**High-Level Design Document**

Mini 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 (e.g., iOS, Android). 3](#_Toc137748519)

[2.1.2 Key features and functionalities of the mobile app. 3](#_Toc137748520)

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

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

[3.1.2 Architecture 4](#_Toc137748523)

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

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

[4 Mobile App Components 6](#_Toc137748526)

[4.1.1 Main components of the mobile app. 6](#_Toc137748527)

[4.1.2 Purpose of each component. 6](#_Toc137748528)

[5 User Interface Design 8](#_Toc137748529)

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

[6 Testing and Quality Assurance 12](#_Toc137748531)

[6.1.1 Testing approach for the mobile app. 12](#_Toc137748532)

[6.1.2 Types of testing to be performed. 13](#_Toc137748533)

[6.1.3 Quality assurance processes and tools to ensure app reliability. 14](#_Toc137748534)

[7 Project Timeline and Resources 15](#_Toc137748535)

[7.1.1 Estimated project timeline, including major milestones. 15](#_Toc137748536)

[7.1.2 Roles and responsibilities of the project team members. 15](#_Toc137748537)

[8 Risks and Mitigation Strategies 15](#_Toc137748538)

[8.1.1 Potential risks and challenges associated with the project. 15](#_Toc137748539)

# Project Overview

### Brief introduction of the project.

Our project aims to develop a Mobile Application Platform that replicates the functionalities of 180 medical patient web portal. This platform will provide a responsive and user-friendly experience for patients and enabling them to conveniently track their medical instruments and supplies.

### Purpose of the project.

The purpose of this project is to empower patients and enhance their experience by providing them with a comprehensive instrument tracking solution.

The key goals of the project include:

* Enhancing Patient Experience: By providing a user-friendly and intuitive interface, the platform aims to empower patients in managing their healthcare journey. Patients will have convenient access to real-time information about their instrument inventory, reorder dates, and order history, leading to a more engaging and personalized experience.
* Improving Patient Engagement: Through personalized dashboards and messages, Patients will have access to essential information about their instrument inventory, reorder dates, and usage history, enabling them to make informed decisions regarding their healthcare needs.

### Goals of the project:

* Design of the UI/UX components for patient portal applications (Mobile and iPad) supporting both Android and iOS as per specification developed during requirement analysis phase.
* The exact scope of work for the Development Phase will be determined by the Requirements Analysis Phase. By the end of development phase, a Minimum Viable Product (MVP) – as scoped in the Requirements phase.
* Design of the Core application tier including:
  + Front End tier for the end-user, accessed via a Mobile application (iPhone & Android) and iPad.
  + Back End API will provide by 180 medicals for both the mobile and iPad applications.
* Deployment to App Store and Play Store.
* Following development, QA testing will be carried out and the application will be submitted for user acceptance testing (UAT). if the App (or any part) fails testing, the CONSULTANT will fix any defects and rectify any functional gaps against the agreed scope at no additional cost during UAT.

The application will support Android version 10.0 and above, and iOS 11 and above

# Project Scope

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

|  |  |
| --- | --- |
| Platforms | Version |
| iOS | 11 and above |
| Android | 10 and above |

### Key features and functionalities of the mobile app.

**User Registration and Authentication:**

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

Secure authentication mechanisms, including username/password

**Personalized Patient Dashboard**:

Customized dashboard for each patient, displaying relevant information such as instrument inventory, usage history, and reorder dates.

Real-time updates on instrument availability and stock levels.

**Seamless Integration:**

Integration with external APIs for accessing information on instrument specifications, usage guidelines, or related educational resources.

**Responsive Design:**

Mobile-friendly design to support various devices, including iPads and smartphones

Responsive user interface for optimal viewing and usability across different screen sizes

# Architecture and Technology Stack

### Overall architecture of the mobile app.

The Mobile Application Platform for patient Instrument Tracking in Healthcare follows a robust and scalable architecture to ensure optimal performance, security, and usability. The architecture consists of the following components:

**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 Mobile Application Platform:**

The technology stack for the Mobile Application Platform for Instrument Tracking in Healthcare will include the following components:

**React Native**:

React Native is a popular JavaScript framework for building cross-platform mobile applications. It allows for the development of native mobile apps using JavaScript and provides a rich set of pre-built UI components.

**TypeScript:**

TypeScript is a statically typed superset of JavaScript that adds optional type annotations. It enhances code quality, improves developer productivity, and helps identify errors early in the development process.

**React Navigation:**

React Navigation is a navigation library for React Native applications. It provides a flexible and customizable solution for handling navigation between screens, including stack navigation, tab navigation, and drawer navigation.

**Axios:**

Axios is a widely used JavaScript library for making HTTP requests from the application to the server. It simplifies the process of sending and receiving data, handling API calls, and managing network requests in the mobile app.

**UI Libraries:**

Various UI libraries compatible with React Native can be utilized to enhance the application's visual appeal and user experience. Library **NativeBase**, components and styles that can be customized to match the application's design requirements.

### Rationale behind the chosen technology stack.

In developing the Mobile Application Platform for Instrument Tracking in Healthcare, we have chosen React Native as the primary technology stack. The rationale behind this selection is based on the following considerations:

**Cross-Platform Compatibility:**

React Native allows us to build a single codebase that can run on both iOS and Android platforms. This cross-platform compatibility significantly reduces development time and effort, as we can leverage a shared codebase to target multiple devices and operating systems. It ensures consistent user experience and functionality across different platforms, minimizing the need for platform-specific development.

**Time and Cost Efficiency:**

By utilizing React Native, we can optimize development resources and streamline the development process. The ability to write code once and deploy it across multiple platforms results in time and cost savings, as we can avoid duplicating efforts for separate iOS and Android development. This efficiency enables faster time-to-market, ensuring the application reaches patients sooner.

**Native-Like Performance:**

React Native allows us to develop mobile applications with native-like performance and user experience. It achieves this by utilizing native components and APIs specific to each platform, resulting in smooth animations, fast response times, and seamless integration with device features. This native performance enhances the overall usability and responsiveness of the instrument tracking application.

**Large Developer Community and Ecosystem:**

React Native boasts a vast and active developer community. This thriving community provides extensive support, resources, and libraries that accelerate development and problem-solving. The availability of a wide range of open-source libraries and components allows us to leverage existing solutions and integrate additional functionalities efficiently.

**Code Reusability:**

React Native's component-based architecture promotes code reusability. By developing reusable components, we can efficiently manage the application's UI elements and logic across different screens. This approach not only simplifies development but also facilitates maintenance and future updates, as changes made to shared components propagate throughout the application.

**Flexibility and Customizability:**

React Native offers flexibility in terms of customization and integration with native modules. It allows us to leverage platform-specific functionalities by incorporating native modules when necessary. This flexibility ensures that we can meet specific requirements, integrate with existing systems, and provide a tailored instrument tracking solution that aligns with your unique needs.

Based on these considerations, React Native emerges as the optimal technology stack for developing the Mobile Application Platform. It enables us to deliver a high-quality, cross-platform application with native-like performance, code reusability, and a vibrant developer community, while ensuring time and cost efficiency in the development process.

# 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

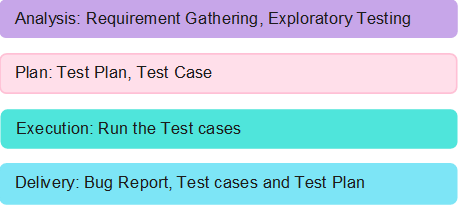
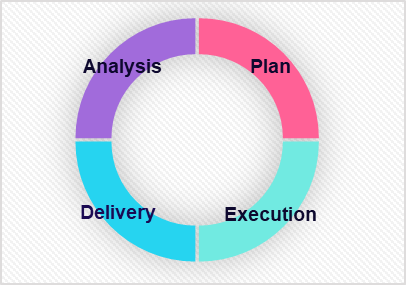
**Regression Testing:**  
Regression testing is responsible for the overall stability and functionality of the existing features. It can be performed on a new build when there is a significant change in the original functionality.  
  
**Outcome:**

* 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:**

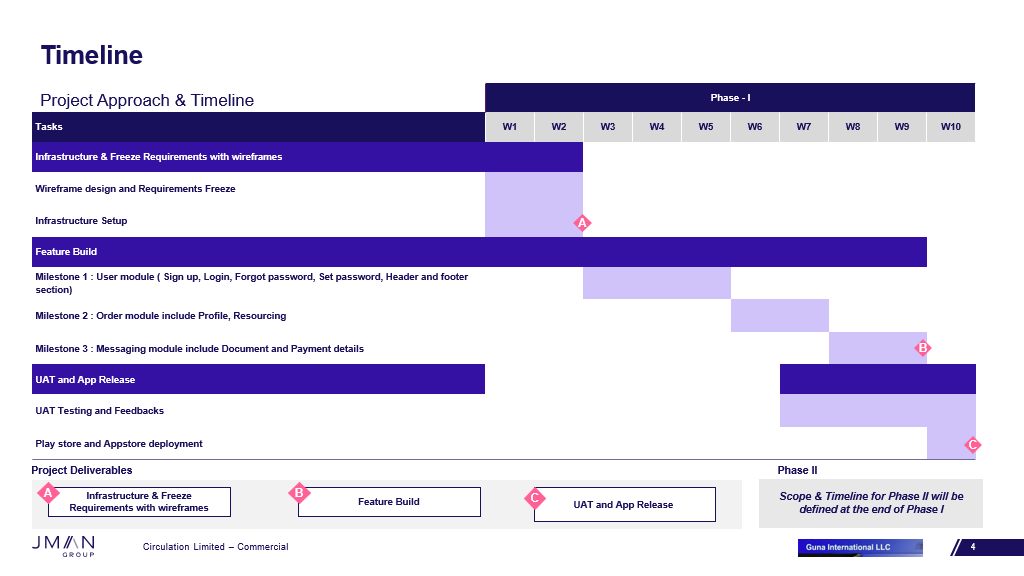
* It ensures that a mobile application functions correctly on different devices and screen sizes.
* It helps identify and resolve compatibility issues early in the development cycle, minimizing user complaints and negative reviews.
* By conducting compatibility testing, developers can ensure a seamless 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