**High-Level Design Document**

JIN Dashboard Module

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 this module will support. 3](#_Toc137748519)

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

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

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

[3.1.2 Architecture 4](#_Toc137748523)

[3.1.3 Technology stack (e.g., programming languages, frameworks, libraries)……………………………………5](#_Toc137748524)

[3.1.4 Rationale behind the chosen technology stack………………………………………………………..………..6](#_Toc137748525)

**4 Dashboard Components…………………………………………………………………………………………7** [4.1.1 Main components of dashboard. 7](#_Toc137748527)

[4.1.2 Purpose of each component. 8](#_Toc137748528)

[5 User Interface Design 9](#_Toc137748529)

[5.1.1 User interface (UI) design approach. 9](#_Toc137748530)

[6 Testing and Quality Assurance 11](#_Toc137748531)

[6.1.1 Testing approach for the modules. 11](#_Toc137748532)

[6.1.2 Types of testing to be performed. 11](#_Toc137748533)

[7 Project Timeline and Resources 12](#_Toc137748535)

[7.1.1 Roles and responsibilities of the project team members. 12](#_Toc137748537)

# Project Overview

### Brief introduction of the project.

Our project aims to develop the dashboard module of JIN Application that replicates the functionalities of web portal. This platform will provide a responsive and user-friendly experience for JMAN member and enabling them to conveniently track their every professional status.

### Purpose of the project.

The purpose of the "JIN App" project is to simplify and enhance the management of personal and professional schedules and tasks for individuals and teams. It aims to provide a centralized platform where users can easily view and organize upcoming birthdays, events, meetings, holidays, and projects. By offering a unified and intuitive interface, the project aims to improve time management, boost productivity, and reduce the risk of missing important dates and deadlines. Additionally, the project facilitates collaboration and communication among team members, making it a valuable tool for both personal and professional use.

The key goals of the project include:

* Enhancing User Experience: By providing a user-friendly and intuitive interface, the platform aims to empower users in managing their professional statuses Users will have convenient access to real-time information about organizational activities, leading to a more engaging and personalized experience.
* Improving User Engagement: Through personalized dashboards and messages, Users will have access to essential information.

### Goals of the project:

* Design of the UI/UX components for Dashboard portal applications (Responsive through every device) 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 JIN web url.
  + Back End API will provide by Sql Workbench Database
* 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)

# Project Scope

### Platforms and devices this module support

|  |  |
| --- | --- |
| Platforms | Version |
| Every Modern Browser | - |
| Every Digital Device | - |

### Key features and functionalities of the module.

**User Registration and Authentication:**

User-friendly registration process for users to login or signup to their respective accounts.

Secure authentication mechanisms, including username/password

**Personalized Patient Dashboard**:

Customized dashboard for each user, displaying relevant information such as Project Allocated, usage history, and many others.

Real-time updates on the home page.

**Seamless Integration:**

Integration with external APIs for accessing information

**Responsive Design:**

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

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

# Architecture and Technology Stack

### Overall architecture of the Dashboard Module.

The Dashboard Module for employees follows a robust and scalable architecture to ensure optimal performance, security, and usability. The architecture consists of the following components:

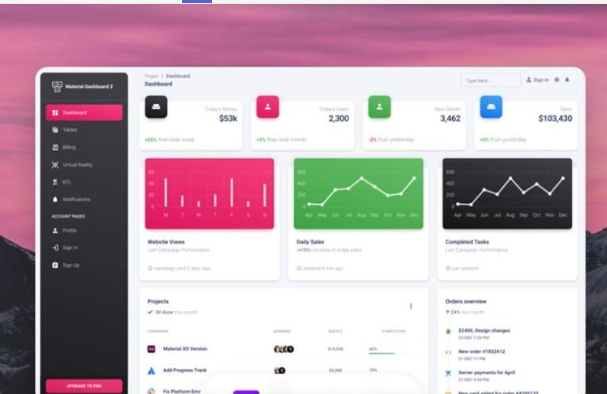
**Client-Side Interface: (JMAN)**

The client-side interface forms the user-facing component of the web components. It provides an intuitive and responsive user interface, allowing employees 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:**

The application logic layer serves as the backbone of the modules, 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

### Architecture(Design Flow) Folder Structure

project-root/

│

├── client/ (React Frontend)

│ ├── public/ (Public assets)

│ │ ├── index.html (HTML template)

│ │ └── ...

│ ├── src/

│ │ ├── components/ (React components)

│ │ │ ├── Dashboard/ (Dashboard components)

│ │ │ │ ├──Upcoming Events Card

│ │ │ │ ├──Upcoming Holidays

│ │ │ │ ├──Upcoming Birthdays

│ │ │ │ ├──Project Running

│ │ │ │ ├──Admin page

│ │ │ ├── Auth/ (Authentication components)

│ │ │ └── ...

│ │ ├── App.js (Main React App component)

│ │ └── ...

│ ├── package.json (Frontend dependencies)

│ ├── package-lock.json (Frontend package lock)

│ └── ...

│

├── server/ (Node.js and Express Backend)

│ ├── controllers/ (API route controllers)

│ ├── models/ (Database models and schemas)

│ ├── routes/ (API route definitions)

│ ├── middleware/ (Custom middleware)

│ ├── config/ (Configuration files)

│ ├── server.js (Express server entry point)

│ └── ...

│

├── database/ (SQL Database Scripts)

│ ├── migrations/ (Database schema migrations)

│ ├── seeds/ (Database seed data)

│ ├── config/ (Database configuration)

│ └── ...

│

├── node\_modules/ (Node.js and Express dependencies)

│

├── .gitignore (Git ignore file)

├── package.json (Backend dependencies)

├── package-lock.json (Backend package lock)

├── README.md (Project documentation)

└── ...

**client/**: This directory contains the React frontend application.

**server/**: This directory contains the Node.js and Express backend application.

**database/**: This directory holds SQL database scripts, including migrations, seed data, and configuration.

**node\_modules/**: This is where Node.js and Express dependencies are installed.

**.gitignore**: A file that specifies what should be ignored by Git version control.

**package.json** and **package-lock.json** in the **client/** and **server/** directories specify frontend and backend dependencies, respectively.

### 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 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.

**Sql Workbench:**

Sql workbench is widely used database for storing data through various sources.

**Node and Express:**

Node.js is known for its non-blocking, event-driven architecture, making it efficient for handling multiple concurrent connections and real-time applications, which is beneficial for interactive dashboards.

Express simplifies the development of RESTful APIs and server-side routing, making it ideal for creating robust backend services that communicate with the SQL database.

Node.js and Express have a large and active developer community, providing access to a wealth of libraries and resources to accelerate development and ensure long-term support.

### Rationale behind the chosen technology stack.

In developing the Dashboard Module, we have chosen below as the primary technology stack. The rationale behind this selection is based on the following considerations:

**1.React(Frontend):**

* **User Experience**: React offers a rich and interactive user interface, making it well-suited for creating a dynamic and user-friendly dashboard.
* **Component-Based**: React's component-based architecture promotes modular and reusable code, enhancing maintainability and scalability.
* **Large Ecosystem**: React has a vast ecosystem of libraries and tools, providing flexibility and efficiency in frontend development.

**2.SQL(Database):**

* **Structured Data**: SQL databases are optimal for managing structured data, which is common in applications involving events, meetings, holidays, and projects.
* **Data Integrity**: SQL databases enforce data integrity through constraints, ensuring reliable and consistent data storage.
* **ACID Properties**: SQL databases guarantee ACID (Atomicity, Consistency, Isolation, Durability) properties, essential for transactional data.

**3.** **Node.js (Backend)**:

* **JavaScript Stack**: Node.js enables a unified JavaScript stack, reducing context switching between frontend and backend development, enhancing productivity.
* **Non-blocking I/O**: Node.js is highly efficient for handling concurrent connections, crucial for real-time features and responsive dashboards.
* **Vibrant Package Ecosystem**: Node.js boasts a rich ecosystem of packages, facilitating rapid backend development.

**4.** **Express (Backend Framework)**:

* **Minimalist Framework**: Express simplifies backend development with a minimalist and unopinionated framework, allowing flexibility in design and architecture.
* **Middleware Support**: Express's middleware architecture enables easy integration of authentication, error handling, and other essential functionalities.
* **Scalability**: Express's lightweight nature makes it suitable for microservices and scalable backend components.

# Dashboard Components

### Main components of the module.

**Frontend (React) Components:**

1. **Dashboard Component:**

**Purpose**: This is the core component responsible for rendering the dashboard interface.

**Functionality**: It displays upcoming birthdays, events, meetings, holidays, and projects, allowing users to view and interact with their data.

**Subcomponents**: Various subcomponents within the dashboard component handle specific sections like birthdays, events, etc.

1. **Authentication Components (e.g., Login, Registration, User Profile)**:

**Purpose**: These components manage user authentication and authorization.

**Functionality**: Users can log in, register, view their profiles, and manage their account settings.

1. Event Creation/Editing Component:

**Purpose**: This component enables users to create and edit events.

**Functionality**: It provides a form for users to input event details, such as title, date, description, and type.

1. **Meeting Management Component**:

**Purpose**: This component handles the creation and management of meetings or appointments.

**Functionality**: Users can schedule, edit, and delete meetings, and view meeting details.

1. **Project Management Component**:

**Purpose**: This component manages projects and their associated tasks.

**Functionality**: Users can create, edit, and delete projects, assign tasks, and track project progress.

1. **Data Display Components**:

**Purpose**: Various components are responsible for displaying data in a user-friendly manner, including lists of birthdays, events, meetings, holidays, and project summaries.

**Functionality**: These components render data fetched from the backend and provide an intuitive user interface for viewing and interacting with the information.

**Backend(NodeJS/Express)-**

1. **API Controllers**:

**Purpose**: These controllers handle incoming API requests from the frontend.

**Functionality**: They validate and process requests related to birthdays, events, meetings, holidays, projects, and user authentication.

1. Model and Database Schema:

**Purpose**: Models define the structure of data in the database.

**Functionality**: They represent tables for birthdays, events, meetings, holidays, projects, and user data, ensuring data consistency and integrity.

1. **Routes**:

**Purpose**: Routes define the API endpoints and their associated controllers.

**Functionality**: They establish the API endpoints for CRUD operations on birthdays, events, meetings, holidays, and projects.

1. **Database Configuration**:

**Purpose**: Configuration files set up the database connection and credentials.

**Functionality**: They enable the application to connect to and interact with the SQL database.

# User Interface Design

### User interface (UI) design approach.

|  |  |
| --- | --- |
| **SignInScreen:** | **Birthday Screen:** |
| A registration form with fields for essential user information, including full name, email address, password, and sometimes additional details like date of birth or profile picture.  **LoginScreen:**      The login page is a crucial component of a web application that allows users to access their accounts. It serves as the entry point for users who already have registered accounts | Creating up coming created the established a connection between frontend backend and database designed the component and fetch data based on birthdays on current date.  **Admin Event Screen:**    Creating the page to upload events on the   created the form for adding the events designed the form accordin to jin template adding information to events table according to the form |
| Creating the admin project Allocation to add and modify the Projects allocation dates and add additional Projects Lists. |  |
|  |  |

# Testing and Quality Assurance

### The stability, functionality, and general quality of a software product are all things that testing and quality assurance work to assure during the software development process. In these procedures, the software is methodically examined and validated at different stages to find errors, flaws, and any variations from the expected behavior. Developers may improve customer happiness, reduce risks, and ultimately deliver a reliable and stable software product to the end users by undertaking extensive testing and quality assurance early in the development cycle.

### Testing approach.

Testing is a crucial aspect of web application development to ensure that the application functions correctly, performs well, and provides a good user experience. Here are some types of testing that are typically performed for a web application::

1. Functional Testing: Test individual components or functions in isolation to ensure they work as expected.
2. Usability 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:

1. **FunctionalTesting:**  
   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 the all the requirements should be met.
* It ensures the proper working of all the functionalities of an application/software/product.

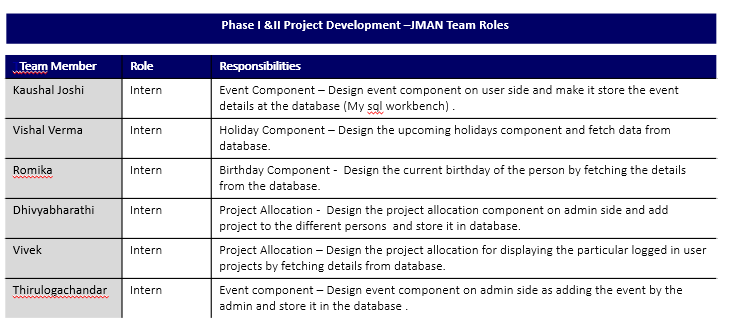
1. **Usability Testing:**
2. Interface (UI) Testing: Check the user interface for consistency, responsiveness, and usability across different devices and browsers.
3. User Experience (UX) Testing: Evaluate the overall user experience, including navigation, user flows, and accessibility.
4. **System Testing:**  
   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**:

* System tests is used to specify how the application should behave

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.

# Project Timeline and Resources

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



**Appendix Title**

Document Title