rotated weekly.

///////////////////////////////////////////////////////////////////////////////////////////////////

# Chapter 1: Project Proposal

### Initial Project Proposal: A Personal Budgeting App

In my first plan, I wanted to make a Personal Budgeting App called BudGapp, which would help people easily and effectively handle their money. The idea was to make a digital partner that would give people the tools they needed to make their own financial decisions and plans.

I picked this plan because it addressed a problem that everyone has money all over the place and does not have clear spending tools. For example, a lot of my friends and I had trouble keeping track of our money when we were in college. We often missed payment due dates, spent too much on things like fun, or forgot to save money for things we needed. Because of these problems, I came up with the idea of an app that would not only make managing personal finances easier but also teach people how to keep good money habits.

The following were planned features for the app:

1. Tracking of Expenses: The app would let users keep track of their daily spending and put it into set or custom groups, like food, rent, activities, and more. It was easy for people to keep track of how much they spent on eating out over the course of a month.
2. Budget Management: Users could set yearly or monthly money goals and see how close they were to being met. For instance, the app would let the user know if their food costs went over a certain limit for that area.
3. Income Management: Users would list their sources of income, such as pay, contract work, and idle income, to get a full picture of their money situation. For instance, users could look at their overall monthly income and costs and change how much they spend based on that.

4. Visual Reports: The app would make graphs, pie charts, and line charts to show how spending and income changed over time. A pie chart might show, for example, that 40% of monthly costs went to rent and 10% to fun things to do.
5. Notifications: Users could stay on top of their financial plans by getting personalized alerts when they have upcoming bill payments, low account amounts, or overspending in certain categories.
6. Savings Tracker: Users could set savings goals and keep track of their progress toward those goals, like saving for a trip or a disaster fund.
7. Multi-Currency Support: This feature lets users from other countries handle their money in more than one currency, with the amounts being converted instantly based on current exchange rates. This function would be useful for people who live abroad or move a lot.

The idea was appealing because it could be used by a lot of people and help them with common money problems they face around the world. A tool that makes planning and managing spending easier seemed like a useful project for people who have trouble keeping their money in order. I was also able to meet the technology standards because

the project would involve user identification, database integration, and data visualization, all of which are skills I was already good at.
In addition, the app might have been useful for users in the long run. For example, someone who uses the app regularly might be able to figure out where they spend too much, which could help them save money and feel safer about their finances. In my own experience, having this kind of tool in college could have kept me from spending too much and helped me better manage my money.

But the budgeting app didn't have enough of a unique feature to make a strong impact as I thought more about the idea and compared it to the goals of innovation and technological progress. Even though it solved a well-known issue, the method did not stand out in a market full of similar products. This understanding was a turning point in my path to a more active and important project.

## Why the Plan Was Turned Down

Even though I really wanted the Personal Budgeting App, the idea did not get accepted. The reasons it was turned down were:

1. "Perceived Simplicity": The people who were reviewing the plan thought it was too simple. They thought the project's scope wasn't as complicated as they thought it would be for this level of study.
2. Excessive Competition: There are already a lot of well-known planning apps on the market, like Mint, YNAB (You Need A Budget), and PocketGuard. Making yet another planning app wouldn't really add anything new or useful to the field.
3. Lack of Differentiation: My idea didn't have anything new or innovative that made it different from other options that were already out there. Even though it was meant to make managing money easier, it didn't have a unique selling point or a way to specifically meet specific needs.
4. This plan did not meet the standards set by the judges, which were that it should show advanced mechanical skills and creative thought. They supported projects that used cutting-edge tools or tried to solve tougher problems.
5. A large group of people: The app wasn't made for a specific group of people; it was made to be used by everyone. Often, projects that have a clear and specific group of users are more effective and easier to defend.

Even though this refusal was disappointing at first, it gave me a great chance to think. It showed how important it was to find a project that not only solved a problem but also showed a lot of academic and creative depth. Having talks with my peers and teachers helped me sharpen my view, which helped me choose a project that was more in line with my goals.

## Project Proposal Accepted: HealthLink, an app for managing medical data

HealthLink, the chosen project, was a web-based app that was made to fix problems with the way medical information is managed. By making a safe and easy-to-use digital tool, it wanted to change the way patients, doctors, and pharmacies work together.

### HealthLink's Goals and Plans

HealthLink wanted to make a unified system that was easy for people to use and that dealt with common problems in the healthcare field. Some of these problems were delays in handling prescriptions, mistakes in patient records, and the fact that paper-based processes were hard to manage. The goals of the application were to

1. Improve Data Accuracy: HealthLink cut down on mistakes caused by misunderstandings or typing in data by hand by letting patients enter and handle their own data.

2. Streamline Workflows: The app was meant to digitize tasks like managing prescriptions and making appointments, which would save time for both patients and healthcare workers.

3. Make it easier to share data safely: HealthLink created a way for approved users to view and share medical information safely.

4. Promote Sustainability: The app helped protect the earth by getting rid of paper notes and prescriptions.

5. Give Patients More Power: Patients had more control over their medical information, which made them feel like they owned it and were responsible for it.

## Important Things About HealthLink

1. User Roles and Permissions: - Patients: Make and handle personal health profiles, share medical records, and get digital copies of medications.
Doctors and consultants can look at patient data, write digital prescriptions, and set up follow-up meetings.  Pharmacists: Check and hand out medicines while making sure they are correct and safe.

2. Secure login: Sensitive medical information was kept safe with multi-factor login and protected data storage.

3. QR Code Integration: Patients could make their own QR codes that would quickly share their medical information with healthcare workers.

4. User-Centric Design: The system was made to work well for a wide range of users, from those who are very tech-savvy to those who aren't.

5. Scalability: The design of the app allowed for future improvements, like adding fingerprint login and data analytics for healthcare trends.

## What Made HealthLink the Best Choice?

1. How it applies to problems in the real world:
HealthLink dealt with important problems in the healthcare field, like managing data

inefficiently and waiting too long to process prescriptions. Patients, healthcare workers, and pharmacists were all supposed to get real benefits from it.

2. The level of technical difficulty:
For the project to work, advanced technologies had to be used, such as database security, API development, and safe methods for sharing data. This level of difficulty showed how technically skilled I am and how well I can solve problems.

3. New ideas:
HealthLink was different from other medical management systems because it had features like QR code integration and the ability to have multiple users. These new features showed that the company was creative and ahead of its time.

4. Growth as a Person:
I learned about new tools and ways of doing things because of the project. For example, I learned about ReactJS for front-end development, MongoDB for database management, and Agile techniques for project management. This experience helped me get better at both professional skills and working with others.

5. Fit with the Evaluation Criteria:
HealthLink met the judges' needs by having a user-centered method, a deep scientific understanding, and real-world use.

## What I Think About HealthLink

It was scary at first to switch from the Personal Budgeting App to HealthLink. The second one required more time and work to learn and had a steeper learning curve. As I worked on the job more, though, I realized it was a really great experience.

The thought of helping to create a solution that would make healthcare easier to use and better for patients was very inspiring. To build HealthLink, technology problems had to be solved, like encrypting data securely and making the user interface easy to use. Every problem that came up was a chance to learn and grow.

The process also showed how important it is to work together. Working closely with a group of people from different backgrounds led to a wider range of ideas and points of view, which in turn made the application stronger and more user centered. It was especially satisfying for me to study what users wanted, create key features, and make sure data was safe. Being able to see the project grow from an idea to a working prototype showed how determination and teamwork can pay off. I also gained a deeper understanding of how technology can make lives better and processes run more smoothly in important areas like healthcare.

HealthLink is a project that showed how resilient, flexible, and creative people can be. Even though it was hard, the trip taught us a lot about how to solve problems in the real world and showed how technology can change old ways of doing things. I have a much better understanding of how important user-centered design and safe data handling are in the healthcare field because of this project.

The project's ongoing method also showed how important it is to keep getting feedback and making things better. The final product met the complex needs of both healthcare workers and patients by taking user feedback into account and trying the app's features thoroughly. This part of the development process made me appreciate Agile methods and how well they work for handling complicated projects even more.

## Moving on to Chapter 2

Now that we know what HealthLink all is about, the next chapter goes over the technical parts of the project, such as the tools, design, and testing methods that made it possible. In this chapter, the steps and decisions that went into making the product are broken down in detail to show how an ambitious idea can be turned into a real answer.

# Chapter 2: The Actual Project

## 2 HealthLink

Our project named HealthLink is a medical web application in which medical information is stored on a database. Within the web application there are three types of user consultant, patient and pharmacy. Each account grants different permissions from script creation to script viewing and script retrieval. This process ends the need for consultants to send scripts via post to patients who then need to deliver this script to a pharmacy with the medication in stock. In addition to this, patients can create their accounts with the correct information to avoid any confusion/spelling errors/typos which plague current medical systems. Our goal for this project as mentioned above is to eliminate errors within patient information due to communication/human error, efficient script delivery instead of waiting for the script to be sent via post as well as eliminating paper waste and to overall allow for patients to have more control over their information and what they need to do.

## 2.1 Technologies

For this project we utilised Gitlab as our choice of DevOps software package. This allowed us to collaborate effectively as well as efficiently and gave us the ability to thoroughly test, amend and code before committing to the main branch of the project by employing our own personal branches for the sections we had to work on.

The languages used throughout this project for both front end and back end are HTML, CSS, JavaScript, and NoSQL queries. These languages were chosen due to our own familiarity with them as a group, their versatility and power thus allowing us to develop our project with the best languages at our disposal without the need to learn a new language or needing to make up for a weaker language.

Throughout our development we used many libraries and packages to allow us to create HealthLink, they are listed as follows along with their purpose and functionality.

ReactJS was used for our front-end development. ReactJS is an open-source front-end JavaScript library which allowed for our user interfaces to be more seamless as well as granting us a structured file architecture as said in *2.2 Architecture*. In tandem with

ReactJS, we employed Vite a dev server as well as a build command to allow for changes to be instantly visualised whilst developing both front-end and back-end, thus allowing for our testing to take place at much faster speeds.

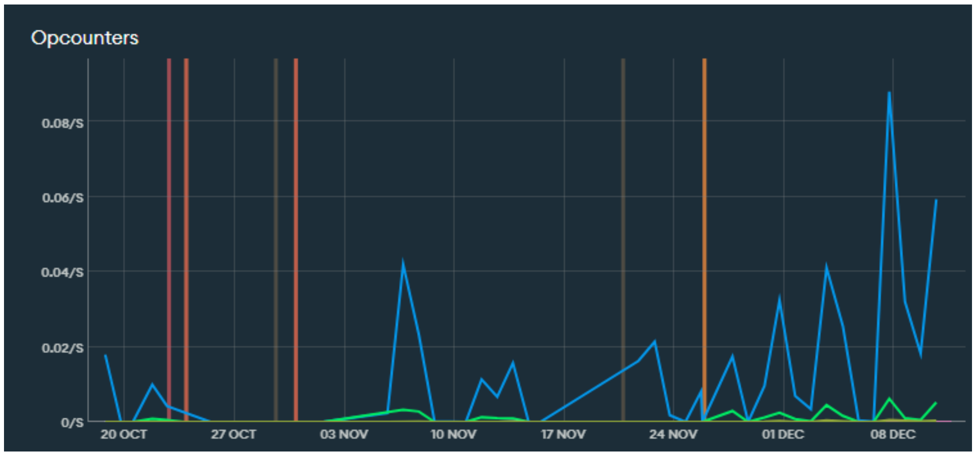
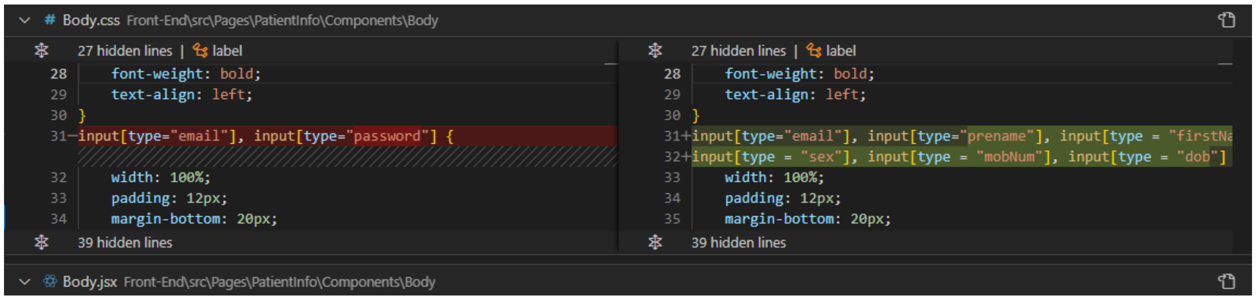


*Figure 2 MongoDB Graph*

The database chosen for the project was MongoDB as well as NodeJS for the connection. MongoDB was chosen due to it being a NoSQL database, being extremely versatile, and having extensive information on table interaction as shown above in *figure 2* with the bonus of it being free. With MongoDB’s intuitive tables and graphs at our disposal we were able to properly set up and test our tables, as we could see how many connections there were to a particular table and how many queries were being sent to it. This really helped with development as it allowed us to identify which tables were being overloaded and what table were redundant. While NodeJS was chosen for its familiarity with the team and its power. These two provided us with the fundamentals to connect our code to the database and to be able to store the information efficiently.


Body.css Front-End\src\Pages\PatientInfo\Components\Body

27 hidden lines | label

```
    font-weight: bold;
    text-align: left;
}
-input[type="email"], input[type="password"] {
    width: 100%;
    padding: 12px;
    margin-bottom: 20px;
```

39 hidden lines

27 hidden lines | label

```
    font-weight: bold;
    text-align: left;
}
+input[type="email"], input[type="prename"], input[type = "firstNa
+input[type = "sex"], input[type = "mobNum"], input[type = "dob"]
    width: 100%;
    padding: 12px;
    margin-bottom: 20px;
```

39 hidden lines

Body.jsx Front-End\src\Pages\PatientInfo\Components\Body

*Figure 2.1 VS Git Changes*

The IDE of choice for development was Microsoft Visual Studio Code. Visual Studio's adaptability, power, and ease of use put it as our number one IDE for development. The ability to visualise our branch changes in detail with information on who the authors of changes were and a line-by-line guide on what exactly was edited as shown above in *figure 2.1* gave an excellent advantage with our team communication and coherence. In addition to this VS Code's built in debugger helped with testing and debugging without the need for an external debugger.

Thunder Client is a VS Code extension that simplifies testing APIs. This extension was used to test back-end API end points, thus refining the project even further.

## 2.2 Architecture

The architecture of HealthLink is mainly shown on the GitLab for how it is structured and the code within it.

Firstly, for the GitLab we had our main branch and then from there each team member had their own branch to work within before merging into the main branch. With this process in place any work which was done could properly be verified and tested before its merged to avoid any errors or accidental deletions.

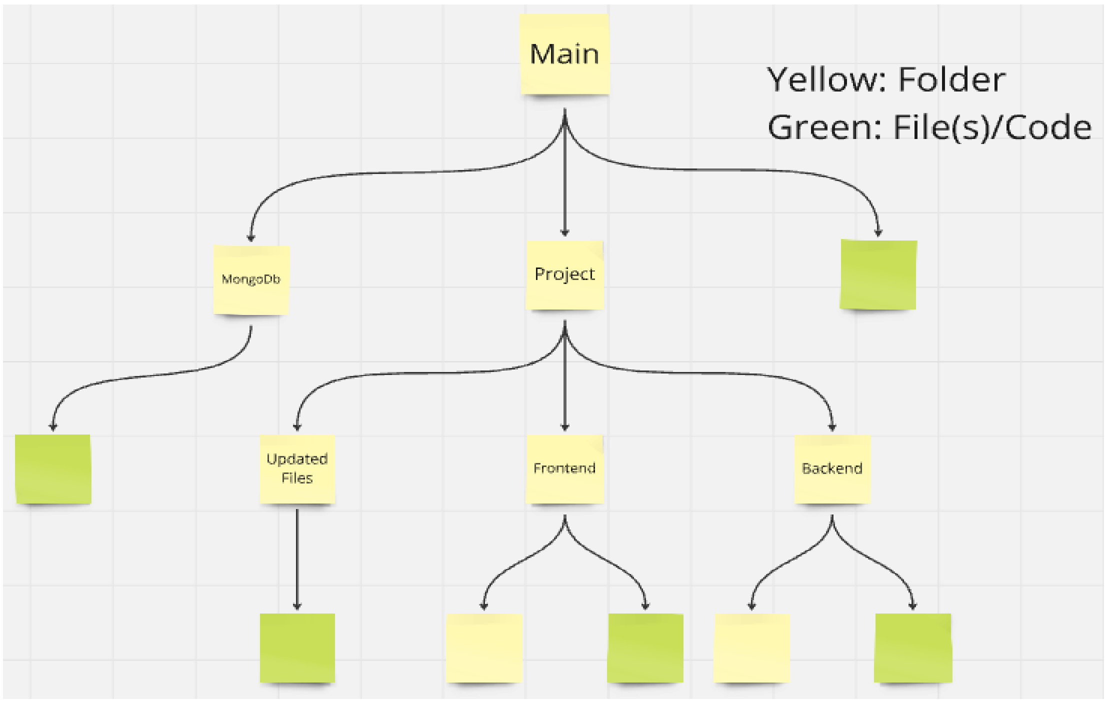


*Figure 2 Main Branch Architecture*

As shown above in *figure 2* for the main branch we had it set up in which there would be a folder “Project” where the code would be divided into “Front-end” and “Back-end” folders. Each with their own respective file structure. This made significantly easier to navigate to the area being worked on whether it was front-end or back-end.

With the implementation of ReactJS as mentioned above in *2.1 Technologies* the architecture of the code was heavily influenced by how ReactJS worked. In addition to this the code was divided into sperate folders named “components” to keep our coherence as shown below.

$$\textit{SignIn} > \textit{Components}$$

$$\left| > \left(\textit{Body} > \textit{Body.css, Body.jsx}\right)\right|$$

$$\left| > \left(\textit{NavBar} > \textit{NavBar.css, NavBar.jsx}\right)\right|$$

*Figure 2.1 SignIn Structure*

In reference to *figure 2, figure 2.1* would be found as one of the yellow boxes in the front-end branch. This type of file structure was heavily influenced by ReactJS as the main ReactJS file had to be in the same branch of the project as the other web pages. Although our familiarity with ReactJS was fresh, this was of storing the files was far more efficient to our earlier design.

Originally within the project folder there were only two folders, “Front-end” and “Back-end”. All relative files were stored in their respective folders. This created confusion on which files belonged to what web page. We then employed divider folders which take the name of a web page to avoid the confusion thus laying the foundation of the architecture we ended up with. Although this solved our problems on navigating the GitLab repository it created new errors with ReactJS not being able to connect each individual page to the main web page. Our solution to these errors was the architecture mentioned above in *figure 2.1*.

## 2.3 Testing

Throughout our development of HealthLink we had to conduct vigorous testing to achieve what we wanted. As mentioned above in *2.1 Technologies* our employment of Vite allowed us to visualise our changes instantly. This was our main form of testing when it came to designing as being able to see what the code added looked like was much more efficient than having to relaunch the webpage every time. This placed Vite not only as a core asset to our development but also as a tool to be taken with us in future development.

With Vite being our main method of testing promptly for front-end design and back-end queries we utilised Thunder Client for our back-end end points testing. As mentioned above in *2.1 Technologies* Thunder Client is an extension for VS Code that serves as an API client for the purpose of testing and debugging. Thunder Client being an extension for VS Code allowed for it to be easily implemented into our tool set for testing.

For the patient information form a fellow student acted as a user to test if the forms worked correctly with different information. This style of testing was used as it allowed for a person outside the development team to use the application without any expectations or knowledge on the code behind it. As well as this the development team also acted as users to test the patient information form. This testing fixed errors such as the form allowing for nothing to be submitted and the form accepting characters as a date of birth.

Visual Studio's built in debugger was used an extensive amount for the debugging of back-end code most notably the early on "backend.js" code for submission, table schema and database connection.

Peer reviews and walkthroughs were a method of testing used in our weekly meetings on Wednesday and Friday to try fix errors and to check code coherency. This was done by either reading out the code line by line to another group member or by allowing another group member to read the code. This was an effective method of testing as it not only fixed errors but also allowed for team members to see how and what other members were doing in detail and to make sure code was being done in a way that was compatible with the rest of the project/area of the project being developed.

////////////////////////////////////////////////////////////////////////////////////////////////////////////////

# Chapter 3: The SCRUM process

## 3.0 Introduction to SCRUM

SCRUM is an Agile software development framework that focuses on teamwork and adaptability. SCRUM allows teams to manage projects by breaking large tasks into smaller, more manageable pieces of work called sprints. Each sprint produces a version of the product increasing in usability ech time, giving the team time to adapt to changes and focus on delivering what matters most to its potential users. SCRUM encourages cooperation, accountability continuous improvement, making this approach to group projects perfect for software developments projects such as this one.

In our project we utilised Scrum to create the HealthLink app, designed to manage patients' medical information. HealthLink lets users store their medical history, prescriptions and personal health data in one place in a secured manner saving time during doctor and clinic visits due to individuals not needing to give the same details again and again.

Using SCRUM, we organised our work into sprint, with each sprint focused on delivering specific features of the app, such as user registration, data encryption for secure information storage and a user-friendly interface in which healthcare professionals could access whatever patient information they required. The iterative nature of SCRUM allowed us to incorporate feedback from potential users early in the development process to ensure that the app met real word needs. Regular team meetings such as weekly stand-ups improved our collaboration throughout the project and allowed us to adapt quickly and easily to challenges that came out way.

## 3.1 Initial Scrum Meeting: Story Map Creation, Releases and Sprints

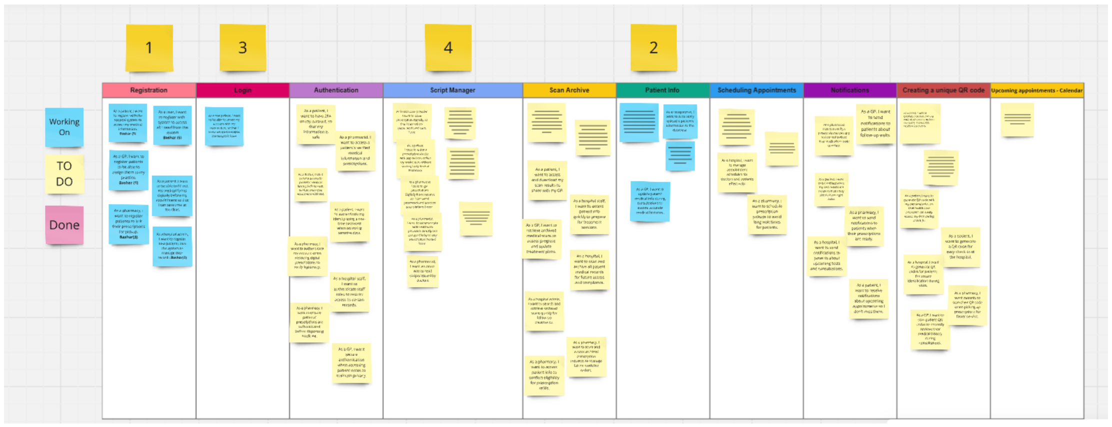
### 3.1.0 Story Mapping

A story map is a diagram that visualises the user's journey by listing key features and functionalities of a project. It helps the group organise development tasks of the project which are split up based on potential users' needs and priorities. By matching features to what different users need, the story map helps keep the development process on track. It makes sure the team focuses on what really matters to users, while also spotting any potential challenges and planning out sprints in a clear, organised way.

In the HealthLink App, the story map was designed based on inputs received from team discussions and brainstorming sessions. We decided the major user roles for our app would be patients, clinic staff, pharmacists and consultants such as doctors. The story map prioritises features in order of importance to enhance the user experience.  Patients would need to create and manage personal health records, complete forms electronically, access their prescriptions and be able to create a unique QR code to share this data electronically. Clinic staff would need to schedule appointments, access medical history and manage patient records. Pharmacists would need to verify prescriptions sent from consultants and doctors, securely manage patient information and link records to patient profiles. Consultants would need to access patients' medical history, manage appointments, create prescriptions and be able to forward them onto the pharmacy.

We decided to divide features into high and low priority. High priority features included: registration, login, patient information, authentication while low- priority features included archival scanning and prescription management.

We organised our story map as follows with columns representing user activities such as registration, login and prescription management. Blue notes indicating what we were working on at the time. Yellow notes indicating what still needed to be done. Pink notes indicating what was completed and the large notes at the top indicating the order in which we tackled each case e.g. we planned to start with registration, secondly go to patient information and thirdly tackle login.



## 3.1.2 Releases and Sprints

In SCRUM, each release is focused on delivering small, valuable improvements to users. The development of the application was divided into four two-week sprints to ensure steady progress.

Sprint Plan

During sprint one we planned to tackle the core functionalities of the application. We aimed to start user registration and login, basic authentication for security and QR code generation for data sharing. Sprint two was aimed at integration and workflow optimisation including patient medical history saving and scanning and integration with pharmacy systems. For sprint three we wanted to get started on advanced features such as notifications for appointments and prescription pickups. Sprint four would be starting prescription management if we had time for combining all our code.

Sprint Details

Sprint 1: Initial Setup of Login and Registration

This sprint's objective was to develop the basics of login and registration and perform initial code testing. The tasks we set out wert to design the initial UI for login and registration, implement basic secure password storage via a database as well as to develop a system for validating user inputs.

Sprint 2: Finalise Login and Registration

At the end of sprint one we realised we had underestimated how long each task would take. We therefore had to commit sprint two to finalising the login and registration systems and hopefully advancing some features. The task in this sprint included ensuring database integration for user registration data as well as conducting end- to – end testing of the registration and login process.

Sprint 3: Patient Information and QR code

For this sprint we planned to build the front – end and back – end for patient information management and enable QR code functionality. We aimed to design and implement the front-end for patient data input and displaying this for users, setting up back-end APIs for securely storing patient information and adding functionality to generate and scan QR codes for data sharing.

Sprint 4: Prescription Managment and combining workload

We firstly aimed to merge our code for the login, registration and patient information together to get the app up and running and secondly begin prescriptions management to allow pharmacists and healthcare providers to manage prescriptions digitally. Tasks for the final sprint included developing a model for adding and updating prescriptions, designing the UI for pharmacists to process, verify and manage prescriptions and lastly conduct end-to-end testing of the prescription management system, including integrating clinic and pharmacy systems.

Conclusion

We found that the story map helped us to divide up the work into sprints. It displayed key feature vertical slicing of registration, login, QR code generation, prescription management, scanning and archiving key patient information and prescriptions,

By following the SCRUM framework, we were able to develop HealthLink in a way that allowed us to stay flexible and deliver meaningful improvements with each release. Story mapping, clear release plans, and well-defined sprint goals helped us stay organised and ensure that every step of the process was focused on delivering real value to users. This approach kept our development process efficient and on track.

## 3.2 Project Estimation: Planning Poker

### 3.2.0 Introduction to Planning Poker

Planning Poker is a team-based estimation method used in Agile to figure out how much effort or time a task might take. The team works together, shares their thoughts, and comes to an agreement on the estimate, making the process more accurate and collaborative. This approach makes sure that everyone in the group understands the amount of time and effect required for each talk by allowing team members to provide their own estimates followed by discussions to reach a mutual consensus. We decided to choose hours as our estimation value.

There are many benefits of planning poker including how every team member contributes their estimates, promoting equal participation and preventing any one person’s opinion from suppressing others. We also found that planning poker reduced bias as independent estimates are provided before group discussions. Finally, we found that it encouraged alignment amongst team members, encouraging mutual understanding of the task's requirements.

### 3.2.1 Process and Application of Planning Poker

Our team used Planning Poker to estimate the effort required for each task related to the HealthLink App development. We used Miro, an online collaboration platform for discussions and estimators. This tool allowed all our team members to input their estimates anonymously and then visually compare them. A detailed task with full descriptions of each user story was then shared within the team for reference during estimation.

The team reviewed each task to make sure that we all understood what was included in all of them. Tasks included:

i. Registration
ii. Login
iii. Authentication
iv. Script Manager
v. Scan Archive
vi. Patient Info
vii. Scheduling Appointments
viii. Notifications
ix. QR Code Generation
x. Upcoming Appointments (Calendar)

Each team member discretely selected their estimate for a task's effort. After all the estimates were revealed, team members discussed the reasoning behind their numbers, and we addressed any significant differences. Using hours as our unit of measurement, those with higher or lower estimates shared potential challenges or simplifications that influenced their estimates. A second round was estimation was conducted after discussions about why everyone chose specific estimates to try to achieve a smaller variance. Final estimates were agreed upon after considering all viewpoints. For example, Random estimated the Registration task would take 30 hours to complete whereas Hunain estimated it at 50 hours. It was finalised at 40 hours of deliberation.
Solidified estimates:

| TASK | FINAL ESTIMATE (HOURS) |
|---|---|
| Registration | 20 |
| Login | 20 |
| Authentication | 35 |
| Script Manager | 50 |
| Scan archive | 20 |
| Patient Information | 40 |
| Scheduling Appointments | 10 |
| Notifications | 15 |
| QR Code Generation | 10 |
| Upcoming Appointments | 25 |

We found that we encountered some challenges while using planning poker. One challenger had unclear requirements. The ‘create a unique QR code’ task initially lacked clear boundaries making it difficult to estimate. To fix this we broke the task up into smaller subtasks. This included research, generating the QR code and integrating it into the system. Estimates were then provided for each of these subtasks and then added together.

As our team includes people from lots of different courses ranging from Computer Science and Software Engineering, Multimedia Design to Quantitative Finance, we all have different levels of front-end and back-end coding experience and understanding. We also had an initial wide range of estimates based on our knowledge of the different subjects. Having open discussions about our different estimates allowed us to align these perspectives, resulting in a common understand of the complexity of each task.

The ‘notifications’ and ‘scheduling appointments’ features shared some overlapping functionalities. Although we did not get to finish this part of our project, if we continue this application in the future, it is something we would make sure to take note of as we could integrate some previous code so we wouldn't have to start from scratch when creating updated features.

## Conclusion

Estimated hours guided task allocation across the four sprints, making it easy to divide the work up amongst ourselves without overwhelming any one person at any time. Planning poker also helped the team identify high and low- priority tasks. It proved to be an invaluable tool for estimating task difficulty and effort required in a collaborative way. It allowed us to identify potential challenges early on in the project. This encouraged mutual understanding of task requirements and ensured realistic sprint planning. By using this method, we were able to structure a smooth and efficient development process for the HealthLink app.

## 3.3 Meeting Minutes

### 3.3.0 Purpose of Meeting Documentation

Meeting minutes are a critical component of the SCRUM methodology. They serve as a formal record for discussions, decisions and progress updates. Minutes ensure alignment within the team.

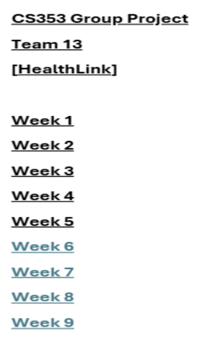
One of the benefits we found from documenting our meetings using minutes was tracking decisions. Meeting minutes allowed the team to capture and recall key decisions such as prioritised tasks and agreed timelines. Clearly defining every team member’s responsibility helped everyone understand their individual tasks. We found that it helped us monitor our progress each week against the sprint backlog and we already had basis for adjusting plans and sprints when we needed to. Lastly, we found that they kept all team members well informed. Anyone who was absent found the notes very helpful in terms of catching on discussions and staying up to date with the rest of the team.

### 3.1.1 Format and Content of Metting Minutes

To ensure clarity and consistency, the team followed a standard format for documenting meeting minutes. Each entry typically included:

- Date: The date of the meeting.
- Participants: Names of attendees and their respective roles.
- Agenda: Topics discussed during the meeting.
- Key Decisions: Summary of agreements reached during the meeting.
- Action Items: Tasks assigned to team members, along with deadlines.

We also created a link to each week's minutes to ensure easy access to each week so that we did not have to scroll and could simply click on each week we wanted to access, see below:



CS353 Group Project
Team 13
[HealthLink]

Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
Week 7
Week 8
Week 9

## 3.1.2 Key Decisions from initial meetings

Week one commenced on 02/10/24, which focused on our project idea. After brainstorming ideas such as a Vinyl app and a budgeting app, the team decided to develop a healthcare app called HealthLink to simplify medical information management. We prioritised features such as QR code functionality to reduce paperwork, integrating patient, clinic and consultant accounts for seamless data access as well as an overall eco -friendly design to eliminate the need for paper prescriptions. Our plan was to research real-world applications and gather feedback from pharmacists and receptionists.

Week two on 09/10/24 contained the initial story map, sprint planning and planning poker. We created a shared story map on Miro to visualise task priorities. We used planning poker to estimate task durations such as registration 40 hours and login 40 hours.

On Week three on 16/10/2024 we decided to split the team into front-end and back-end teams with Randon and Mac working on the wireframes, Hunain, Bashar and Vladislav worked on the back end for registration and login while Megan and Conall began the back end for patient information. During this sprint we decided on MongoDB to be our database and React for front-end frameworks. We also created a GitLab group, and all team members were added.

### 3.1.3 Teams' experience and Conclusion

We created detailed minutes each week such as these. Maintaining meeting minutes proved essential in keeping the team aligned and ensuring transparency throughout the group. The following outcomes were directly attributed to detailed and structured meeting documentation.

Previous meeting minutes helped track progress and monitor the completion of tasks such as setting up GitLab and creating wireframes.

Minutes identified areas where collaboration could be improved, such as aligning frontend and backend teams. An issue we had with this is one person was doing front-end and back -end code for patient information which was a heavy workload for one person whereas three people were working on registration and login for the backend. This in turn was too many people for one task as then there was more than one back – end code structure for login for example. This was also inefficient. Through Weekly scrums we were able to bring forward any issues such as someone feeling overwhelmed about their workload. We were able to align front-end and back-end teams going forward to make sure the project ran more smoothly and was more efficient.

Scrum retrospective discussions documented in the minutes allowed the team to adjust sprint timelines, balance workloads and set realistic future goals.

Well-structured and detailed meeting minutes played a vital role in keeping the team aligned with objectives and responsibilities. The minutes facilitated effective decision making, enhanced collaboration, and contributed to the smooth development of the HealthLink App

## 3.2 Burndown Chart

A burndown chart is a graphical representation used in SCRUM to track the progress of a sprint or project. It plots the remaining work measures in hours, tasks or story points against time, represented by days or sprints on the x-axis. These types of charts allow teams to monitor their progress toward completing the sprint backlog and to determine if they are on track to achieve their sprint goals. A burndown chart typically included an 'ideal progress line' which represents the planned rate of work completion as well as an 'actual progress line; depicting the actual progress of completed work. The objective of the 'actual progress line' is to closely follow the 'ideal line' signifying the team is on schedule and performing as planned.

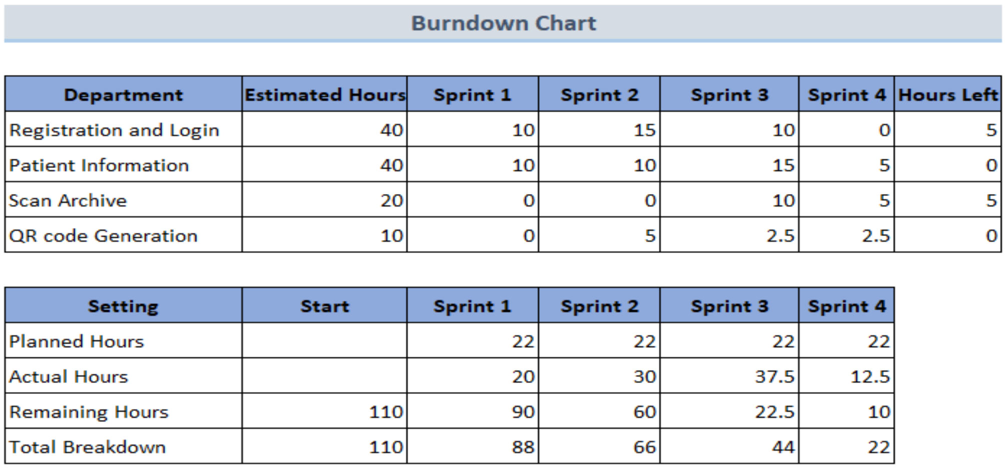
### 3.2.1 Creation and Usage

In order to create the burndown chart for our project we followed these steps. Firstly, we had to input the data. Task estimation was carried out using Planning Poker sessions, refined through team discussions. Tasks and their estimated hours were allocated as follows:

i. Registration and Login: 40 hours
ii. Patient Information: 40 hours
iii. Scan Archive: 20 hours
iv. QR Code Generation: 10 hours

These estimated hours were divided across four sprints based on task complexity and priority. We used spreadsheet software to create the burndown chart. Planned hours, actual hours completed, and remaining hours were updated and entered into the table during each sprint. We used bars for planned and actual hours per sprint and lines for remaining work, with a dashed line indicating ideal progress or 'total breakdown', The chart was able to show us deviations between planned and actual progress, allowing the team to identify whether they were lagging behind schedule or on track.

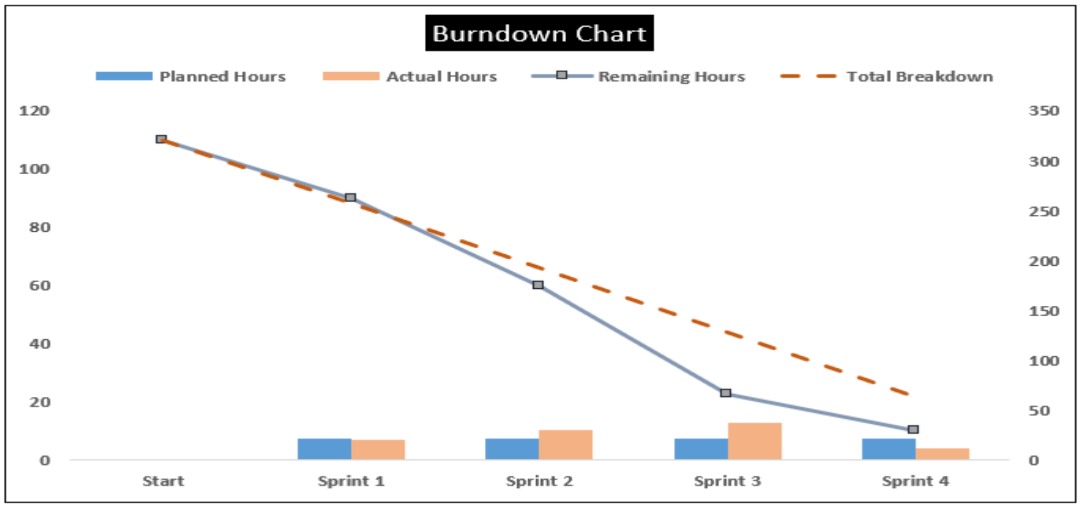
### 3.2.2 Visualisation



Burndown Chart

| Department | Estimated Hours | Sprint 1 | Sprint 2 | Sprint 3 | Sprint 4 | Hours Left |
|---|---|---|---|---|---|---|
| Registration and Login | 40 | 10 | 15 | 10 | 0 | 5 |
| Patient Information | 40 | 10 | 10 | 15 | 5 | 0 |
| Scan Archive | 20 | 0 | 0 | 10 | 5 | 5 |
| QR code Generation | 10 | 0 | 5 | 2.5 | 2.5 | 0 |

| Setting | Start | Sprint 1 | Sprint 2 | Sprint 3 | Sprint 4 |
|---|---|---|---|---|---|
| Planned Hours | | 22 | 22 | 22 | 22 |
| Actual Hours | | 20 | 30 | 37.5 | 12.5 |
| Remaining Hours | 110 | 90 | 60 | 22.5 | 10 |
| Total Breakdown | 110 | 88 | 66 | 44 | 22 |



3.2.3 Insights from the Chart

The burndown chart provided the following insights throughout the project:

1. Underestimation of Workload:
   In Sprint 3, actual hours (37.5) significantly exceeded planned hours (22), primarily due to the complexity of tasks such as Patient Information and Scan Archive. This spike highlighted an underestimation of the effort required, prompting the team to adjust their estimates for the following sprints. These changes resulted in smoother progress in Sprint 4.
2. Sprint Scope Adjustments:
   Tasks like QR Code Generation were redistributed evenly across later sprints after observing workload imbalances in Sprint 2 and Sprint 3. The team reduced low-priority items, such as additional features for QR Code Generation, to focus on completing core features like Registration and Login.
3. Team Efficiency:
   A steady decline in remaining hours across the sprints demonstrated improved team collaboration and alignment.
4. Blocker Identification:
   The chart highlighted points where actual progress diverged from planned progress. For example, in Sprint 3, delays occurred due to MongoDB integration issues and difficulties synchronising the front-end and back-end. Addressing these blockers early prevented similar issues in later sprints.

### 3.2.4 Conclusion

The burndown chart was important throughout our project in tracking progress, identifying underestimations or overestimations in workload and adjusting sprint plans accordingly. By comparing planned versus actual progress, the team ensured alignment on goals and made informed decisions to keep the project on track. This tool created a structured development process for the HealthLink app, enabling the team to achieve their sprint goals effectively.

### 3.3.0 Refining the Process of Story Maps

The creation of the story map for the HealthLink App began with brainstorming sessions that highlighted core features and user roles. The initial categories included Registration, Login, Prescription Management, QR Code Generation, and Patient Information. These categories were designed to align with the primary user roles: Patients, Clinic Staff, Pharmacists, and Consultants. Refining the story map was an iterative process conducted during regular SCRUM meetings. The refinements included:

1. Integrating User Feedback:
   Insights from potential users, such as pharmacists and healthcare staff, led to adjustments ensuring the app met real-world needs. We used User Acceptance Testing here. Megan was able to connect with two pharmacists who both agreed that a singular place to store patients' information and prescriptions would less time consuming and eco-friendly compared to the current method of printing, faxing and manually storing prescriptions. Conall, who has seen first-hand that getting patients to fill out multiple forms upon each visit to a clinic found that this was inefficient, and a better method had to be found.

2. Prioritisation Adjustments:
   Tasks were reprioritised based on technical difficulty and immediate value to users. Lower-priority features, such as QR Code Generation, were broken down into smaller subtasks and distributed across sprints with lighter workloads.

3. Decomposing Tasks:

Large features, such as Patient Information, were decomposed into subtasks, including the creation of data input forms such as the ‘registration’ and ‘login’ features and implementing data security measures.

### 3.3.1 The Evolution of the Story Map During Development

The story map evolved significantly during development, reflecting the team’s deeper understanding of the project’s scope and technical constraints. Key developments included:

1. Addition of Details:

   Early changes to the story map outlined broad tasks. Over time, details were added, such as the required fields for registration (e.g., name, email, and password) and the specific roles for different users (patients, clinics, and pharmacies).

2. Incorporation of Blockers:

   Challenges such as MongoDB integration issues and front-end-back-end synchronisation were addressed by refining the story map. Tasks were divided into parallel tracks for front-end and back-end development, enabling the team to tackle issues systematically.

3. Alignment to Sprint Goals:

   The story map was aligned with the sprint structure to ensure an efficient workflow. For example, high-priority stories like Registration and Login were completed in the first sprint. Features such as Scan Archive and Patient Information were deferred to later sprints.

### 3.3.2 The Role of the Story Map in aligning development goals

Story maps played a critical role in aligning the team on development goals and ensuring a clear and organized process. Story maps provided a comprehensive overview of the project scope, so that team members could understand how individual tasks contributed to the app's overall functionality. By organising activities according to user roles and priorities, the story map promoted collaboration between front-end and back-end teams. The story map helped define sprint backlogs, categorising features based on urgency and complexity. This ensured the team worked efficiently toward achieving consistent progress.

### 3.3.3 Conclusion

The ongoing refinement of story maps enabled the team to adapt to new challenges, integrate feedback, and maintain alignment on development objectives. This process ensured that the development of the HealthLink App was well-structured, user-oriented, and aligned with real-world needs.

## 3.4 Testing

Each person had access to run the project on their own laptops and to test their new features that they implemented. This was done through their own branch. Once branches were merged to the main branch on GitLab, everyone had access to test the code themselves and see if it might have impacted any development they were working on and already merged to the main branch. Testing of code as a group would take place after the Stand-up portion of the SCRUM meetings.

## 3.5 Version Control

Version control in software development is applied to organize, control, keep track of different versions of the software, and keeping a history of changes of the software being developed. This is particularly helpful when there are multiple members in the development team working on different sections of the application. This also allows members to experiment with the project without the worry of unintended errors they might produce which could potentially damage the progression of development. For our version control policy, we were provided access to GitLab via the University. We set up repository in the main branch and created a branch per person stemming from the main. Each person was allowed to pull updates from the main branch into their own and once they completed a feature, they would commit their progress to their branch and merge the branch with the main for the progress to appear in the main branch. This gave us room to play with the repository in our own branches without the fear of breaking the main branch.  Any merge conflicts that did occur was addressed at our next meeting.

## 3.6 Future releases and sprints

Future sprints would include enhanced security. If we were to continue on with the project, we would love to implement biometric login options such as fingerprints or facial recognition. We would be able to create data analytics providing monitoring tools for healthcare providers to track patient trends and lastly enhance compatibility with additional pharmacy systems. UAT feedback emphasized the need for faster prescription access and improved data privacy, shaping future development priorities. However, implementing biometric security and integrating new APIs could require additional expertise.

## 3,7 Team communication and management

The main tools we used for communication were Microsoft teams for announcement. We also used Miro for collaboration through story mapping and planning poker. We used GitLa for our code. We all had our own individual branch and were able to commit our code to the main branch when it was ready and approved by all members of the group.

The team adapted well to these tools, though an over-reliance on Miro for documentation such as the user stories at the start of the project created in efficiencies as we found it got messy quickly and switched to Microsoft word for taking the minutes. Weekly stand-ups and retrospectives improved alignment and productivity.

## 3.8 Remote Working and AI Code generation

Remote collaboration created challenges such as misaligned schedules and delays in feedback loops due to different timetabling schedules. We mitigated these issues by creating clear deadlines, flexible updates and flexible meeting schedules. Anytime someone wasn’t able to attend an additional meeting during the week, they were able to attend via a Microsoft teams call.

GitHub Copilot and ChatGPT assisted in generating boilerplate code and debugging. This reduced development time, particularly for repetitive tasks. However, there were some limitations as it required human oversight to ensure code quality.

## 3.9 Conclusion

The SCRUM framework provided a structured yet flexible approach to developing the HealthLink app. Key lessons included the importance of regular feedback loops, adaptive sprint planning, and robust testing practices. These insights not only ensured a functional app but also prepared the team for future Agile projects.

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

# Chapter 4: My Contribution

As I talk about my main addition to the development of the HealthLink program in this part, I will talk about what I did. Some of the things that were done were making wireframes, doing in-depth study on front-end frameworks, trying the Patient Registration Page in different development settings, and making an overview presentation of the project.

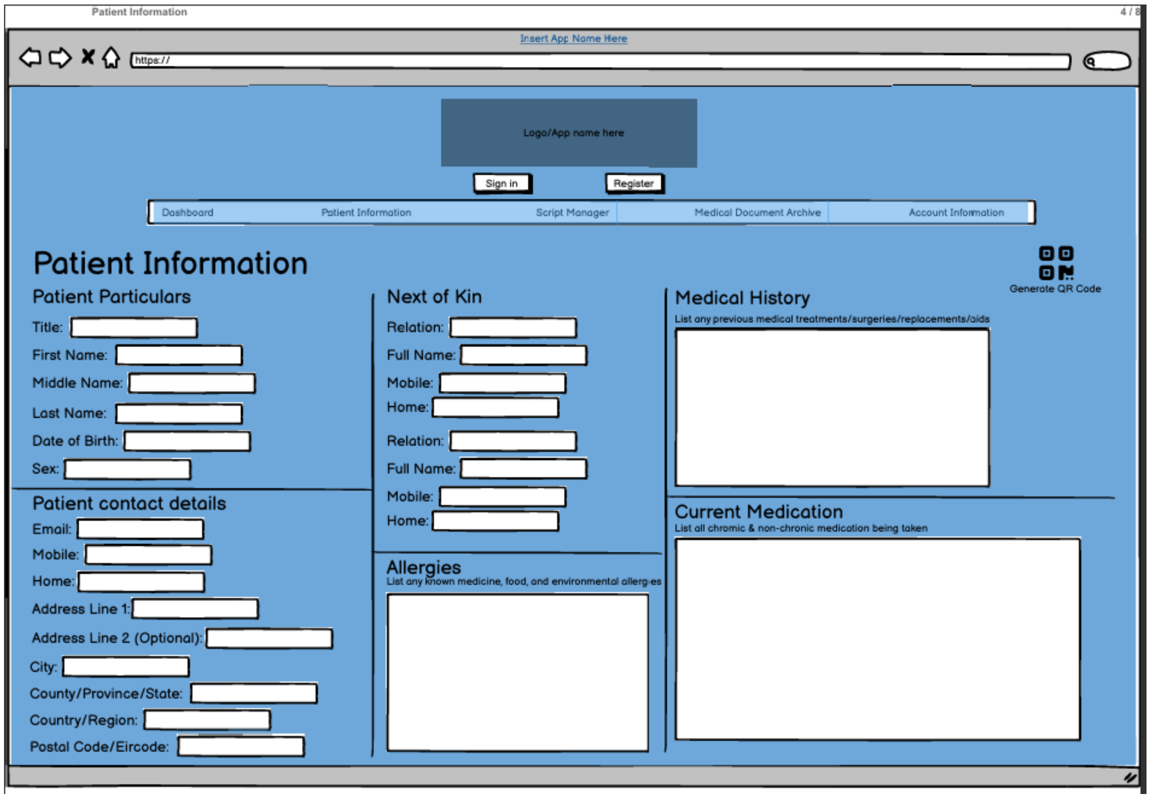
## Wireframes Development

The first wireframes for the HealthLink app were made by me and Randon working together at the preparation of the project. Wireframes are an important part of building a user-centered design. These versions helped us see how the app would be laid out and make sure that the user interface (UI) was easy for all stakeholders—patients, advisers, and pharmacists—to understand and use.

We made the wireframes with Balsamiq, a tool for designing interfaces together. The main pages had the following:

1. The Home Page has easy-to-use browsing and a summary of what HealthLink can do.
2. Patient Registration Page: This page lets users enter their personal information, protected login information, and medical information.
3. Prescription Management Page: This page is for pharmacists to check medications and keep track of them.
4. A "dashboard" is a central place where users can view their info and take the steps they need to.
5. The "Appointment Scheduling Page" makes it easier to have meetings with doctors and hospitals.

With each new version of the design, team talks, and virtual user experiences were used to get feedback. This iterative method made sure that the project goals were met, and that performance was better. Accessibility standards were carefully considered to make sure that the design could work for people with different levels of technical knowledge.


Patient Information 4 / 8
Insert App Name Here
https://
Logo/App name here
Sign in Register
Dashboard | Patient Information | Script Manager | Medical Document Archive | Account Information

Patient Information
Generate QR Code

Patient Particulars
Title:
First Name:
Middle Name:
Last Name:
Date of Birth:
Sex:

Patient contact details
Email:
Mobile:
Home:
Address Line 1:
Address Line 2 (Optional):
City:
County/Province/State:
Country/Region:
Postal Code/Eircode:

Next of Kin
Relation:
Full Name:
Mobile:
Home:
Relation:
Full Name:
Mobile:
Home:

Allergies
List any known medicine, food, and environmental allergies

Medical History
List any previous medical treatments/surgeries/replacements/aids

Current Medication
List all chronic & non-chronic medication being taken

## Research on AngularJS vs. ReactJS

I looked at both AngularJS and ReactJS side by side to find the best framework for HealthLink's front end.

### AngularJS

Google takes care of AngularJS, which is a complete platform for building dynamic web apps. It has built-in tools for data linking and dependency injection that cut down on the code that needs to be written. The Model-View-Controller (MVC) design of AngularJS makes it good for large-scale apps.

Advantages:

- Two-way data linking makes sure that the model and view are always in sync.
- Gives you a full structure with strong tools and features.
- Great data and help from the community.

Cons:

- It's harder to learn because of complicated ideas like directives and dependency injection.
- Performance problems with big, changing apps because they need a lot of processing power.

**ReactJS**

ReactJS is a JavaScript tool for making UI elements. It was made by Facebook. Through virtual DOM and a component-based design, it works on giving users a quick and dynamic experience.

Pros:

- Using the virtual DOM for efficient processing.
- Code can be used more than once with component-based design.
- A large community and the ability to work with many tools.
- Scalability for apps that need to be updated often.

Cons:

- It needs extra tools for managing routes and state.
- It can be hard for beginners to get the hang of JSX code.

Making a Choice:

ReactJS was picked because it met the needs of the project. It worked better with

HealthLink's goals of smooth user experience and growth because it was efficient, flexible, and had a large community. Also, ReactJS's component-based structure made it easy to make UI components that could be used again and again, which was a key part of the app's flexible design. Its fame and support from the community made our decision even stronger, giving us access to a huge number of tools and updates.

## Testing the page for registering patients

I was in charge of checking the Patient Registration Page, which is an important part of getting new users started. During the testing process,

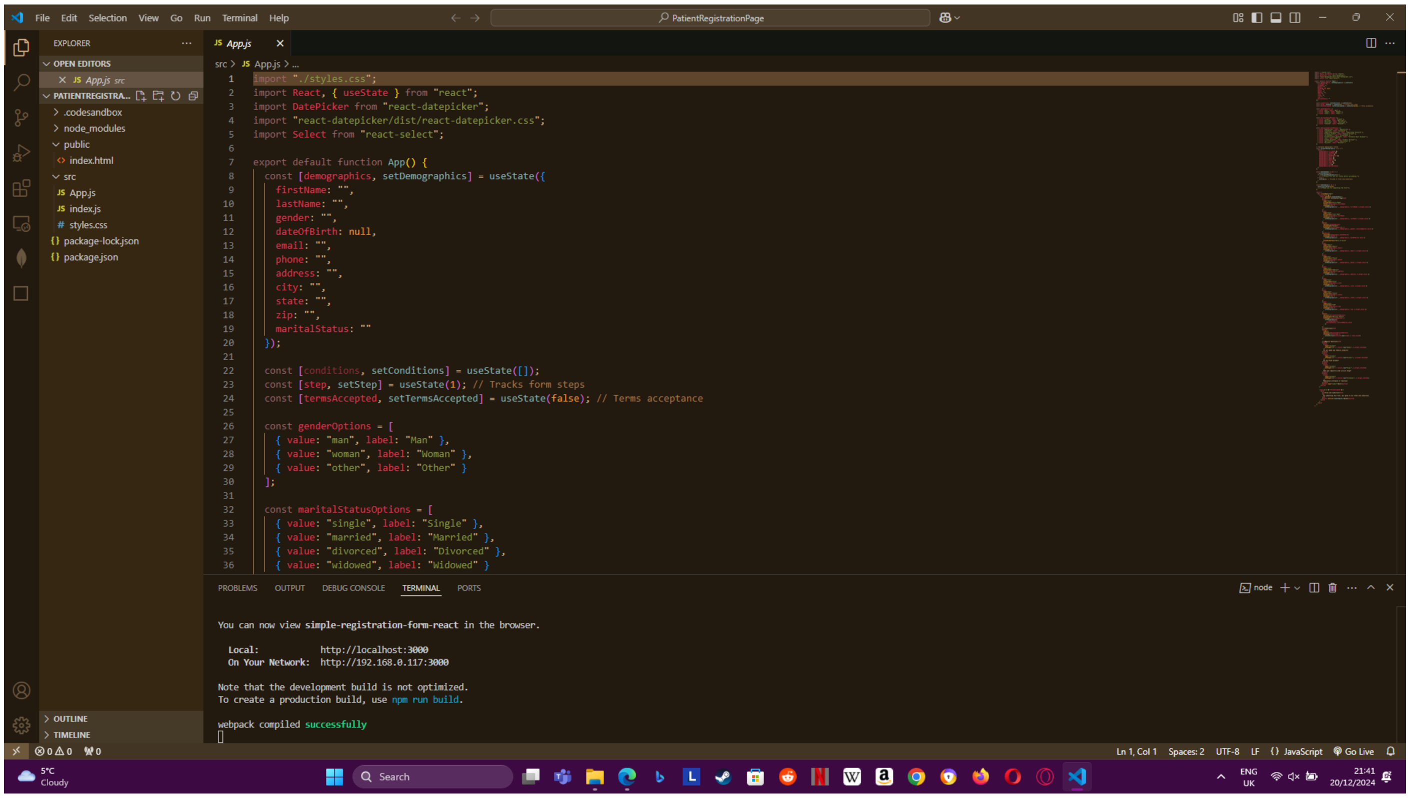
1. Development Environments:

- VSCode is used for deeply analyzing and combining with other parts. Breakpoints, stack traces, and other debugging tools were very helpful in finding and fixing problems.
- CodeSandbox: Perfect for quickly trying separate parts without a lot of setups. This let them make prototypes and fix bugs quickly while the software was being built. I learned this in my other module CS385 (Mobile Web Development).

2. Real-life test cases:
- Making sure that necessary inputs are contained in form areas (like name, email address, and password).
- Making sure that errors related to inappropriate data entries, like fields that are empty or forms that are wrong, are handled correctly.
- Testing ways to store and get info through API addresses.
- Making sure it works on all devices and websites to make sure it's always compatible.

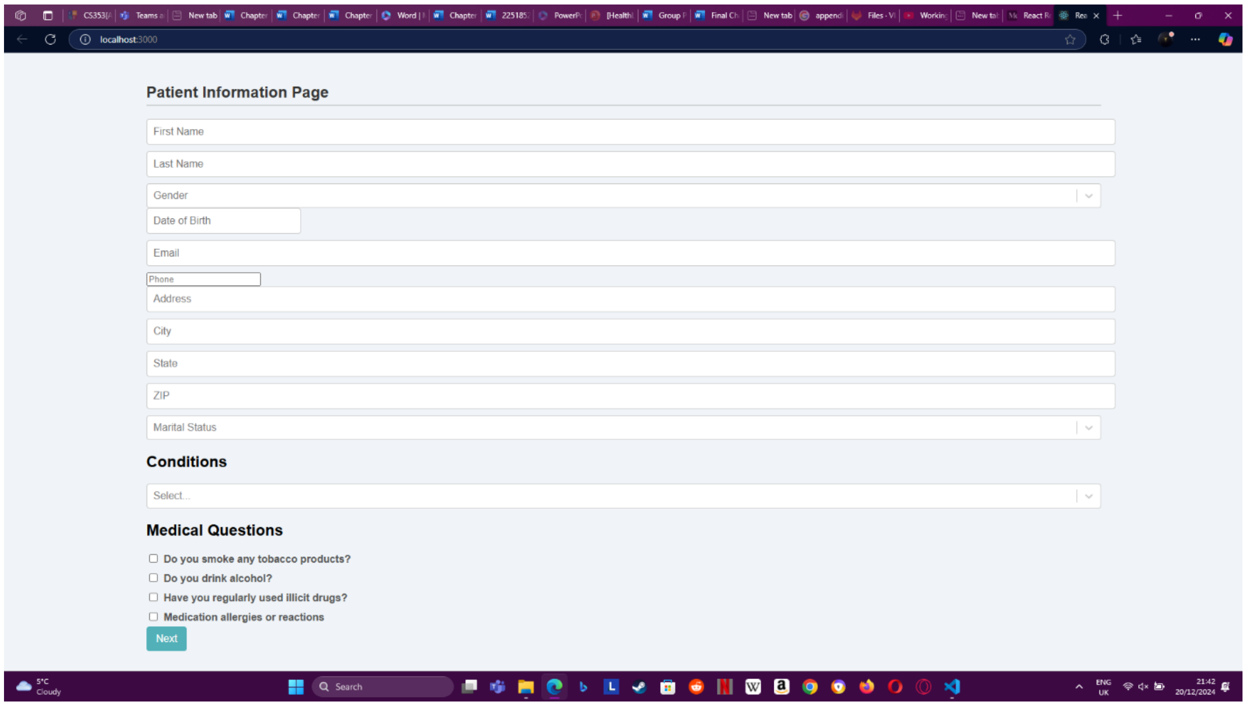
3. Tools and Methods for Testing:
- Used Chrome DevTools to keep an eye on API calls, look at DOM parts, and fix problems on the front end.
- Unit testing with Jest was used to make sure that React components worked as expected and that they were fully functional.
- Held physical testing meetings with team members to make sure the form was error-free and easy for people to use in the real world.
- Stress testing the form by sending in large amounts of data and trying to register multiple

users at the same time was also part of the testing. This showed where improvements needed to be made, like how the information was handled and how quickly the front end responded.



```
import "./styles.css";
import React, { useState } from "react";
import DatePicker from "react-datepicker";
import "react-datepicker/dist/react-datepicker.css";
import Select from "react-select";

export default function App() {
  const [demographics, setDemographics] = useState({
    firstName: "",
    lastName: "",
    gender: "",
    dateOfBirth: null,
    email: "",
    phone: "",
    address: "",
    city: "",
    state: "",
    zip: "",
    maritalStatus: ""
  });

  const [conditions, setConditions] = useState([]);
  const [step, setStep] = useState(1); // Tracks form steps
  const [termsAccepted, setTermsAccepted] = useState(false); // Terms acceptance

  const genderOptions = [
    { value: "man", label: "Man" },
    { value: "woman", label: "Woman" },
    { value: "other", label: "Other" }
  ];

  const maritalStatusOptions = [
    { value: "single", label: "Single" },
    { value: "married", label: "Married" },
    { value: "divorced", label: "Divorced" },
    { value: "widowed", label: "Widowed" }
```

```
You can now view simple-registration-form-react in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.0.117:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```



localhost:3000

**Patient Information Page**

First Name

Last Name

Gender

Date of Birth

Email

Phone

Address

City

State

ZIP

Marital Status

**Conditions**

Select...

**Medical Questions**

- ☐ Do you smoke any tobacco products?
- ☐ Do you drink alcohol?
- ☐ Have you regularly used illicit drugs?
- ☐ Medication allergies or reactions

Next

## Things that will be in the final application

## Several of the features I worked on made it into the final version of the app:

- Secure Registration System: Made sure that the page for registering users correctly caught and verified their information. Encrypted private data in line with best practices in the business.
- User Feedback Mechanisms: Integrated tools for dynamic error messages and real-time form validation, which make the experience of the user better. This included live comments on how strong the password was and how the entry was formatted.
- If you want responsive design, Made the UI work better on different screen sizes so it can be used on computers, tablets, and phones. This had to be tested thoroughly using mobile design tools and simulations.
- Better Accessibility: Added ARIA (Accessible Rich Internet Applications) roles and labels to make the site easier for people who use assistive tools to use.

## Project Presentation

[HealthLink]

CS353 Project Overview

Team 13

Tired of always filling in forms at the Doctor's rooms?

I made the final presentation overview for the group. The point of this lecture was to summarize:

- The project's goals are to show how important HealthLink is to current healthcare.
- Technical Achievements: describing the tools and methods used, such as why ReactJS and MongoDB were chosen.
- User Journey: Showing how clinicians, patients, and pharmacists use the app together in a smooth process.
- Dealing with problems like merging clashes and design changes is what "Challenges and Solutions" is all about. I talked about how we solved problems with React's component structure and state management as an example.

Key features, like registering users and making appointments, were also shown in real time during the talk. The technical answers were helped by pictures and screenshots of wireframes and code snippets. This all-around method made sure that parties, both technical and non-technical, could understand how the project would affect them.

////////////////////////////////////////////////////////////////////////////////////////////////////////////

# Chapter 5: Summary

The process of making HealthLink was thorough and educational, and it has had a long effect on my career and personal growth. This chapter looks back at the process of making HealthLink, looking at what went well, what went wrong, and what was learned. A lot was learned about joint software development, Agile methods, and how technology can be used to solve problems in the real world through this project.

## Success and Achievements in the Project

One of the biggest successes of the project was putting HealthLink's main features into action, like registering users, managing medical data, and adding the new QR code. These features fixed real-life problems with the way healthcare was done, making a system that worked better for patients, doctors, and pharmacies. We were very smart to choose ReactJS for the front end and MongoDB for the database. These tools gave us the flexibility, speed, and community help we needed to meet the needs of the project.

The SCRUM framework was very important for setting up the project. Because SCRUM is ongoing, comments and improvements could be made all the time. This made sure that the app met user needs. The fact that our team was able to change and improve based on sprint reviews showed that this method worked. Before the project was over, the team had made a prototype that worked and could be improved in the future.

## Working Together and How Teams Work

One of the best things about the project was working with people from different backgrounds. Each person brought their own skills and ideas to the table, which made the growth process better overall. Communication and teamwork tools that work well, like Microsoft Teams and GitLab, were very important for keeping track of tasks and being open.

The team members took turns being the SCRUM Master. This way, everyone had a chance to lead, which helped everyone feel responsible for the project and learn how to handle it. Weekly stand-ups and retrospectives made it easier for people to talk to each other openly, which helped find problems quickly and solve them. There were some problems, like plans that weren't lining up right and different levels of technical knowledge, but the team's dedication to helping each other and learning got them through them.

## Contributions and Learning on Your Own

I worked on the project by making wireframes, doing in-depth study on front-end frameworks, and putting the Patient Registration Page into action. It was especially useful to compare AngularJS and ReactJS to find the best framework for our needs. The study I did helped me learn more about things to think about when designing software, like scale, user experience, and community support.

By testing the Patient Registration Page, I got to use a number of different tools and methods, such as Jest for unit testing and Chrome DevTools for debugging. I also learned a lot about the concepts of accessible and adaptable design. These things have helped me get better at technology and made me value user-centered design even more.

Putting together the project's end presentation was also fun and showed how important it is to communicate clearly. To summarize scientific accomplishments and show off the application's features, I had to turn complicated information into easy-to-understand words, which is a skill that comes in handy in any work setting.

## Problems and Answers

It wasn't easy to finish the job. Managing the merging of front-end and back-end parts was a big problem that sometimes-caused delays. These problems showed how important it is to carefully plan and communicate with other teams. Story maps and regular SCRUM meetings helped solve these problems by making sure that goals were aligned, and relationships were made clear.

Another problem was that they didn't give themselves enough time to do some things, like adding features to QR codes and integrating databases. The Planning Poker estimate method helped the team change their goals and better use their resources in the next sprints. These events made me realize how important it is for project managers to use flexible planning and always be looking for ways to make things better.

## How it will affect future career goals

This project has had a big impact on my job goals because it gave me real-world practice with software development and project management. I've learned how to handle difficult jobs with ease thanks to using Agile methods, especially SCRUM. For future projects, I will use the skill of being able to match technology needs with customer needs obtained.

The experience also showed how technology can change the way we solve problems in the real world. Working on HealthLink made me even more interested in making apps that do useful things for people. The focus of the project on healthcare has made me want to look into more options in this field, where technology can make things more efficient, easier to get to, and better for patients.

## Thoughts on Working Together and AI Tools

Working with others on the team was a great way to learn. The different backgrounds and areas of knowledge made it easier for people to share their thoughts and work together. There were times when people on the team didn't agree, but their dedication to fair conversation and problem-solving made sure that differences were handled in a healthy way.

When AI tools like GitHub Copilot and ChatGPT were added, they helped a lot with writing standard code and fixing bugs. The development process was sped up and routine jobs were cut down, which freed up the team to work on bigger problems. But the experience also showed how important it is to have human review to make sure of the quality of the code and keep creative control.

## What I Learned and What I Hope For

When I think about the project, I remember a few important lessons:

1. Planning is Important: For handling complicated projects well, you need to plan carefully and make sure that everyone knows what their job is.
2. Adaptability: Being able to change your mind when problems come up out of the blue is what keeps growth and new ideas coming.

3. Continuous Feedback: Feedback loops that happen on a regular basis help both the product and the team do better.
4. User-Centered Design: Solutions that put user needs first are more successful and have a bigger impact.

I would suggest the following for upcoming projects:

- Giving testing and merging more time in case problems come up that were not expected.
- Improving the way writing is done to make sure it is clear and easy to find.
- Looking into other ways AI tools can be used to improve processes while still critically evaluating the results they produce.

## Final Thoughts

The process of making HealthLink was life-changing because it involved technical problems, working together as a team, and getting results that mattered. The project not only helped me get better at managing and using technology, but it also helped me learn more about how technology can be used to solve problems in the real world. When I think about the future, the things I've learned from this project will definitely affect how I make software, work with others, and come up with new ideas.

## References:

- https://www.agilealliance.org/agile101/
- https://www.infoworld.com/article/2334751/a-brief-history-of-the-agile-methodology.html
- https://www.atlassian.com/agile#:~:text=Whereas%20the%20traditional%20%22waterfall%22%20approach,at%20the%20heart%20of%20agile.
- https://agilemanifesto.org/iso/en/manifesto.html
- https://www.geeksforgeeks.org/agile-manifesto-for-software-development/
- https://www.atlassian.com/agile/scrum
- https://www.scrum.org/resources/what-scrum-module
- https://www.geeksforgeeks.org/scrum-software-development/
- https://www.coursera.org/articles/scrum-roles-and-responsibilities
- Best Budgeting Apps of December 2024
- Build a React Budget Tracker App – Learn React & Context API with this Fun Project
- https://leobit.com/blog/how-to-build-a-budgeting-app-opportunities-challenges-and-practical-tips/
- https://radixweb.com/blog/react-vs-angular
- Angular vs. React: A side-by-side comparison | Hygraph
- https://codesandbox.io/blog/announcing-vs-code-support-for-sandboxes
- https://youtu.be/ZJ1sNiTZw5M?feature=shared
- https://medium.com/@ericapantojacs/react-registration-form-d298b3b7e75d
- Scrum and Task Estimation slides on Moodle

### Screencast and Project Repo Supplier: VLADISLAV IVASKEVYCH

**vladislav.ivaskevych.2021@mumail.ie**

## Appendix:

Appendix: Supplementary Materials

Appendix A: SCRUM Framework Overview

The SCRUM framework was foundational to the HealthLink project. Below are the key elements of SCRUM implemented in the project:
- Roles: Scrum Master, Product Owner, and Development Team.
- Artifacts: Product Backlog, Sprint Backlog, and Burndown Charts.
- Ceremonies: Sprint Planning, Sprint Review, Sprint Retrospective, and Daily Scrum.

This structure enabled continuous feedback and iterative development.

Appendix B: HealthLink Technology Stack

The following technologies were used for the development of the HealthLink application:
1. Frontend: ReactJS, HTML, CSS, JavaScript.
2. Backend: NodeJS.
3. Database: MongoDB.
4. Development Tools: Visual Studio Code, GitLab, Thunder Client, and Vite.

Appendix C: Sample User Story Mapping

The initial user story map prioritized features for different stakeholders (patients, consultants, pharmacists):
- High Priority Features:
  - Registration and Login System.
  - Patient Information Management.
  - QR Code Generation.
- Low Priority Features:
  - Archival Scanning.

- Notification System.

Appendix D: Key Testing Methods

Testing was critical in ensuring functionality:
- Unit Testing: Conducted using Jest to validate React components.
- Integration Testing: Focused on API connections and database queries.
- User Acceptance Testing: Conducted with real-world stakeholders for feedback.

Appendix E: Planning Poker Estimation Summary

The Planning Poker technique facilitated effective task estimation:
- Registration: 20 hours.
- Login: 20 hours.
- Patient Information: 40 hours.
- QR Code Generation: 10 hours.

These appendices provide additional insights into the methodologies, tools, and frameworks that guided the successful development of the HealthLink project.