Meal Mate

Installation and Deployment Manual

Group 6:

Avhad, Dipali Vijay

Upadrista, Sarath

Barka, Archana

Dharavath, Harika

Gowru, Kundan Sai Datta Prasad

2024

## **1. Audience Definition**

This deployment manual is expressly designed for technical personnel tasked with the release and management of a mobile application developed using React Native with Expo, utilizing Firebase for backend functionality and incorporating the OpenAI API. The content herein is directed toward the following key roles:

### **System Administrators:**

Individuals in this role are required to:

* Be proficient with mobile operating systems and their app distribution platforms (Apple App Store for iOS and Google Play Store for Android).
* Be adept in using command-line interfaces (CLI) and scripting to facilitate deployment processes.
* Understand version control systems, especially Git, for accessing the application’s codebase on GitHub and managing deployment versions.

### **IT Support Staff:**

These team members should have:

* Basic proficiency in JavaScript and the React Native framework for addressing deployment-related issues.
* Knowledge of cloud services, particularly Firebase, to manage backend configurations.
* Experience with API management, with a focus on securely integrating and maintaining the OpenAI API key within the app.

### **DevOps Engineers:**

Professionals in this category will:

* Oversee the setup and maintenance of continuous integration and continuous deployment (CI/CD) pipelines, if used.
* Handle release cycles, including environment configurations across development, staging, and production stages.
* Monitor post-deployment application performance.

### **Quality Assurance (QA) Engineers:**

QA personnel must:

* Execute thorough testing according to deployment scenarios to verify application functionality before market release.
* Understand the deployment pipeline sufficiently to replicate test environments accurately.

### **Additional Requirements for the Audience:**

All individuals engaged in the deployment process should ensure that they:

* Have access to the application’s GitHub repository, which contains the full codebase and documentation.
* Possess the necessary permissions for Firebase and any other cloud services utilized by the application.
* Have the authority and credentials to publish applications to the Apple App Store and Google Play Store.

The manual presumes that the intended readership is technically savvy and capable of comprehending and implementing the deployment instructions provided. This entails familiarity with the application’s infrastructure, the GitHub repository structure, and the overall deployment workflow.

## **2. Platform-Specific Deployment Instructions**

The deployment of our React Native application via Expo to both iOS and Android platforms requires distinct procedures and tools. This section details the platform-specific steps necessary for a successful deployment to each environment. System administrators and IT personnel will need to follow these instructions carefully to ensure the application is correctly configured and published.

### **For iOS Deployment:**

1. **Access the GitHub Repository:**
   * Clone the repository to your local machine using **git clone [repository-url]**.
   * Ensure you switch to the appropriate branch if needed using **git checkout [branch-name]**.
2. **Install Dependencies:**
   * Run **npm install** or **yarn** to install all required dependencies.
3. **Expo Configuration:**
   * Log in to your Expo account using the Expo CLI with **expo login**.
   * Update the **app.json** file with the correct iOS bundle identifier and Expo configuration.
4. **Build the iOS App:**
   * Use the Expo CLI to build your iOS app with **expo build:ios**.
   * Choose either a simulator build (for testing) or an archive (for uploading to the App Store).
5. **Upload to App Store Connect:**
   * Once the build is completed, use Transporter (available on the Mac App Store) to upload the **.ipa** file to App Store Connect.
   * Ensure you have a valid Apple Developer Account and have set up all the necessary certificates and provisioning profiles in advance.
6. **Test with TestFlight:**
   * Before releasing to the public, use TestFlight to distribute the app to testers for a final check.
7. **Publish to the App Store:**
   * After testing, submit the app for review via App Store Connect.
   * Once approved, set a release date or release it manually to the App Store.

### **For Android Deployment:**

1. **Access the GitHub Repository:**
   * Clone the GitHub repository to your local system as described in the iOS section.
2. **Install Dependencies:**
   * Same as the iOS section, run **npm install** or **yarn** to install the required dependencies.
3. **Expo Configuration:**
   * Similar to iOS, ensure you are logged into Expo and the **app.json** has the correct Android package name and configuration.
4. **Build the Android App:**
   * Create an Android build using **expo build:android**.
   * Choose either an APK or Android App Bundle (AAB) depending on your preference for the Google Play Store.
5. **Upload to Google Play Console:**
   * Use the Google Play Console to upload the build artifact (.apk or .aab).
   * Ensure you have a Google Developer Account with all the required configurations done in the Play Console.
6. **Testing with Internal Test Track:**
   * Utilize the Internal Test Track feature in the Play Console to share the app with internal testers.
7. **Publish to the Google Play Store:**
   * Once testing is concluded, roll out the release to production through the Google Play Console.
   * Monitor the release dashboard for any issues during the rollout.

**Note:** Be sure to increment the version number in **app.json** for each new build submitted to either the Apple App Store or Google Play Store to avoid version conflicts and ensure proper update rollout.

## **3. Prerequisite Installation**

The deployment of our React Native application with Expo, Firebase, and the OpenAI API requires the installation of several key prerequisites. These prerequisites are essential for setting up the development environment on various operating systems.

### **For MacOS:**

1. **Node.js and npm:**
   * Install using Homebrew with the command: **brew install node**.
   * Confirm the installation with **node -v** and **npm -v**.
2. **Git:**
   * Install with Homebrew: **brew install git**.
   * Verify the installation with **git --version**.
3. **Expo CLI:**
   * Install using npm: **npm install -g expo-cli**.
   * Confirm by running **expo --version**.
4. **Firebase CLI:**
   * Install with npm: **npm install -g firebase-tools**.
   * Check the installation with **firebase --version**.
5. **Watchman (Optional):**
   * For better performance of the file-watching service, install Watchman: **brew install watchman**.

### **For Windows:**

1. **Node.js and npm:**
   * Download and install Node.js from the [official website](https://nodejs.org/), which includes npm.
   * Confirm the installation with **node -v** and **npm -v**.
2. **Git:**
   * Download and install Git from [Git's official site](https://git-scm.com/).
   * Verify the installation with **git --version**.
3. **Expo CLI:**
   * Install using npm: **npm install -g expo-cli**.
   * Confirm by running **expo --version**.
4. **Firebase CLI:**
   * Install with npm: **npm install -g firebase-tools**.
   * Check the installation with **firebase --version**.
5. **Windows Build Tools:**
   * Install with npm from an administrator PowerShell or CMD: **npm install --global --production windows-build-tools**.

### **For Ubuntu:**

1. **Node.js and npm:**
   * Install using the package manager: **sudo apt update && sudo apt install nodejs npm**.
   * Confirm the installation with **node -v** and **npm -v**.
2. **Git:**
   * Install with **sudo apt install git**.
   * Verify the installation with **git --version**.
3. **Expo CLI:**
   * Install using npm: **npm install -g expo-cli**.
   * Confirm by running **expo --version**.
4. **Firebase CLI:**
   * Install with npm: **npm install -g firebase-tools**.
   * Check the installation with **firebase --version**.
5. **Build-Essential:**
   * Install using **sudo apt-get install -y build-essential**.

**Note:** After installing these prerequisites, ensure that all environment variables are properly set before proceeding with the deployment. Additionally, secure the OpenAI API key by storing it in an environment variable or a secure secrets management service; do not include it in the public code repository.

## **4. Configuration Instructions**

Proper configuration of the environment and application is critical before proceeding to the deployment phase. Below are the steps for setting up the necessary configuration on MacOS, Windows, and Ubuntu systems.

### **Environment Configuration:**

1. **Environmental Variables:**
   * Securely store the OpenAI API key and any other sensitive credentials as environmental variables.
   * On MacOS and Ubuntu, add environment variables by editing **~/.bash\_profile** or **~/.bashrc**. On Windows, use the system properties to set environment variables.
2. **Firebase Project Configuration:**
   * Initialize your Firebase project by running **firebase init** and follow the prompts to select the Firebase services your project will use.
3. **Expo Configuration:**
   * Edit the **app.json** file in your project root to include the necessary Expo and React Native configurations like your application name, icon, version, and build number.
4. **API Configuration:**
   * Configure the OpenAI API integration by referencing the environmental variable you set for the API key.
5. **Network Configuration:**
   * Ensure your network configuration allows for outbound connections to both Firebase and the OpenAI API.

### **Application Configuration:**

1. **Configuring App Manifest (app.json):**
   * Specify your application icons, splash screens, and other related settings required by Expo.
2. **Firebase Service Configuration:**
   * Set up Firebase services such as Authentication, Firestore, Storage, and Functions by configuring the appropriate files (**firebase.json**, **.firebaserc**).
3. **Runtime Environment Setup:**
   * For React Native, configure any runtime environment settings, including global variables and third-party service integrations within your application.
4. **Security Permissions:**
   * Define the necessary permissions in your project's **AndroidManifest.xml** for Android and **Info.plist** for iOS.

Once all configurations have been properly set up, proceed to test the environment and application to ensure that all services are operational and interacting as expected.

## **5. Deployment Scripts or Code Snippets**

To streamline the deployment process, automate repetitive tasks using scripts. Here are examples of deployment scripts and code snippets that can be utilized:

### **Expo Deployment Scripts:**

**Expo Publish:**

1. expo publish
   * This command is used to publish your JavaScript and assets to an Expo hosting service, which can be rolled out instantly to users across all platforms.

**Expo Build:**

1. expo build:android -t apk

expo build:ios

* + These commands generate standalone binaries for Android (APK) and iOS. You can also build an Android App Bundle (AAB) by changing **t apk** to **t app-bundle**.

**Note:** Be sure to test these scripts in your staging environment before using them in your production environment. Additionally, handle versioning and keep track of your build artifacts to manage your releases effectively.

## **6. Testing and Troubleshooting**

Ensuring that the React Native application operates as expected across all target platforms is critical to a successful deployment. This section provides guidance on testing methodologies and troubleshooting strategies.

### **Testing Instructions:**

1. **Local Testing:**
   * Start the application locally using **expo start**. This will open a development server and display a QR code.
   * Test the application on physical devices by scanning the QR code with the Expo Go app.
   * Use simulators and emulators for iOS and Android to test without physical devices.
2. **Automated Testing:**
   * Implement automated unit tests with Jest or similar frameworks to validate individual functions.
   * Use end-to-end testing frameworks like Detox to simulate user behavior and test the application as a whole.
3. **User Acceptance Testing (UAT):**
   * Define test cases that cover all user stories.
   * Conduct UAT with a closed group of end-users to validate the user experience and functionality.
4. **Performance Testing:**
   * Monitor application performance and responsiveness using Expo's profiling tools or React Native's built-in profiler.
   * Use Firebase Performance Monitoring to gain insights into the performance characteristics of your app.
5. **Security Audits:**
   * Regularly audit your app for security vulnerabilities, ensuring that data handling, storage, and communication are secure.
   * Check that all Firebase and OpenAI API keys and other sensitive credentials are not exposed and are properly utilized within environmental variables.

### **Troubleshooting Common Issues:**

1. **Build Errors:**
   * Verify that all dependencies are correctly installed and linked.
   * Check for any compiler or linker errors in the native code that may occur when building for iOS or Android.
2. **Runtime Errors:**
   * Use the debug mode in Expo and React Native to step through the code and identify any runtime errors.
   * Analyze the stack trace of any crashes to pinpoint the cause of the error.
3. **Connectivity Issues:**
   * Ensure that the application can connect to Firebase and that the correct network permissions are in place.
   * Test the network requests to the OpenAI API to verify that the requests are correctly formed and that the API key is valid.
4. **Deployment Setbacks:**
   * If deployment to Firebase hosting or functions fails, check the Firebase CLI output for error messages.
   * When deploying to the App Store or Google Play, ensure that you meet all their guidelines and review the error messages from the respective console if the upload fails.
5. **User Feedback:**
   * After deployment, collect user feedback for any unexpected behavior or bugs.
   * Monitor crash report logs and analytics to quickly respond to and fix issues.