**High-Level Design Document**

Final Project

Contents

[1 Project Overview 2](#_Toc137748514)

[1.1.1 Brief introduction of the project. 2](#_Toc137748515)

[1.1.2 Purpose of the project. 2](#_Toc137748516)

[1.1.3 Goals of the project: 2](#_Toc137748517)

[2 Project Scope 3](#_Toc137748518)

[2.1.1 Platforms and devices the mobile app will support. 3](#_Toc137748519)

[2.1.2 Key features and functionalities of application. 3](#_Toc137748520)

[3 Architecture and Technology Stack 3](#_Toc137748521)

[3.1.1 Overall architecture of the application. 3](#_Toc137748522)

[3.1.2 Technology stack (e.g., programming languages, frameworks, libraries). 4](#_Toc137748524)

[3.1.3 Rationale behind the chosen technology stack. 5](#_Toc137748525)

[4 User Interface Design 8](#_Toc137748529)

[4.1.1 User interface (UI) design approach. 8](#_Toc137748530)

5 **Full stack Application …………………………………………………………………………………………… 7**

5.1.1 Front-End Approach ……………………………………………………………………………………………… .9

5.1.2 Back-End Approach ………………………………………………………………………………………………...9

6 **Data Engineering…………………………………………………………………………………………………...9**

6.1.1 Data Generation ……………………………………………………………………………………………………..9

6.1.2 Data Warehouse ………………………………………………………………………………………………......10

**7 Data Science…………………………………………………………………………………………………….…11**

7.1.1 Data Science Approach ………………………………………………………………………………………….12

7.1.2 Model Development ………………………………………………………………………………………………12

# Project Overview

### Brief introduction of the project.

Our project, the "Timesheet and Feedback System," is designed to streamline essential administrative tasks for employees. By simplifying timesheet creation and feedback submission while implementing role-based access and project-specific customization, we aim to enhance efficiency and accountability within the organization. With automated reminders ensuring timely completion, our platform facilitates seamless communication and fosters a culture of continuous improvement.

### Purpose of the project.

To simplify timesheet creation and feedback submission processes, enhancing efficiency and accountability for employees.

The key goals of the project include:

* Efficiency enhancement: By streamlining timesheet creation and feedback submission processes, the project aims to reduce the time and effort required for administrative tasks, thus increasing overall organizational efficiency and productivity.
* Communication optimization: Through the implementation of role-based access controls and project-specific feedback customization, the project seeks to facilitate more targeted and meaningful communication between employees and administrators. This not only enhances accountability but also promotes a culture of transparency and collaboration within the organization.

### Goals of the project:

* Streamline Administrative Processes: Simplify timesheet creation and feedback submission procedures to minimize manual effort and increase productivity.
* Ensure Compliance and Accountability: Implement mandatory submission requirements for timesheets and feedback to ensure adherence to organizational policies and foster accountability among employees.
* Development of Core Web Application Components:
  + Front End Interface: Design a user-friendly web interface for timesheet creation and feedback submission, optimized for accessibility and ease of use across different browsers.
  + Backend Implementation: Develop robust backend systems to handle data processing, storage, and user authentication for seamless functionality of the web application.
* Browser Compatibility:
  + Ensure compatibility of the web application with popular web browsers such as Chrome, Firefox, and Safari, to provide a consistent experience for all users.

By simplifying administrative tasks, promoting accountability, fostering collaboration, optimizing resource allocation, and enhancing employee satisfaction, the project aims to drive efficiency and productivity while nurturing a culture of continuous improvement within the organization.

# Project Scope

### Platforms and devices the mobile app will support (e.g., iOS, Android).

|  |  |
| --- | --- |
| Platforms | Version |
| Windows, macOS, Linux | Latest Version |
| Android, IOS | 10 and above, 11 and above |

### Key features and functionalities of the application.

**User Registration and Authentication:**

User-friendly registration process for patients to create their accounts.

Secure authentication mechanisms, including username/password

**Timesheet Creation:**

Users can easily create and submit timesheets, detailing their work hours and tasks completed..

Real-time updates on instrument availability and stock levels.

**Feedback Submission:**

Users can provide feedback on various aspects such as projects, tasks, and team collaboration, with questions tailored based on their roles and project involvement.

**Role-Based Access Control:**

Access to timesheets and feedback features is controlled based on users' roles within the organization, ensuring data security and privacy.

**Project-Specific Feedback:**

Feedback questions are customized based on the specific project an employee is working on, allowing for targeted and relevant feedback.

**Automated Reminders:**

Automated notifications remind users of pending timesheets and feedback submissions, reducing the likelihood of missed deadlines.

**User-Friendly Interface:**

The application features an intuitive and responsive web interface, optimized for ease of use across various devices and screen sizes.

# Architecture and Technology Stack

### Overall architecture of the application.

The architecture and technology stack for the web application will be designed to ensure scalability, reliability, and performance. Here's an outline of the proposed architecture and technology stack:

**Client-Side Interface: (JMAN)**

The client-side interface forms the user-facing component of the mobile application. It provides an intuitive and responsive user interface, allowing patients to access their personalized dashboards, track orders and receive message. The interface is designed to support multiple devices, including iPhone, iPads and Android devices, ensuring a seamless user experience across different screen sizes and resolutions.

**Application Logic Layer: (180-Medical)**

The application logic layer serves as the backbone of the mobile application, encompassing the core business logic and functionality. It handles the processing of user requests, orchestrates data retrieval and storage, and performs necessary calculations and validations.

### Architecture

### 

### Technology stack (e.g., programming languages, frameworks, libraries).

**Technology Stack for the Application Platform:**

The architecture and technology stack for the web application will be designed to ensure scalability, reliability, and performance. Here's an outline of the proposed architecture and technology stack:

**Frontend Framework:**

**React.js**: A popular JavaScript library for building user interfaces. It offers a component-based architecture, making it easy to develop interactive and responsive frontend applications.

**Backend Framework:**

Node.js: A runtime environment for executing JavaScript code on the server-side. Node.js provides non-blocking, event-driven architecture, ideal for building scalable and real-time web applications.

**Database:**

MongoDB: A NoSQL database that offers flexibility and scalability for storing and managing semi-structured data. MongoDB's document-oriented model allows for easy integration with Node.js applications.

For the data engineering aspect of the project, Snowflake will serve as the cloud-based data warehouse, providing scalability and performance for handling large volumes of data. dbt (data build tool) will be utilized for data transformation and modelling, allowing for the creation of data pipelines and the orchestration of data workflows.

**Snowflake**:

**Data Storage**: Snowflake will store all relevant data, including timesheet records, feedback submissions, user information, and project details, in a centralized and scalable manner.

**Data Processing:** Snowflake's data processing capabilities will be leveraged for performing complex queries, aggregations, and analytics on the stored data, enabling efficient data retrieval and analysis.

**Data Integration:** Snowflake supports seamless integration with various data sources and tools, facilitating data ingestion from multiple sources into the data warehouse.

**dbt (Data Build Tool):**

**Data Transformation:** dbt will be used for data transformation tasks, including cleaning, structuring, and modelling the raw data stored in Snowflake. It allows for the creation of SQL-based transformation pipelines, enabling the development of curated datasets and analytical models.

**Data Orchestration:** dbt's workflow management capabilities will orchestrate the execution of data transformation processes, ensuring the consistency and reliability of the data pipelines.

**Data Documentation:** dbt provides features for documenting the data transformation logic and business rules, enabling better understanding and collaboration among data engineers and analysts.

For the data science aspect of the project, machine learning models will be developed to analyse feedback data and derive insights for improving employee performance and project outcomes.

**Data Science (Machine Learning):**

Model Development: Machine learning algorithms will be employed to analyse feedback data and identify patterns or trends related to employee performance, project effectiveness, and overall organizational dynamics.

**Feature Engineering:** Data scientists will perform feature engineering to extract relevant features from the feedback data, such as sentiment analysis, topic modelling, or user behaviour metrics, to train predictive models.

**Model Training and Evaluation:** Supervised or unsupervised learning techniques will be applied to train machine learning models on historical feedback data. The models will be evaluated using appropriate performance metrics to assess their effectiveness and generalization ability.

**Model Deployment:** Deployed models will provide actionable insights and recommendations to stakeholders, enabling data-driven decision-making for improving organizational processes and performance.

### Rationale behind the chosen technology stack.

In our endeavour to create an efficient Timesheet and Feedback System, React.js stands as our technology of choice. With its dynamic user interface capabilities and streamlined development workflows, React.js ensures a seamless user experience and enhanced organizational productivity. The rationale behind this selection is based on the following considerations:

**Scalability**: The selected technologies, such as Node.js for backend and Snowflake for data warehouse, are inherently scalable, allowing the web application to handle growing data volumes and user loads efficiently.

**Performance**: With technologies like React.js for frontend and MongoDB for database, the application can deliver fast response times and smooth user experiences, enhancing overall performance.

**Flexibility**: The architecture supports flexibility in development, allowing for easy integration of new features and adaptation to changing business requirements.

**Cross-platform Compatibility**: React.js for frontend development and Node.js for backend allow for building applications that are compatible with multiple platforms, ensuring wider reach and accessibility.

**Cost Efficiency**: Utilizing open-source technologies like React.js, Node.js, and MongoDB can result in cost savings by reducing licensing fees and development time.

**Community Support**: The selected technologies have large and active developer communities, providing extensive support, resources, and libraries for development and problem-solving.

**Ease of Development**: The component-based architecture of React.js and the event-driven architecture of Node.js simplify development efforts, enabling faster time-to-market for the web application.

**Reliability**: Snowflake's cloud-based data warehouse ensures data reliability and availability, minimizing the risk of data loss or downtime.

**Data Processing and Analysis**: The combination of Snowflake for data storage and dbt for data transformation enables efficient data processing and analysis, empowering organizations to derive valuable insights from their data.

**Futureproofing**: By choosing modern and widely adopted technologies, the web application is well-positioned for future scalability, updates, and enhancements, ensuring its longevity and relevance in the rapidly evolving technological landscape.

Overall, the chosen architecture and technology stack offer a robust foundation for building a scalable, reliable, and high-performance web application that meets the organization's needs and objectives effectively.

# Mobile App Components

### Main components of the mobile app.

* react-native- 0.71
* native-base
* react-native-signature-capture
* react-navigation/native
* react-navigation/native-stack
* moment
* react-native-date-picker
* react-native-splash-screen
* react-native-vector-icons
* axios
* react-native-modal
* react-native-element-dropdown
* Formik
* react-native-async-storage/async-storage

### Purpose of each component.

#### React Native

React Native brings React's declarative UI framework to iOS and Android. With React Native, you use native UI controls and have full access to the native platform.

**Declarative**. React makes it painless to create interactive UIs. Declarative views make your code more predictable and easier to debug.

**Component-Based.** Build encapsulated components that manage their state, then compose them to make complex UIs.

**Developer Velocity**. See local changes in seconds. Changes to JavaScript code can be live reloaded without rebuilding the native app.

**Portability**. Reuse code across iOS, Android, and other platforms.

#### Native Base

Native Base is a mobile-first, accessible component library for building a consistent design system across android, iOS & web.

#### React Native Signature

The react-native-signature-capture plugin is a React Native package that allows you to capture and draw signatures within your mobile application. It provides a simple and customizable signature capture view where users can sign using their finger or a stylus.

#### React Navigation / Native

The react-navigation/native package is a popular navigation library for React Native applications. It provides a set of components and navigation methods to help you easily implement navigation and routing within your mobile app.

#### React Navigation / Native stack

The react-navigation/native-stack package is an extension of the react-navigation/native package that provides a stack-based navigation system for React Native applications. It offers a simple and declarative way to implement stack navigation, where screens are stacked on top of each other and can be pushed or popped from the stack.

#### Moment

The moment package is a widely used and powerful JavaScript library for handling and manipulating dates and times. It provides an extensive range of functionalities to parse, format, manipulate, and display dates and times in various formats.

#### React Native Splash screen

The react-native-splash-screen package is a popular React Native library that allows you to add a splash screen or launch screen to your mobile application. A splash screen is the initial screen that appears when the app is launched, providing an opportunity to display a logo, branding, or loading indicator while the app is initializing.

#### React Native Vector Icons

The react-native-vector-icons package is a widely used React Native library that provides a collection of customizable vector icons for use in your mobile applications. It allows you to easily integrate high-quality icons into your app's UI and customize their appearance to fit your design requirements.

#### React Native Axios

The react-native-axios package is a lightweight JavaScript library that provides an easy-to-use API for making HTTP requests in React Native applications. It is based on the popular axios library and offers a simplified syntax for performing HTTP operations such as GET, POST, PUT, DELETE, and more.

#### React Native Modal

The react-native-modal package is a versatile and customizable library for creating modals in React Native applications. Modals are commonly used to display overlays, alerts, pop-ups, or any other content that needs to be presented on top of the main app screen.

#### React Native Element Dropdown

React Native Element Dropdown is a library that provides a customizable dropdown component for React Native applications. This library simplifies the process of creating dropdown menus and provides a variety of options to customize the dropdown to match the design and functionality of your application.

#### Formik

The Formik npm package is a popular and flexible form management library for React applications. It simplifies the process of building and handling forms by providing a set of powerful features and utilities.

#### React Native Async Storage

The react-native-async-storage package, also known as @react-native-async-storage/async-storage, is a widely-used npm package for storing and retrieving data in React Native applications. It provides a simple and efficient interface to persist key-value pairs asynchronously on the device.

# User Interface Design

### User interface (UI) design approach.

|  |  |
| --- | --- |
| **Splash Screen:**  A screen shot of a cell phone  Description automatically generated with medium confidence | **Walk Through Screen:**  A screen shot of a cell phone  Description automatically generated with medium confidence |
| 1. The mobile splash screen serves as the initial visual introduction to the app, representing its branding and visual identity. 2. It provides feedback to the user that the app is loading and initializing, improving the perception of a smooth transition. 3. The duration of the splash screen should be optimized to balance loading time and user experience, avoiding unnecessary delays. 4. Incorporating brand messaging and a clear call-to-action on the splash screen can enhance user engagement and guide them towards key app features. 5. Designing the splash screen to adapt to different screen sizes and orientations ensures a consistent and visually pleasing experience across devices. | 1. Walkthrough screens provide a guided introduction to the app's features, functionality, and user interface. 2. They help new users understand how to navigate and interact with the app, reducing the learning curve. 3. Walkthrough screens can showcase key app benefits, highlight important features, and set user expectations. 4. Interactive elements, such as swiping or tapping, can be incorporated into walkthrough screens to engage users and encourage exploration. 5. Designing visually appealing and intuitive walkthrough screens can leave a positive first impression and increase user retention. |

# Testing and Quality Assurance

Testing and quality assurance are essential processes in software development that aim to ensure the reliability, functionality, and overall quality of a software product. These processes involve systematically examining and validating the software at various stages to identify defects, bugs, and any deviations from the expected behavior. By conducting thorough testing and quality assurance, developers can uncover and rectify issues early in the development cycle, enhancing user satisfaction, minimizing risks, and ultimately delivering a stable and dependable software product to the end users.

### Testing approach for the mobile app.

When it comes to testing a mobile application, several approaches can be employed to ensure its quality and functionality. Here is the testing approach for a mobile application:

1. Requirement Analysis: Understand the requirements of the mobile application thoroughly, including its intended functionality, target audience, supported platforms (iOS, Android, etc.), and any specific device requirements.
2. Test Planning: Develop a comprehensive test plan that outlines the testing objectives, scope, test environments, test cases, and testing techniques to be used. Determine the types of testing to be performed.
3. Test Environment Setup: Prepare the necessary test environments, including physical devices, emulators, simulators, or cloud-based testing platforms. Ensure that the test environments closely match the actual user environment.
4. Functional Testing: Verify that the mobile application meets the specified functional requirements. Test various features and functionalities of the application, such as user interface, navigation, user input validation, data processing, and integration with backend services.
5. User Interface Testing: Test the mobile application's user interface (UI) to ensure consistency, responsiveness, and adherence to platform-specific design guidelines. This involves checking for proper alignment of elements, accurate rendering of fonts and images, correct color schemes, and intuitive navigation across different devices and screen sizes.

### Types of testing to be performed.

The following types of testing should be carried out in the project:

**Functional Testing:**  
In functional testing, each function in the application tested by giving the input value, determining the output, and verifying the actual output with the expected value. This testing examines the

**Outcome**:

* It ensures that the customer or end-user is satisfied.
* It ensures the all the requirements should be met.
* It ensures the proper working of all the functionalities of an application/software/product.

1. **Usability Testing:**  
   Conduct thorough compatibility testing across different mobile platforms, screen resolutions, and device configurations to identify and address compatibility issues proactively.  
     
   **Outcome**:

* User satisfaction: Measure the level of user satisfaction and identify any areas where users may face difficulties or frustrations while interacting with the mobile application.
* Navigation and intuitiveness: Evaluate the ease of navigation within the mobile application and assess how intuitive the user interface is for performing tasks and accessing different features.
* Learnability: Assess how quickly users can understand and learn to use the mobile application, especially for first-time users, by evaluating the clarity of instructions and the simplicity of the interface.
* Error frequency and severity: Identify the frequency and severity of user errors encountered during usability testing, noting any critical or recurring issues that may hinder user experience.

1. **System Testing:**  
   System Testing (ST) is to evaluate the complete system the system's compliance against specified requirements. In System testing, the functionalities of the system are tested from an end-to-end perspective.   
     
   **Outcome**:

* It covers end to end testing
* System tests is used to specify how the application should behave

1. **Retesting:**  
   Re-testing is executing a previously failed test against new software to check if the problem is resolved. After a defect has been fixed, re-testing is performed to check the scenario under the same environmental conditions.   
     
   **Outcome:**

* It verifies that the problem has been resolved and that everything is operating as intended
* It raises the applications or product’s quality

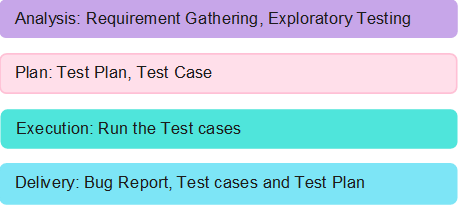
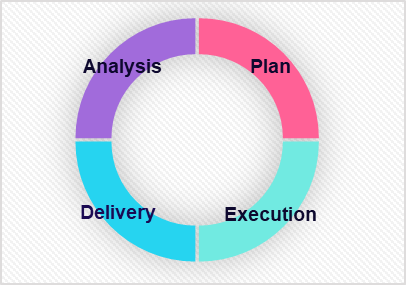
**Regression Testing:**  
Regression testing is responsible for the overall stability and functionality of the existing features. It can be performed on a new build when there is a significant change in the original functionality.  
  
**Outcome:**

* Regression testing ensures a fix does not adversely impact the existing functionality
* Regression testing outweigh the risks of skipping it

1. **Compatibility Testing:**  
   Compatibility Testing is a type of testing to check whether your software is capable of running on different hardware, operating systems, applications or Mobile devices.  
     
   **Outcome:**

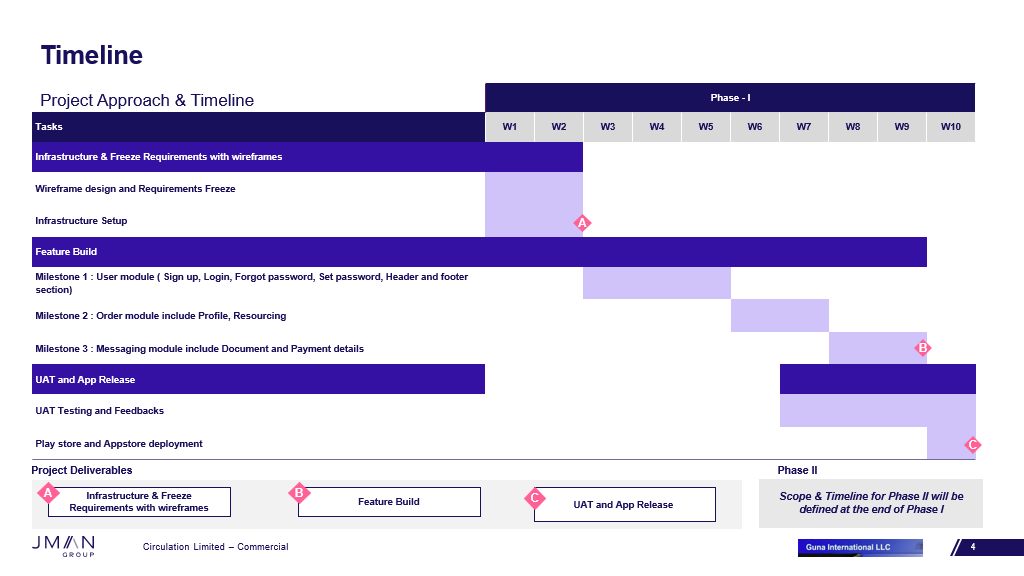
* It ensures that a mobile application functions correctly on different devices and screen sizes.
* It helps identify and resolve compatibility issues early in the development cycle, minimizing user complaints and negative reviews.
* By conducting compatibility testing, developers can ensure a seamless user experience across various platforms, enhancing customer satisfaction.
* The outcome of compatibility testing is a robust and reliable mobile application that works consistently across a wide range of devices, improving its market acceptance and user adoption.

### Quality assurance processes and tools to ensure app reliability.

The objective of the test is to define the goals and purpose of the testing effort. It aims to provide a comprehensive and focused statement of what is to be accomplished through testing. The test objectives serve as a guiding principle for the testing activities and ensure that they are aligned with the overall objectives of the project  
  


# Project Timeline and Resources

### Estimated project timeline, including major milestones.



### Roles and responsibilities of the project team members.

A screenshot of a computer

Description automatically generated with low confidence

# Risks and Mitigation Strategies

### Potential risks and challenges associated with the project.

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO | Risk / Challenges | Impact | Mitigation Plan |
| 1 | If there is a delay in API | Project deliverables will be delayed | Need to have all the API ahead of development team |
| 2 | Application version limitation |  | App will support only versions which is above |

**Appendix Title**

Document Title