COMP3850 Project Deliverable Certificate

Name of Deliverable	*Project Plan, Quality Manual, Requirements/Scoping, Design & Testing Document*
Date Submitted	*29 / 04 / 2021*
Project Group Number	*21*
Rubric stream being followed for this deliverable (highlight one) *Note: the feasibility study has the same rubric for all streams.*	***SOFTWARE Rubric*** *GAMES Rubric* *CYBERSECURITY Rubric* *DATA SCIENCE Rubric*

We, the undersigned members of the above Project Group, collectively and individually certify that the above Project Deliverable, as submitted, **is entirely our own work**, other than where explicitly indicated in the deliverable documentation.

INITIALS	SURNAME	GIVEN NAME	STUDENT NUMBER	SIGNATURE *(IN-PERSON OR DIGITAL)*
FT	*Tesoriero*	*Flynn*	*45621365*	*Flynn Tesoriero*
SM	*Mishra*	*Shivansh*	*45854319*	*S MISHRA*
IL	*Lee*	*Isaac*	*45526249*	*LZJ*
MN	*NAQVI*	*Maham*	*45559708*	*Maham Naqvi*
SK	*Kamath*	*Smriti*	*44489935*	*Smriti Kamath*
MP	*Pavlov*	*Mikhail*	*43659691*	*MikhailPavlov*

NB: please write all details clearly (if handwritten).

List of tasks completed for the deliverable and activities since last deliverable certificate with totals for each individual team member and whole team *(copy individual total row for each member and copy pages if more pages needed)*

Performed by *(Student Names)*	Duration *(hrs)*	Complexity *(L, M, H)*	Name of task	Checked by *(Initials)*
Everyone	1	M	Met with IBM & Other Teams, discussed project (20th April)	FT
Everyone	1	M	Met with IBM & Other Teams, discussed project (27th April)	FT
Everyone	0.5	H	Met with team members to discuss progress (28th April)	FT
Flynn Tesoriero	1	L	Created document outline, headings, notes and formatted sections	SK
Flynn Tesoriero	1	H	Incorporated deliverable 2 feedback	SK
Everyone	15	H	Worked on prototype development	FT
Everyone	0.5	M	Updated revision history	SM
Flynn Tesoriero	0.5	M	Wrote handover requirements section	SK
Flynn Tesoriero	1	H	Wrote prototype section	SK
Flynn Tesoriero	2	M	Created use case diagrams and descriptions	SK
Flynn Tesoriero	1	H	Wrote user stories	SK
Flynn Tesoriero	3	H	Created ER diagram/data definitions	SK
Flynn Tesoriero	1	H	Created window navigation diagram	SK
Smriti Kamath	1	M	Researched Architectures	FT
Smriti Kamath	1	L	Purpose and Audience for SDD	FT
Smriti Kamath	1	M	Software and hardware architecture formulated	FT
Smriti Kamath	1	M	Explained the Concurrent processes	FT
Smriti Kamath	1	M	Briefly described the Trade-offs	FT
Smriti	2	M	Formulted User interface strategy	SM
Isaac Lee	1	M	Wrote System Design Document	FT
Isaac Lee	2	M	Created Activity Diagram	FT
Isaac Lee	2	M	Created Class Diagram	FT
Isaac Lee	2	M	Created State Diagram	
Shivansh Mishra	2	M	Worked on Test Plans	SK
Shivansh Mishra	2	M	Created Test Cases	FT
Maham NAQVI	2	M	Worked on Test Plans	SK
Maham NAQVI	2	M	Created Test Cases	FT
Shivansh Mishra	1	M	Completed and edited the Testing Document	FT
Maham NAQVI	1.5	M	Proofread and editions sections to ensure consistency	SM
Mikhail Pavlov	0.5	L	Added further test cases	FT

Mikhail Pavlov	1	L	Document review, minor edits	FT
Isaac Lee Total	25			
Shivansh Mishra Total	23			
Flynn Tesoriero Total	28.5			
Mikhail Pavlov Total	19.5			
Maham Naqvi Total	23.5			
Smriti Kamath Total	23			
Team Total	142.5			

Sankofa: A blockchain-based healthcare architecture platform

Project Plan, Quality Manual, Requirements/Scoping, Design & Testing Document

Project Plan

Revision History

Revision	Date	Person(s)	Changes
1	1/04/2021	All	Initial document
2	29/04/2021	All	Added handover requirements. Clarified team organisation and structure, as well as project resources. Expanded process model justification. Numbered assumptions and moved to Analysis And Design section.

Statement of Purpose/Scope/Description

The repercussions of the COVID-19 global pandemic have revealed the difficulties and problems associated with handling and sharing of sensitive healthcare data between different groups of organisations. One significant problem that requires addressing is the ability to track vaccine rollout plans and of its recipients. The Sankofa Healthcare Framework (SHF) project, therefore, at its core seeks to provide an efficient solution to this problem by leveraging blockchain technology and providing a single framework for the secure sharing and storage of healthcare data. This project strongly focuses on providing a reliable and verifiable means of tracking the vaccine rollout through the use of identity access management (IAM) and granular data access to ensure that users can only view, change and delete the data that they have been allowed access to. This project deliverable, in particular, focuses on the project development and quality management plan of the designed product, and also outlines the software requirements elicitation and specifications (both functional and non-functional) required to successfully design and implement the new software product in the healthcare industry.

The scope of this development project is very broad in terms of the clients of the system and the user audience involved. Some of these can be narrowed down to the healthcare organisations especially those that are responsible for monitoring and controlling the vaccine rollout, the general public receiving the vaccine, the security organizations responsible for storing sensitive medical data, the project manager and the software development team, and the sponsors of the project i.e the IBM team.

Risk Management

The importance of risk management to software development is greater now than ever before. This importance stems from the growing role that the software now plays in delivering value to customers. Research supports that most software projects fail due to unrealistic expectations set, poor management and other development, operational and external risks that are not acknowledged during the development process. Therefore, it is crucial in order to maximise the success of the project that the practitioner is aware of the risks in the pursuit of delivering software projects. These potential risks that may affect the Sankofa Healthcare Framework (SHF) project have been identified and recorded in the risk matrix/ table provided below which covers in detail the risks /hazards with their description, probability, severity, consequences and mitigation strategies suggested to eliminate the impact of known risks to an optimum level.

Risk Matrix Table

REF/ ID	RISK	DESCRIPTION	PROBABILITY	CONSEQUENCES	RISK LEVEL	MITIGATION STRATEGY
RI001	Inherent Schedule Flaws	Given the intangible nature and uniqueness of software development, it is inherently difficult to estimate and schedule a development timeline which may include unrealistic expectations from tight schedules.	**Frequent**; Likely to occur immediately or in a short period of time; expected to occur.	The consequences may include unsuccessful projects, poor quality projects, late delivered projects with poor quality work and unsatisfied clients.	**High**	Apply Agile methodologies, and ensure the maximum involvement of all team members in planning and development, receive feedback at all stages from the client starting from the earliest ones, involve the Stakeholders. Another solution may be an emergency expansion of the team to increase the development speed, however, this can significantly affect the project budget.
RI002	Incorrect Budget Estimation	With the wrong or untimely budget management, the project can be halfway completed or go far beyond the agreed cost. The main causes of cost risks in software development include incorrect initial budget calculation, no reserve funds, unplanned project scope expansion.	**Likely**; Quite likely to occur in time.	The consequences may include over-budget projects, unsuccessful or incomplete projects with not enough reserve budgets to cater to more important crucial product features or functionalities.	**High**	It is necessary to maintain constant control of the budget and development process to mitigate this risk. When introducing additional functionality or features – or the need for any new changes – calculate the cost at the discussion stage and reserve more budget for high priority requirements.
RI003	Inadequate Requirements	Requirements are incomplete, late or informal with insufficient detail due to bad timing or uninvolved user community.	**Likely;** Quite likely to occur in time.	The consequences include poor productivity and performance, failed projects, waste of money and time, failure to meet stakeholder expectations or requirements.	**Very High**	Apply agile methodologies, divide the project into different phases broken into smaller pieces and work with the unknown, make carefully planned assumptions based on the best knowledge on the domain, and communicate effectively.

RI004	Changing requirements	There are many changes during the lifespan of a project – the concepts, requirements, a number of tasks in a sprint and priorities all can change over the course of the project. Sometimes, the client may change their mind midway through the completion of a given module, which can drive up the cost as well.	**Frequent** Likely to occur immediately or in a short period of time; expected to occur	Overloaded or underloaded sprints, abandoned incomplete tasks, complete or partial revision of the software product, changes to the schedule, incomplete or extended sprints, sudden need of adding more people to the team, failure to deliver on time, increased budgets costs.	**Very High**	When any changes are made, carefully analyze what impact they will have on the current state of the project, how much effort it will take, or if there is a risk of delays. The detailed analysis will allow for an efficient division of tasks, updating priorities, and providing the client with accurate information on what can or cannot actually be delivered.
RI005	Unclear Business Problem / Success Criteria	Often the project may be moving forward while solving the wrong problem without stopping to ensure that it is appropriate for the true need, or even that the true need is understood. Another gap in understanding the business need is the lack of clear criteria to measure the success rate.	**Frequent** Likely to occur immediately or in a short period of time; expected to occur	It is difficult and highly unlikely to be successful without clarity on what is being developed. Projects experiencing this problem are unable to articulate what constitutes success for the project, meet the project goals or stakeholder and client expectations.	**High**	Conduct detailed analysis, questionnaires, interviews, ask the right questions and try to get ahead of any assumptions or miscommunication. Establish clear roles and deadlines for project members, with correct success criteria and requirements, map out a chain of communication.
RI006	Operational / Management Risks	Day-to-day operational activities might hamper due to improper process implementation, conflicting priorities, or a lack of clarity in responsibilities.	**Occasionally** May occur in time	High levels of operating losses and costs over the course of time, untimely and quick degradation of the product, poorer performance, obsolescence of the product over time.	**Moderate**	Set achievable timeframes and a sustainable work pace during your project estimates to avoid burn-out of staff, appoint a dedicated product manager who is directly involved and regularly collaborates with the team.
RI007	Poor Code Quality & Technical Risks	Poor quality code is difficult to read, or make changes to and can occur when projects are rushed. Technical risks can occur due to constant requirement changes, technology with insufficient functionality, too complex a project.	**Occasionally** May occur in time	Increased risk of technical debt, reduced functionality of software, code highly susceptible to cybersecurity attacks,	**Moderate**	Use flexible risk management, do intensive testing and review of source code, establish clear coding standards and guides, implement well-defined user acceptance criteria, use technology that is not in development stages with sufficient support available.

RI008	Unavoidable External Risks	External risks are dangerous due to their low unpredictability. They are outside the control of the project team and its host organisation and can include key vendor going bankrupt, economic upheaval, the fast growth of a competitor, implementation of new government regulations, or changes in consumer behaviour and priorities.	**Seldom** Not likely to occur but possible	Recession, increase in interest rates, low product selling and usage due to other competitor advantages, product becoming unlawful and restricted from deployment, complete cut-off, inflation	**Low**	It is crucial that developers have the best business analyst in the area of their market to back-up ideas and eliminate the risks created by these external factors. Other mitigation strategies may include cutting costs or diversifying the client base so that revenue is not solely reliant on one segment or geographic region, and using insurance

Resource Management

Project Resources

A great number of resources are available to the SHF project. First and foremost are the people resources available. As a team of six, the Admin UI team is aligned to deliver a successful element of the SHF project: the administration user interface. Supporting the Admin UI team are mentors from IBM, who will help guide the project and offer assistance when required. Weekly meetings with the IBM team and the Admin UI team helps to ensure that everyone is aligned and a common understanding of the project and progress can be achieved. Mentors from Northwestern University also assist in this guidance process and further help provide direction and alignment.

In terms of hardware resources, each team member has access to their own personal laptop. Further, Macquarie University provides computers should one be required. Working as a distributed/remote team, the hardware required to successfully deliver the project is simply a computer such as a laptop. This is made possible through a number of softwares and technologies.

One such software resource that the team shall leverage is IBM cloud. This service allows the system to run and be developed in a cloud environment, meaning the project can be worked on, tested and improved from anywhere in the world. The use of cloud-based version control software, GitLab, also allows this to be achieved. Visual Studio Code is the Integrated Development Environment (IDE) that shall be used as it is open-source, extensible and free. Finally, Slack is another software that shall be used as it allows fast and efficient communication, both inside the Admin UI team and externally to other teams and IBM.

A number of open-source software frameworks will also be used for the SHF system. These include React, which is a front-end JavaScript framework. Another open-source framework that will be leveraged is Carbon UI, which is a user interface framework developed by IBM that speeds up the development process and provides a collection of UI components that can be easily used.

These resources shall all be combined and leveraged to successfully deliver the project through holistic and systematic asset management systems that cohesively combine physical, financial and contractual attributes of software to enable cost efficient timely solutions and minimize operational risks to the organization.

Many of these software resources are freely available through the Macquarie University login portals, or the provided IBM Cloud environment.

Team Organisation/Structure

The following table delineates the organisation and structure of the Admin UI team. The team has been structured and roles assigned to take advantage of the strengths each team member contributes to the team.

Name	**Roles/Responsibilities**
Flynn Tesoriero	Project Manager
Shivansh Mishra	UX Analyst
Isaac Lee	Project Analyst/System Analyst
Maham Naqvi	Risk Analyst/Administrator
Smriti Kamath	Business Requirements Analyst
Pavlov Mikhail	Functional Analyst
Ron Majumdar (IBM)	Subject Matter Expert

The team members bring a variety of experience in both software engineering and business. These various skill sets combine to create a high-performing team with experience in all the necessary project areas. These skills include requirements gathering and analysis, user interface design, software development, as well as marketing and operations experience.

The external team members and mentors from IBM greatly enrich this team with their vast knowledge and experience, helping to fill any skill gaps and provide assistance. IBM can be communicated with via the provided Slack channels and through the weekly support meetings.

Project Schedule

Tasks/Activities/Phases

A number of tasks and activities are involved in the successful development, delivery and completion of the SHF project. One such product, which has already been delivered, is the Feasibility Study. The aim of this study was to ensure that the proposed system was viable and addressed a problem. The study was structured to follow a process of identifying the problem the system aimed to solve, as well as assessing the opportunities, current situation and benefits of the system. Then a solution was proposed in the study. This study was undertaken by all members of the group, with different members roles contributing to different parts of the study. The division of this can be found in the below Gantt chart. This document was delivered on 11/03/2021.

Another essential task was the delivery of the Project Plan, Quality Manual and the Software Requirements Specification (SRS) documents. These documents (this current document) provide a clear plan for moving forward, a clear picture of quality expectations and processes, as well as the requirements that must be addressed in the system. All team members were involved in developing these documents, with the task allocation available in the below Gantt chart. These documents were delivered on 1/04/2021.

The updating of the documents discussed above is critical as the project progresses, especially as the problem domain and solution becomes more understood, it is important for these documents to reflect the most up-to-date information. As such, the Project Plan, Quality Manual and SRS documents shall be updated and re-delivered by 29/04/2021 to reflect the most up-to-date information. By this same date, the first iteration of the project shall also be delivered. This shall include a prototype or minimum viable product of the administration user interface system. Document around the design of the interface shall also be delivered, along with testing documentation. In line with the previous deliverables, the entire team shall contribute to these products, with the allocation of tasks available on the Gantt chart.

Following this, updated design and test cases shall be delivered on 20/05/2021. These products shall be worked on according to the allocation of their previous version. An updated iteration of the product shall also be delivered on this date, incorporating new changes and feedback from the project sponsor. This iteration, as with the first iteration, shall be contributed to by the whole team.

Following this, a reflective report and presentation shall be delivered on 5/06/2021. The entire team shall be involved in these activities, especially as they are reflective in nature. Following this, the final iteration of the product shall be delivered after this date, but before Thursday of Week 16. This final iteration shall incorporate all of the feedback from the project sponsor and adhere to the documentation produced, such as the requirements of the system. As with previous iterations, the entire group shall be working on this activity.

We will go through Agile Development as the golden standard to the SHF Administration UI project. It is a collaborative decision-making process between clients and team members, in order to meet its requirements and provide solutions. The method gathers user requirements

for software development based on practices, values and principles. This will allow team members to commit to take feedback and continue to make improvements.

There are four values of Agile Development: Communication, Simplicity, Feedback and Courage. Those are the importance of the methodology created short and long-term which act together on software development.

The first stage is Communication. The collaboration between IBM team and team members is an essential component that requires consistent updates that are prone to errors by developers to meet tight deadlines.

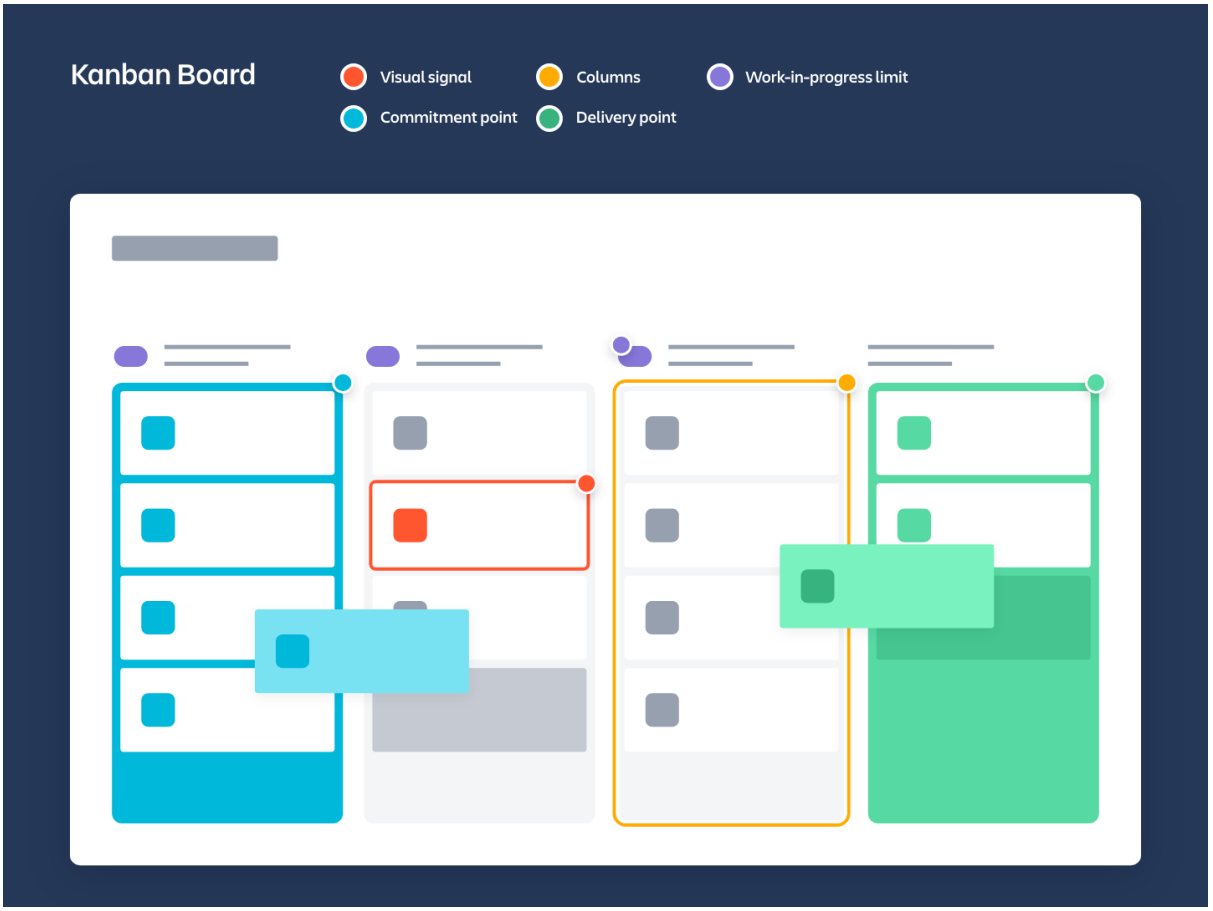
The second stage is Simplicity, having it can bring better clarity by doing the simplest approach when developing a project. It makes team members understand and receive a clearer focus and objective of the project at a basic level.

The third stage is Feedback. Every design will always have feedback when it comes to programming or developing a user interface. It is crucial to have important concrete feedback, and clients will always occur within a given time frame when it is needed. The clients can provide critical functional feedback for all developers to implement better solutions. It gives better adjustments and enables progression during tight schedules at an early stage.

The fourth stage is Courage. Team members having the courage to respond to critical feedback allows them to accomplish better goals. It can be a high-risk which is high-reward, giving better experimentation in an innovative way to its objective. As a result, this will strengthen the relationship and trust among team members to achieve the client's requirements in terms of improvements for the project. Even if the team members have to change the code or implement new solutions, it shows encouragement to strive for a better goal.

Implementing the Agile methodology as well as Kanban instead as opposed to the traditional Software Development Life Cycle is well suited to develop the SHF project, especially given the fluid and ambiguous nature of the requirements. This approach will allow rapid changes through the project phases. It also gives the flexibility for developers to adapt to new information.

We decided to use the Kanban board which is an agile project management methodology to maximise work efficiency to reach optimal goals on a daily basis. The board gives an easy to interpret visualisation, which gives team members total visibility of what task they need to complete at a given time. This was chosen over Scrum due to its simplicity and less rigidity, especially for a small team with varying schedules.



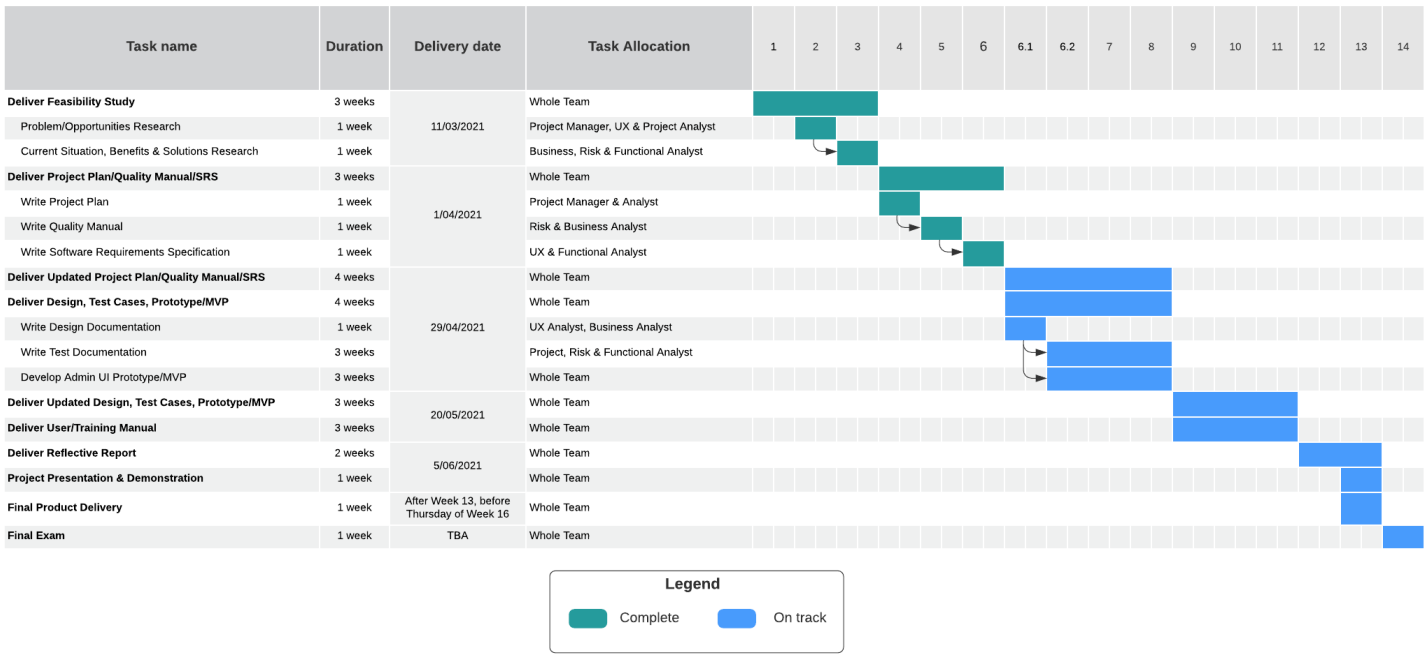
Source: https://www.atlassian.com/agile/kanban/boards
The features of a Kanban board (as above) include:

- Visual Signals are memos where all team members can write on the dashboard to stay on the same page and keep updated on the current status of the project.
- The workflow has a column like a timeline where it displays information to everyone the specific activity such as To-Do, In-Progress, Completed List, etc, shown on the diagram below:
- Work In Progress can only go up to three cards maximum in a column as team members are required to finish the tasks in order to proceed to the next. This kind of method can expose bottlenecks in the workflow. It can also be an indicator if you committed too much to a specific section.
- Kanban board has the commitment point where it has the backlog where clients and team members share ideas for the projects and ready to commit the project when it's finalised
- At the end of the workflow is to take cards to the delivery point as fast as possible. The purpose is to decrease lead time as much as possible

The advantage of Kanban is it's always on the dashboard, which always displays information to clients and team members on what tasks need to be completed. It optimises the software itself to help companies to learn and keep track of progress. Having digitised compared to physical, it keeps it tidy and saves space in the workplace. It is also accessible remotely.

Timeline

Task name	Duration	Delivery date	Task Allocation
Deliver Feasibility Study	3 weeks	11/03/2021	Whole Team
Problem/Opportunities Research	1 week		Project Manager, UX & Project Analyst
Current Situation, Benefits & Solutions Research	1 week		Business, Risk & Functional Analyst
Deliver Project Plan/Quality Manual/SRS	3 weeks	1/04/2021	Whole Team
Write Project Plan	1 week		Project Manager & Analyst
Write Quality Manual	1 week		Risk & Business Analyst
Write Software Requirements Specification	1 week		UX & Functional Analyst
Deliver Updated Project Plan/Quality Manual/SRS	4 weeks	29/04/2021	Whole Team
Deliver Design, Test Cases, Prototype/MVP	4 weeks		Whole Team
Write Design Documentation	1 week		UX Analyst, Business Analyst
Write Test Documentation	3 weeks		Project, Risk & Functional Analyst
Develop Admin UI Prototype/MVP	3 weeks		Whole Team
Deliver Updated Design, Test Cases, Prototype/MVP	3 weeks	20/05/2021	Whole Team
Deliver User/Training Manual	3 weeks		Whole Team
Deliver Reflective Report	2 weeks	5/06/2021	Whole Team
Project Presentation & Demonstration	1 week		Whole Team
Final Product Delivery	1 week	After Week 13, before Thursday of Week 16	Whole Team
Final Exam	1 week	TBA	Whole Team



A large view of the above Gantt Chart timeline can be viewed here: https://lucid.app/documents/view/e2d87830-d28a-43dc-8223-39492ce9b9f7

Handover Requirements

A success project handover is a critical step in the successful delivery of a software project. The advent of cloud-based project management and collaboration tools has meant that project handovers can be completed with minimal friction. Source code, algorithms and scripts are stored in a Git repository hosted by IBM in IBM Cloud. Not only does this make it easy to collaborate with many different teams and stakeholders, but it also makes the handover of project source code simple. As IBM already has access to this source code, no extra steps are required to ensure the project sponsor can access these materials. Testing tools and environments are also stored within IBM Cloud, making it trivial for IBM to access these resources. Documentation such as test cases and other information is stored in Box, a shared cloud storage service. This shared cloud storage space contains all relevant documentation for the project, accessible to all stakeholders. This ensures that the handover process is seamless. Information on project progress, issues and features remaining to be built is stored within the project management tools of the GitLab repository. As with the source code, this repository is accessible to all stakeholders involved with the project. In summary, all aspects of the project are already accessible to IBM, making the handover process seamless. No specific requirements or tasks exist to ensure the project is handed over successfully, as the project is in a sense continually handed over to the client through being stored in a cloud-based environment.

Quality Manual

Revision History

Revision	Date	Person(s)	Changes
1	1/04/2021	All	Initial document
2	29/04/2021	All	Added review, audits and testing timelines.

This quality manual sets out the quality standards applied to the SHF Administration UI project. This manual sets out the key concepts behind quality control and management. It then goes on to explain the review, audit and testing processes. Tools for managing quality are then discussed, as well as tracking and change management. Communication tools are then discussed. Finally, conflict resolution and negotiation processes are covered, as well as standards, templates and forms.

Quality Control and Management

Quality control and management are essential components of successfully developing and launching a software system. Quality Management involves setting down agreed-upon standards and specifications. These standards and specifications ensure that everyone is aware of expectations and follow a consistent set of rules. A number of quality management standards are set out in this quality management, such as following a code style guide, using Git branches effectively and utilising linting tools.

While quality management is concerned with preventing quality breaches, quality control seeks to detect quality defects before they become a greater issue. Quality control, therefore, involves the evaluation of the project against a set of standards. Quality Control standards and procedures are also discussed in this quality manual, namely the use of code reviews, audits and automated testing.

The application of both quality control and quality management standards, processes and procedures will help ensure that the product produced is of a high level of quality. This is essential, given the critical nature of the product. As a system handling healthcare-related information, it is of the utmost importance that the highest level of quality standards are achieved. Errors in the information conveyed, or the display of unauthorised personal information would have greatly detrimental effects. To this end, the system will need to conform to the US Health Insurance Portability and Accountability Act (HIPAA) rules around healthcare information security, due to its application to US-based healthcare providers. Further, the system will also comply with HITRUST CSF, a data security and privacy framework. By ensuring compliance with these regulatory frameworks and standards, the quality of the system can be assured.

Reviews and Audits, Testing

A form of Quality Control (QA) is undertaking reviews, audits and testing.

In the context of this software development project, reviews take the form of code reviews of newly developed code. Before code can be incorporated into the *master* branch of the project, code must be reviewed for quality, security and bugs. Having another developer review code before it becomes part of the system helps provide a fresh perspective and can also reveal better or more efficient ways of accomplishing a specific task, as well as providing quality control. Code reviews will follow the practices set forth by Google in their *Engineering Practices Code Review Developer Guide*. This process document sets forth concepts which code should be reviewed against, such as design, functionality, complexity and documentation. The process document also details the process of a code review, from submitting a pull-request for the newly changed code, how to provide feedback on the code and approving pull-requests. It is important to note that first and foremost, code should follow the style guide (discussed later in this quality manual) to ensure that the code meets a consistent standard.

While code reviews are performed before code becomes part of the master version of a system, code audits are performed on existing code. Code audits are essential as their aim is to catch bugs before they have an impact on the end-users of a system. As such, code reviews are essential in maintaining the quality of a system and form an important part of quality control. It is recommended that code reviews are undertaken on the system at a regular interval, such as monthly or quarterly. These reviews shall be undertaken by different developers each time. This not only ensures that the code is reviewed from a fresh perspective each time but also helps developers working on the project become more familiar with the codebase. A static analysis tool is also recommended, which will analyse the code at regular intervals to detect bugs and potential issues proactively and automatically.

Code testing also forms an integral part of quality control. Code testing involves writing unit tests to verify that each component and function of a system produces the correct output for different types of output. These tests are written as the functions they test are written, ensuring that tests cover the entirety of the codebase. Tests are especially useful in testing how different components of the system handle different types of input, particularly unexpected input. As such, automated unit testing is critical in avoiding unexpected cases in the system, and ensuring that these cases are handled gracefully with minimal impact to the user.

These reviews, audits and testing should be undertaken swiftly. Reviews shall be completed within a week of being received, as should audits. Testing should be undertaken before code is committed to the repository and is included in development time. These short timeframes ensure that changes can be rapidly made.

Tools for Managing Quality

Previously Quality Control measures have been discussed. These measures are focused on maintaining quality after code has been written or changes to the system have been made. Quality Management (QM) focuses on proactively avoiding quality issues by following a set of standards, procedures and policies. In the context of the SHF project, a number of tools for managing quality have been implemented, such as the application of a code style guide, the use of a linting software and utilising a Git branching workflow.

Given the flexibility of modern programming languages, it is essential that developers use the same coding styles to ensure consistency and learnability throughout a codebase. This is especially relevant to JavaScript, HTML & CSS, the languages that will be used in this system. This is because these languages provide high levels of flexibility and each developer has their own coding style. This can lead to codebases that are fragmented, hard to understand and inconsistent, making it harder for future developers to comprehend and learn quickly. To remedy these issues, a coding style guide shall be used to ensure that the code is consistent throughout the codebase. These style guides will be enforced through the code review and auditing processes, as discussed previously. These style guides form the core reference material for developers working on the project and helps avoid ambiguity by providing a consistent reference manual. Style guides include guidance on how a developer should structure their code and files, and name variables and functions, amongst other topics. As multiple frameworks and programming languages will be used by the project, a style guide will exist for each language/framework. For React, the frontend JavaScript framework, the Airbnb React Style Guide shall be used. This style guide has become the industry standard style guide for React project, meaning many developers are already familiar with this style guide. This means future developers will find it easier to adhere to the React coding styles for this project. For HTML and CSS, the Google HTML/CSS Style Guide shall be used. Similar to the Airbnb style guide, this guide places the project in line with industry standards.

An important complementary tool to style guides is using linting software. Linting refers to automatically applying certain rules to code as it is being written. These rules uphold the concepts in the style guide and help ensure that developers are following the style guide automatically. This further helps in ensuring a high level of code quality by creating consistent rules that are enforced automatically.

Another key tool for managing quality is following a standard Git workflow. Git is the industry-standard code change tracking software. Having a number of developers and teams working on a single codebase can quickly create complexity around version control, necessitating a centralised system to manage changes. The use of Git will be discussed in the following section of this quality manual, though the branching policy shall be discussed in this section. As developers make changes to the code of a system, it is important that these changes cannot be accidentally committed to the master version system before the necessary reviews, audits and tests have been completed. To achieve this, a git branching workflow is to be followed. Instead of working from a single version of code, branches can be created to allow each developer to work on aspects of the code independently before submitting a pull request to have this code be integrated into the master version of the system. As many teams are working on the various aspects of this project, a branch of the

master code exists for each team. This allows teams to work independently without breaking other teams' code, but also incorporate the changes other teams have made when the code is ready (reviewed, approved etc.). In addition to maintaining a branch for each team, a branch shall be created for each new feature being implemented in the system. While this feature is in development, changes will be committed to this feature branch. Isolating the feature like this allows other developers to work on their own features in their own feature branch, whilst also ensuring that the code in the teams' branch does not contain half-built features that may not yet be fully functional. When a feature is complete, a pull request shall be made, and after the necessary approvals, this feature will be incorporated into the team's branch, and later the master branch. This branching workflow is an industry-standard approach to collaborative development and is referred to as the *Git Feature Branch Workflow*. Implementing this branching workflow helps ensure that code quality is appropriately managed and tracked and helps ensure that the necessary approvals are made before code is committed to the production environment of the system.

Tracking/Change Management

As discussed in the previous section, the use of version control is essential in a large software project as the SHF system. Git allows each developer to work on the system independently, as well as collaborating with each other through the use of branches. The branching workflow for this project was discussed in the previous section, with the *Feature Branch Workflow* being chosen as the standard for this project. Git also offers a number of other advantages that benefit the project. One such advantage is being able to audit which developers made a particular change, which is helpful during the audit and review process. Git also aids in resolving merge conflicts, which occur when conflicting changes were made to a single file. Git includes a system by which these conflicts can be resolved.

The Git source control system for this project will be hosted in GitLab, a popular git hosting service. GitLab provides a web GUI for graphically managing the branches and pull requests in the project, whilst also serving as a cloud-based repository. This means the code is backed up and accessible anywhere. GitLab also contains a project management and task tracking system. This system shall be used to track development progress, tasks and bugs that need addressing. Tracking these inside GitLab ensures that everyone working on the project has visibility over the priorities and the next work items to complete.

In addition to utilising a branching workflow, the project also follows a commit message convention. Commit messages are added by developers working on the project each time they publish a collection of code changes to their respective branch. This message is a summary of what was changed and why. Utilising a tool called *Commitlint*, these commit messages must follow a predefined convention to automatically categorise and better describe the contents of each commit. This makes it easier for other developers to interpret each commit more quickly and assists with quickly understanding the changes made to the project.

Changes to the scope of the deliverables outlined in this document that are deemed likely to have an adverse impact on the timeline of the project shall be conducted in accordance with the process below:

1. Input: Need of change
2. Describe and document the need
3. Document the suggested change in the Change Log
4. Document justification and impact of change
5. Present the change to the group via agreed communication channels to provide approval or non-approval
6. If approved:
    a. Document approval in the changelog
    b. Update plan and other documentation if required
    c. Inform involved project members
7. If not approved, document in the changelog and attach documents.

Communication

Communication is essential to the success of the SHF project. Given the many developers on the admin UI team, as well as the other teams and their members working on the project, it is essential that a common communication platform exists. Slack has been chosen for this project. Each team working on the SHF project has their own channel, which is used to communicate within the team. A larger Slack channel containing the team members from all teams is also used, allowing cross-team communication and announcements. This structure allows the remote teams to communicate effectively, especially given the time zone differences across teams and the project sponsor.

Expectations around communication also exist, such as providing updates on what each team member is contributing and any questions they may have. Communication is also conducted through GitLab, where discussions are had within the project management tool. By recording communication on each task inside GitLab, it is ensured that all communication related to a specific task is in a single place. This helps each team member to keep track of the project as it evolves and forms a single source of information when a team member begins to work on a particular task.

Further, communication is also carried out through git commit messages and pull requests. Similar to recording communication on a work item in GitLab projects, communication about a particular commit or pull request can be found in a single location. By using these tools, the conversation around a particular component of the project can be centralised and easily found by whoever needs to access a particular piece of information, such as a pull request, work item or a commit.

Segmenting communication through these different channels helps reduce information overload and ensure that knowledge is easily shared and found within the project.

Conflict Resolution/Negotiation

It is essential to the success of the project that everybody working on the project is aligned and on the same page. At times conflicts are unavoidable and can be a positive experience for all involved if resolved and negotiated correctly. It's essential that all members of the team are heard, respected and treated equally. Group discussion and collaboration are key to the success of the project.

If a conflict does occur, start by working out exactly what the conflict is. It's important to understand the source and essence of a conflict before attempting to solve it. This applies to all parties in the conflict, and a good faith attempt from all parties should be made to follow this guidance. Once the source of conflict is understood by all parties, the parties shall meet (virtually) to discuss the conflict and attempt to come to an agreement. All sides and opinions should be heard, and a good faith attempt should be made to come to an agreement. Each party should be willing to compromise in the interest of the project's success. If discussions between the parties involved in the conflict should fail to provide a resolution, a wider group discussion shall occur between all members of the group. Those not involved in the conflict shall mediate the discussion and attempt to steer the conflicting parties in the direction of a resolution. Should this fail, the issue shall be escalated to either the project sponsor for their input and for a final decision.

This process for resolving conflicts is adapted from Google's Standard of Code Review. It is hoped this process will provide a respectful and positive negotiation and conflict resolution tool should such conflict occur. Additional information can be found in the following section.

Standards/Templates/Appendices/Forms

Quality Control and Management Standards
HIPAA: https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html
HITRUST CSF: https://hitrustalliance.net/product-tool/hitrust-csf/
ISO 13485:2016: https://www.iso.org/standard/59752.html
ISO 27001: https://www.iso.org/isoiec-27001-information-security.html

Code Review, Audits & Testing Guides
Google Engineering Practices Code Review Developer Guides:
https://google.github.io/eng-practices/review/
https://google.github.io/eng-practices/review/reviewer/standard.html

Code Style Guides
Airbnb React Style Guide: https://airbnb.io/javascript/react/
Google HTML & CSS Style Guide: https://google.github.io/styleguide/htmlcssguide.html
Commitlint Documentation: https://commitlint.js.org/

Tracking/Change Management Standards
Git Feature Branch Workflow:
https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow

Conflict Resolution/Negotiation Guides
Google Respect Guides:
https://chromium.googlesource.com/chromium/src/+/master/docs/cr_respect.md
https://chromium.googlesource.com/chromium/src/+/master/docs/cl_respect.md

Requirements Document/ Scoping Document (SRS)

Revision History

Revision	Date	Person(s)	Changes
1	1/04/2021	All	Initial document
2	29/04/2021	All	Sorted definitions alphabetically. Clarified documentation locations and forms. Improved functional and usability requirements consistency and clarity.

1. Introduction

IBM, through a collaboration with Northwestern University and Macquarie University, would like to develop a platform for sharing healthcare information. The platform would allow the secure sharing of healthcare information between various parties, ensuring the validity and accuracy of the information shared. The platform would be initially used for the purpose of tracking the COVID-19 vaccine rollout.

1.1 Purpose

The purpose of the Sankofa Healthcare Framework (SHF) is to provide a platform for the sharing of healthcare information between a multitude of stakeholders. The platform has a specific focus on tracking the vaccine rollout, facilitating the tracking and verification of vaccinations for particular individuals.

This Software Requirements Specification (SRS) document outlines the purpose, description and requirements of the SHF project, particularly from the Admin User Interface (Admin UI) perspective. The document follows the structure set out by IEEE standard 29148:2011 [1].

1.2 Scope – including a context diagram (Level 0 Data Flow Diagram)

As the Administration User Interface (Admin UI) team on this project, the scope is restricted to the administration interface components of the system. As such, the primary product that will be produced by this team is a user interface that will allow the administration of the system.

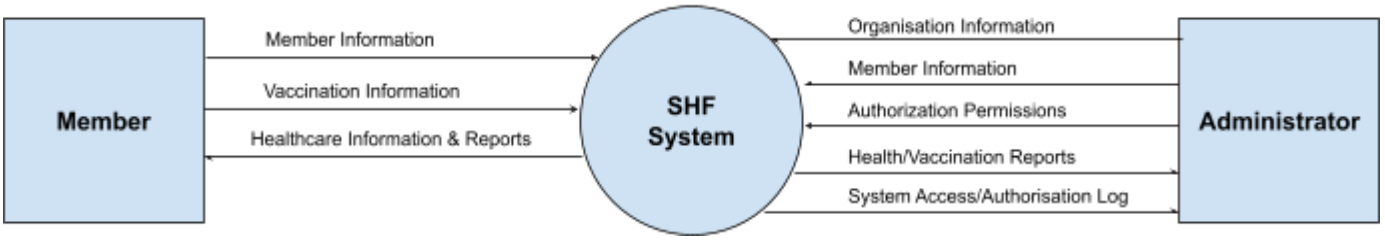
As identified above, the system will provide a framework for sharing healthcare information between numerous parties, particularly vaccination information. A number of teams are working on each component of the system, such as the vaccination user interface, API's, security, blockchain and DevOps.

The objective of the project is to ease the sharing of information between healthcare providers and provide a secure and verifiable way of tracking the vaccine rollout. This will provide a number of benefits, such as increasing the security and transparency of shared healthcare information, as well as creating a secure means of checking if a person has received their vaccinations. A goal of this project is to expand its functionality to allow the

tracking and sharing of various different types of healthcare information. Another goal is to onboard many different organisations at varying levels, such as government, healthcare providers and hospitals.

A number of technologies are being used by the SHF project, most notably blockchain. For the administration user interface component, React will be used as the frontend JavaScript framework. Carbon UI will be used to form the user interface.

Two primary actors are applicable for the administration UI: administrators and members. These are represented in the below Level 0 Data Flow Diagram (context diagram). Administrators have elevated permissions and can create, edit and delete organisation and member information, as well as controlling authorisation permissions. Members will be able to view information stored in the system that they are authorised to access, as well as create, edit and delete this information.



Level 0 Data Flow Diagram/Context Diagram

Above is a Level 0 Data Flow diagram showing the two primary actors interacting with the system. An additional diagram can be seen in the appendix below, depicting a wider context for the interaction between the various system components and their functions.

1.3 Definitions, Acronyms, and Abbreviations

API: Application Programming Interface An interface through which developers can connect different elements of a system together.
DevOps: Development Operations Automation and integration of development processes that allow software to be built and tested faster and more efficiently. [2]
SHF: Sankofa Healthcare Framework An open and secure healthcare data platform.
UI: User Interface The graphical interface through which the user interacts with the system.

1.4 References

[1] ISO - International Organization for Standardization, “ISO/IEC/IEEE 29148:2011,” ISO, 2011. https://www.iso.org/standard/45171.html (accessed Mar. 20, 2021).

[2] Atlassian, “What is DevOps? | Atlassian,” Atlassian, 2016. https://www.atlassian.com/devops (accessed Mar. 22, 2021).

1.5 Overview/Document Convention/Intended audience

This Software Requirements Specification (SRS) begins with a short product perspective and a summary of the product functions. The different classes of users will then be discussed, as well as their characteristics. The environment in which the system operates in, as well as documentation, will then be discussed. The various requirements of the software will then be discussed and broken down into functional, design/implementation, usability and non-functional requirements.

This document follows the conventions set out in IEEE standard 29148:2011 [1]. The intended audience for this document is key stakeholders, developers and contractors involved in the project. Reading and understanding this document is critical in ensuring that all stakeholders understand the requirements of the SHF project and the relevant functionality and expectations around the framework.

2. Overall Description

The following section contains a general overview of the product.

2.1 Product Perspective

The SHF administration user interface functions as a component of the large SHF project. The SHF Admin UI will interact with the wider system through the administration API, which will then in turn interact with the database and blockchain systems. The Admin UI will be accessible through a web browser and will allow the functionality defined in this document to be achieved.

2.2 Product Functions

The SHF project as a whole will allow healthcare information to be shared between a multitude of providers. Chiefly, the system will allow the tracking and verification of vaccinations. As such, the system will allow the registration, approval, confirmation and scheduling of vaccinations. In terms of the SHF Admin UI, the product will allow the authentication of users and the creation, modification and deletion of organisations and their members. The Admin UI will also allow the control of member permissions, controlling the information that members can create, modify and delete.

2.3 User Classes and Characteristics

The following types of users exist in the SHF system:

1. Super Administrator
   The super administrator has absolute control over the SHF system and can add, remove and control the privileges of other administrators and members. IBM admin and relevant superiors.

2. SHF Administrator
   An SHF administrator has control over the organisations on the SHF system. They can add, remove and control the permissions of each organisation and its members. These users would typically belong to key members of the healthcare industry.

3. Organisation Administrator
   An organisation administrator has control over the members of their organisation on the SHF system, allowing them to add, remove and update the permissions of their members. These users would typically be senior managers or administrators inside healthcare providers such as hospitals.

4. Member
   A member of an organisation can access the resources that have been assigned to them by their organisation administrator. This user may include people such as doctors, nurses, the university, pharmacists and officials from major health providers

A number of considerations exist around these users, such as:

- Varying levels of computer literacy should be taken into account
- Confirmation should be asked before changing or removing information
- It should be assumed that not all users will understand how to properly use the system initially and accommodate for these mistakes
- The integrity of the data stored by the system is of paramount importance
- The user interface should be easy to understand
- The system should validate and correct improperly entered data

2.4 Operating Environment

The system must be accessible from a wide variety of devices, including desktop computers, laptops, tablets and mobile devices. These devices will run a variety of operating systems and support various input interfaces, such as mouse and keyboard, and touch. The SHF must support all of these devices and maintain portability between them. The most up-to-date version of the data stored in the system must be accessible from any of these devices at any time.

2.5 User Documentation

Documentation will be provided inside the application (online help), as well as contextual help such as popups and tooltips. Documentation will also be provided through the User Training Manual provided upon project delivery. Documentation for developers is included throughout the codebase through comments in the code. A wiki in the Git repository is also used to provide documentation, help and explanations around development decisions.

3. Requirements

This section contains the requirements for the SHF platform administration UI.

3.1 Functional Requirements

R1 - Registration/Authentication
These functional requirements deal with the registration and authentication of users in the SHF admin UI.

R1.1 The system shall allow administrators and members to log in through an email and password
R1.2 The system shall allow new accounts to be created
R1.2.1 The system shall not allow newly created accounts to be used until they are approved by an SHF Administrator
R1.3 The system shall allow passwords to be reset through a secure email process.
R1.4 The system shall log registration and authentication history for all users
R1.5 The system shall utilise pagination, sorting and filtering to provide the administrator with control over the displayed registrations.

R2 - Organisation Management
These functional requirements are concerned with the management of organisations in the SHF admin UI. Only users who hold an Organisation admin role or greater can edit organisations and add members

R2.1 The system shall only allow organisation admins to edit and add members to the organisation they are assigned
R2.2 The system shall allow SHF Admins and Super Admins to create organisations, as well as add additional members to any organisation
R2.3 The system shall allow Organisation Admins or SHF Admins to control the authorisation level for each member.
R2.3.1 The system shall allow Organisation admins to edit the authorisations for the members of their organisation
R2.3.2 The system shall allow SHF admins and super admins to edit the member authorisations for any member in any organisation
R2.4 The system shall log all changes to organisations and their members, as well as authorisation changes

R3 - Member Management
These functional requirements are concerned with the management of members in the SHF admin UI.

R3.1 The system shall allow only members to access, create, edit and the information they are authorised
R3.2 The system shall log queries to the information each member accesses
R3.3 The system shall only allow members to create, edit and delete the information they are authorised
R3.4 The system shall log queries to the information each member creates, edits or deletes.

3.2 Design and Implementation Requirements/Constraints

R4 Time Constraints

This project will need to be completed within the semester at MQU university

- **R4.1** The system shall be completed within time deadline i.e 13 weeks
- **R4.2** The system shall be in a handover friendly state and reproducible

R5 Technical Constraints

The project shall be in accordance with IBM's standard practice in order to make it easy for them to continue to work on.

- **R5.1** The system shall be written in ReactJS
- **R5.2** The system shall use Carbon UI design system
- **R5.3** The system shall be written using JavaScript
- **R5.4** The system shall be able to incorporate Blockchain technology

R6 Accessibility

- **R6.1** The system shall be accessible through all web browsers and operating systems.
- **R6.2** The system shall be accessible on slower network connections regardless of the platforms it was accessed on.
- **R6.3** The system shall not need demanding hardware.
- **R6.4** The system shall be accessible at all times.

3.3 Usability Requirements

R7 The system shall be user friendly and easy to use by following consistent and recognisable interface patterns and flows
R8 The system shall indicate the data is loading through a shimmer or loading spinner
R9 The system shall respond within .5s of user actions
R10 The system shall be able to run on a low-speed CPU (less than 4 cores & 1.5Ghz), such as a notebook laptop
R11 The system shall support a range of modern browsers, such as Chrome, Firefox, Safari and Edge
R12 The system shall handle errors with the adequate response
R13 The system shall display helpful information with steps to solve an issue if a problem occurs
R14 The systems shall be easy to learn and follow user interface conventions
R15 The system shall provide tooltips and contextual help
R16 The system shall validate input before submitting it to the backend

3.4 Other Non-functional Requirements

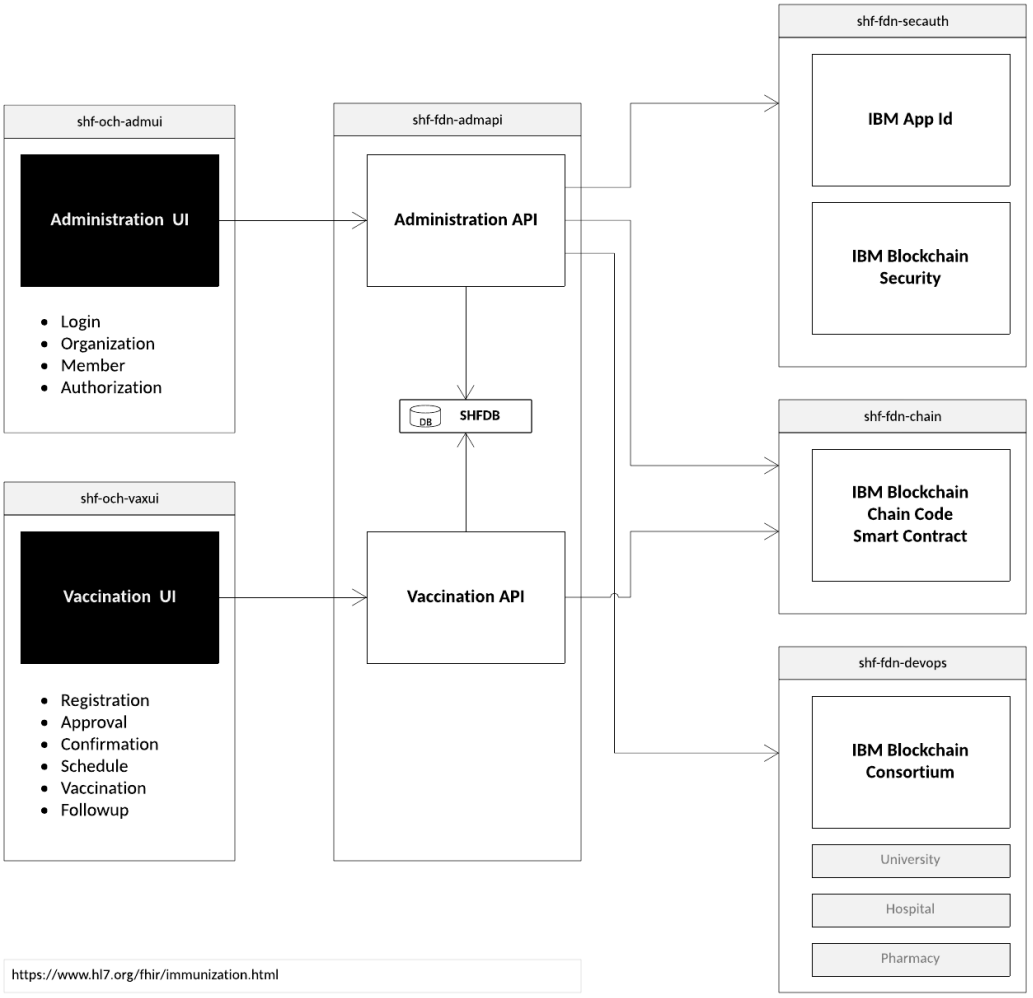
R17. Security
These security requirements involve ensuring the integrity and confidentiality of records in the SHF system. The system shall support the user access classes identified previously.

R17.1 The system shall allow users to be authenticated before accessing the framework
R17.2 The system shall allow users to only be able to access content according to their access privilege level
R17.3 The system shall protect records from unauthorized access or modification
R17.4 The system shall provide end-to-end encryption with all user interactions.
R17.5 The system shall be highly secure and and shall not susceptible to any security breach attacks
R17.6 The system shall have remote data backup servers incase of any potential data breaches or data loss.

R18. Availability
The system shall be available at all times.
R18.1 The system shall be available constantly, 99% uptime.
R18.2 The system shall not have redundancies and failovers by maintaining an offsite backup server

R19. Scalability
The system shall be scalable and allow for users to access the system concurrently.
R19.1 The system shall be able to cope with spikes in access requests 300% above the median level
R19.2 The system shall be able to scale up as members and organisations increase

4. Appendix



Personas:

- Super Admin
- SHF Admin
- Org Admin
- Member

Client Feedback

Meeting date and time:
30/03/2021, 9am-10am and 01/04/2021 12:00am.

Feedback received:

- Inclusion of diagrams providing greater system context and clarity
- Clarification on definitions, acronyms and abbreviations

- AdminUI, looks good but consistency is key since design elements can be easily tweaked. A few points:
- First, keep it focused on a simple use-case - the framework was for a 3-org network involving hospitals, pharmacies, and universities. I didn't see any mention of a university in your report. Mentioning government complicates it further as it varies by country.
- Second, Covid-19 is certainly a use-case, but this framework should apply to all vaccinations. Again, don't complicate matters with a super complex, highly variable vaccination program like COVID.
- Third, keep consistent with users:
    - Superadmin - host, IBM
    - SHFadmin - the Orgs - hospital system, university, pharmacy chain
    - Orgadmin - specific hospital or clinic associated with SHF admin
    - Member - nurse, pharmacist, doctor, university department associated with Orgadmin

Team response/action points:

- Inclusion of extra diagram to provide greater system context and clarity
- Editing and addition of definitions, acronyms and abbreviations
- Simplified document to generalise the app to any vaccine taking COVID-19 as an inspiration rather than the end goal
- Removed details about government to allow scalability to multiple nations and to avoid limiting or overcomplicating the project
- Simplified the user cases with the feedback from the sponsor

Analysis & Design Document

Revision History

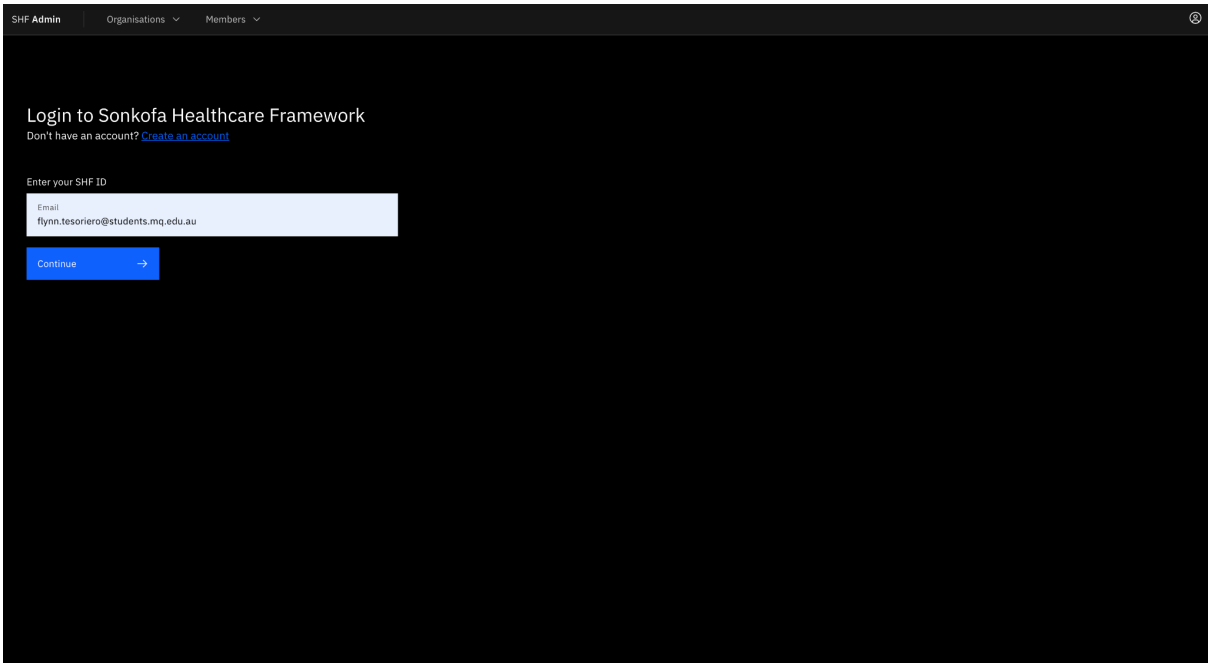
Revision	Date	Person(s)	Changes
1	29/04/2021	All	Initial document. Added assumptions, prototype, use case diagram, use case descriptions and user stories.

Assumptions

1. Resources shall be available to the team (i.e. IBM Cloud will not experience downtime).
2. Team members actively contribute and attend meetings.
3. Other teams working on the SHF project deliver their iterations so that components of the application can be linked.
4. The initial scope of the project is static.
5. The resources available to the team shall be altered.
6. The allocation of resources shall be followed.
7. IBM mentors will be available to the team and willing/able to assist.
8. The backend architecture of the system shall be able work seamlessly with the front-end UI/ UX design.
9. The data servers storing sensitive health information shall be able to be accessed in the back-end system.
10. The system shall be able to share health data with only authorized health organizations to track Covid-19 vaccination recipients.
11. Other teams within the SHF project shall complete their respective components of the system, such as the API and blockchain backend.
12. Future teams will build on the work of the current teams to expand the scope of the system and allow more use cases to be added.

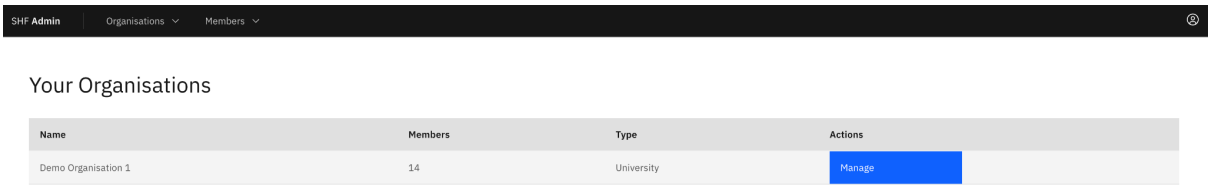
Prototype

The team has made significant progress on the prototype so far. As being part of the team working on the Administration User Interface (Admin UI) of the Sankofa Healthcare Framework (SHF) project, we were tasked with building the web interface used to administrate and manage the SHF system. This included building the Login Screen, Organisation List Screen, Organisation Management Screen , as well as User Management Screens. As a number of other teams are working on different aspects of the SHF project, such as the API, database and blockchain components, dependencies on other teams exist for the system to work efficiently as a whole. For example, the Admin UI cannot be connected to the API service until the relevant API endpoints have been completed. Similarly, the API service cannot be connected to the backend database until the blockchain component has been finalised. As such, the first increment of the Admin UI utilises stubs and sample data to replicate the expected functionality until the other dependent components have been completed. Screenshots of the progress made on various screens is included below:



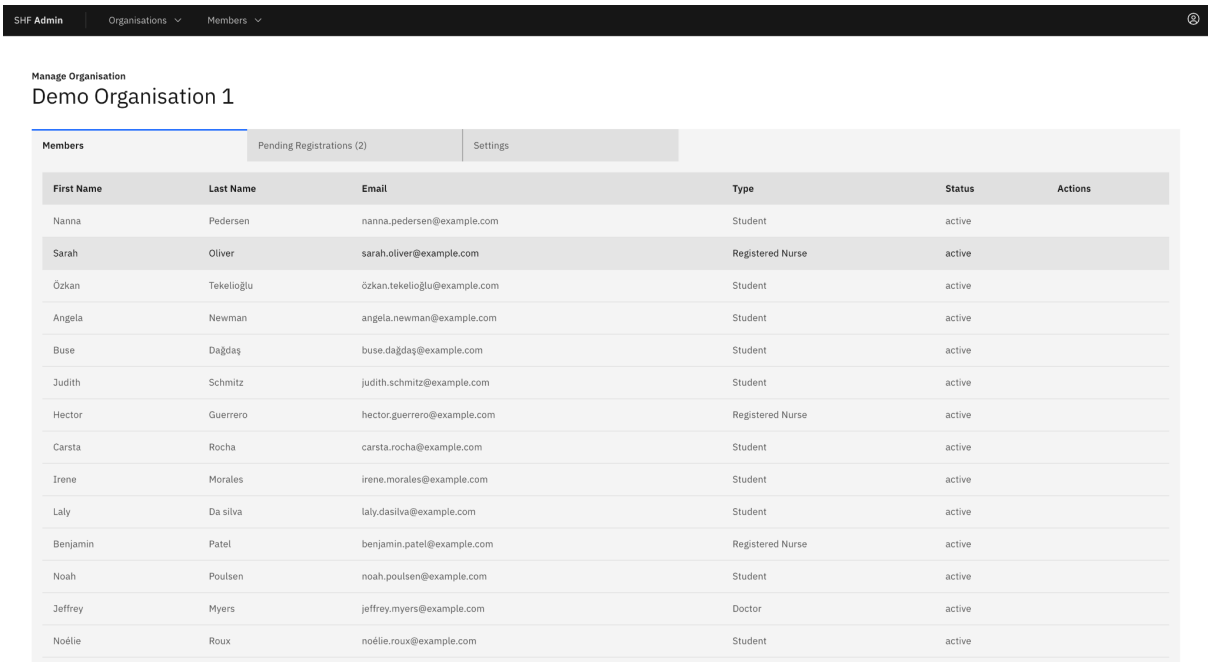
Login Screen

The login screen provides authentication for the Admin UI. It allows users to input an email and password before gaining access to the system. Email validation has been included to ensure the email address entered is valid. Future increments will include account creation functionality.



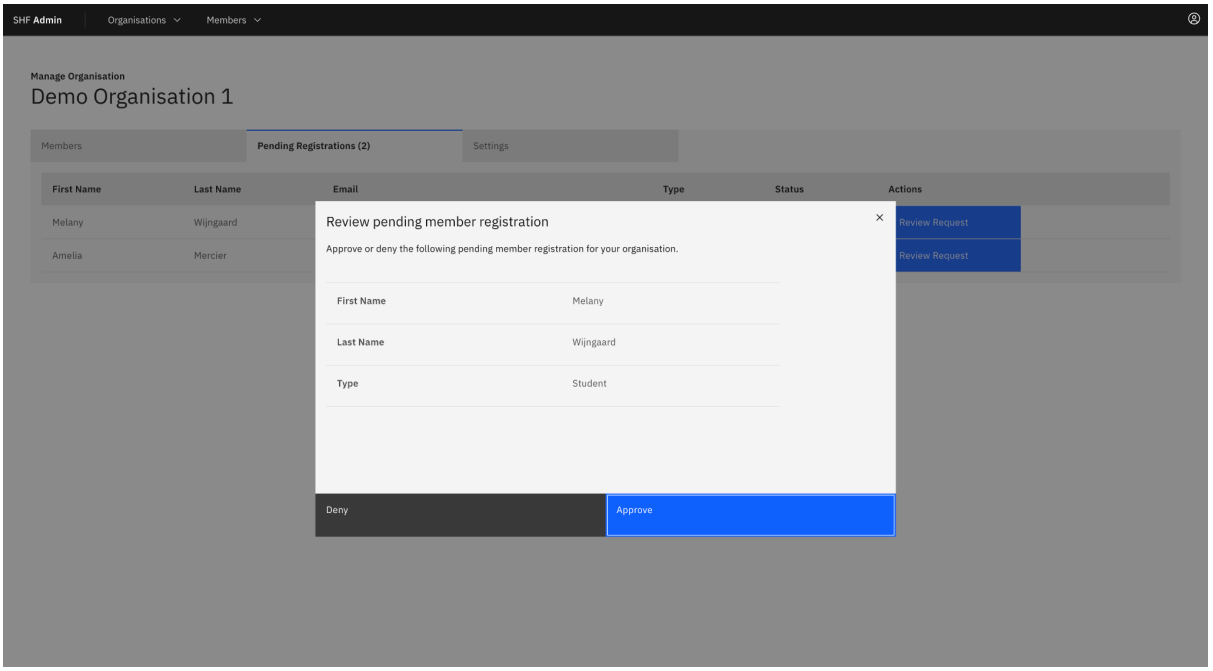
Organisations List Screen

The organisations list screen provides a list of the organisations a user is permitted access to. Users in the Admin UI can have a number of different roles, allowing them to have granular permissions and access to different functions of the system. In this example, the user has access to the *Demo Organisation 1*. Other users will have access to different organisations, and at different levels of permissions. The interface provides summary information about the organisation(s) that the users are permitted access to, as well as a “Manage” button to manage the organisation.



Organisation Management Screen

The organisation management screen provides an authorised user to view the members of an organisation, approve pending registrations, as well as configure settings for the organisation. In this screen, a summary of each of the members of an organisation is displayed in a table. In future increments, a number of actions will be able to be completed on each member, such as editing or deleting them. The following modal popup provides insight into the pending registration approval process.



Member Review Modal

The member review modal popup provides a means for an administrator to provide approval to users who have requested to join a particular organisation, such as a student joining a University. When an administrator selects to view a particular application, they can view summary information for the pending member registration. They can then either approve or deny this request.

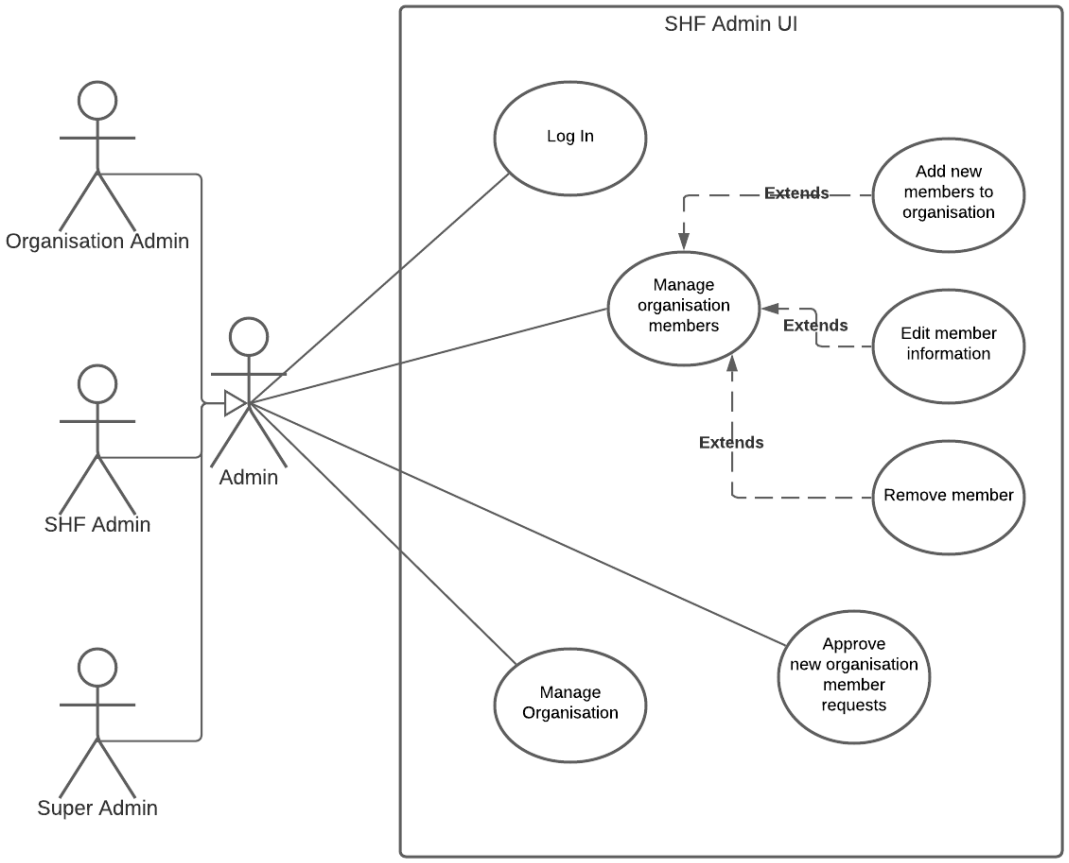
Future iterations of the prototype will build on what has been developed so far. Future features to be built will include a create account screen, as well as user permission management functionality, allowing organisation administrators granular control over the permissions of users.

Sponsor Meeting & Feedback

The team meets with IBM, the project sponsor, each Tuesday at 8am. Throughout these meetings, the progress of each team is discussed. In the meeting on the 27th April, the prototype and various pieces of feedback were discussed. Overall, the IBM team was pleased with the progress of the teams so far. One significant piece of feedback given was the idea to add pagination to the tables through the user interface. By adding pagination, the length of the page is reduced, as well as the size of data required. This leads to improved efficiencies and makes the interface easier to navigate for the end users. Furthermore, the IBM team also recommended adding filtering and sorting options to the tables through the user interface. This allows the users to find the specific information they are looking for more quickly, and makes the interface more usable. IBM team also provided feedback and guidance to the team through providing various diagrams, models and frameworks over the course of the development. These have helped the team guide the development of the application. The IBM team is also in the process of reviewing the latest updates to the prototype, with feedback expected shortly.

The feedback provided so far, and future feedback, is promptly implemented into the prototype. For example, the feedback around particular menu structures and screen designs have helped formulate how the prototype currently looks like. Further feedback around adding filtering, sorting and pagination to pages and tables is currently being implemented. The team follows a process of responding positively to feedback and noting down the details in the feedback received. The team then discusses the feedback and makes a decision on whether to implement the feedback into the prototype. Most typically, almost all of the feedback is implemented. The person providing feedback is then notified that the feedback has been received and then actioned.

Use Case Diagram



Use Case Descriptions

Use Case 1: Log In

Use Case	Log In
Goal	To successfully authenticate with the system to gain access to the resources that the user is permitted to access.

<a longer statement of the goal in context if needed>		
Preconditions <what we expect is already the state of the world>	The user has navigated to the login screen of the system. The user is a registered user of the system.	
Success End Condition <the state of the world upon successful completion>	The user is successfully authenticated with the system and can access the resources they are permitted to.	
Failed End Condition <the state of the world if goal abandoned>	The user stays on the login screen and is not permitted access to the system.	
Primary Actors **Secondary Actors**	Administrator User SHF Admin System, SHF Authentication API Service	
Trigger <the action upon the system that starts use case>	The opens the system web page and is presented with the log in screen.	
Description / Main Success Scenario <the steps of the scenario from trigger to goal delivery and any clean up after. Indicate substeps using numbering>	**Step**	**Action**
	1.	The system displays the login page welcoming the user to the system.
	2.	The user enters their email address.
	3.	The system verifies that the email address is a valid format.
	4.	The system prompts the user to enter their password.
	5.	The user enters their password.
	6.	The system verifies the password is a valid format and length.
	7.	The system makes a request to the Authentication API.
	8.	The Authentication API confirms the email and password combination is valid.
	9.	The system redirects the user to the Admin landing page and saves the logged in state.

	10.	The use case ends.
Alternative Flows <a: condition causing branching> <a1: action or name of sub use case>	**Step**	**Branching Action**
	3.a	The user enters an invalid email address.
	3.a.1	The system displays an invalid email error message.
	6.a	The user enters an invalid password (i.e. too long, too short, etc.).
	6.a.1	The system displays an invalid password error message.
	8.a	The user enters an incorrect email/password combination.
	8.a.1	The system displays an incorrect email/password error message.

Use Case 2: Manage Organisation

Use Case	Manage Organisation
Goal <a longer statement of the goal in context if needed>	To successfully view the details of an organisation and make changes to its name, description or other settings if required.
Preconditions <what we expect is already the state of the world>	The user is successfully authenticated with the system. The user has permission to manage the organisation.
Success End Condition <the state of the world upon successful completion>	The user has successfully made the changes to the organisation they wish to perform.
Failed End Condition <the state of the world if goal abandoned>	The user was unable to make changes to the organisation, and its state remains unchanged.
Primary Actors **Secondary Actors**	Admin User SHF Admin System, SHF API Service
Trigger <the action upon the system that starts use case>	When the user selects the manage organisation button on a particular organisation.

Description / Main Success Scenario <the steps of the scenario from trigger to goal delivery and any clean up after. Indicate substeps using numbering>	**Step**	**Action**
	1.	The system displays the relevant information for the organisation the user selected to manage.
	2.	The user navigates to the Settings tab.
	3.	The system displays the relevant settings for the organisation.
	4.	The user makes the necessary changes to this settings form, such as name, description or other toggles.
	5.	The system validates the input from the user as they are editing.
	6.	The user selects the Update button.
	7.	The system validates the input a final time.
	8.	The system sends a request to the API to update the necessary fields in the database.
	9.	The API response confirms the database has been updated.
	10.	The system displays a message to the user that the settings have been updated.
	11.	The use case ends.
Alternative Flows <a: condition causing branching> <a1: action or name of sub use case>	**Step**	**Branching Action**
	4.a	The user does not have permission to modify organisation settings.
	4.a.1	The system disallows the editing of the settings fields and displays a warning tooltip.
	5.a	The user enters invalid input.
	5.a.1	The system displays an error message and prompts the user to correct input.
	9.a	The API responds with an error message.
	9.a.1	The system shows an error message that the settings could not be updated.

Use Case 3: Manage organisation members

Use Case	Manage Organisation Members

Goal <a longer statement of the goal in context if needed>	To manage the members of an organisation by adding, removing or editing their information.	
Preconditions <what we expect is already the state of the world>	The user is authenticated with the system. The user has permission to manage the organisation. The user has permission to manage the members of the organisation.	
Success End Condition <the state of the world upon successful completion>	The organisation's members have been updated.	
Failed End Condition <the state of the world if goal abandoned>	The organisation's members have not been updated and the system state remains unchanged.	
Primary Actors **Secondary Actors**	Admin User SHF Admin System, SHF API Service	
Trigger <the action upon the system that starts use case>	The user selects an organisation to manage by clicking the Manage Organisation button.	
Description / Main Success Scenario <the steps of the scenario from trigger to goal delivery and any clean up after. Indicate substeps using numbering>	**Step**	**Action**
	1.	The system displays the relevant information for the organisation the user selected to manage.
	2.a	The user selects the Add New User button.
	2.a.1	The system displays the Add New User modal popup.
	2.a.2	The user fills the first name, last name, birth date and user type fields.
	2.a.3	The system validates the input as the user enters the data.
	3	The user clicks the Save/Confirm button.
	4	The system validates the data again.
	5	The system sends the data to the API service.
	6	The API service responds confirming the data has been updated in the database.

	7	The system displays a message to the user confirming the organisation users have been updated.
	8	The use case ends.
Alternative Flows <a: condition causing branching> <a1: action or name of sub use case>	**Step**	**Branching Action**
	2.b	The user selects a user to edit.
	2.b.1	The system displays the Edit User modal popup.
	2.b.2	The user edits the fields, such as first name, last name, birth date and user type.
	2.b.3	The system validates the input as the user enters the data.
	2.c	The user selects the delete user button.
	2.c.1	The system confirms that the user requested to be deleted is correct.
	4.a	The data entered is incorrect or invalid.
	4.a.1	The system displays an error message to the user.
	6.a	The API responds with an error message.
	6.a.1	The system shows an error message that the settings could not be updated.

Use Case 4: Approve new organisation member request

Use Case	Approve new organisation member request
Goal <a longer statement of the goal in context if needed>	To approve a request by a new member to join an organisation.
Preconditions <what we expect is already the state of the world>	The user is authenticated with the system. The user has permission to manage the organisation. The user has permission to manage the members of the organisation.
Success End Condition <the state of the world upon successful completion>	A new organisation member has been approved or denied and the system updated.

Failed End Condition <the state of the world if goal abandoned>	A new organisation member was not approved or denied and the system state remains unchanged.	
Primary Actors **Secondary Actors**	Admin User SHF Admin System, SHF API Service	
Trigger <the action upon the system that starts use case>	The user selects an organisation to manage by clicking the Manage Organisation button.	
Description / Main Success Scenario <the steps of the scenario from trigger to goal delivery and any clean up after. Indicate substeps using numbering>	**Step**	**Action**
	1.	The system displays the relevant information for the organisation the user selected to manage.
	2.	The user navigates to the Pending Registrations tab.
	3.	The system displays a table of the pending member registrations.
	4.	The user selects the Review Request button.
	5.	The system displays the Review Member Application modal popup with the relevant pending member's information.
	6.	The user selects the Approve button.
	7.	The system communicates the request to the SHF API service.
	8.	The API service confirms the database was updated.
	9.	The modal popup closes and the pending member registration is removed.
	10.	The use case ends.
Alternative Flows <a: condition causing branching> <a1: action or name of sub use case>	**Step**	**Branching Action**
	6.a	The user selects the Deny Request button.
	8.a	The API responds with an error message.
	8.a.1	The system shows an error message that the settings could not be updated.

User Stories

1. As an organisation administrator, I want to log into the system, so that I can view and manage my organisations.
2. As an organisation administrator, I want to view a list of organisations that I have permission to view and edit, so that I have visibility on what I can manage in the SHF system.
3. As an organisation member, I want to submit a request to become an organisation member, so that my health information can be tracked.
4. As an organisation administrator, I want to be able to approve new members requests for my organisation, so that I can ensure that only those who belong to my organisation have access.
5. As an organisation administrator, I want to be able to add new organisation members, so that I can keep my organisation up to date.
6. As an organisation administrator, I want to be able to update my organisation members, so that the organisation is kept up to date.
7. As an organisation administrator, I want to be able to remove members from my organisation, so that if a member leaves I can ensure they no longer have access.
8. As an organisation administrator, I want to be able to manage my organisation information, so that the information reflected to users is correct.
9. As an organisation administrator, I want to be able to sign out of the system, so that I can ensure that unauthorised access is avoided.
10. As an organisation administrator, I want to be able to view the members of my organisation, so that I can ensure the list is up to date.

System Design Document

Revision History

Revision	Date	Person(s)	Changes
1	29/04/2021	All	Initial document. Added current sections, and their content & diagrams.

Purpose

The System Design Document tracks and records the necessary information required to effectively define the system architecture and design of the product. This document describes the proposed “Sankofa Healthcare Framework (SHF) project” at its architecture level in order to allow organizations to be able to track vaccine rollout plans and of its recipients in a safe and secure manner for the current COVID-19 Vaccine and any future vaccines. It is integral for the architecture to be inclusive therefore accessible to a large audience as the vaccine rollout is relevant to most if not all individuals. This system will be dealing with highly sensitive and private information therefore security will need to be in forefront. This SDD will talk about the software and hardware architectures needed to achieve the systems goals with focus on the UI aspect, data management strategies,

noteworthy tradeoffs and any concurrent processes. The SDD also briefly goes over the Package diagram, a foundational guide for further implementation of the system. All the design documents are incrementally and iteratively produced during the system development life-cycle.

Intended Audience and Viewership

This document is intended to be used by Administrative staff (Super Admin, SHF Admin, Organisation Admin, etc) who are our clients, Software developers, Project managers, lead members (liaisons) and perhaps any concerned government health departments.

System Architecture

The Admin UI will be accessible through a web browser and will allow functionality as defined in this document. The architecture helps define a solution to meet all the technical and operational requirements, with the common goal of optimizing for performance and security.

The suggested architecture for this system would be layered N-tier architecture. This approach is commonly used as it is built around the database and the healthcare industry tends to store their data naturally in tables. The code is arranged in a manner that allows data to enter the top layer until it reaches the bottom which will be the database. Each layer will have a specific task which will involve checking for consistency and reformatting the data while keeping it secure as needed by the system. It is common for different programmers to work on different layers independently.

The main advantages of the layered architecture is separation of concerns, which lets each layer focus solely on its role. This makes the system more maintainable, testable, updatable and generally makes it easier to divide and delegate working on the system. A proper layering of the architecture will mean other layers will not be affected by changes made to another layer. This allows for shared access and also bypassing of layers for speed which is integral for the system discussed. This approach is best for new applications that need to be built quickly like SHF with need for strict maintainability and testability standards.

In terms of hardware architecture for this system as it using block chain software the level of Blockchain leverages the power of decentralization, that means, the collective computing power of all the nodes in the networks will always be larger than the computing power of a single player who wants to take the network down or change the data in the blockchain. Any computer with sufficient computing abilities to run a blockchain should add to the overall computing power of the network having no complicated hardware requirements . Admin users would constitute Light clients and require less storage usually within a few gigabytes, but also offer less functionality since they don’t hold all transaction information. It would further benefit the system’s security if their clients, the admin could be treated as hybrid clients allowing to fraud proof the blockchain. If a gateway is found to be dishonest the administrator has means to submit a fraud-proof blockchain allowing to penalize the misbehaviour and gateway risk to lose their stake.

The suggested computer specifications for an administrator would be

- Memory : 8GB ram
- Processor: 6-core, Intel Core i5 Processor, 3.0 GHz, or better
- Graphics Accelerator Intel HD Graphics, or discrete graphics card
- Network Adapter or Modem: Integrated 10/100/1000 Ethernet Adapter
- Operating System: Windows 10 Enterprise (can be upgraded to from any version of Windows 10 at no charge), or macOS Mojave 10.14.X or newer

Storage/persistent data strategy

The storage and data strategy revolves around blockchain as the system uses blockchain to strategically structure the data without need for central authority. The solution to make cloud storage faster and more secure is using blockchain. Blockchain hosts the continuously growing number of records SHF systems will need to hold. The database stores records in blocks rather than collating them in a single file. Each record is then chained to the next block in a linear chronological order using cryptographic signatures. This will allow for a clear log of changes made to the records to be visible to administrators. This allows for the information to be reliable and to some extent transparent. Blockchain acts like a ledger that can be shared with relevant permission put in place. These ledgers can be used in multiple countries and intuitions as required by SHF and the rollout of vaccines across many nations and integral use in travel decisions and restrictions. This ledger is encrypted such that only authorized parties can access the data. Since the data is shared, the records cannot be tampered.

Concurrent Processes or Data

Most of the transactions performed through the SHF system shall not be concurrent, avoiding the issues associated with concurrent processes and data processing. Notwithstanding, an example of a concurrent process within the SHF system is loading the application from the server when the user visits the system for the first time through their web browser. The user's web browser will load the different scripts and resources for the application concurrently to speed up loading time. The user's web browser handles this process, meaning that the SHF system does not have to handle this process. Another example of concurrent processing within the application is the process of loading data. When a user requests a new screen in the application, not only does the application begin loading the elements on screen, such as the buttons, images and tables, but it also sends a request to the API server to load the data for the view. This process happens concurrently, meaning both the loading of the application and the data happen simultaneously. This concurrent process is handled by the frontend user interface framework, React. By using React, the Admin UI team doesn't have to explicitly program and handle this process, a key benefit of using the React framework.

User Interface Strategy

The user interface is primarily a graphical user interface accessible through a web browser. This user interface is the most familiar to end users, whilst also providing the most flexibility to offer the greatest level of functionality. User interface for web browsers is no longer limited to desktop computers but has moved way beyond to access through portable devices. The

strategy employed by the system is to strive to design usable interfaces to cater for the diverse requirements of Portable devices and comply with the technical requirements of the devices and the desires of the manufacturers while remaining functional. The need for differentiated design techniques stems from the diverse characteristics of portable devices such as reduced hardware capabilities and lack of input devices.

The user strategy is to anticipate mistakes and set up channels to correct and identify them. In the SHF system wrong usernames and passwords, wrong record entries, invalid answers will be identified and alternate solutions will be provided with cross cultural awareness. The system is set to be used in multiple nations and must take into account cross-cultural differences and translate and if possible in the future localize the way a user interface is set-up to the location its being accessed from to be more widely accepted.

Fast feedback is a must as well. The system should be able to quickly reach to user interactions and provide feedback in constructive manner immediately even if it is just an acknowledgment. A reactive user interface is a great way to keep the user engaged and feeling ‘safe’. With simple messages such as your changes have been saved recorded at each step to improve accountability of actions as well as reaffirm the user the system is working as planned

Placement and size of action buttons is to be well thought out as action buttons need careful consideration as the system deals with highly sensitive and private information. Action buttons should be clear with a large font to ensure lesser automated no-thought clicking. A confirmation of changes at each stage despite being time consuming must be implemented to make the system secure especially on the admin end.

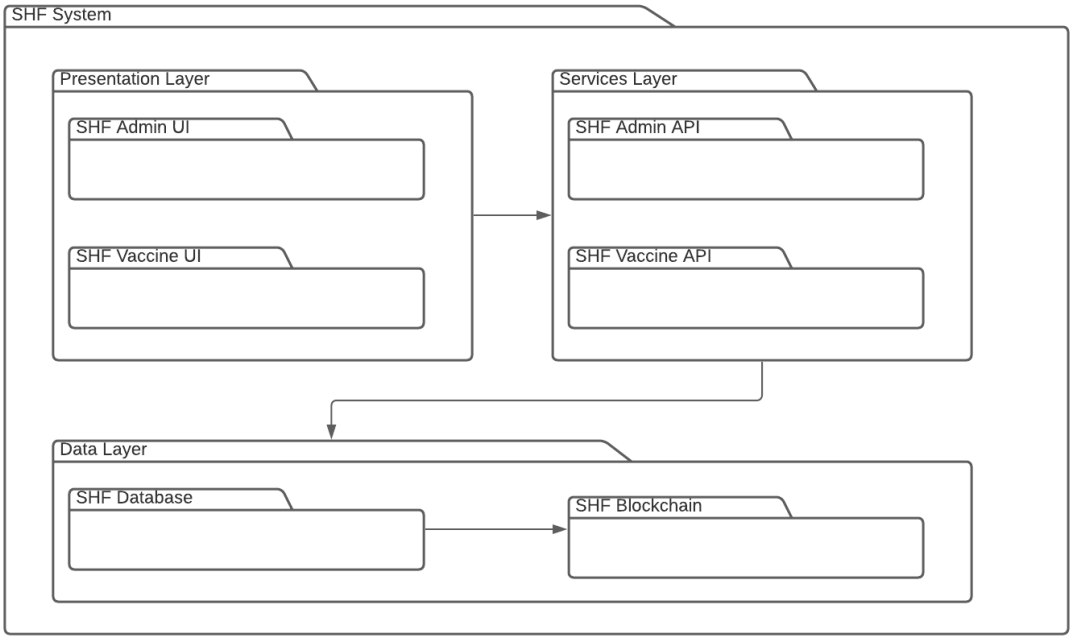
The user interface will be a mirrored version of platforms the users are likely to be already familiar with login, registration, change management, setting pages being similar to industry standards. There will be no attempt to reinvent the wheel but rather make it as simplistic and familiar to not add any extreme cognitive load as the users try to understand the how-to’s of this SHF.

Design Decision Choices and Trade-offs

The decision to develop the system as a web application as opposed to a desktop application has a number of benefits and trade-offs. Benefits include the ability to run the application on almost any modern web browser, meaning the system is accessible to a broad range of users without the need to download specialist software or use a particular platform. It also means that users don’t have to download updates, as the web application is always using the latest version of the system. Further, choosing to develop the system as a web application means that a single code base can be maintained, instead of requiring a separate code base for each operating system. Notwithstanding, this does create a few trade-offs. One such trade-off is the security issues associated with having the system accessible over the internet. As such, the team will have to ensure that the system meets the highest standards in web application security. Developing a web application also results in less features and tools available to the development team, as compared to native apps. Web applications also typically perform slower and require more computing resources than native

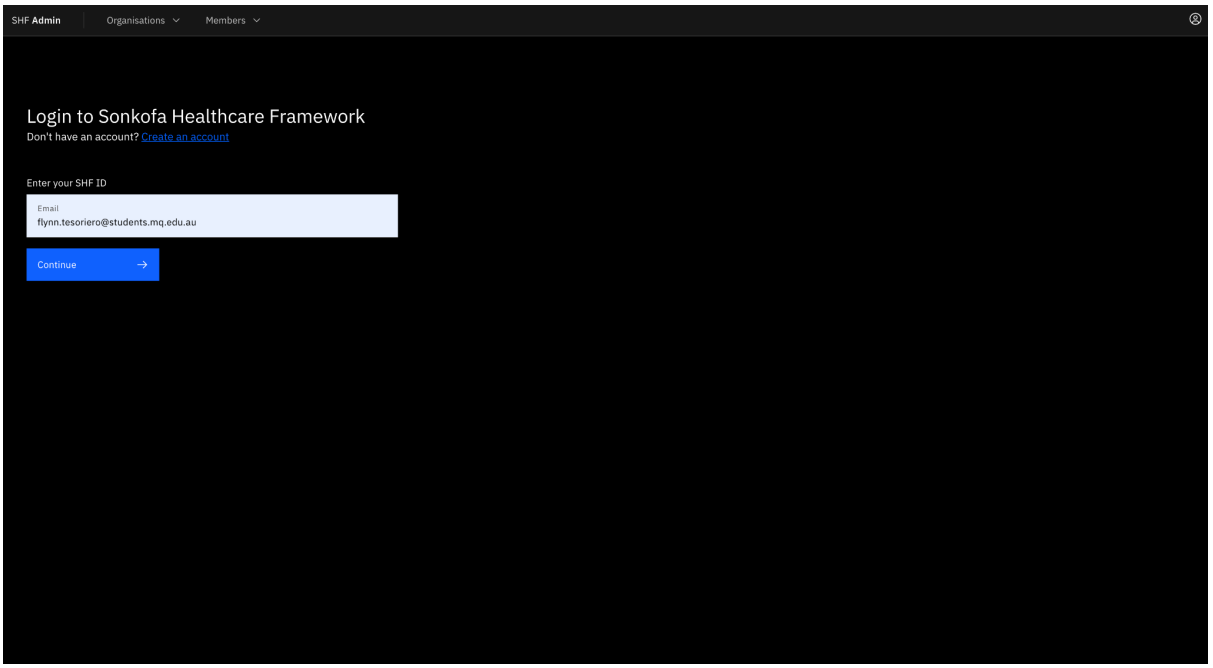
apps. This being said, the benefits outweigh the trade-offs, and developing the system as a web application is the best decision moving forward.

Another decision is the use of Carbon UI as a user interface framework. This UI framework has the benefit of making it quick and effective to build great user interfaces that are consistent and uphold the latest design trends. These benefits come at a cost of customizability and flexibility. For example, developing components not included within the Carbon UI package can be difficult. Notwithstanding, using the Carbon UI framework has a greatly positive impact on the project by decreasing development time, so this tradeoff is necessary.

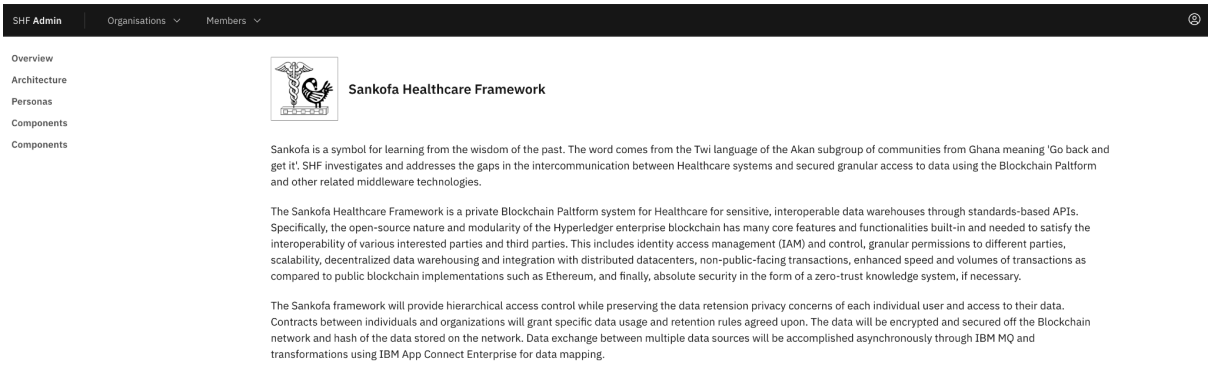


Included above is a package diagram, providing an overview of the entire SHF system. This diagram shows the relationship between the various packages/components of the SHF system and the flows between them. To summarise the diagram, the SHF Admin UI and Vaccine UI sit inside the presentation layer. These packages communicate with the Admin API and Vaccine API respectively. These API services sit at the services layer, and communicate with the Data layer, where the Database and Blockchain reside. Note that the Admin UI team is responsible for the Admin UI component of the system, while other teams are responsible for each other component.

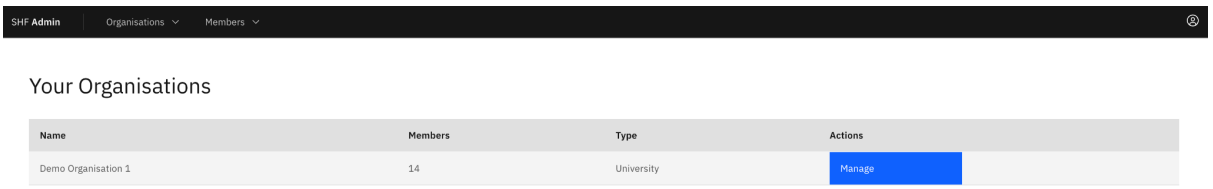
User Interface Layouts



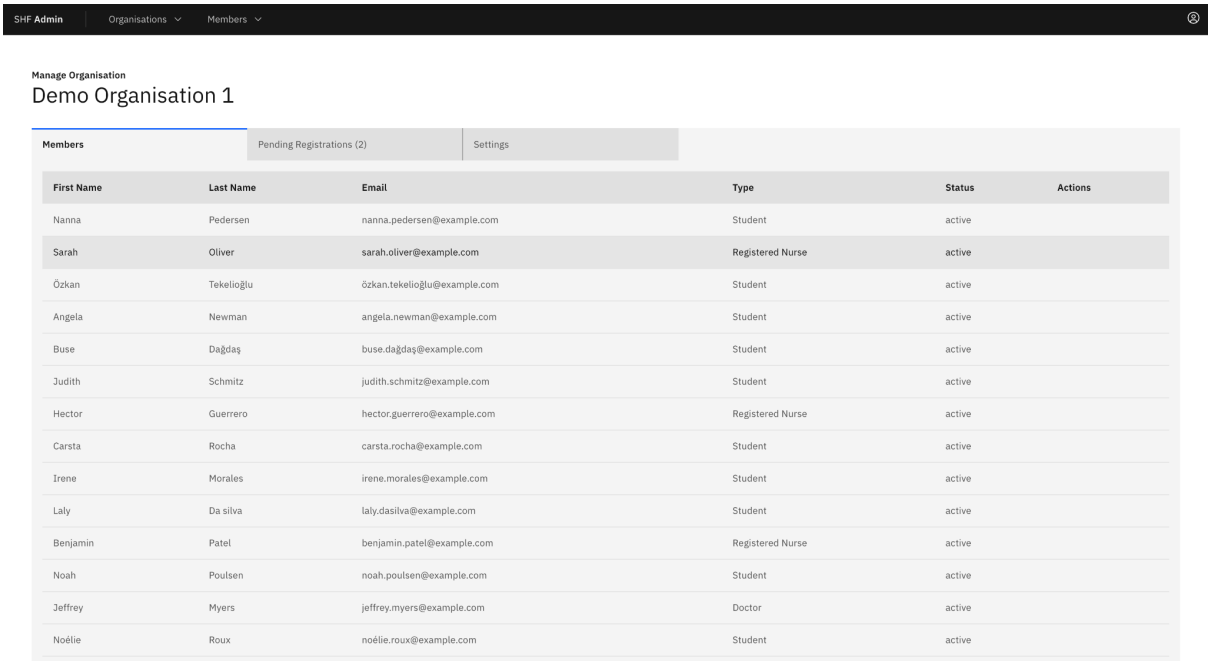
Admin Login Screen



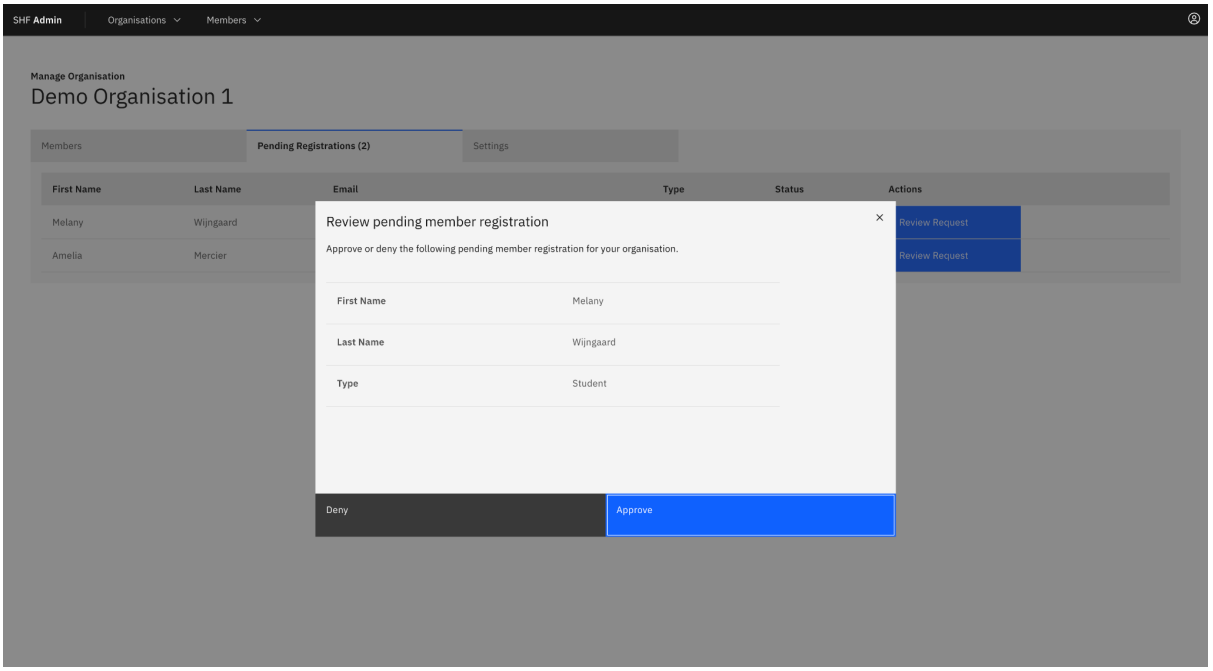
Admin Landing Page with project overview



List of admin user organisations



List of admin user organisation members

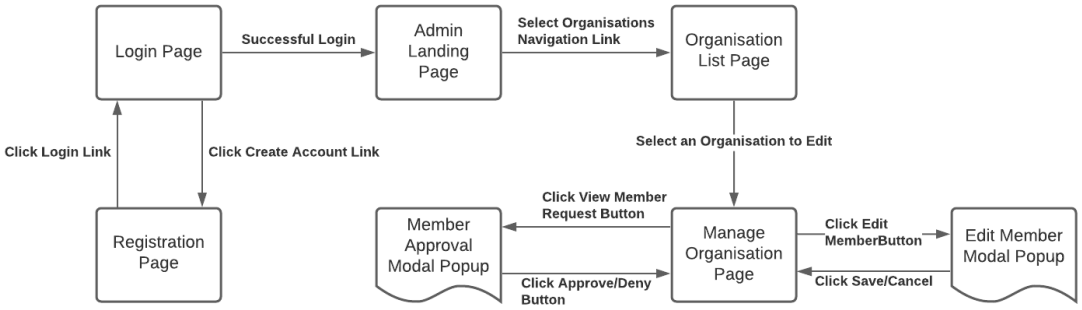


New organisation member approval screen

Report Layouts

No reports are included within the scope of this project.

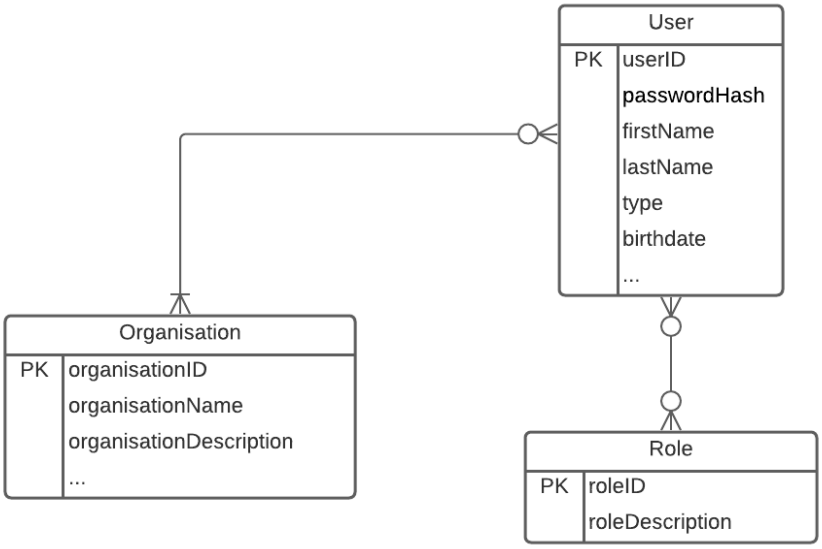
Window Navigation Diagram



The above Window Navigation Diagram shows the various screens in the prototype thus far and the triggers between then. The user begins at the Login page, where they can select the create account link to move to the registration page. Clicking the login link will move them back to the login page. After inputting a correct email address and password combination, the user is moved to the Admin Landing Page. Selecting the Organisations link in the navigation bar, the user is then moved to the Organisations List page, where they can view a list of the all the organisations they have permission to access. From there, the user can select a particular organisation to manage. Upon selecting an organisation, the user is moved to the Organisation Management page for that particular organisation. Here the user can complete a number of actions, such as approving members who have requested access

to the organisation, as well as editing, adding or deleting members. Future iterations of the prototype will expand upon this diagram to provide more functionality within the system.

Data Definitions

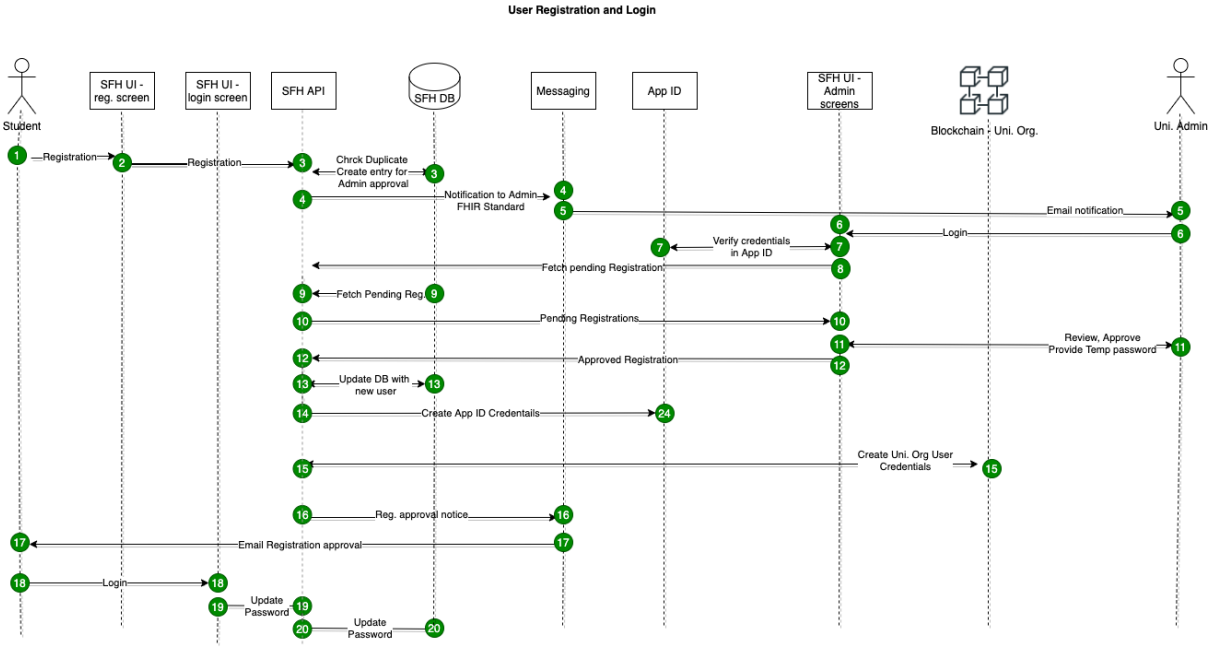


The above ER diagram provides a simplified overview of the current system. Note that as the Admin UI team, the primary concern is the development of the administration user interface. The data for the user interface shall be delivered via an API endpoint in JSON format. As such, the Admin UI team is not critical concerned with the storage, structure and retrieval of data, instead simply requiring a JSON data model to be ingested into the Admin UI. Nevertheless, above is an entity relationship diagram providing an overview of the current storage model. User details are stored in a User entity, along with other information such as their first name, last name, their type, as well as their birthdate. Other information is also stored in this table, but has been truncated for brevity. An organisation entity also exists, which stores the details of the organisation(s) that an admin user belongs to. A role entity also exists, which contains the information for the role(s) an administrator user can perform, such as read, write, edit and delete operations on organisations and their users.

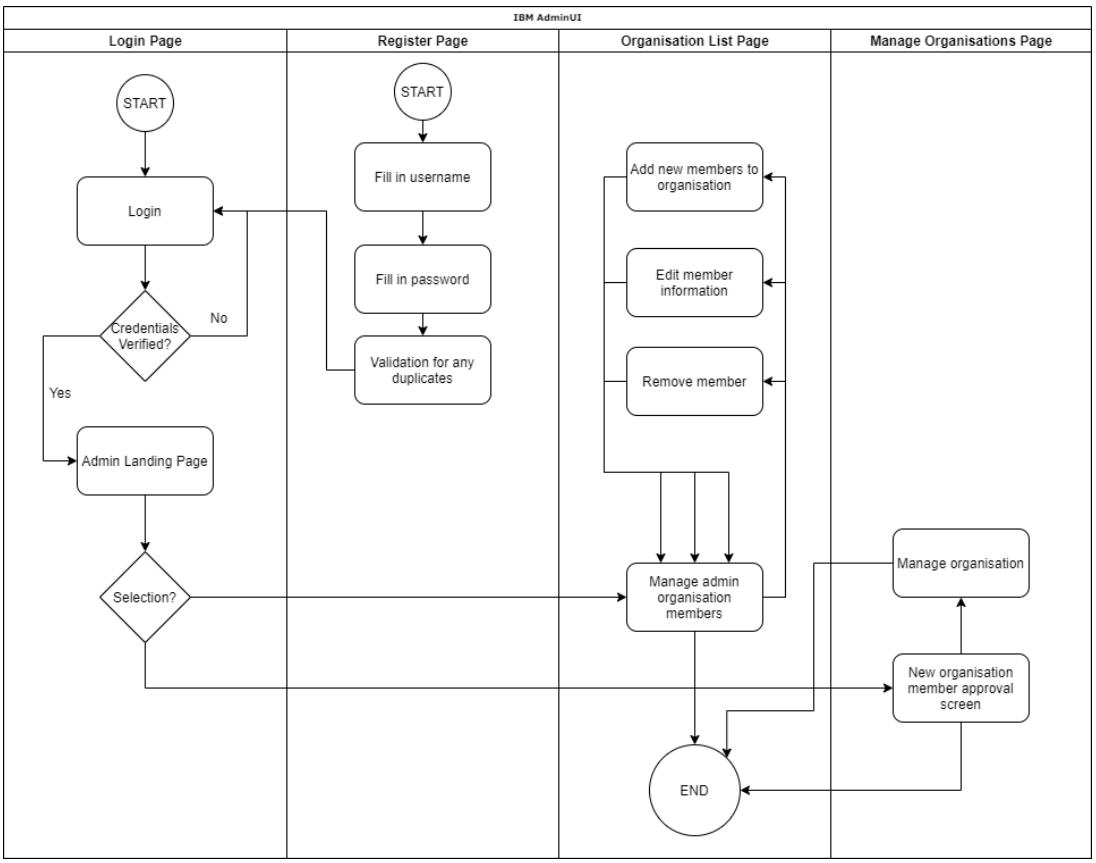
Field Name	**Type**	**Example**
userID	integer	5697
passwordHash	string	@^#TGUYFGW^273fsf
firstName	string	Jeremy
lastName	string	Clarkson
type	string	student
birthdate	date	19-04-1986
roleID	integer	1353

roleDescription	string	View self health data
organisationID	integer	64373
organisationName	string	Macquarie University
organisationDescription	string	A university in Sydney, Australia.

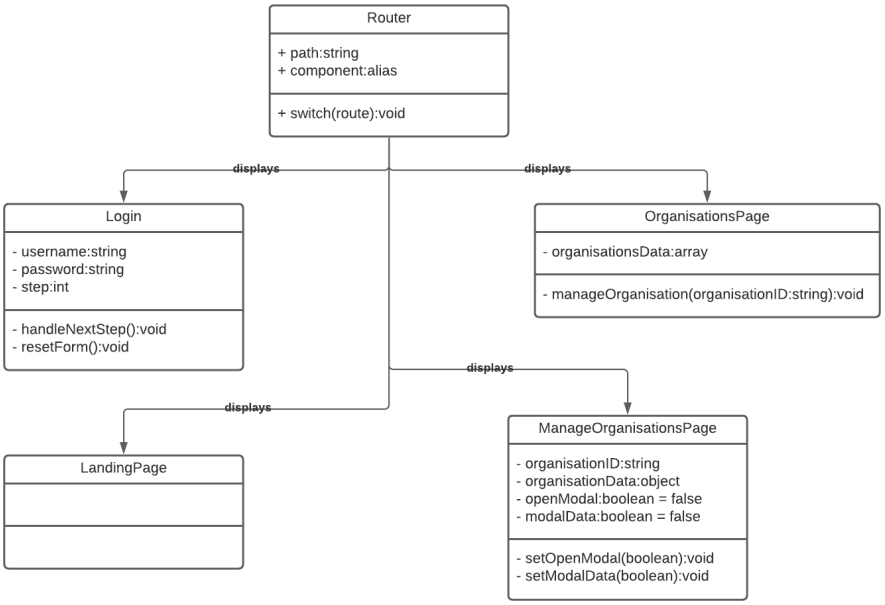
Activity/Sequence Diagram



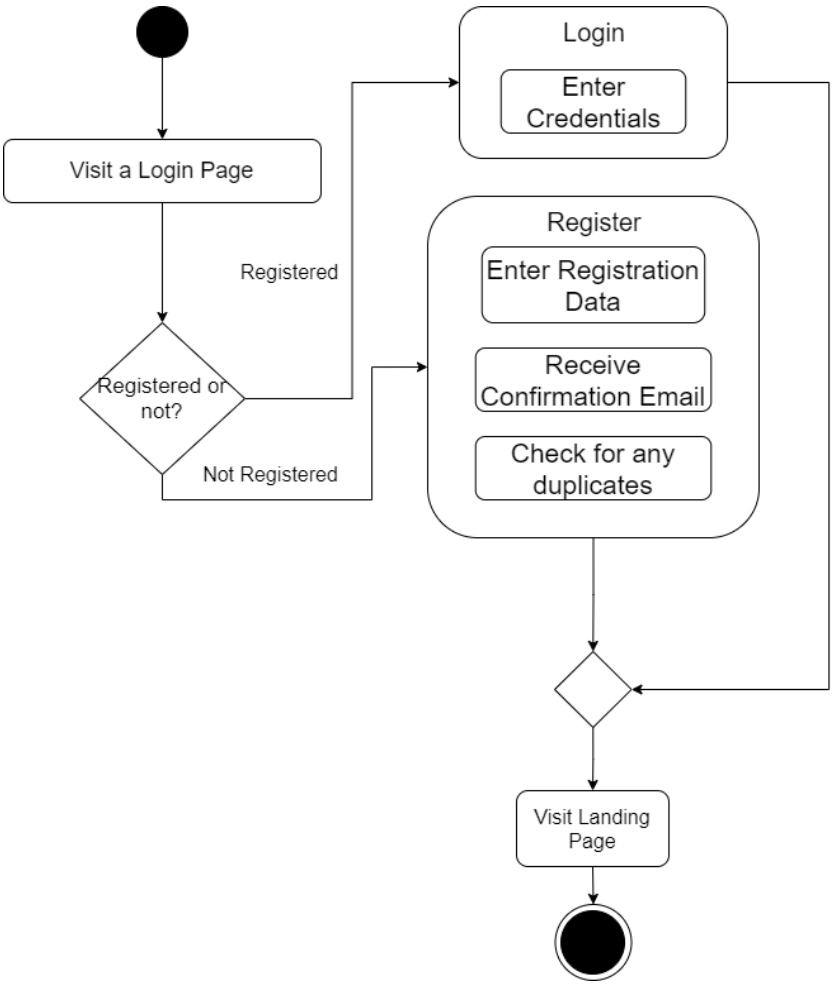
The above sequence diagram provides an overview of the user registration and login flow for the entire SHF system. The flows between the SHF Admin UI and Administrator actor are of particular importance to this team. Note that although the Admin UI is not being developed in an object-oriented language, other aspects of the system are, hence the inclusion of the above sequence diagram.



Class & State Diagrams



The above class diagram shows an overview of the primary classes of the Admin UI react application. The diagram depicts the React Router which controls the display of each of the component views. Each component view is depicted with the relevant attributes and methods.



The above state diagram depicts the varying states of the login flow. The states possible include whether the user is registered on not, and the relevant states that these can include.

Testing Document

Revision History

Revision	Date	Person(s)	Changes
1	29/04/2021	All	Initial document. Added current test plans and specifications.

Test Plans

This section will provide detailed test procedures that will be conducted for each of the test case specifications listed above. This project is using an agile approach for design and testing, with weekly iterations. At the end of each week, a specific functionality is tested and delivered. The testing environment is chosen to be a closed environment and the type of testing is chosen to be “white-box testing” as we do have access to the source code and files.

It is acknowledged that healthcare frameworks / systems are found all across the globe so the new system should be able to work efficiently with the proposed Sankofa Healthcare Framework system architecture. The unit testing, integrated testing, and system testing are chosen as they are specifically designed for code testing to allow careful analysis of the test cases. The users required for testing will be the system and software development team that will carefully interact with the new system and verify if the fit criteria is met and if there are any bugs to be eliminated. The hardware interfaces required for testing will include any personal computer / laptop that meets minimum hardware and network requirements needed to load the application. The software interfaces required for testing will include the SHF application development IDE, GitLab, Carbon Design Systems, React, and any supported web browser. For unit testing, component testing, integration testing and system testing, automated system tests will be run that will provide the correct working environment to test the different classes and methods. When the SHF system is fully integrated and ready to be tested for User Acceptance Testing (UAT), guided walkthroughs will be done to understand the user requirements and expectations out of the system.

To provide maximum test coverage, all the testers will be interacting with the system to check for optimum system response and to ensure that the system is easy to use and accessible by all users. For stress, load and performance testing, a group of test users will interact with the system to check how the system performs under stressful conditions. At the beginning, some instruction manuals and documentation will be provided to the test users to checklist each of the requirements they are made responsible for to be testing. Supporting documentation will also be required for testing each of the subsystems to provide information about both inner and outer workings of the system. A copy of this document will be provided to the client, the stakeholders, other development teams, and other intended audience. Each of the tests will be validated against the requirements enlisted and then verified if the system functions match the expected functionality.

Repeatability tests and regression tests will be conducted for verification purposes. Milestone tasks will be generated and the start date, end date and effort for each project milestone achieved will be recorded. The features to be tested will essentially be the features of the new SHF system that include the correct functioning of the Login Page, Admin Landing Page, Organisation Page and the Member Management Page. For each of the tests, a priority and severity list will be generated. The priority list will record different priority levels for each of the test cases such as “must fix”, “should fix”, “fix when have time”, and “low priority”. The severity list will record the bug severity levels as “Critical” if the bug causes non-recoverable conditions like system crashes, database or file corruption and potential data loss, “High” if the bug causes serious problems like lack of functionality, or error messages, “Medium” for incorrect functionality for a small system component or a process, or “Minor” for documentation errors. If any system modifications are made, the test will be conducted again until correct expected functionality from the system is achieved.

Test Case Specifications

Test Case Identifier	Test Description	Input Specifications	Output Specifications
TC001	The SHF system is tested to check if it's running properly and there are no compilation errors.	Running the application after publishing it	The application is loaded on the web.
TC002	The login screen is tested for existing users	Entering valid user credentials	The user gets redirected to the Admin Landing Page.
TC003	The login screen is tested for invalid credentials.	Entering invalid user credentials	The user gets an error message prompting “@” missing.
TC004	The Admin Landing Page is loaded.	Logging in with the correct credentials.	The overview of the Sankofa Healthcare Framework is displayed.
TC005	The Organisation List Page is loaded.	Clicking the “Organisation” tab on the Admin Landing Page	The organisations’ information is displayed.
TC006	The Organisation Management is tested.	Clicking the “Manage” button under the “Actions” tab.	The member details are listed.
TC007	Member can be added to organisation	User adds member to organisation	New member exists under organisation

TC008	Existing member details can be edited	User updates member details to reflect change in data	The user can view the update data against the member record
TC009	Member can be removed from organisation	User removes member from organisation	Member no longer exists under organisation

References

Blair-Early, A. and Zender, M., 2008. User interface design principles for interaction design. Design Issues, 24(3), pp.85-107.

Brody, P, Klimas, T, Ghumman, D, Champion de Crespigny, A & Garg, R 2017, 'How this technology could impact the CFO', *EY*, vol. 1, no. 1, pp. 3–4.

H. Davenport, T 2018, The 2 Types of Data Strategies Every Company Needs, Harvard Business Review.

Jagne, J., Smith-Atakan, A.S.G. Cross-cultural interface design strategy. Univ Access Inf Soc 5, 299–305 (2006). https://doi.org/10.1007/s10209-006-0048-6

Karampelas, P 2009, 'Web User Interface Design Strategy: Designing for Device Independence', Springer-Verlag Berlin Heidelberg, vol. Part I, no. LNCS 5614, pp. 515–524.

Wilson, C 2020, *How to Design a Web Application: Software Architecture 101*, Educative: Interactive Courses for Software Developers.