MACQUARIE University
SYDNEY·AUSTRALIA

COMP3850 Project Deliverable Certificate

Name of Deliverable	*Project Plan, Quality Manual & Requirements/Scoping Document*
Date Submitted	*01 / 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	1.5	H	Met with IBM & Other Teams, discussed project architecture and UI overview (16th March)	FT
Everyone	1.5	H	Met with IBM & Other Teams, Discussed UI & Menu structure (23rd March)	FT
Everyone	1	H	Met with IBM & Other Teams, discussed SRS & UI (30th March)	FT
Flynn Tesoriero	1.5	L	Created document outline, headings, notes and formatted sections	SM
Flynn Tesoriero	2	H	Researched sample SRS and read IEEE standards document	SM
Flynn Tesoriero	3	H	Worked on SRS purpose, scope, definitions, references and overview	SM
Flynn Tesoriero	3	H	Worked on SRS product perspective, functions, user classes, environment and documentation	SM
Smriti Kamath	3	M	Functional requirements brainstorming and research	FT
Smriti Kamath	2	M	Non-Functional Requirements research	FT
Smriti Kamath	1.5	L	Design and Implementation Constraints	FT
Smriti Kamath	2	M	Usability Requirements	FT
Maham NAQVI	1.5	M	Worked on the Statement Of Purpose, Scope, Description	FT
Maham NAQVI	3	H	Brainstormed on the potential development risks	FT
Maham NAQVI	3	M	Created the risk matrix table includes the risk, description, probability/likelihood, effect/consequence and risk level	FT
Maham NAQVI	2	H	Researched and worked on risk mitigation	FT
Shivansh Mishra	3	M	Functional requirements brainstorming and research	SM
Isaac Lee	3	M	Tasks/Activities/Phases	SM
Flynn Tesoriero	3	H	Formulated Quality Manual	SM
Isaac Lee	1	M	Timeline	SM
Shivansh Mishra	2	M	Non-Functional Requirements research	SK
Shivansh Mishra	1.5	L	Design and Implementation Constraints	SK
Shivansh Mishra	2	M	Usability Requirements	SK

Mikhail Pavlov	1.5	L	Devised change management process	FT
Shivansh Mishra	1	M	Incorporated Feedback	FT
Smriti Kamath	1	M	Incorporated Feedback	FT
Flynn Tesoriero	3	H	Created timeline, allocated resources	SK
Maham NAQVI	1	M	Added more requirements in the requirements section	FT
Isaac Lee Total	8			
Shivansh Mishra Total	13.5			
Flynn Tesoriero Total	19.5			
Mikhail Pavlov Total	5.5			
Maham Naqvi Total	14.5			
Smriti Kamath Total	13.5			
Team Total	74.5			

Sankofa: A blockchain-based healthcare architecture platform

Project Plan, Quality Manual & Requirements / Scoping Document

Project Plan

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.

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 (External)	Subject Matter Expert

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.

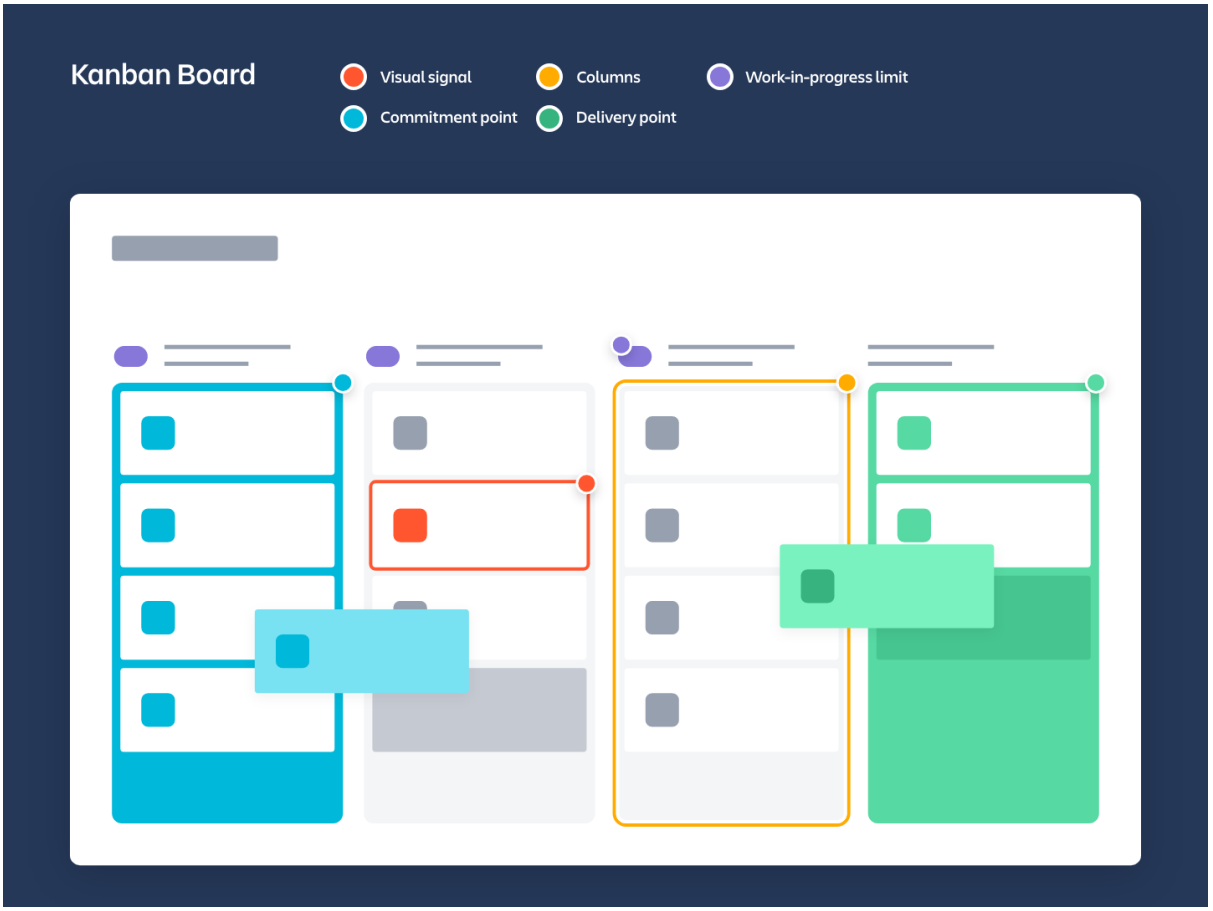
The second stage is Simplicity, having it can bring better clarity by doing the simplest approach when developing a project. It makes team members understand and receive a clearer focus and objective of the project at a basic level.

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.



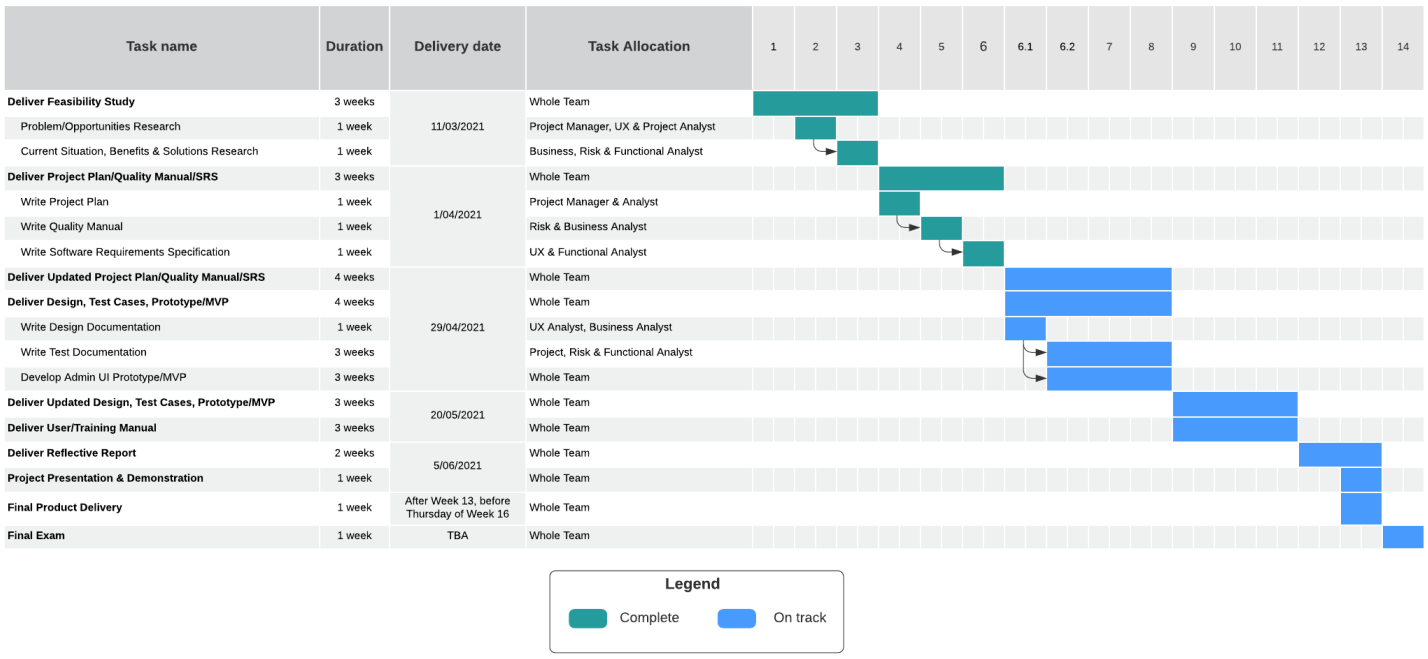
Source: https://www.atlassian.com/agile/kanban/boards
The features of a Kanban board (as above) include:

- 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

Assumptions

- Resources shall be available to the team (i.e. IBM Cloud will not experience downtime)
- Team members actively contribute and attend meetings
- Other teams working on the SHF project deliver their iterations so that components of the application can be linked
- The initial scope of the project is static
- The resources available to the team shall be altered
- The allocation of resources shall be followed
- IBM mentors will be available to the team and willing/able to assist
- The backend architecture of the system shall be able work seamlessly with the front-end UI/ UX design
- The data servers storing sensitive health information shall be able to be accessed in the back-end system
- The system shall be able to share health data with authorized health organizations to track Covid-19 vaccination recipients

Quality Manual

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.

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)

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.

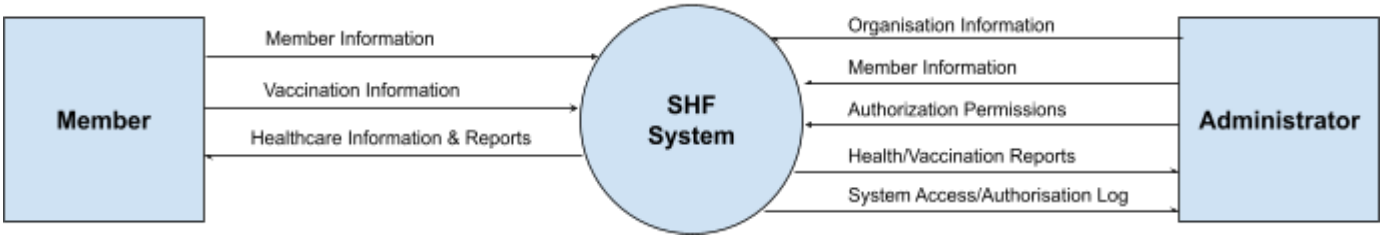
As identified above, the system will provide a framework for sharing healthcare information between numerous parties, particularly vaccination information. A number of teams are working on each component of the system, such as the vaccination user interface, API's, security, blockchain and DevOps.

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]
UI: User Interface The graphical interface through which the user interacts with the system.
SHF: Sankofa Healthcare Framework An open and secure healthcare data platform.

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.

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 must 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 must be written in ReactJS
- **R5.2** The system shall use Carbon UI design system
- **R5.3** The system must be written using JavaScript
- **R5.4** The system must 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 should 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.
R8 The system shall indicate the data is loading through a shimmer or loading spinner
R9 The system shall respond immediately to user actions
R10 The system shall be able to run on a low-speed CPU, such as a notebook laptop
R11 The system shall support a range of modern browsers
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

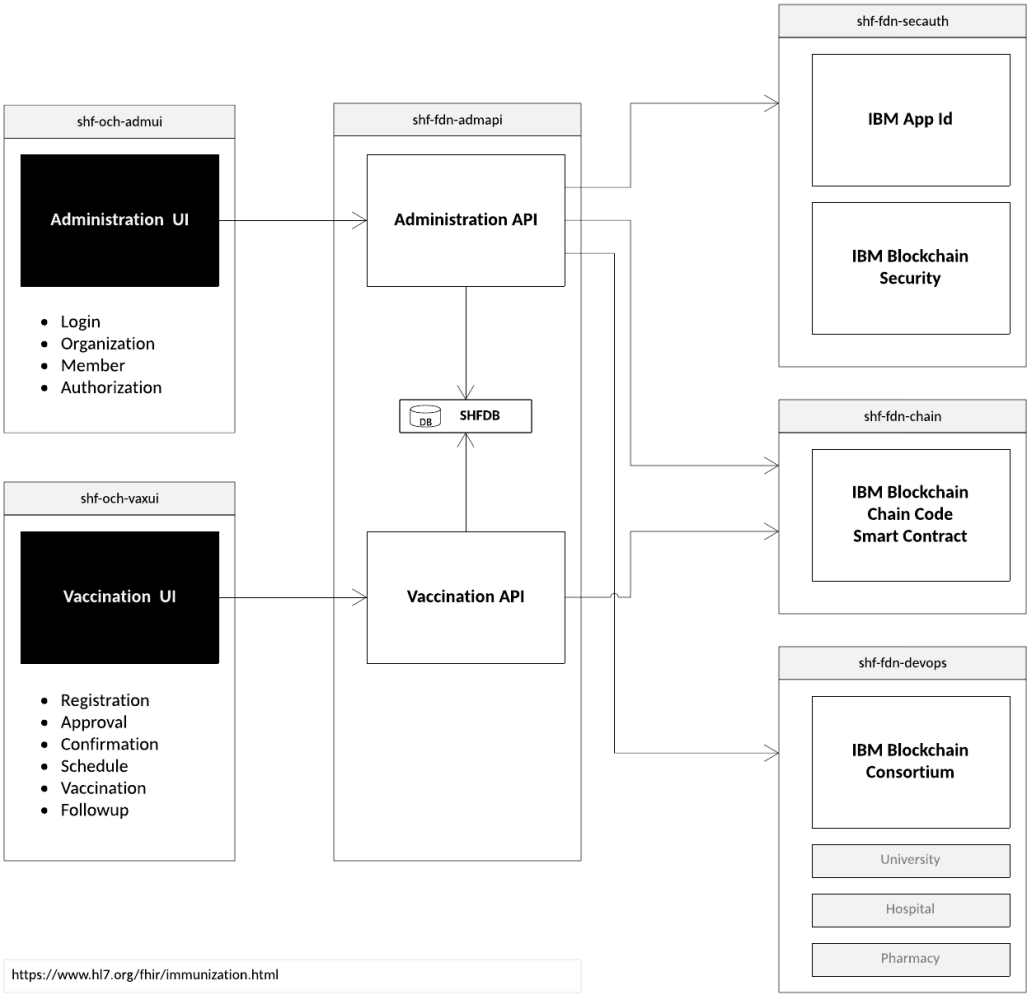
R17. Security
These security requirements involve ensuring the integrity and confidentiality of records in the SHF system. The system shall support the user access classes identified previously.

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 must 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 must 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
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