# ■ Frontend Tech Stack & Development Document

## ■ Project Overview

This document provides a detailed overview of the technologies, frameworks, and tools used for developing the client’s web and mobile applications. The project involves modern, scalable, and high-performance technologies focused on providing an optimized user experience across all platforms.

## ■ Frontend Web Development

### 1. Technologies Used

The web application is developed using React.js, which enables a component-based architecture for better scalability and maintainability. The following technologies are used:

• HTML5 – For structuring and organizing web content.
• CSS3 / Tailwind CSS – For creating responsive, clean, and visually appealing designs.
• JavaScript (ES6+) – For handling logic and interactivity within the application.
• React.js – The primary frontend library used to build Single Page Applications (SPAs).
• React Router DOM – For handling client-side routing and navigation without reloading pages.
• React Hooks / Context API – For efficient state management across the application.

### 2. Development Tools

• Node.js – Used for running JavaScript outside the browser and managing dependencies.
• npm / yarn – Package managers used to install and manage libraries.
• Vite / Create React App – For project scaffolding and efficient development builds.
• VS Code – The primary Integrated Development Environment (IDE).
• Git & GitHub – For version control and collaborative development.

### 3. UI/UX & Libraries

• Framer Motion / Lottie – For animations and transitions.
• React Icons / Lucide Icons – For icons and design consistency.
• Axios / Fetch API – For API calls and data integration.
• Formik / Yup – For form handling and validation.
• React Toastify / Snackbar – For user notifications and alerts.

## ■ Mobile App Development (React Native + TypeScript)

### 1. Technologies Used

The mobile application is built using React Native with TypeScript to ensure type safety, better code readability, and enhanced developer experience. The combination allows cross-platform app development for both Android and iOS using a single codebase.

• React Native – Framework for building native mobile apps using React and JavaScript.
• TypeScript – Superset of JavaScript that adds static typing and better error checking.
• NativeWind – Tailwind CSS-inspired styling library for React Native, allowing utility-based styling with speed and consistency.
• React Navigation – Handles navigation and screen transitions across the mobile app.
• Axios / Fetch API – Used for API communication and data retrieval.
• AsyncStorage / SecureStore – For local data storage and session management.

### 2. Development Tools

• React Native CLI / Expo – Used for building and testing mobile apps.
• Android Studio / Xcode – For emulator setup and debugging.
• VS Code – For development and TypeScript integration.
• Git & GitHub – For version control and team collaboration.

## ■ Development Workflow

1. Clone or pull the latest code from GitHub.
2. Create a new branch for each feature or bug fix.
3. Implement the feature and test it locally.
4. Push the code and create a Pull Request for review.
5. After approval, merge the branch into the main branch.
6. Deploy to staging or production environments.

## ■ Team Roles

• Frontend Developer – Builds and maintains web interfaces using React.js.
• Mobile Developer – Develops Android and iOS apps using React Native and TypeScript.
• UI/UX Designer – Designs user interfaces and prototypes.
• QA Engineer – Tests features and ensures product quality.

## ■ Conclusion

The project leverages a modern and efficient tech stack that includes React.js, React Native with TypeScript, and NativeWind for styling. This combination ensures a unified design system, faster development cycles, and a consistent experience across both web and mobile platforms.