

UKZN HEALTH HIVE

|  |  |  |
| --- | --- | --- |
|  | **GROUP MEMBERS** | **STUDENT NUMBER** |
|  | **CHAHAL DHILRAJ** | **218027455** |
|  | **SIPHEPHELO MYENI** | **217014127** |
|  | **KEMERA MOODLEY** | **221109210** |
|  | **NEVANIA NAIDOO** | **217009198** |
|  | **NKOSINGIPHILE MBHELE** | **217010094** |
|  | **ARUSHAN NAICKER** | **220053537** |

Table of Contents

[UKZN HEALTH HIVE 1](#_Toc169560518)

[Abstract 3](#_Toc169560519)

[Introduction 3](#_Toc169560520)

[Software Development Methodology 5](#_Toc169560521)

[Agile Methodology 5](#_Toc169560522)

[Scrum Framework 7](#_Toc169560523)

[Software development team structure 9](#_Toc169560524)

[Project Management Plan 12](#_Toc169560525)

[Project Scope 12](#_Toc169560526)

[Project Objectives 13](#_Toc169560527)

[Work Breakdown Structure 14](#_Toc169560528)

[Project Schedule 17](#_Toc169560529)

[Software Development Approach 18](#_Toc169560530)

[**Object-oriented approach** 18](#_Toc169560531)

[System Version Control 20](#_Toc169560532)

[Human-Computer Interaction (HCI) Strategy Implementation 21](#_Toc169560533)

[Key Principles 21](#_Toc169560534)

[Strategy Components 22](#_Toc169560535)

[Implementation Plan 23](#_Toc169560536)

[Knowledge Management System 23](#_Toc169560537)

[Key Components and Achievements 24](#_Toc169560538)

[Software Testing Plan 24](#_Toc169560539)

[Objectives 25](#_Toc169560540)

[Scope 25](#_Toc169560541)

[Testing Phases 25](#_Toc169560542)

[Test Environment 27](#_Toc169560543)

[Test Data 28](#_Toc169560544)

[Test Execution 28](#_Toc169560545)

[Defect Management 28](#_Toc169560546)

[Reusability and maintenance 29](#_Toc169560547)

[Modular Design 29](#_Toc169560548)

[Adherence to SOLID Principles 29](#_Toc169560549)

[Utilization of Design Patterns 29](#_Toc169560550)

[Code Reusability Practices 30](#_Toc169560551)

[Maintainable Code Practices 30](#_Toc169560552)

[Version Control and Branching Strategies 30](#_Toc169560553)

[Refactoring and Technical Debt Management 30](#_Toc169560554)

[Dependency Management 31](#_Toc169560555)

[Documentation and Knowledge Sharing 31](#_Toc169560556)

[Performance Monitoring and Optimization 31](#_Toc169560557)

[Usability and Functionality of the System 32](#_Toc169560558)

[Conclusion 34](#_Toc169560559)

[Appendices 35](#_Toc169560560)

[References 39](#_Toc169560561)

## 

# Abstract

This academic defence document aims to highlight the utilization of the various tools, strategies, and methodologies during the software development of a system for the UKZN health science department. The system aims to automate a portion of the clinical administration tasks to record and manage patient data efficiently. During this project, the agile scrum methodology will be used to allow for tasks to be effortlessly allocated, and productive daily meetings to be conducted. For efficiency, our project will be supported by the following advanced tools: Microsoft SQL server that will serve as the database host for our project, ensuring data integrity and security, GitHub that will act as the version control for our project and assist in tracking all the changes as well as any modifications made in the course of working on the project, and finally Jira, that will be used for task organization, sprint management, and issue tracking.

In this way, the systematic approach established by the team ensures that the goal to work with the client to create a satisfactory and efficient system, within the desired specifications and to reach a positive result ultimately, will be upheld.

# Introduction

In today's digital age, the need for efficient and secure data management systems is paramount, particularly in the healthcare sector. Our group identified a significant gap in the College of Health Science’s current system, which heavily relies on manual processes for recording and managing clinical data. This not only posed challenges in data security and organization but also placed an undue burden on clinical administrators responsible for maintaining extensive paper-based records.

Dr. Urisha Naidoo, a respected faculty member, emphasized the critical need for a digital solution to streamline this process. Her vision was to empower students with a platform that would enable them to record patient data in a secure and organized way. This need for digital transformation became even more pressing within the College of Speech and Language Therapy, where clinical records were traditionally managed through a labor-intensive paper-based system. The clinical administrator faced considerable challenges in ensuring the accuracy, security, and accessibility of these records, often leading to inefficiencies and potential risks of data breaches.

Recognizing these issues, our group proposed the development of an innovative system designed to reduce the manual workload and enhance the management of clinical documentation. The proposed system aims to provide a structured and secure approach to storing patient data, thereby ensuring that records are not only easily accessible but also protected against unauthorized access. By transitioning from a paper-based to a digital system, we sought to modernize the administrative processes and support the clinical staff in delivering better care to patients.

The digital system aims to achieve several key objectives:

* **Efficiency**: Automating the data entry and management processes significantly reduces the time and effort required by the clinical administrator, allowing them to focus on more critical tasks.
* **Security**: Implementing advanced security protocols ensures that the patient data is protected from unauthorized access and potential breaches, maintaining patient confidentiality.
* **Accessibility**: Authorized personnel can access patient records from any location, enabling more flexible and responsive patient care and academic activities.
* **Organization**: A digital system provides a more structured approach to data management, facilitating easier retrieval and analysis of patient records. This enhances the overall workflow for both students and faculty.
* **Reporting and Analytics**: The system includes robust reporting features that allow for detailed analysis of patient data, disorders, and trends. These reports can help identify patterns, support research, and inform clinical practices and educational content.

**Innovations in Reporting**

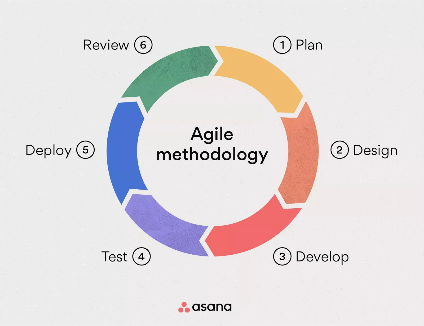
One of the standout features of our system is its comprehensive reporting and analytics capabilities. The system can generate detailed reports based on various criteria, such as patient demographics, types of disorders, and treatment outcomes. These reports provide valuable insights into patient trends and can highlight areas for improvement in clinical practices.

* **Patient Data Reports**: These reports can summarize patient demographics, visit histories, and treatment plans. Such reports are useful for both clinical and administrative purposes, helping to ensure that patient care is consistently documented and reviewed.
* **Disorder Analysis**: By categorizing and analyzing data on specific disorders, the system can identify prevalent conditions and track the effectiveness of different treatment approaches. This information is crucial for both teaching purposes and clinical decision-making.
* **Trend Identification**: The system's analytics tools can detect trends over time, such as increases in certain types of disorders or changes in patient demographics. Identifying these trends can inform resource allocation, training needs, and research initiatives.
* **Innovative Reporting Features**: Beyond basic reporting, the system can integrate data visualization tools to present data in more engaging and interpretable formats. Charts, graphs, and dashboards can provide quick insights and support data-driven decision-making.

Dr. Naidoo's endorsement and active involvement in this project highlight its importance and relevance to the College of Health Science. By developing this system, we aim to not only meet the immediate needs of the college but also set a precedent for other departments facing similar challenges. Our project represents a strategic effort to enhance the operational efficiency and security of the College of Health Science's clinical documentation processes. By leveraging technology, we aim to create a more organized and effective environment for both students and faculty, ultimately contributing to the overall improvement of healthcare education and patient care. The success of this system could serve as a model for broader adoption across the institution, promoting a more efficient, secure, and modern approach to managing clinical records.

# Software Development Methodology

## Agile Methodology



**Figure 1: Agile Methodology**

The methodology chosen by Control Alt Elite was an Agile-Scrum methodology. Agile and Lean Startup are two contemporary methodologies of interest when considering short (often less than one year) term software productions. Agile, along with its sister-like versions (Extreme Programming, Scrum, Lean Software Development, Rational Unified Process, and Kanban), embraces change rather than following a plan. It utilizes short iterations, eliminates waste, responds to change, and continually welcomes new requirements. Agile focuses on quick responses to change along with incremental development avoiding big bang specifications, designs, or coding; you instead work collaboratively (in tight-knit teams) with the customer to ensure that business values are upheld (Al-Saqqa et al., 2020). The Agile approach is about being able to prioritize quickly to deliver those things that will bring the greatest overall value to the company. The agile approach involves the customer in the requirements process so that the software can meet the needs of the company, its users, and the market opportunities. Although Agile is suitable for short-term projects, Agile is not a silver bullet for solving all software project issues, and major concerns about the use of Agile are still present in the software industry (Kasauli et al., 2021). Business as usual is over. The only way to avoid seeing the same things, sharing the same information, and understanding, and becoming creatively blind to the answers is to work in teams. Teams who research, ideate and make decisions together. The key to driving meaningful innovation of value creation and value fixation is concurrent trans-disciplinary ideation and decision-making.

Continuous and comprehensive quality systems are the foundation of ISO 9001. In the pursuit of narrative simplicity, the normal systems required to achieve quality have been more implicit than explicit. The adoption of the explicit ISO 9001 requirements, such as maintaining documented information or the requirement to meet objectives for continual improvement, may be a signal that there has been a loss of sensitivity in the architectural scaffolding that is required for the successful delivery of quality. This paper's objective is to present the results of a survey of experienced consultants in quality management (Wilson and Campbell, 2020).

Agile methodology was the optimal approach for developing the medical repository system due to several reasons:

1. **Iterative Development**: Agile allows for incremental development, enabling continuous feedback from stakeholders like Dr. Naidoo and clinical administrators. This iterative process ensures that the final product aligns closely with user requirements and evolving needs.
2. **Adaptability**: The dynamic nature of healthcare demands a flexible approach to development. Agile methodologies embrace change, allowing for adjustments to be made throughout the development cycle in response to emerging challenges or opportunities.
3. **Collaborative Approach**: Agile promotes collaboration among multidisciplinary teams, including developers, clinicians, and administrators. This collaborative environment fosters a deeper understanding of user needs and facilitates effective communication throughout the development process.
4. **Focus on Delivering Value**: Agile prioritizes features based on their value to end-users, ensuring efficient resource allocation. This approach enables the team to deliver a working product in the shortest time possible and address the immediate pain points while creating a foundation for future enhancements.
5. **Risk Mitigation**: Agile methodologies emphasize early and frequent testing, reducing the likelihood of critical issues being discovered late in the development cycle. This proactive approach to risk management is particularly crucial in healthcare software, where patient safety and data security are paramount.

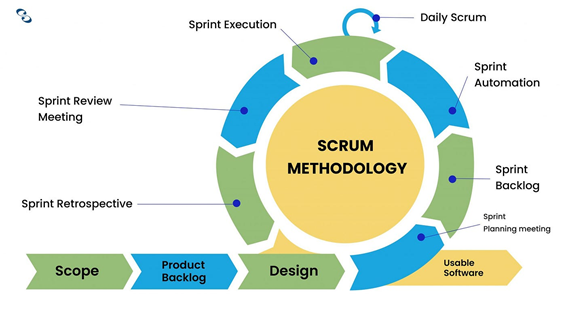
By leveraging Agile methodology, our team was able to develop a medical repository system that not only met the College of Health Sciences' immediate needs but also positioned it for future growth and innovation in data management and patient care.

## Scrum Framework

In the development of our document repository system for the College of Health Science, we adopted the Scrum framework. Scrum is an agile and iterative process used for the project management framework used for developing new information technologies. Scrum can be used in other fields as well, but the main usage of Scrum is in product development. It is based on concepts such as Agile and Lean Software Development in its philosophy and is focused on managing tasks rather than managing needs. The full set of Scrum components includes Roles, ceremonies, and Artifacts. Our team comprised a Scrum Master, a Product Owner, Developers, Testers, and Business Analysts. Each role had specific responsibilities and contributed uniquely to the successful completion of the project.

The main advantage of this framework is the ability to build flexible software products that adapt to new requirements. The challenges of deploying this experience are narrowed down to requiring an "Attitude Change" in the development team, which encompasses requests for a "cultural" mindset and the acquisition of specific practical development skills. (Zayat and Senvar, 2020)

Scrum is a simple framework used to help teams collaboratively develop complex products. Scrum consists of the Scrum Teams and their associated roles, events, artefacts, and rules. Each component within the framework serves a specific purpose and is essential to Scrum's success and usage. The rules of Scrum bind together the roles, events, and artefacts, governing the relationships among them and the output of the work. The Scrum framework is designed from the ground up to optimize predictability and control risks of continuous development in a complex environment with rapidly changing and evolving technology (Zayat and Senvar, 2020).



**Figure 2: Scrum methodology**

Delays and cost overruns are common, and initial customer confidence becomes a disappointment at the delays and demands of the finished product. No other wide area computing medium can exist and grow in this manner. The initial intent is that Scrum is like playing Rugby with software. It's a form of web development project management. Scrum projects accept that everything about the development project and its computer application is subject to the harsh realities of life. Scrum projects are like interfaces between external customer interests and vendor interests. Each project is built around a core team of individuals with good ideas and the strength of will and flexibility to implement them. They always give reliable estimates of their effort. They live by mandated technological practices and live in time walls. Most importantly, these individuals always work as a unit and work in small teams (Opelt et al., 2023).

The software industry has spent over 40 years developing many distinct models (e.g., logical progressions) identifying the essential people, processes, tools, and technologies for custom software development as contracts are negotiated. These models have relatively sophisticated procedures for specifying, building, inspecting, and installing the software. However, due to the industry's problems with large software development projects, the people, processes, tools, and technologies are no longer enough.

Agility involves seeking outcomes with flexibility and adaptability. It is an adaptive strategy rather than a predictive planner. In the software company, the commitment to agility requires the re-evaluation and retooling of virtually all software processes. The ability to develop quality software in an unpredictable environment is the key to success. This leads to this paper's topic, namely the Scrum approach to system design.

The addition of new roles can increase compliance and raise the quality of the final software. Thus, the cost of obtaining compliance can be decreased by detecting and repairing the defect at the higher level of the software development life cycle. When software quality is increased by more frequent activities related to validation, the risk of overlooking major defects is low. Moreover, the additional backlog considerably facilitates the re-estimation of the number and size of user stories. Since the Scrum approach provides visibility and transparency through the daily Scrum, both the domain-specific aspects of the UKZN Health Hive and validation activities are appropriately performed.

The Scrum framework has accumulated a lot of attention from both academic and industrial perspectives. Its popularity is based on its pragmatic, empirical foundations. However, the benefits of adopting Agile practices should meet the challenges for proper diffusion in the regulated environment of the healthcare industry. As a result, the transition of any organizational change in this domain should balance both challenges and benefits. In addition, the importance of the healthcare industry justifies additional refinement or extension of the Scrum framework.

Through iterative sprints, regular feedback loops, and continuous improvement practices, the team was able to deliver a secure, organized, and user-friendly document repository system. This system significantly reduced manual work for the clinical administrator, improved data security, and provided a structured approach to managing clinical documentation, thereby meeting the project's goals and the expectations of Dr. Urisha Naidoo and the College of Health Science.

# Software development team structure

**1. Scrum Master**

The Scrum Master is responsible for ensuring the integrity of the Scrum process, including the facilitation of Scrum meetings. The Scrum Master does not have authority over the members of the project team but instead works to mediate and manage the sprint’s challenges. This includes ensuring a proper balance of stakeholders' interactions with the project team and maintaining a focus on the goals of the current sprint (Shastri et al., 2021). Leaders encourage change, and Scrum Masters are expected to paint a picture of the current big picture—removing barriers, forming leadership qualities, and maintaining the role of the methodologist. Scrum Masters offer new ideas, process changes, and best practices to the team. They also strive to improve the speed of the development team while working to increase the flow of issues addressed by the team. Scrum Masters also ensure collaboration throughout a development sprint and manage the availability of testing resources and project designers with the project owner (Morandini et al., 2021).

**Role and Responsibilities:**

* **Facilitation**: The Scrum Master facilitated all Scrum ceremonies, including Sprint Planning, Daily Standups, Sprint Reviews, and Sprint Retrospectives.
* **Removal of Impediments**: They identified and removed any obstacles that could hinder the team's progress.
* **Coaching**: They coached the team on Scrum practices and ensured adherence to Agile principles.
* **Shielding the Team**: They protected the team from external distractions and interference.

**Contribution to the Project: The Scrum Master played a crucial role in maintaining the project's flow.** By facilitating efficient meetings and removing blockers, they ensured that the development process was smooth, and the team could focus on delivering high-quality increments of the document repository system.

**2. Product Owner**

The project owner is a representative from the business side of the project and is often in charge of gathering requirements from stakeholders. They must ensure that requirements are well understood by the development team. The project owner has the final say in priorities and acceptance of the features of the system. The project owner’s primary responsibility is to maximize the return on investment of the development effort, being responsible for creating product epics and defining the required user features. The project owner champions development progress, where it is expected that a project owner committed to the project will always be available to the team. (McGraw & Harbison, 2020)

**Role and Responsibilities:**

* **Backlog Management**: The Product Owner maintained the Product Backlog, ensuring it was prioritized and up to date.
* **Stakeholder Communication**: They acted as the primary liaison between stakeholders and the development team, gathering requirements and feedback.
* **Vision and Goals**: They communicated the product vision and goals to the team and ensured that the delivered increments aligned with stakeholder expectations.

**Contribution to the Project:** The Product Owner was instrumental in defining the features and functionalities of the document repository system. By prioritizing the backlog based on business value and stakeholder feedback, they ensured that the most critical features were developed first, aligning the final product with the needs of Dr Urisha Naidoo and the College of Health Science.

**3. Developers**

A software developer is a professional who designs, writes, tests and maintains computer programs and applications. This role involves the application of engineering principles to software development, ensuring that programs are functional, efficient, and meet user requirements. Software developers work in various programming languages and are responsible for both the front-end and back-end aspects of software. According to Sommerville (2011), software developers must possess a deep understanding of programming concepts, software engineering practices, and the software development lifecycle to create robust and maintainable software systems. They collaborate with other stakeholders, including project managers, testers, and users, to deliver high-quality software solutions (Pressman, 2014).

**Role and Responsibilities:**

* **Design and Implementation**: Developers were responsible for designing, coding, and implementing the features outlined in the Product Backlog.
* **Collaboration**: They worked closely with testers and business analysts to ensure a cohesive development process.
* **Code Reviews and Refactoring**: They participated in code reviews and regularly refactored code to improve its quality and maintainability.

**Contribution to the Project:** The developers were the backbone of the project, transforming requirements into a functional document repository system. They implemented features such as secure data storage, user authentication, and the ability to upload and retrieve documents, ensuring the system met technical and functional requirements.

**4. Testers**

A software tester is a professional responsible for evaluating and verifying that software applications function as intended, ensuring the quality and reliability of the product. This role involves designing and executing test cases, identifying defects, and reporting them for resolution. Software testers employ various testing methodologies, including manual and automated testing, to examine the software from different angles and ensure it meets specified requirements. According to Myers, Sandler, and Badgett (2011), software testing is a critical component of the software development lifecycle, aimed at detecting and preventing defects early in the development process to enhance the overall quality of the software. Furthermore, Beizer (2003) emphasizes that effective software testing requires a combination of technical skills, attention to detail, and a systematic approach to uncover hidden issues that might compromise software performance and user satisfaction.

**Role and Responsibilities:**

* **Test Planning and Execution**: Testers created and executed test plans to verify that the system met the defined requirements.
* **Bug Identification and Reporting**: They identified, documented, and reported bugs to the development team.
* **Regression Testing**: They performed regression testing to ensure that new changes did not introduce new defects.

**Contribution to the Project:** Testers ensured the quality and reliability of the document repository system. By conducting thorough testing at each iteration, they identified issues early, allowing developers to address them promptly. This iterative testing process helped maintain a high standard of quality throughout the project lifecycle.

**5. Business Analysts**

A business analyst is a professional who acts as a liaison between stakeholders and the project team, ensuring that business requirements are accurately captured and translated into functional specifications for technical implementation. Their role encompasses gathering, analyzing, and documenting business needs, processes, and workflows to facilitate the development of effective business solutions. Business analysts employ various techniques, such as SWOT analysis, requirements elicitation, and process modelling, to understand and address organizational challenges. According to IIBA (2015), the business analyst's primary responsibility is to drive business improvement and innovation by aligning business needs with IT capabilities, thereby enhancing operational efficiency and decision-making. Additionally, Cadle, Paul, and Turner (2014) highlight that successful business analysts possess strong analytical skills, effective communication, and a comprehensive understanding of both business operations and IT environments.

**Role and Responsibilities:**

* **Requirements Gathering**: Business analysts gathered requirements from stakeholders and translated them into detailed user stories and acceptance criteria.
* **Process Documentation**: They documented business processes and workflows to ensure a clear understanding of system functionality.
* **Facilitating Communication**: They acted as a bridge between the technical team and non-technical stakeholders, ensuring mutual understanding.

**Contribution to the Project:** Business analysts played a key role in defining the scope and requirements of the document repository system. Their detailed documentation and clear communication helped ensure that the developers and testers understood the functional requirements, leading to a product that met the needs of the College of Health Science.

**Collaboration and Project Completion**

The collaboration between these roles was essential to the successful completion of the document repository system. The Scrum Master facilitated effective communication and process adherence, the Product Owner ensured alignment with stakeholder needs, the Developers built the system, the Testers ensured its quality, and the Business Analysts clarified requirements and processes.

# Project Management Plan

## Project Scope

The scope of this project is to create a comprehensive and user-friendly web-based system designed to replace the existing manual process used by the clinical administration in the School of Speech and Language Therapy for managing patient documents. This new system will streamline the workflow, reduce the administrative burden, and enhance the overall efficiency and security of managing patient records. The project will involve the following key components:

1. **Website Development**:
   * Develop a secure, responsive, and intuitive website that supports various devices and browsers.
   * Implement a user-friendly interface that simplifies the process of uploading, accessing, and managing patient documents.
   * Incorporate features such as role-based access control, search functionality, and document categorization.
2. **Security Enhancements**:
   * Implement robust security measures, including encryption, user authentication, and access controls to protect patient information.
   * Comply with relevant data protection regulations, such as HIPAA or GDPR, to ensure legal and ethical handling of patient data.
3. **User Training and Support**:
   * Provide comprehensive training for clinical administrators and other users to ensure smooth adoption of the new system.
   * Develop user manuals and help resources to assist users in navigating and utilizing the website effectively.
4. **Ongoing Maintenance and Updates**:
   * Establish a maintenance plan to regularly update the website, address any technical issues, and incorporate user feedback for continuous improvement.

## Project Objectives

This project's objectives are to develop a digital document management system that significantly enhances the efficiency, security, and user experience for managing patient records in the School of Speech and Language Therapy. The specific objectives include:

1. **User-Friendly Interface**:
   * Design a website with an intuitive interface that simplifies document uploading, retrieval, and management for clinical administrators.
   * Ensure that the system is accessible to users with varying levels of technical proficiency.
2. **Enhanced Security**:
   * Implement advanced security protocols to safeguard patient information, including encryption, multi-factor authentication, and role-based access control.
   * Ensure that the system complies with all relevant data protection regulations to maintain the confidentiality and integrity of patient data.
3. **Efficient Document Management**:
   * Streamline the process of uploading, organizing, and retrieving patient documents to reduce the administrative workload.
   * Incorporate features such as automated document categorization, tagging, and search functionality to enhance document management efficiency.
4. **Data Integrity and Reliability**:
   * Ensure the accuracy and integrity of patient data during and after the migration process.
   * Implement backup and recovery mechanisms to prevent data loss and ensure the reliability of the system.
5. **Comprehensive Training and Support**:
   * Provide thorough training sessions and resources for clinical administrators and other users to facilitate the transition to the new system.
   * Offer ongoing technical support to address any issues and ensure the system operates smoothly.
6. **Continuous Improvement**:
   * Establish a feedback loop to gather user input and continuously enhance the system based on user needs and technological advancements.
   * Plan for regular updates and maintenance to keep the system secure and up to date with the latest features and best practices.

By achieving these objectives, the project aims to create a robust, secure, and user-friendly document management system that improves the efficiency and effectiveness of clinical administration in the School of Speech and Language Therapy.

## Work Breakdown Structure

**Sprint 1: Planning and Requirements**

1. **Create a Development Plan**
   * Define project goals and objectives
   * Identify key milestones and deliverables
   * Allocate resources and assign roles
2. **Gather Requirements**
   * Conduct stakeholder interviews
   * Document user stories and acceptance criteria
   * Prioritize requirements based on business value
3. **Analyze Requirements**
   * Review and refine user stories
   * Create use case diagrams
   * Validate requirements with stakeholders

**Sprint 2: System Development**

**Initial Development and Setup**

1. **Create MVC Solution**
   * Set up the project repository and structure
   * Implement initial MVC architecture
   * Create basic views and controllers
2. **Add Login Security**
   * Implement user authentication
   * Set up user roles and permissions
   * Test login functionality

**Core Functionality Development**

1. **Add Clinical Record Functionality**
   * Develop models and controllers for clinical records
   * Create views for adding and viewing clinical records
   * Implement CRUD operations for clinical records
2. **Add Doctor Functionality**
   * Develop models and controllers for doctor information
   * Create views for adding and viewing doctor details
   * Implement CRUD operations for doctor data
3. **Doctor Testing**
   * Write and execute test cases for doctor functionality
   * Identify and fix bugs
   * Validate data integrity
4. **Patient and Document Testing**
   * Write and execute test cases for patient and document management
   * Identify and fix bugs
   * Validate data integrity

**Sprint 3: Enhancement and reporting**

**Reporting and UI Enhancements**

1. **Create Reporting**
   * Design report templates
   * Implement backend logic for generating reports
   * Test report generation with sample data
2. **Enhance UI and UX**
   * Improve navigation and layout
   * Apply consistent styling and themes
   * Conduct usability testing and make improvements

**Site Clean-Up and Validation**

1. **Clean Up Site**
   * Refactor code for readability and maintainability
   * Remove unused code and dependencies
   * Optimize performance
2. **Validation Enhancements**
   * Add input validation for forms
   * Implement error handling and notifications
   * Test validation rules and error scenarios

**Sprint 4: Testing and Quality Assurance**

1. **End-to-End Testing**
   * Perform end-to-end testing of the entire system
   * Ensure all components work together seamlessly
   * Fix any integration issues

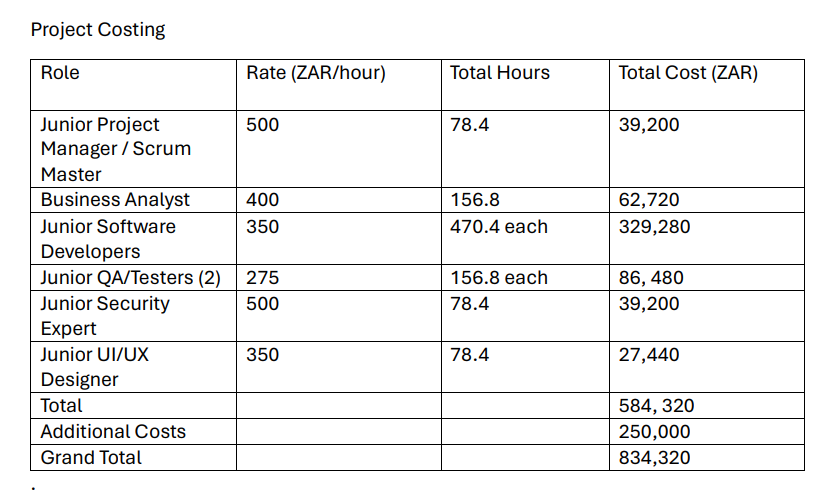
**User Analysis and Deployment**

1. **User Analysis Testing**
   * Conduct user acceptance testing (UAT)
   * Gather feedback from real users
   * Make final adjustments based on feedback
2. **Deployment**
   * Prepare deployment scripts and documentation
   * Deploy the system to the production environment
   * Monitor deployment and address any issues

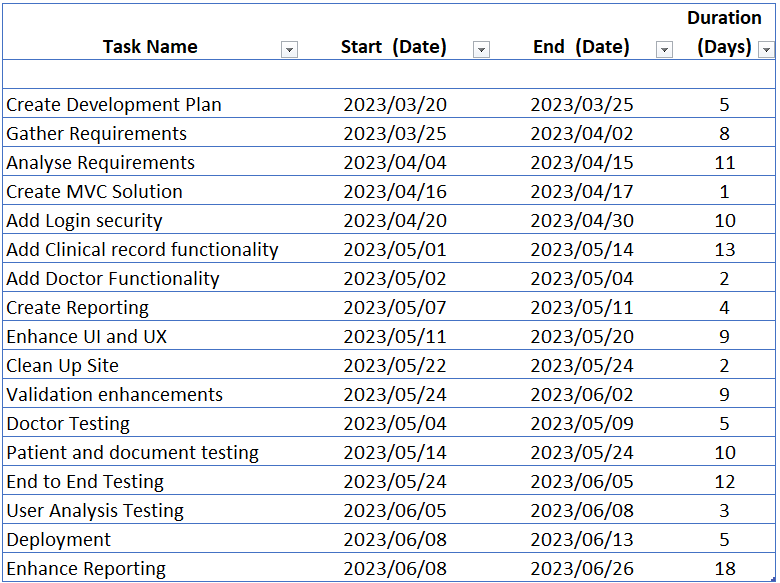
**Reporting Enhancements and Project Closure**

1. **Enhance Reporting**
   * Add advanced reporting features (e.g., filters, export options)
   * Improve report presentation and layout
   * Validate enhanced reporting functionality
2. **Project Closure**
   * Conduct a final review of the project
   * Document lessons learned
   * Celebrate the team's success

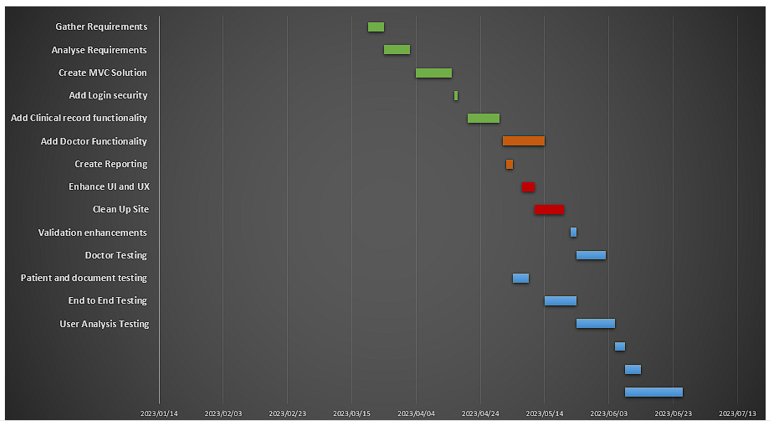
**Project Costing**

 **Figure 7: Project cost breakdown**

## Project Schedule



**Figure 8: Project schedule breakdown**

 **Figure 9: Project schedule - Gantt Chart**

# Software Development Approach

## Object-oriented approach

The object-oriented approach focuses on establishing the behavior and structure of software systems through small, self-contained modules known as objects, which consist of both processes (methods) and data (attributes). This approach emphasizes the use of classes to define the blueprint of these objects, allowing for the encapsulation, inheritance, and polymorphism that collectively enhance the modularity, reusability, and maintainability of software systems (Satzinger, J.W et al., 2015). The object-oriented method was used for our system to efficiently manage and record patient records and data while creating a patient management system. The system was divided into several objects, such as Patient, Clinic, and Records, Doctors and security, each of which included its own set of pertinent parameters and operations. Team members were able to work independently on components because of this modular framework, which improved communication and sped up development. Inheritance allowed related classes to share similar methods and characteristics, which encouraged consistency and reuse of code. Encapsulation allowed us to protect internal data, making our system easier to debug and update. Polymorphism made it possible to handle various file kinds and formats flexibly, guaranteeing that the system could readily adjust to changing needs. Additionally, by enabling the addition of new features without requiring major modifications to the current system, this technique promotes scalability.

Incorporating a code-first approach into our object-oriented development process further enhanced the efficiency and effectiveness of our patient management system. The code-first methodology emphasizes writing code for the data models first before designing the database schema. This approach allowed us to define our objects and their relationships programmatically, ensuring that the system's architecture was driven by the actual code rather than being constrained by predefined database structures. By focusing on the code first, we could easily refactor and adapt our models to changing requirements without extensive database migrations, making the system more adaptable and resilient (Rangarajan et al., 2020).

The benefits of this combined object-oriented and code-first approach were manifold:

**Enhanced Modularity**: By breaking down the system into smaller, manageable parts and organizing them into classes with specific behaviors, the team was able to work on different components simultaneously without interference. This led to a more efficient development process, timely delivery, and minimized technical issues.

**Improved Flexibility**: Our system could easily adapt to changing requirements and new features. The code-first approach allowed us to make changes at the code level without worrying about the underlying database schema, facilitating quicker iterations and updates.

**Increased Maintainability**: Encapsulation protected internal data, making the system easier to debug and update. By keeping the implementation details hidden within objects, we ensured that changes in one part of the system did not adversely affect other parts.

**Scalability**: The use of inheritance and polymorphism enabled us to extend the system's functionality without significant modifications. New classes could be created by inheriting properties and methods from existing ones, promoting code reuse and consistency.

**Efficient Collaboration**: The modular structure allowed team members to work independently on different components, improving communication and collaboration. Each member could focus on developing specific objects, contributing functional pieces to the overall system.

The object-oriented approach, combined with the code-first methodology, proved to be the best choice for our patient management system. It not only enhanced the system's modularity, flexibility, and maintainability but also facilitated a more efficient development process, timely delivery, and high-quality output. By focusing on the code first, we ensured that our system could readily adapt to changing needs, providing a robust and scalable solution for managing patient records and data (Bushong et al., 2021).

# System Version Control

For our system version control, we have decided to use Git from GitHub. GitHub is a cloud-based hosting service that enables developers to use the version control system Git to organize and save their code. It makes it possible for several developers to collaborate effectively, track changes, and work on the same project at once (Maria Webb, 2024). GitHub, on the other hand, is a hosting provider that will allow us to host our Git project on their remote server (GitHubDocs). Utilizing Git for version control in the development of the patient management system allows us to manage and track changes to the code base, collaborate efficiently, and maintain a detailed project history. Here’s how UKZN Health Hive achieved version control during the development of the patient management system:

* Creation of repository: A repository named UKZN Health Hive (figure 10) was created on GitHub for the system, this repository serves to house the MVC solution for the UKZN Health System, which was also the primary location for storing the project's code and history.
* Cloning the repository: Each team member cloned the repository to their local machine, creating a local copy of the project. This enabled us to work on the code independently and make changes without affecting the main repository.
* Each system feature that the team members were working on had its own branch established. As a result, we were able to complete several project components concurrently without obstructing one another's progress. Branches also aid in maintaining the stability and cleanliness of the main branch, commonly referred to as "main" or "master"(GitHubDocs).
* Every team member was required to commit code changes to their local repository as they were made. When changes are committed, a snapshot of the code is created at that point of the team, which enables the team to follow the project's history and roll back to earlier iterations as needed.
* To ensure that the central repository remained current and to make their modifications visible to the rest of the team, each person working on a branch had to push their finished product to the central repository on GitHub.
* When it came time to combine the modifications into the main branch, each individual made a pull request, which let us examine the changes, offer comments, and accept or request more changes before the code was merged.
* After a pull request was examined and accepted, the modifications were combined with the main branch. This guarantees that the most recent, stable version of the project is always included in the main branch (Ben Straub, 2014).
* To maintain the most recent versions of their local repositories, each team member regularly pulled and fetched updates from the central repository while the system was being built. Assuring that everyone is using the most recent version of the code and preventing disagreements are two benefits of this (Ben Straub, 2014).

|  |  |
| --- | --- |
| **Steps** | **Process** |
| Step 1 | Creation of repository |
| Step 2 | Cloning the repository |
| Step 3 | Create Branches |
| Step 4 | Commit and Push changes |
| Step 5 | Create Pull requests |
| Step 6 | Merge changes |

**Table 1.1 System Version Control Steps**

# Human-Computer Interaction (HCI) Strategy Implementation

The objective of our HCI strategy was to enhance the end-users experience in using the UKZN Health Hive , a web-based application developed using the Model-View-Controller (MVC) architecture and the Identity Framework for security. The system allows CRUD operations for clinics, doctors, patients, and clinical records. Our focus was on improving usability, accessibility, and overall user satisfaction.

## Key Principles

**1. User-Centered Design (UCD):** We placed the end-user at the center of the design process and continuously gathered user feedback to refine the system.

**2. Consistency:** We ensured consistent design and behavior across the system to minimize learning curves and prevent errors.

**3. Accessibility:** We designed the system to be accessible to all users, including those with disabilities.

**4. Efficiency:** We optimized workflows to minimize the time and effort required for users to complete their tasks.

## Strategy Components

1. **User Research and Personas**
   * **Conducted User Interviews and Surveys:** We gathered insights into the needs, preferences, and pain points of different user groups (e.g., doctors, clinic administrators, patients).
   * **Developed User Personas:** We created detailed personas to represent key user groups, guiding our design decisions.
2. **Information Architecture**
   * **Simplified Navigation:** We organized information logically and used clear labels to ensure intuitive navigation.
   * **Implemented Breadcrumbs and Search Functionality:** We provided breadcrumbs for easy backtracking and a robust search feature to quickly locate records.
3. **User Interface Design**
   * **Consistent Layouts and Elements:** We used a consistent layout with familiar UI elements (buttons, forms, tables) to reduce cognitive load.
   * **Responsive Design:** We ensured the system was fully responsive, providing a seamless experience across various devices (desktops, tablets, smartphones).
   * **Visual Hierarchy:** We used visual cues (size, color, contrast) to guide users’ attention to important elements and actions.
4. **Interaction Design**
   * **Streamlined Workflows:** We optimized common workflows (e.g., creating a new patient record) to minimize steps and reduce redundancy.
   * **Feedback Mechanisms:** We provided immediate feedback for user actions (e.g., success/failure messages, loading indicators).
   * **Error Prevention and Handling:** We implemented validation checks and clear error messages to prevent and resolve mistakes efficiently.
5. **Accessibility**
   * **Adhered to WCAG Standards:** We ensured compliance with Web Content Accessibility Guidelines (WCAG) to support users with disabilities.
   * **Keyboard Navigation:** We ensured that all functionalities were accessible via the keyboard.
   * **Screen Reader Compatibility:** We designed the interface to work seamlessly with screen readers.
6. **Security and Privacy**
   * **Role-Based Access Control (RBAC):** We implemented RBAC to ensure users only access information relevant to their roles.
   * **User Consent and Transparency:** We clearly communicated data usage policies and obtained explicit consent from users.
7. **Usability Testing and Iteration**
   * **Regular Usability Testing:** We conducted regular usability tests with real users to identify pain points and areas for improvement.
   * **Iterative Design Process:** We used an iterative design approach, incorporating user feedback and test results into continuous improvements.
8. **Training and Support**
   * **Onboarding Tutorials:** We provided interactive tutorials and guides to help new users get acquainted with the system.
   * **Help Center and Support:** We offered a comprehensive help center with FAQs, user manuals, and access to support.

## Implementation Plan

**Phase 1: Research and Planning**

* + We conducted user research (interviews, surveys).
  + We developed user personas and journey maps.

**Phase 2: Design and Prototyping**

* + We designed the information architecture and UI mockups.
  + We created interactive prototypes for key workflows.

**Phase 3: Development and Testing**

* + We implemented UI and interaction designs.
  + We conducted usability and accessibility testing.

**Phase 4: Deployment and Training**

* + We rolled out the system with comprehensive onboarding and training materials.
  + We established support channels.

**Phase 5: Monitoring and Iteration**

* + We gathered ongoing user feedback.
  + We regularly updated and refined the system based on user input and technological advancements.

By implementing this HCI strategy, we created a UKZN Health Hive that is intuitive, efficient, and satisfying to use, ultimately enhancing the overall user experience and adoption rate.

# Knowledge Management System

To extend our system’s transaction processing capacity and enhance decision-making, we integrated Power BI into our workflow. This integration provides advanced analytics and knowledge management capabilities, transforming raw data into actionable insights.

## Key Components and Achievements

1. **Data Integration and ETL Processes**:
   * We implemented robust ETL (Extract, Transform, Load) processes to gather data from various sources, ensuring a consistent and up-to-date data repository.
   * Data from transactional systems, logs, and external APIs is continuously fed into our central data warehouse, providing a holistic view of our operations.
2. **Power BI Dashboards and Reports**:
   * We developed interactive Power BI dashboards that provide real-time insights into key performance indicators (KPIs), transaction volumes, and system performance metrics.
   * Custom reports tailored to different stakeholders (e.g., management, operations, development) offer detailed analysis and facilitate data-driven decision-making.
3. **Automated Alerts and Notifications**:
   * Power BI is configured to send automated alerts and notifications based on predefined thresholds and anomalies, enabling proactive management and quick resolution of potential issues.
4. **Advanced Analytics and Predictive Modeling**:
   * Leveraging Power BI’s advanced analytics capabilities, we implemented predictive models to forecast transaction trends, identify potential bottlenecks, and optimize resource allocation.
   * Machine learning models integrated with Power BI help predict user behavior, enhance customer experience, and improve operational efficiency.
5. **Knowledge Management and Collaboration**:
   * Power BI’s collaborative features allow team members to share insights, comment on reports, and collaborate on data analysis, fostering a culture of continuous improvement and knowledge sharing.
   * Documentation and data dictionaries are integrated within Power BI, providing context and ensuring consistent understanding of data across the organization.
6. **Performance Monitoring and Optimization**:
   * Continuous performance monitoring through Power BI dashboards enables real-time tracking of system health and transaction processing capacity.
   * Regular performance reviews and optimizations based on Power BI insights ensure our system remains efficient and responsive.

# Software Testing Plan

This testing plan outlines the strategies and processes for testing the UKZN Health Hive. The testing phases include Unit Testing, System Testing, Alpha Testing, and Beta Testing. The functionalities to be tested include CRUD operations for Doctors, Patients, Clinics, and Clinical Records, and integration with Azure Blob Storage and the Identity Framework for security.

## Objectives

* Ensure all functionalities work as expected.
* Identify and fix bugs or issues.
* Validate the integration with Azure Blob Storage.
* Ensure the Identity Framework provides robust security.
* Verify system performance and scalability.

## Scope

**Functionality to be tested:**

* CRUD operations for Doctors, Patients, Clinics, and Clinical Records.
* Integration with Azure Blob Storage.
* Security enhancements using the Identity Framework.

## Testing Phases

Phase 1: Unit Testing

**Objective:** Verify the functionality of individual components or units of the software.

**Tools:** xUnit, NUnit

**Scope:** Test the smallest parts of the application, such as methods and classes.

Test Cases:

1. **Doctors**
   * Create Doctor
   * Read Doctor details
   * Update Doctor details
   * Delete Doctor
2. **Patients**
   * Create Patient
   * Read Patient details
   * Update Patient details
   * Delete Patient
3. **Clinics**
   * Create Clinic
   * Read Clinic details
   * Update Clinic details
   * Delete Clinic
4. **Clinical Records**
   * Add Clinical Record to Azure Blob Storage
   * Read Clinical Record
   * Update Clinical Record
   * Delete Clinical Record
5. **Identity Framework**
   * User Authentication
   * Role-based Authorization
   * Password Management

**Strategy:** Develop and run unit tests for each function. Use mock objects to simulate interactions with external systems such as Azure Blob Storage and the Identity Framework.

Phase 2: System Testing

**Objective:** Verify that the complete and integrated software works as intended.

**Tools:** Selenium

**Scope:** Test the entire system, including all integrations and interactions.

Test Cases:

1. **End-to-End Scenarios**
   * User Registration and Login
   * Creating and Managing Doctors, Patients, and Clinics
   * Adding, Viewing, Updating, and Deleting Clinical Records
2. **Integration Testing**
   * Interaction between CRUD operations and Azure Blob Storage
   * Validation of security protocols via the Identity Framework
3. **Performance Testing**
   * Response time for CRUD operations
   * Load testing for multiple concurrent users
   * Stress testing the system’s handling of large volumes of data

**Strategy:** Perform automated and manual testing on the fully integrated system. Validate that all components interact correctly and meet performance requirements.

Phase 3: Alpha Testing

**Objective:** Conduct internal testing to identify bugs before releasing the software to external users.

**Scope:** Focus on usability, functionality, and performance under normal usage conditions.

**Participants:** Internal testers, developers, and QA team members.

Strategy:

* Conduct thorough testing on a staging environment.
* Simulate real-world scenarios and usage patterns.
* Collect feedback from testers and identify critical issues.
* Fix identified issues and re-test.

Phase 4: Beta Testing

**Objective:** Conduct external testing with a limited number of actual users to identify any remaining issues in a real-world environment.

**Scope:** Focus on overall user experience, functionality, and performance.

**Participants:** Selected group of end-users.

**Strategy:**

* Release a beta version to selected users.
* Collect feedback on usability, bugs, and performance.
* Monitor the system for any critical issues.
* Address feedback and issues reported by beta testers.
* Conduct follow-up tests based on beta user feedback.

### Test Environment

* Visual Studio
* Azure DevOps for CI/CD
* Local databases and Azure Blob Storage simulation
* Staging server with a similar configuration to the production
* Actual Azure Blob Storage for integration tests
* Secure network for testing the Identity Framework

## Test Data

* Dummy data for Doctors, Patients, and Clinics.
* Sample clinical records in various formats.
* Test user accounts with different roles and permissions.

## Test Execution

**Unit Testing Execution:**

* Frequency: Daily during development
* Responsibility: Developers

**System Testing Execution:**

* Frequency: After each major feature is developed
* Responsibility: QA Team

**Alpha Testing Execution:**

* Frequency: Once system testing is complete
* Responsibility: Internal team

**Beta Testing Execution:**

* Frequency: Before final release
* Responsibility: Selected external users and QA Team

### Defect Management

* Use Azure DevOps for tracking and managing defects.
* Prioritize defects based on severity and impact.
* Regular triage meetings to review and assign defects.
* Ensure all critical defects are resolved before moving to the next testing phase.

This testing plan ensures that the UKZN Health Hive is thoroughly tested at every stage of development, from individual units to the complete integrated system. By implementing unit testing, system testing, alpha testing, and beta testing, we aim to deliver a robust, secure, and user-friendly application.

# Reusability and maintenance

## Modular Design

We adopted a modular design approach, breaking down our system into distinct, independently manageable components:

* **Separation of Concerns**: We restructured our code base to ensure each module handles a specific aspect of functionality. This restructuring has minimized interdependencies, making it easier to update and maintain individual components.
* **Micro-services Architecture**: For parts of our application where scalability and independent deployment were crucial, we implemented a micro-services architecture. Each service operates as a standalone unit, communicating with others through well-defined APIs.

## Adherence to SOLID Principles

To enhance code quality and maintainability, we rigorously applied SOLID principles:

* **Single Responsibility Principle**: Every class and module now has a single responsibility, reducing complexity and increasing clarity.
* **Open/Closed Principle**: We ensured that our components are extensible without requiring modifications to existing code.
* **Liskov Substitution Principle**: Our components are designed to be replaceable with their subtypes, promoting flexibility and reuse.
* **Interface Segregation Principle**: We created client-specific interfaces, preventing the implementation of unnecessary methods.
* **Dependency Inversion Principle**: Our architecture relies on abstractions rather than concrete implementations, enhancing flexibility and testability.

## Utilization of Design Patterns

We standardized the use of design patterns across our code base:

* **Creational Patterns**: Patterns like Factory and Singleton have streamlined our object creation processes.
* **Structural Patterns**: Adapter, Composite, and Decorator patterns have helped us manage complex object structures efficiently.
* **Behavioral Patterns**: We utilized Strategy, Observer, and Command patterns to manage communication and behavior among objects.

## Code Reusability Practices

We implemented several practices to maximize code reusability:

* **Libraries and Frameworks**: We developed and standardized the use of internal libraries and frameworks encapsulating common functionalities.
* **Reusable Functions/Methods**: Our code base now features numerous small, reusable functions and methods, each performing a single task.
* **Generic Programming**: We employed generics to create components that operate with any data type, increasing their versatility.

## Maintainable Code Practices

Our team adopted best practices to ensure long-term code maintainability:

* **Consistent Coding Standards**: We established and strictly adhered to a unified coding standard.
* **Comprehensive Documentation**: Each component is thoroughly documented, detailing its purpose, usage, and expected behavior.
* **Code Reviews**: Regular code reviews have become integral to our development process, ensuring code quality and facilitating knowledge sharing.
* **Automated Testing**: We implemented a robust suite of unit tests, integration tests, and end-to-end tests, ensuring code reliability and facilitating safe refactoring.
* **CI/CD Pipelines**: Our continuous integration and deployment pipelines automate testing and deployment, ensuring our components are always in a releasable state.

## Version Control and Branching Strategies

We standardized our version control and branching strategies to streamline development and collaboration:

* **Version Control Systems (VCS)**: We use Git for tracking changes, managing versions, and collaborating effectively.
* **Branching Strategy**: Our adoption of the GitFlow branching strategy has organized our feature development, releases, and hotfixes, ensuring a smooth workflow.

## Refactoring and Technical Debt Management

To maintain code quality over time, we adopted a proactive approach to refactoring and technical debt management:

* **Regular Refactoring**: We schedule regular refactoring sessions to improve code structure and performance.
* **Technical Debt Management**: We continuously identify and address technical debt, preventing it from accumulating and impacting productivity.

## Dependency Management

We implemented robust dependency management practices:

* **Dependency Injection**: Our use of dependency injection has decoupled components, improving testability and flexibility.
* **Package Management**: We standardized the use of package managers like NuGet to manage external dependencies and ensure consistent versioning.

## Documentation and Knowledge Sharing

We prioritized documentation and knowledge sharing across our team:

* **Comprehensive Documentation**: Our documentation includes architecture diagrams, API references, and user guides, ensuring all team members have access to necessary information.
* **Knowledge Sharing**: Regular knowledge-sharing sessions and detailed documentation of key decisions and best practices have fostered a culture of continuous learning and improvement.

## Performance Monitoring and Optimization

We established a performance monitoring and optimization framework:

* **Monitoring Tools**: We deployed monitoring tools to track the performance of our components in production.
* **Regular Optimization**: We routinely review and optimize component performance, ensuring they remain efficient and responsive.

The successful implementation of this strategy has resulted in a more modular, maintainable, and reusable codebase. Our development process is now more efficient, and our system components are easier to manage, scale, and extend. This strategic approach has not only improved our current operations but also positioned us well for future growth and innovation.

# Usability and Functionality of the System

As part of our ongoing efforts to enhance the effectiveness and user-friendliness of our Clinical Document Record System, we have conducted a comprehensive usability questionnaire. This questionnaire was designed to gather helpful feedback from users who interacted with the system. The aim was to identify areas of strength in the system and opportunities for improvement within the system’s interface and functionality.

The questionnaire was distributed via a Google Form and completed by a diverse group of 19 participants. These individuals were selected based on their first-time interaction with the system and their varied roles within the healthcare ecosystem, ensuring a well-rounded perspective on the system’s usability and functionality.

The feedback collected will help guide the improvements to the clinical system. It will also serve as a critical component of the academic defence document.

(Refer to appendix: Results from the questionnaire for each question)

Question 1: Rate the overall experience of navigating through the system based on your permissions.

The results from question 1 show that when asked to rate the overall experience of navigating through the system based on their permissions, most participants (68.4%) gave the highest rating of 5, which suggests they had a positive experience. Ratings 3 and 4 each received three responses (15.8%), which shows that some users encountered challenges when navigating the system or saw room for improvement.

Question 2: It is easy to learn how to use the system.

The results from question 2 indicate that the majority of participants (68,4%) strongly agree that the system is easy to use from their rating of 5. However, a small group of participants (21.1 %) gave it a rating of 3 suggesting that the system was easy to use, but there were issues encountered, while even a smaller percentage (10.5%) gave it a rating of 4.

Question 3: Did you find the ability to add documents, pertaining to patients, easier than the manual method?

The results indicate that when participants were asked if they found the ability to add documents pertaining to patients easier than the manual method, the majority responded positively. Specifically, (68.4%) rated it a 5, suggesting they found the electronic method much easier than the manual method. A smaller group, (21.1%), rated it a 3, indicating a moderate ease of the ability to add documents, and (10.5%) rated it a 4, showing they found it somewhat easier.

Question 4: Were you only able to access the work you had permission to?

The pie chart indicates that most participants, (89.5%), answered “Yes” to the question “Were you only able to access the work you had permission to?” This suggests that the permission settings in the system are in place and that most users are accessing only the work they are authorized to. A small percentage, (10.5%), answered “No,” indicating that they could access work beyond their permissions. This could point to potential issues with permission settings that need to be addressed to ensure proper role-based access control.

Question 5: The system is error-prone.

The horizontal bar graph reflects participants’ use of the system’s error frequency. Here’s a breakdown of the responses:

* (31.6%) the system as 1, suggesting they strongly disagree with the statement that the system is error-prone.
* (10.5%) gave a rating of 2, indicating disagreement, though not as strong.
* The largest group, (36.8%), rated it as 3, showing moderate agreement with the statement.
* (15.8%) rated it as 4, which suggests they agree that the system has some errors.
* Only (5.3%) rated it as 5, indicating strong agreement that the system is error-prone.

Question 6: The system enables good decision-making.

The bar graph indicates that many of the participants, (78.9%,) believe that the system enables good decision-making, as they have given it the highest rating of 5. A smaller portion, (21.1%), rated it a 3, suggesting they think it’s somewhat effective but not strongly so. No participants rated it poorly (1 or 2), and none were neutral (4). This suggests a generally positive perception of the system’s effectiveness in facilitating good decisions.

Question 7: The system responds in a timely manner to my requests.

The pie chart shows that a majority of the participants, (89.5%), are satisfied with the system’s response time, indicating they find it timely. A small minority of participants, (10.5%), do not find the system’s response time satisfactory, which is indicated by the “Maybe” segment. Overall, this suggests the system is perceived as efficient in response time.

Question 8: Do you think the current functionality will replace the manual method?

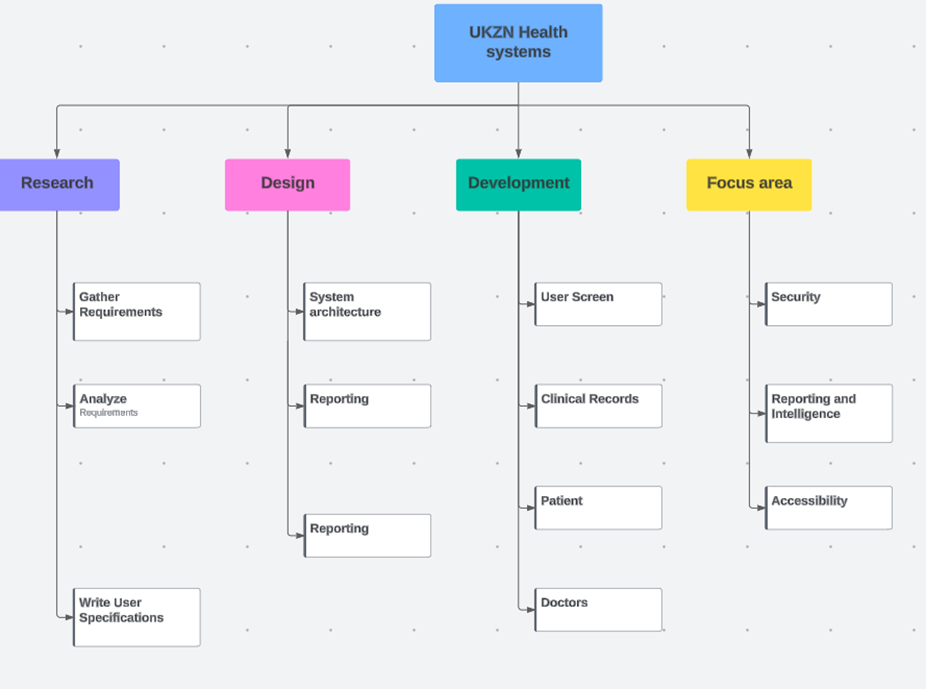
The horizontal bar graph suggests that most participants, (63.2%), strongly believe that the current functionality of the system can replace the manual process that is already in place, as indicated by the highest rating of 5. A smaller group, 21.1%, seems somewhat convinced with a rating of 3. Only one respondent (5.3%) gave the lowest rating of 1, showing strong dissatisfaction with the system’s ability to replace manual processes.

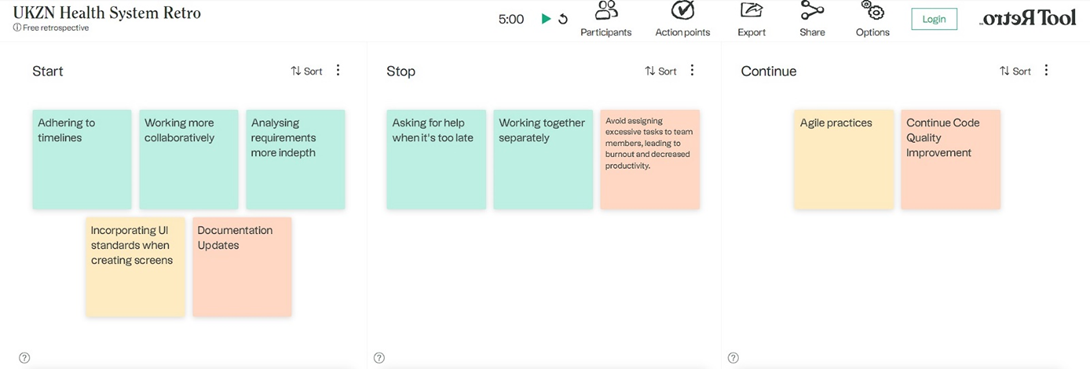
# Conclusion

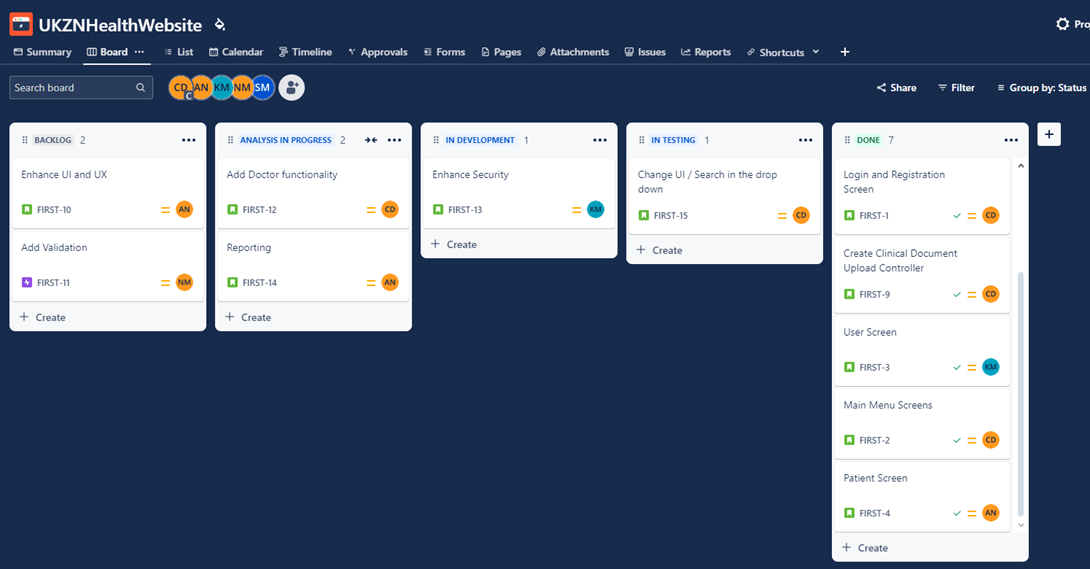
From the usability and functionality studies done on the Clinical Document Record System, it can be seen that there are improvements in terms of Human-Computer Interaction and the functionality of the system. The final paper-based survey, with which a wide and inclusive group of subjects took part, showed that most of them considered the system easy to use and more efficient than the previous manual methods. The majority of the users remarked that the system helped them to make respectable decisions and queried their demands adequately.  
  
Due to the effectiveness of the strategies described, the goal of obtaining a scalable, sustainable and reusable code has been achieved, which in turn helps to organize the work of developers efficiently. The standards set in terms of code quality, maintainability and performance monitoring not only improve the current outcomes but prepare the system well for the future.

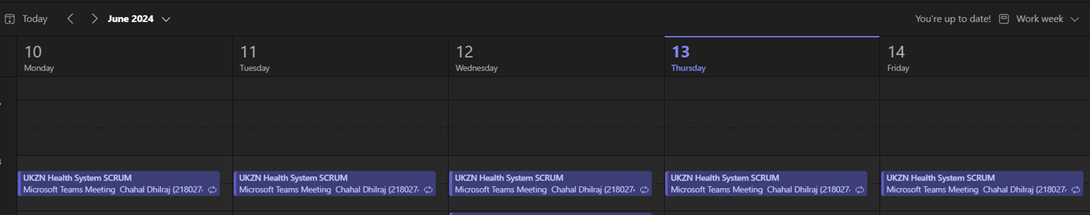
In conclusion, the systematic approach towards the enhancement of the Clinical Document Record System, based on the results of tests, user feedback, and continuous optimization has been successful. The system is now considered a robust, secure, and user-friendly application that is well-equipped to meet the demands of its users and adapt to future challenges.

# Appendices

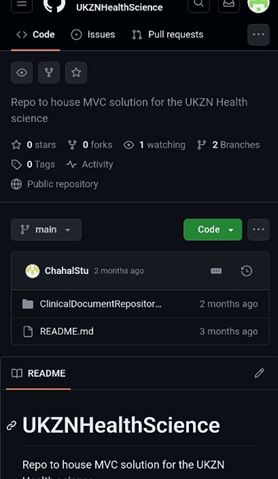
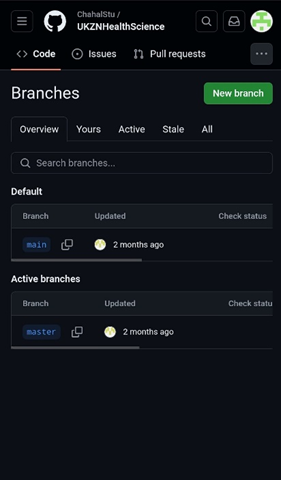
  
**Figure 3: Work Breakdown Structure**

 **Figure 4: Retros Project Board**

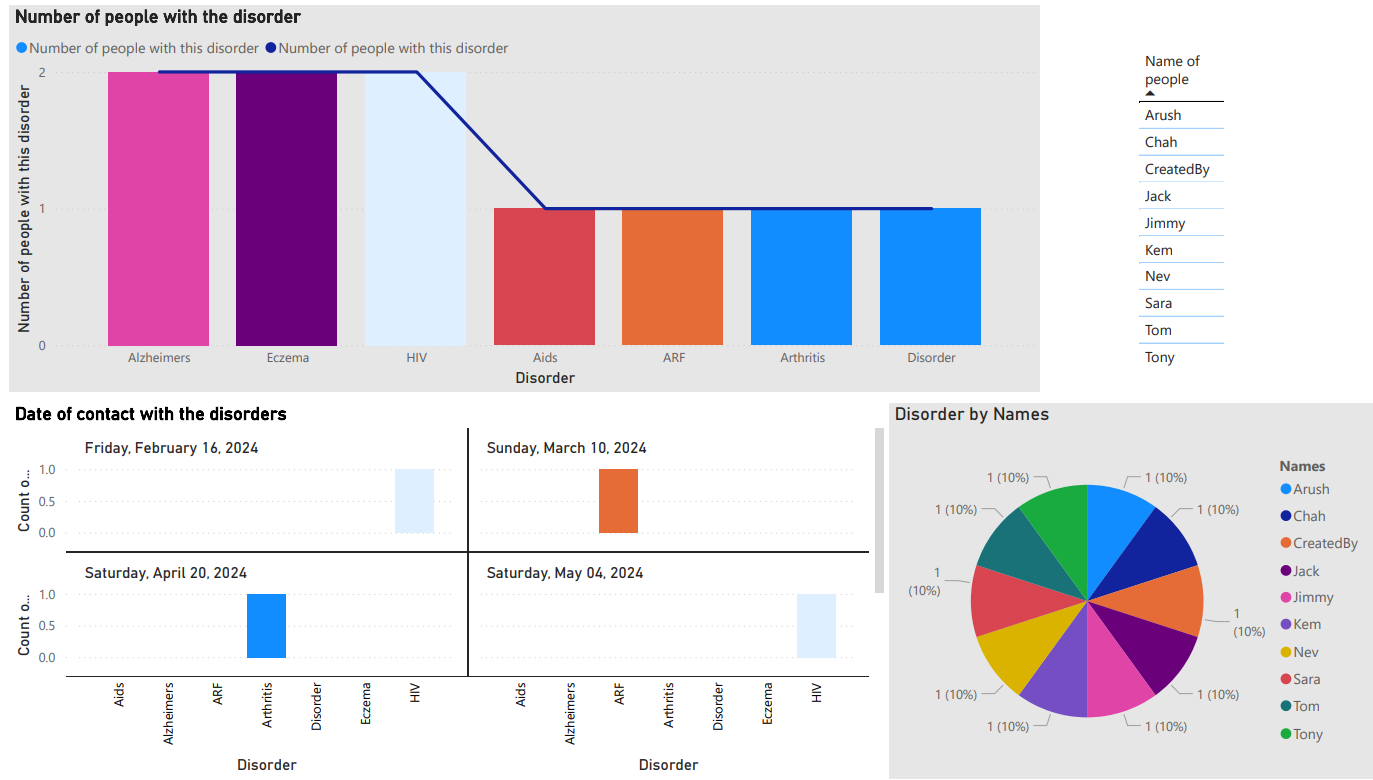
**Figure 5: Jira – Project Management Tool**



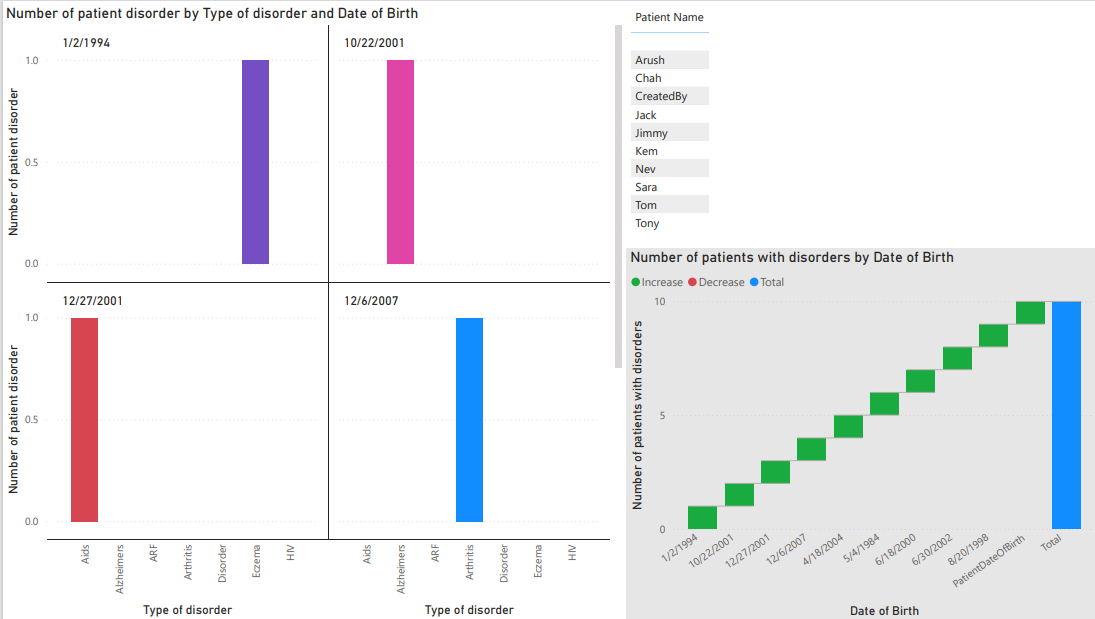
**Figure 6: Daily Scrum meeting**



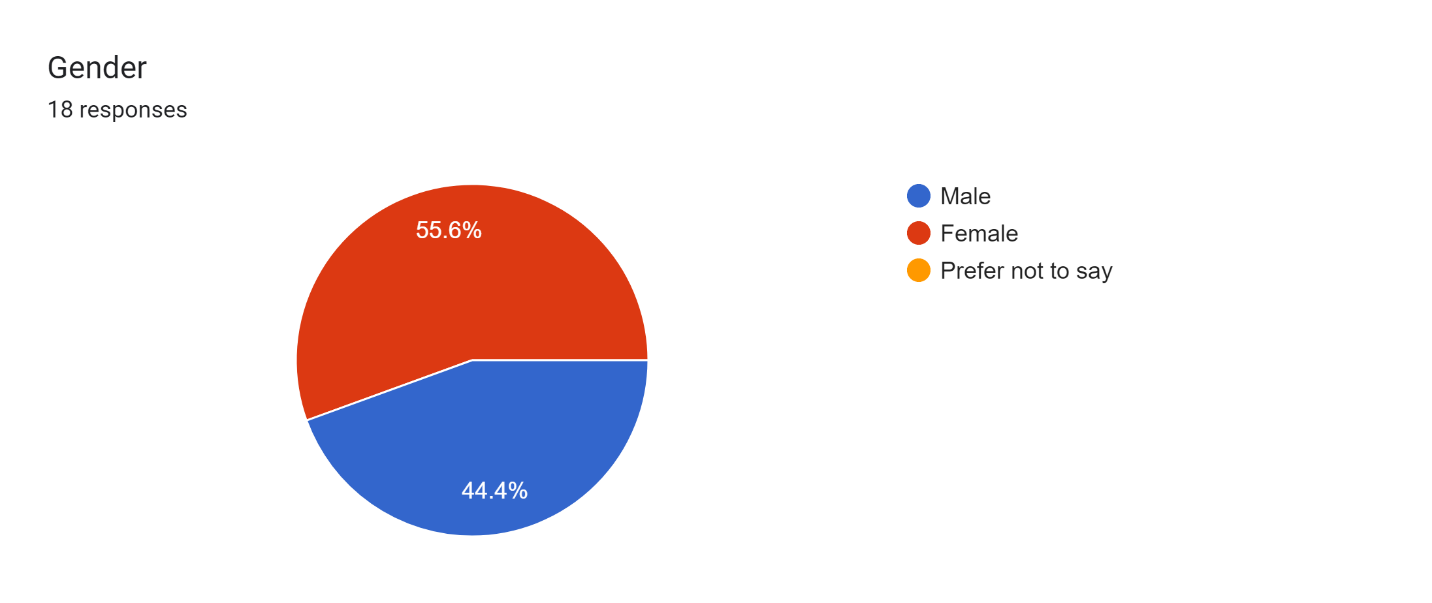
**Figure 10: GitHub – Repository Figure 11: GitHub - Branch**



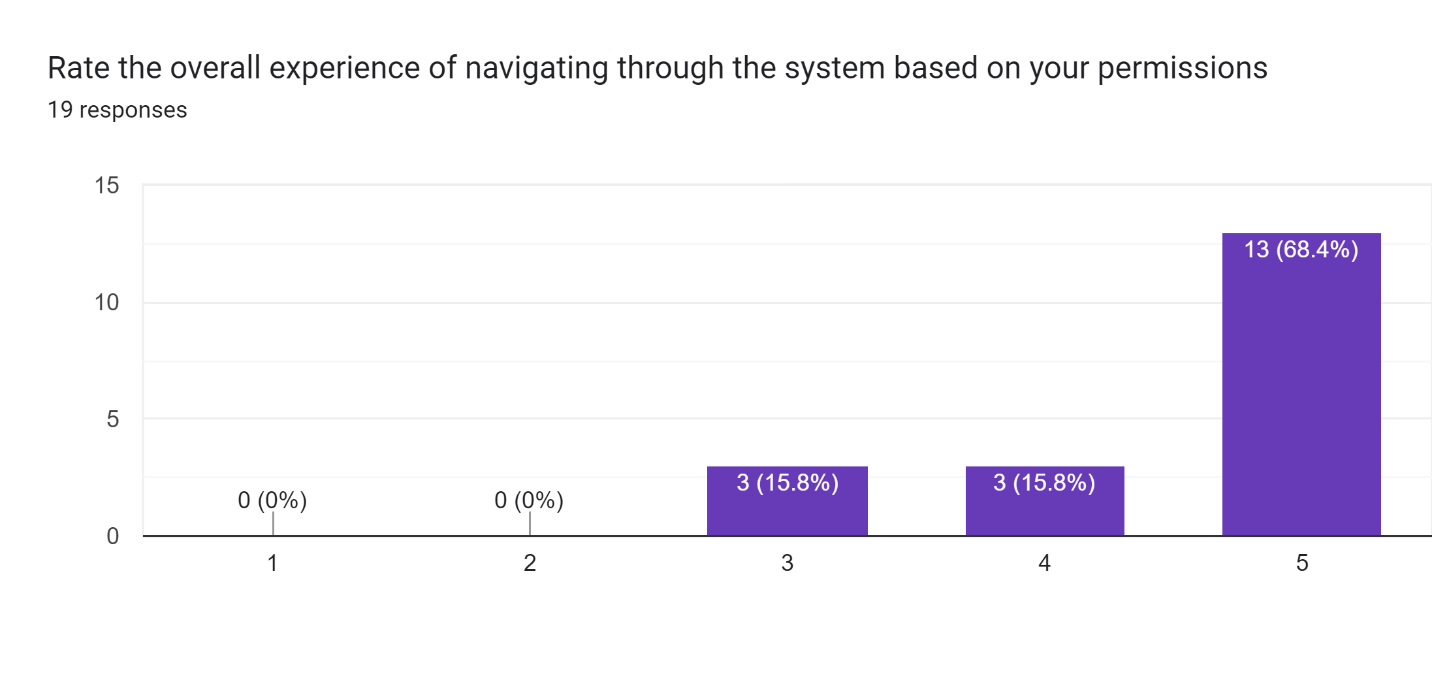
**Figure 12: Power BI Reports**

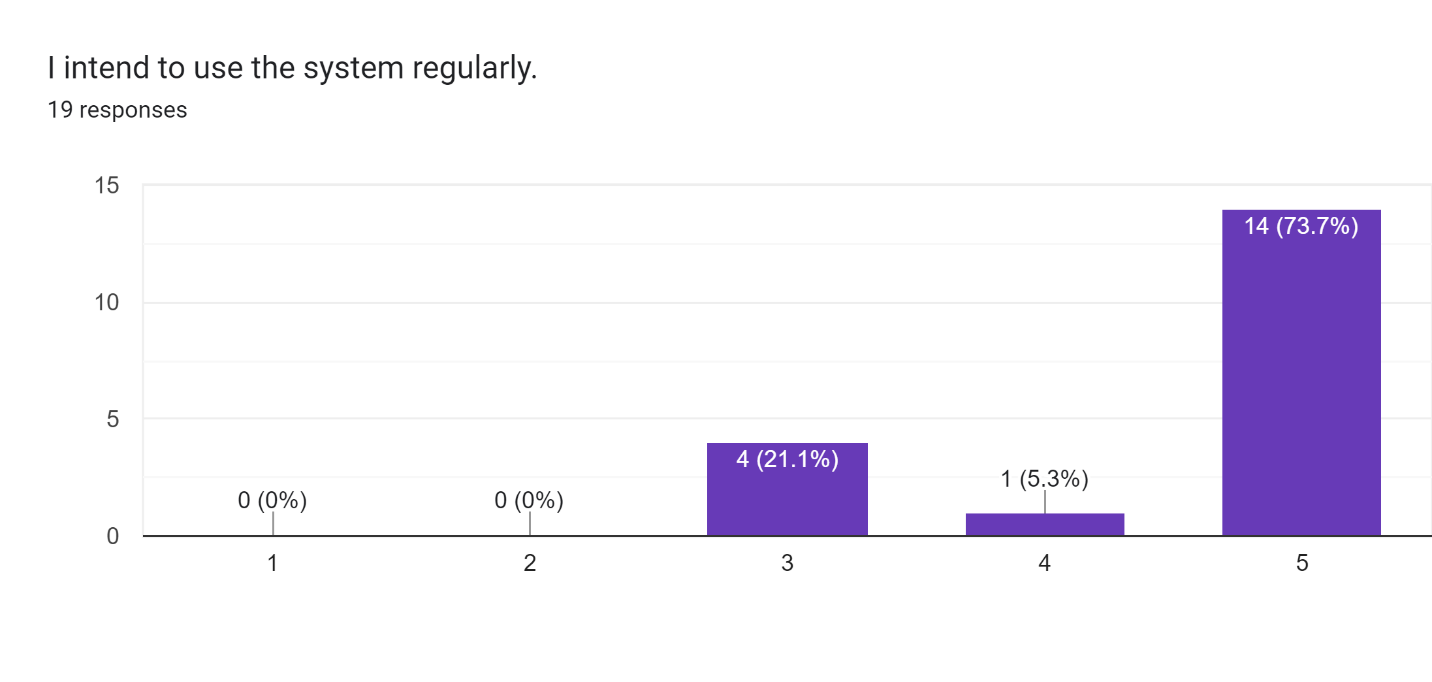


**Figure 13: Power BI Reports**

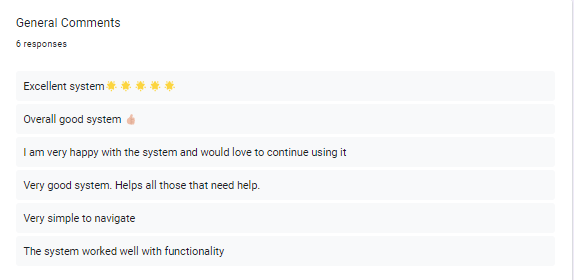


**Figure 14: Graph showing Gender response on Usability survey**

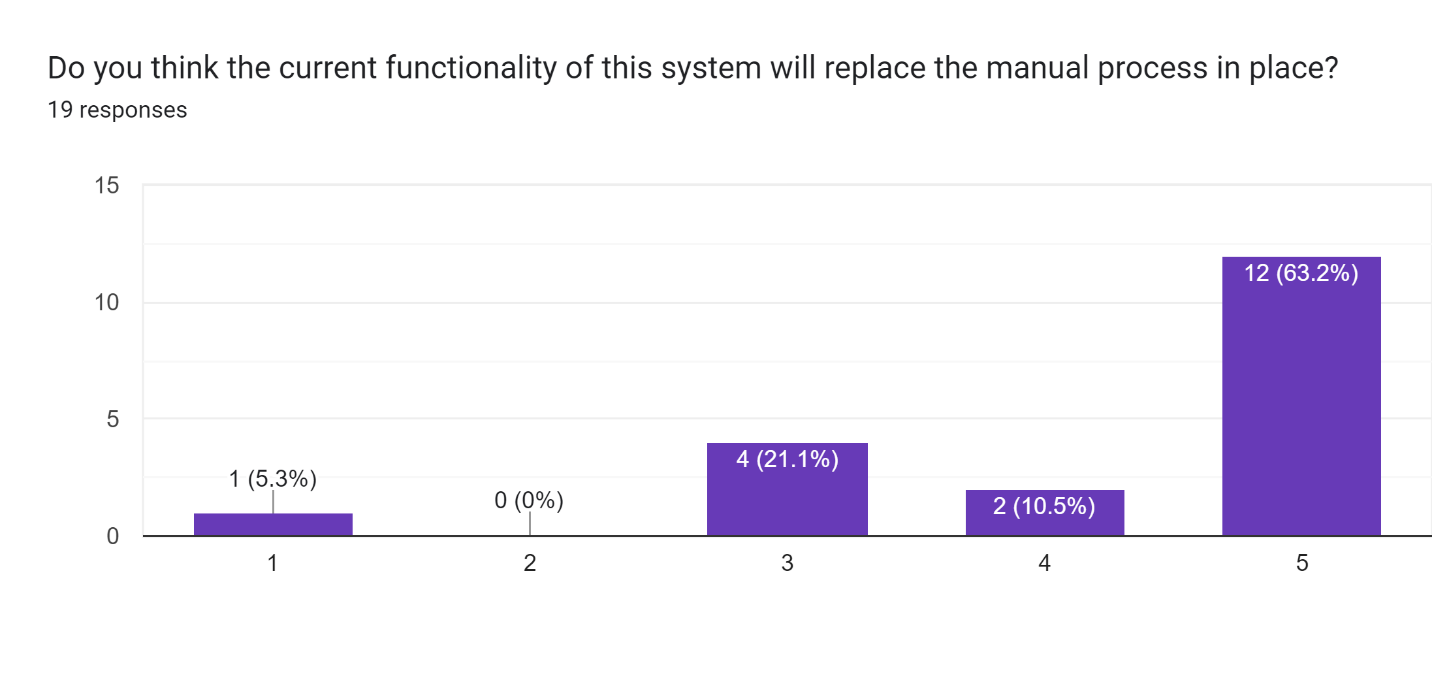




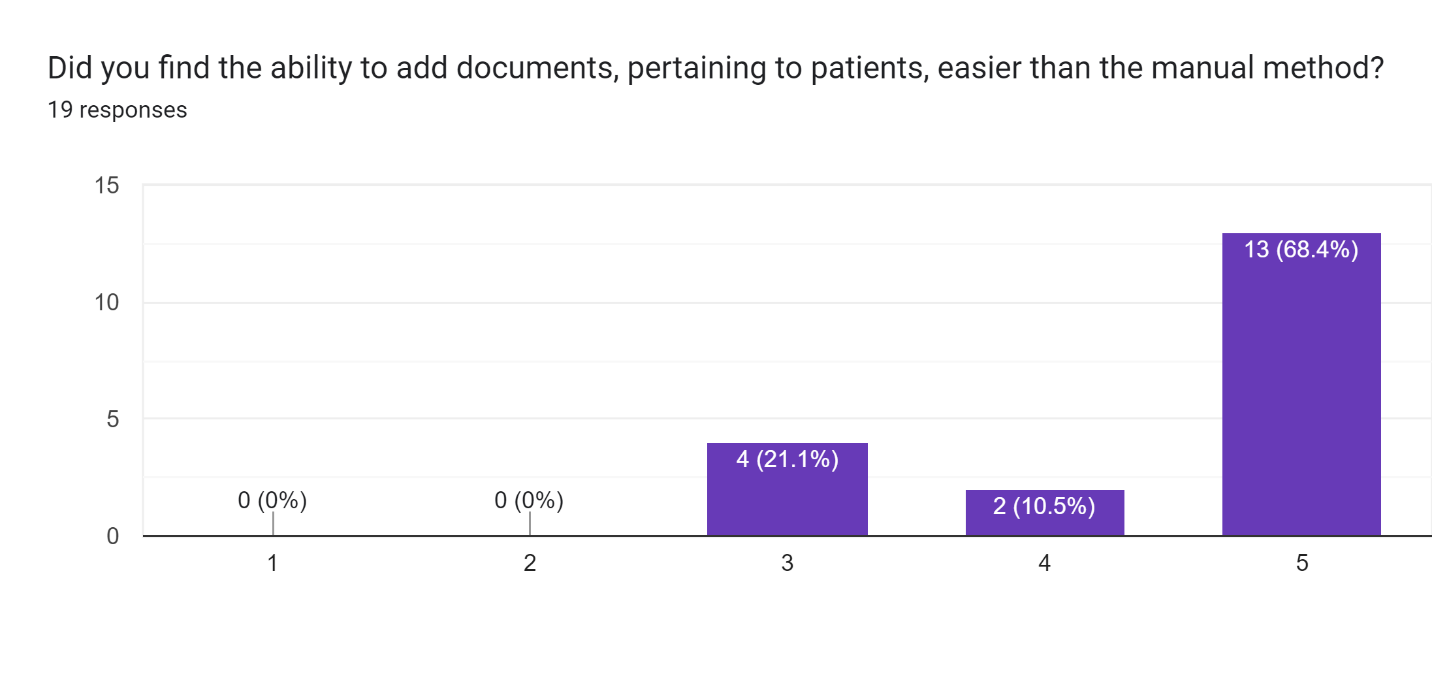
**Figure 15: Graphs showing Overall navigation experience**



**Figure 16: General Comments from Usability Survey**



**Figure 17: Graph showing manual work replacement**



**Figure 18: Graph showing functionality success rate**

### References

Al-Saqqa, S., Sawalha, S., & AbdelNabi, H. (2020). Agile software development: Methodologies and trends. International Journal of Interactive Mobile Technologies, 14(11). [semanticscholar.org](https://pdfs.semanticscholar.org/2fef/154748093288894dbd0b98db1b9b54731c71.pdf)

Beizer, B. (2003). *Software Testing Techniques* (2nd ed.). Dreamtech Press.

Bushong, V., Abdelfattah, A. S., Maruf, A. A., Das, D., Lehman, A., Jaroszewski, E., ... & Bures, M. (2021). On microservice analysis and architecture evolution: A systematic mapping study. Applied Sciences, 11(17), 7856. [mdpi.com](https://www.mdpi.com/2076-3417/11/17/7856/pdf)

Cadle, J., Paul, D., & Turner, P. (2014). *Business Analysis Techniques: 99 Essential Tools for Success* (2nd ed.). BCS, The Chartered Institute for IT.

International Institute of Business Analysis (IIBA). (2015). *A Guide to the Business Analysis Body of Knowledge (BABOK Guide)* (3rd ed.). IIBA.

Kasauli, R., Knauss, E., Horkoff, J., Liebel, G., & de Oliveira Neto, F. G. (2021). Requirements engineering challenges and practices in large-scale agile system development. Journal of Systems and Software, 172, 110851. [sciencedirect.com](https://www.sciencedirect.com/science/article/pii/S0164121220302417)

McGraw, K. L. & Harbison, K. (2020). User-centered requirements: the scenario-based engineering process. [[HTML]](https://www.taylorfrancis.com/books/mono/10.1201/9781003064138/user-centered-requirements-karan-harbison-karen-mcgraw)

Morandini, M., Coleti, T. A., Oliveira Jr, E., & Corrêa, P. L. P. (2021). Considerations about the efficiency and sufficiency of the utilization of the Scrum methodology: A survey for analyzing results for development teams. Computer Science Review, 39, 100314. [[HTML]](https://www.sciencedirect.com/science/article/pii/S1574013720304147)

Myers, G. J., Sandler, C., & Badgett, T. (2011). *The Art of Software Testing* (3rd ed.). John Wiley & Sons.

Opelt, A., Gloger, B., Pfarl, W., & Mittermayr, R. (2023). Agile contracts creating and managing successful projects with Scrum. [[HTML]](https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118640067)

Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill.

Rangarajan, S., Liu, H., & Wang, H. (2020). Web service QoS prediction using improved software source code metrics. Plos one. [plos.org](https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0226867&type=printable)

Shastri, Y., Hoda, R., & Amor, R. (2021). Spearheading agile: the role of the scrum master in agile projects. Empirical Software Engineering. [[HTML]](https://link.springer.com/article/10.1007/s10664-020-09899-4)

Sommerville, I. (2011). *Software Engineering* (9th ed.). Addison-Wesley.

Wilson, J. P., & Campbell, L. (2020). ISO 9001: 2015: the evolution and convergence of quality management and knowledge management for competitive advantage. Total Quality Management & Business Excellence, 31(7-8), 761-776. [[HTML]](https://www.tandfonline.com/doi/abs/10.1080/14783363.2018.1445965)

Zayat, W., & Senvar, O. (2020). Framework study for agile software development via scrum and Kanban. International journal of innovation and technology management, 17(04), 2030002. [researchgate.net](https://www.researchgate.net/profile/Wael-Zayat/publication/341318650_Framework_Study_for_Agile_Software_Development_Via_Scrum_and_Kanban/links/6211f935eb735c508ae3c161/Framework-Study-for-Agile-Software-Development-Via-Scrum-and-Kanban.pdf)