NJALA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

SCHOOL OF TECHNOLOGY

MSC. COMPUTER SCIENCE SEMESTER 3

COURSE TITLE:

CSD634-MOBILE SOFTWARE DEVELOPMENT WITH REACT NATIVE AND JAVASCRIPT

WORK TITLE: SEMESTER 3 PROJECT

Hotel Booking

SUBMITED TO: Dr. **A.J. FOFANAH**

SUBMITTED BY:

Valentine Mann Bangalie-76100

Musa Marrah-21306

DATE: 2024

Table of contents

Abstract

The hotel booking application is designed to streamline the reservation process for users, providing a seamless experience from search to confirmation. By integrating modern technologies like React Native and JavaScript, the application aims to offer a user-friendly interface that caters to the needs of travelers looking to book accommodations efficiently. The app is equipped with advanced features such as biometric authentication, ensuring secure and quick access to personal accounts and bookings. This project focuses on delivering a high-performance, scalable solution that meets the demands of today's mobile users.

Objectives of the project

- User-Friendly Interface: Develop an intuitive and responsive interface that allows users to easily search, filter, and book hotels.
- Secure Authentication: Implement biometric authentication (fingerprint) to enhance the security of user accounts and streamline the login process.
- Real-Time Availability: Provide real-time hotel availability and booking status through seamless integration with backend services and APIs.
- Personalized Experience: Offer personalized recommendations based on user preferences and past booking history.
- Cross-Platform Accessibility: Ensure the application is accessible on both iOS and Android platforms through the use of React Native.

Target Audience

The primary target audience for this hotel booking application includes:

- Frequent Travelers: Individuals who travel frequently for business or leisure and require a reliable platform to book accommodations.
- Tech-Savvy Users: Users who prefer mobile applications for managing their bookings and value features like biometric authentication for added security.
- Young Professionals: Millennials and young professionals who seek quick and convenient solutions for travel arrangements.
- Hotel Enthusiasts: Users interested in exploring various accommodation options, including luxury and boutique hotels, with detailed information and reviews.

Technologies Used

- React Native: For developing a cross-platform mobile application that works seamlessly on both iOS and Android devices.
- JavaScript: The core programming language used for developing the logic and functionality of the application.
- APIs: Integration with third-party APIs for fetching hotel data, real-time availability, user authentication, and payment processing.
- Biometric Authentication: Integration of fingerprint recognition technology for secure user login and access.
- Database: A backend database to store user data, booking history, and hotel details, ensuring quick retrieval and processing.

This write-up provides a comprehensive overview of the key aspects of the hotel booking project, outlining the objectives, target audience, and technologies used in its development.

Application Design

Wireframes

The initial phase of the application design involves creating wireframes for all major screens of the hotel booking application. Wireframes serve as the blueprint for the app's layout and structure, providing a clear visual guide for developers and designers. The key wireframes include:

Home Screen:

- Features a search bar for users to input their destination, check-in, and check-out dates.
- Prominent buttons for accessing user profile, bookings, and customer support.
- Display of featured hotels or special offers.

Search Results Screen:

- A list of hotels matching the user's search criteria.
- Filters and sorting options (e.g., price range, star rating, distance).
- Each hotel listing includes a thumbnail image, name, rating, price per night, and a brief description.

Hotel Details Screen:

- A detailed view of the selected hotel, including multiple images, amenities, room types, and guest reviews.
- Options to choose room type and add extras (e.g., breakfast, airport transfer).
- "Book Now" button leading to the booking confirmation page.

Booking Confirmation Screen:

- Display of the selected room details, total cost, and payment options.
- Input fields for user details and payment information.
- Biometric authentication prompt (fingerprint) for secure and quick confirmation.

User Profile Screen:

- Access to personal information, booking history, and saved preferences.
- Options to update profile details, manage payment methods, and log out.

Booking History Screen:

- A list of past and upcoming bookings, with options to view details, modify, or cancel reservations.

User Interface (UI)

The UI design focuses on creating a visually appealing and user-friendly interface that aligns with modern design trends while maintaining a professional look. Key aspects of the UI design include:

- Color Scheme: A consistent and appealing color palette that reflects the brand identity, with primary colors for action buttons and neutral tones for backgrounds.
- Typography: Clear and readable fonts for all text, with varying sizes to establish a visual hierarchy.
- Iconography: Simple and intuitive icons to enhance navigation and provide visual cues without overwhelming the user.
- Responsive Design: Ensuring that the UI adapts smoothly across different screen sizes and resolutions, providing a consistent experience on both iOS and Android devices.

User Experience (UX)

The UX design is centered on providing a smooth, intuitive, and enjoyable experience for the user. The primary UX considerations include:

- Easy Navigation: Simple and straightforward navigation paths, with clearly labeled buttons and menu options, allowing users to move between screens with minimal effort.
- Quick Access: Fast loading times and minimal input requirements, with features like autofill for user details and saved preferences, enhancing convenience.
- Personalization: Customizable user profiles that allow users to save preferences, receive tailored recommendations, and view personalized content based on past interactions.
- Security and Trust: Biometric authentication for secure access to accounts and bookings, along with clear communication of privacy policies and data protection measures.
- Error Handling: User-friendly error messages and guidance to help users quickly resolve issues, such as incorrect input or payment failures.

By focusing on these aspects of application design, the hotel booking application aims to deliver a seamless experience that meets the needs of its users, encouraging repeat usage and customer loyalty.

Development

Setup

The development phase begins with setting up the project environment. This includes:

- Development Environment: project is initiated using React Native CLI, with Node.js installed as the runtime environment. Package managers like npm or Yarn are used to handle dependencies.
- Version Control: Git is used for version control, with a repository set up on GitHub (or a similar platform) for collaboration and code management. Branching strategies ensure organized development, with branches for features, bug fixes, and releases.
- Testing Framework: Jest is integrated for unit testing, ensuring that each component functions correctly as the application evolves.

Components

The application is built using a modular approach, where each screen and feature is divided into reusable components. Key components include:

- Search Bar: A component that handles user input for searching hotels by location, date, and other filters.
- Hotel Card: A component that displays a hotel's image, name, rating, and price, used in the search results and recommended hotels.
- Hotel Details: A detailed view component that shows comprehensive information about a selected hotel, including amenities and reviews.
- Booking Form: A form component that captures user information and payment details during the booking process.
- Profile: A component that displays and allows users to edit their profile information, view booking history, and manage preferences.

Each component is designed to be highly reusable and customizable, enabling a consistent look and feel across the application while allowing for flexibility in design and functionality.

State Management

For managing the application's state,

React's Context API is utilized, providing a way to share state across components without passing props manually. This is particularly useful for:

- User Authentication State: Managing user login status and storing user information securely across the application.

Booking State: Storing the user's selected hotel, room type, and booking details throughout the booking process.

- UI State: Handling UI elements such as loading spinners, error messages, and navigation states.

For more complex state management needs, **Redux** could be considered, especially for handling large-scale state management across multiple screens and components.

Navigation

The application's navigation is handled using React Navigation, a popular library in the React Native ecosystem. The navigation structure includes:

- Stack Navigation: For handling transitions between screens like the Home, Search Results, Hotel Details, and Booking Confirmation screens.

Tab Navigation: For managing bottom tab navigation between core features such as Home, Bookings, and Profile.

- Drawer Navigation: (Optional) For providing quick access to additional options like Settings, Help, and Logout from a side drawer menu.

Each navigation type is configured to ensure smooth transitions and a seamless user experience.

API Integration

The application integrates with several external APIs to fetch real-time data and enhance functionality:

Hotel Data API: To retrieve hotel listings, availability, pricing, and detailed information.

Payment Gateway API: For processing payments securely within the app.

Geolocation API To assist users in finding nearby hotels based on their current location.

Authentication APITo manage user login, registration, and secure session handling.

API requests are handled using Axios or Fetch API, with proper error handling and data validation.

Data Storage

To manage data storage and retrieval efficiently, the following solutions are implemented:

Local Storage: Async Storage is used for storing small amounts of data locally on the device, such as user preferences, authentication tokens, and search history.

SQLite: For more complex local data storage needs, **SQLite** is integrated to manage larger datasets like booking history and hotel information offline.

**Backend Database: Integration with a backend database (e.g., Firebase, MongoDB) ensures that user data, bookings, and hotel details are stored securely and can be accessed across devices.

Data synchronization strategies ensure that local and remote data are kept consistent.

Authentication

User authentication is a critical component of the application, ensuring secure access and data protection. The authentication process includes:

Biometric Authentication: Users can log in using fingerprint recognition, adding a layer of security and convenience.

Token-Based Authentication: Upon successful login, the application receives a token (e.g., JWT) from the server, which is stored securely using Async Storage for session management.

Social Login Integration: (Optional) Users can log in using social media accounts like Google or Facebook, providing more flexibility in accessing the app.

Testing

Unit Testing

Unit testing is a crucial step in the development process to verify that individual components and functions of the hotel booking application work as expected. The focus of unit testing includes:

Component Testing: Each core component, such as the Search Bar, Hotel Card, and Booking Form, is independently tested to ensure they function correctly. For example, tests check that the Search Bar handles user input properly and that the Hotel Card displays accurate information.

Functionality Testing: Key functions that process data, like sorting hotel search results or calculating total booking costs, are tested to ensure they return the correct outputs under various conditions.

Edge Case Handling: Tests are designed to cover edge cases, such as invalid input or no available search results, ensuring that the application handles these situations gracefully.

Unit testing is conducted using Jest and React Native Testing Library, with tests automated to run during the development process, ensuring that any issues are caught early.

Integration Testing

Integration testing ensures that different components and modules of the application work together seamlessly. Key aspects of integration testing include:

API Integration: Tests verify that the application correctly communicates with external APIs, such as retrieving hotel data, processing payments, and managing user authentication. For instance, the test ensures that the Hotel Details component accurately pulls and displays data from the backend.

Component Interaction: Testing the interaction between components, such as ensuring that selecting a hotel from the Search Results Screen correctly navigates to the Hotel Details Screen with the appropriate data passed through.

State Management: The application's state management system is tested to ensure that state changes trigger the correct updates in the user interface, maintaining consistency across different screens and user actions.

Integration testing is essential for validating that all parts of the application work together as intended, with a combination of automated and manual testing techniques used.

User Testing

User testing is conducted to gather real-world feedback and ensure the application meets user needs and expectations. The process involves:

Beta Testing: A beta version of the application is released to a select group of users who represent the target audience. These users perform typical tasks, such as searching for hotels, making bookings, and navigating the app, to identify any usability issues or bugs.

Usability Testing: Observing users as they interact with the application to identify pain points, confusing navigation, or features that need refinement.

Feedback Collection: Gathering detailed feedback through surveys, interviews, and in-app feedback mechanisms. This feedback is analyzed to identify areas for improvement, which are then prioritized in future updates.

User testing is a vital step to ensure that the application provides a smooth, intuitive experience and meets the needs of its users.

User Manual

A user manual is created to guide end-users on how to use the hotel booking application effectively. The manual includes:

Getting Started: Instructions for downloading, installing, and setting up the application on both iOS and Android devices.

Feature Guides: Step-by-step instructions for performing key tasks within the app, such as searching for hotels, viewing details, making bookings, and managing user profiles.

Troubleshooting: Solutions to common issues that users might encounter, such as difficulties logging in, payment errors, or problems with booking confirmations.

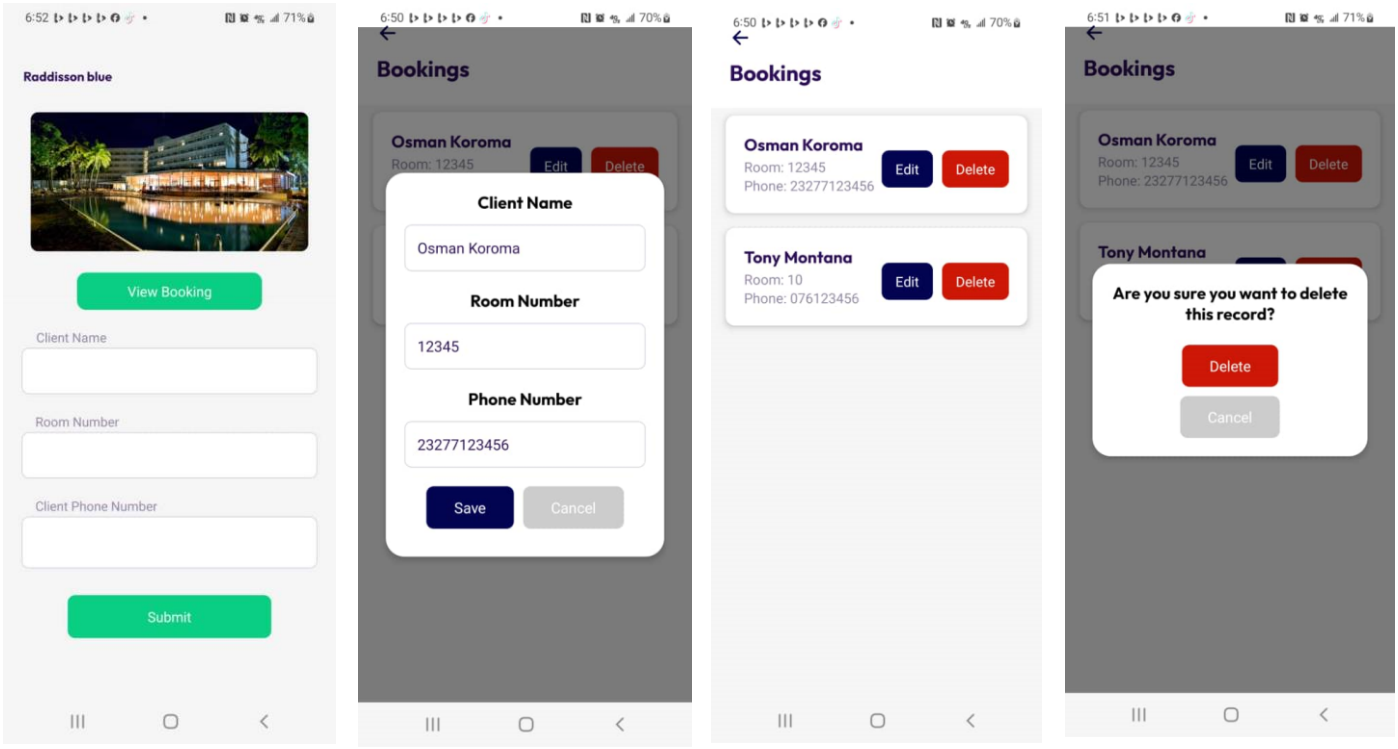
Frequently Asked Questions (FAQ): A section addressing common questions and providing quick answers to help users navigate the app more efficiently.

The user manual is designed to be clear and easy to follow, available both within the app and as a downloadable PDF.

Technical Documentation

Technical documentation provides an in-depth overview of the application's architecture, components, and the technologies used. It is intended for developers and technical stakeholders and includes:

Architecture Overview: A detailed description of the application's architecture, including diagrams that illustrate the flow of data between the frontend, backend services, and external APIs.



Component Breakdown: Documentation of each major component, including its purpose, how it interacts with other components, and any dependencies it may have.

Technology Stack: An overview of the technologies and tools used in the project, such as React Native for the frontend, Node.js for backend services, and various APIs for data retrieval and processing.

API Documentation: Detailed information about the APIs integrated into the application, including endpoints, request/response formats, authentication methods, and error handling procedures.

Deployment Instructions: A guide for setting up the development environment, building the application, and deploying it to production environments, including configuration settings for both iOS and Android platforms.

Technical documentation serves as a comprehensive resource for developers, providing the necessary information to understand, maintain, and extend the application in the future.

This write-up outlines the testing strategies and documentation practices that ensure the hotel booking application is reliable, user-friendly, and well-supported for both users and developers.

The authentication process is designed to be secure, seamless, and user-friendly, with options for password recovery and account management.

Error Handling

Effective error handling ensures a smooth user experience even when issues arise. Key error-handling strategies include:

Try-Catch Blocks: Used extensively in API calls and asynchronous operations to catch and manage errors gracefully.

User Feedback: When an error occurs, user-friendly messages are displayed, guiding users on how to resolve the issue or providing reassurance (e.g., “Please check your internet connection and try again”).

Logging: Errors are logged locally or sent to a remote server for monitoring, helping developers identify and fix issues in future updates.

Fallbacks: Default actions or fallback data are provided in case of API failures, ensuring the app remains functional even if some services are temporarily unavailable.

By implementing robust error handling, the application minimizes disruptions and maintains a high level of reliability and user trust.

This write-up covers the core aspects of the development process, ensuring that the hotel booking application is built on a solid foundation with a focus on performance, security, and user experience.