The Shamiyana App

Aakash Maheshwari (B21CS002)
Naman Goyal (B21CS048)

Motivation

The Shamiyana on our campus receives numerous orders, particularly in the evening. Thus, an app through which we can place our orders and get an expected ready time can be quite useful.

Solution

Here is a software requirement specification for the app we want to make, which includes a high-level design (HLD) and a low-level design (LLD):

High-level design (HLD)

The high-level design of the app includes all of the system's components, interfaces, and architecture. The mobile application will serve as the client, and a web application will serve as the server. This web application will manage Shamiyana's menu, orders, payments, and customer feedback.

Components

- **Mobile app client**: The mobile app will be available for Android devices (for a start) and will include the user interface (UI) for ordering food, making payments, and tracking waiting time.
- **Web application server**: The web application will be hosted on a cloud server and will include the server-side logic for managing the restaurant's menu, orders, payments, and customer feedback.

Interfaces

1. **User interface**: The mobile app UI will include the following screens:
2. **Home screen**: Displays the restaurant's menu and search options.

3. **Item screen**: Displays the details of a selected item and allows users to customize it.
4. **Cart screen**: Displays the items added to the cart and allows users to modify the order.
5. **Payment screen**: Allows users to make online payments using UPI or digital wallets.
6. **Order tracking screen**: Allows users to track the status of their orders, and get an expected waiting time.
7. **Feedback screen**: Allows users to leave feedback on their orders and rate the restaurant's food and service.
8. **API interface**: The mobile app will communicate with the web application server using a RESTful API that supports CRUD operations for the restaurant's menu, orders, payments, and customer feedback.

Low-level design (LLD)

The low-level design of the proposed app includes the specifications of the app's components, including the data structures, algorithms, and functions used to implement the app's features.

Mobile app client

The mobile app client will be developed using native Android and iOS frameworks and will include the following components:

1. **Authentication module**: This module will manage user authentication and securely store user credentials.
2. **Menu module**: This module will fetch the restaurant's menu from the web application server and display it to the user using a list view or grid view. The module will allow users to search for items and filter them by category or price.
3. **Order module**: This module will handle the order creation and modification process and store the order details locally on the device. The module will allow users to add or remove items from the cart, customize items, and view the order summary.
4. **Payment Module**: This module will manage the online payment process through an external payment gateway. The module will encrypt the user's payment information and send it to the payment gateway in a secure manner.
5. **Order tracking module**: This module will fetch the status of the user's orders from the web application server and display them to the user using a list view or map view. The module will update the order status (waiting time) in real-time using push notifications.
6. **Feedback Module**: This module will allow users to provide feedback on their orders and rate the quality of the restaurant's food and service. The feedback will be stored locally on the device and securely transmitted to the web application server.

Web application server

The web application server will be built with modern web technologies like Firebase and have the following parts:

1. **Authentication module**: This module will handle the user authentication process and store the user's credentials securely. The module will use OAuth2.0 authentication for secure and seamless login.
2. **Menu Module**: This module will manage the restaurant's menu using a database, such as Firebase Database (a NoSQL database), and will expose a RESTful API for the mobile app to retrieve and modify menu data. The API will provide endpoints for menu item CRUD operations, such as GET /menu, POST /menu, PUT /menu/id, and DELETE /menu/id.
3. **Order module**: This module will manage the restaurant's orders using a database and expose a RESTful API for the mobile application to create and modify orders. The API will support CRUD operations on orders, such as POST /orders, PUT /orders/id>, and GET /orders/id>.
4. **Payment Module**: This module manages the online payment process through a third-party payment gateway, such as Razorpay. The module will encrypt and securely transmit the user's payment information to the payment gateway. The module will also take care of payment-related concerns.
5. **Order tracking module**: This module will update the status of the user's orders in real-time using push notifications. The module will also store the order status history and expose a RESTful API for the mobile app to fetch the status of a specific order or all orders.
6. **Feedback Module**: This module will store the ratings of the restaurant's food and service, as well as the user-submitted feedback. The module will store feedback in a database and expose a RESTful API for the mobile app to submit and retrieve feedback data.

Data structures and algorithms

The proposed app will use the following data structures and algorithms to implement its features:

- **Array**: Used to implement the cart data structure in the mobile app client.
- **Hash table**: Used to store and retrieve the menu data and order data in the web application server.
- **Sorting algorithm**: Used to sort the menu items by price or name in the mobile app client and the web application server.

Functions and APIs

The proposed app will use the following functions and APIs to implement its features:
Authentication functions: Used to handle user authentication, such as login, logout, and password reset.

- **Menu APIs**: Used to fetch and modify the restaurant's menu data, such as getMenuItem, addMenuItem, updateMenuItem, and deleteMenuItem.
- **Order APIs**: Used to create and modify orders, such as createOrder, updateOrder, and getOrderStatus.
- **Payment APIs**: Used to handle online payments, such as initiatePayment, verifyPayment etc.
- **Notification APIs**: Used to send push notifications to the mobile app client, such as sendOrderStatusUpdateNotification and sendFeedbackReplyNotification.
- **Feedback APIs**: Used to handle customer feedback, such as submitFeedback and getFeedback.

Software Quality Product Aspects

The functionality will be mainly based on two aspects of Software Product Quality, namely Security and Usability.
1. Security: The reason that security is going to be an integral part of the application we are making is because of the integration of an Online Payment Gateway. This requires serious security measures to ensure that the user details are kept safe. Also, storing the user credentials is a matter of security in order to protect the personal information of the user.
2. Usability: The application we are developing is going to be a new introduction for the facilities available on the college campus. Being user friendly will make it more popular among the intended users and thus increase the adaptation rate of the app. Thus making a usable app will help it be accepted by people more easily.

Modelling

We plan to use the **Iterative Model** in order to develop this app. The reason is that this will allow us to start a new iteration of the app in case we find errors during the testing or later phases. And since there are just two people in the team, we plan to use the Series Implementation of the Iterative Model.

Timeline

Week 1: Making of SRS.

Week 2: Designing the skeleton of the application.

Week 3: Completing the UI of the app (Programming).

Week 4: Planning the database structure for storing the user info, menu and placed orders. Completing the user authentication and sign-in / sign-out process. Completing the Authentication Module.

Week 5: Fetching the information about the menu, item availability, from the database to be displayed on the UI. Enabling the option to add desired items to the cart in desired quantity. Completing the Menu Module.

Week 6: Unit Testing for the modules completed so far (Auth and Menu).

Week 7: Buffer Week (In case some tasks were not completed on time).

Week 8: Preparing the bill for the items added to cart. And start working on integrating the payment gateway.

Week 9: Finish the payment process, and add the order to the database on successful payment completion. Completing the Order and Payment Module.

Week 10: Unit Testing for Order and Payment Modules. Integration and System Testing and finishing the app.