Dining Reservations UML Diagrams

Justin Gangan

October 14, 2024

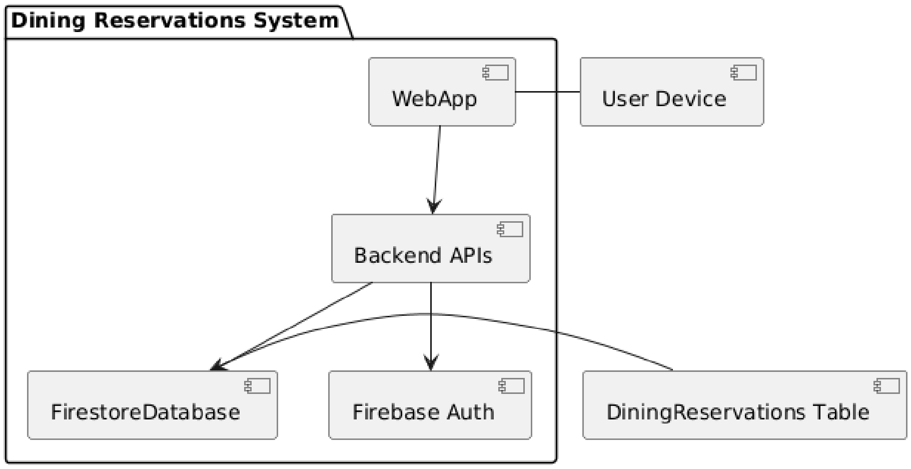
This document contains the UML diagrams related to the Dining Reservations system, including the component, class, activity, sequence, and ERD diagrams. Each diagram is explained in detail.

1 Component Diagram

Description: The component diagram illustrates the high-level architecture of the dining reservation system. The system is divided into three major components:

- **Frontend (React App)**: Provides the user interface where users can create, update, view, or delete their dining reservations.
- **Backend API (Firebase Functions)**: Acts as the intermediary between the frontend and the database, processing reservation requests, interacting with the database, and sending confirmation messages to users.
- **Database (Firebase Firestore)**: Stores user and reservation data, including reservation times, dates, venues, and the user's unique ID.

This diagram shows how the components communicate with each other through HTTP requests and database operations.

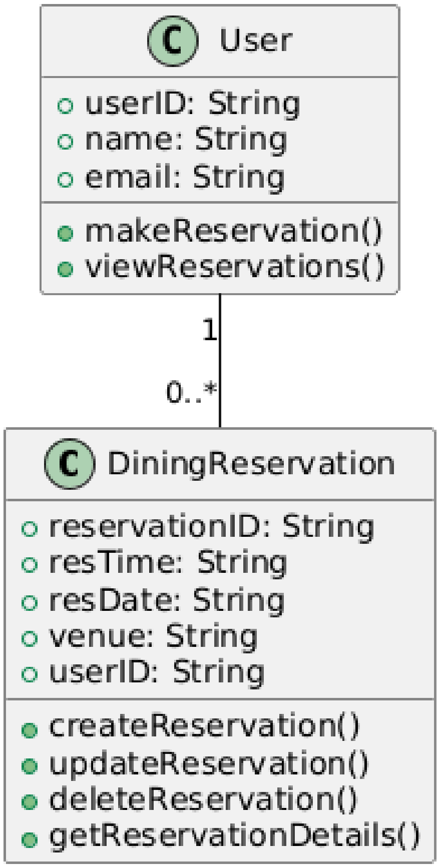


2 Class Diagram

Description: The class diagram provides a structural overview of the system's main entities:

- The ‘User‘ class represents the users of the system, with attributes such as ‘userID‘, ‘name‘, and ‘email‘.
- The ‘DiningReservation‘ class stores information about individual reservations, including ‘reservationID‘, ‘resTime‘, ‘resDate‘, ‘venue‘, and ‘userID‘ (which serves as a foreign key to the ‘User‘ class).

The class diagram also highlights the one-to-many relationship between users and their reservations, where a single user can make multiple reservations.

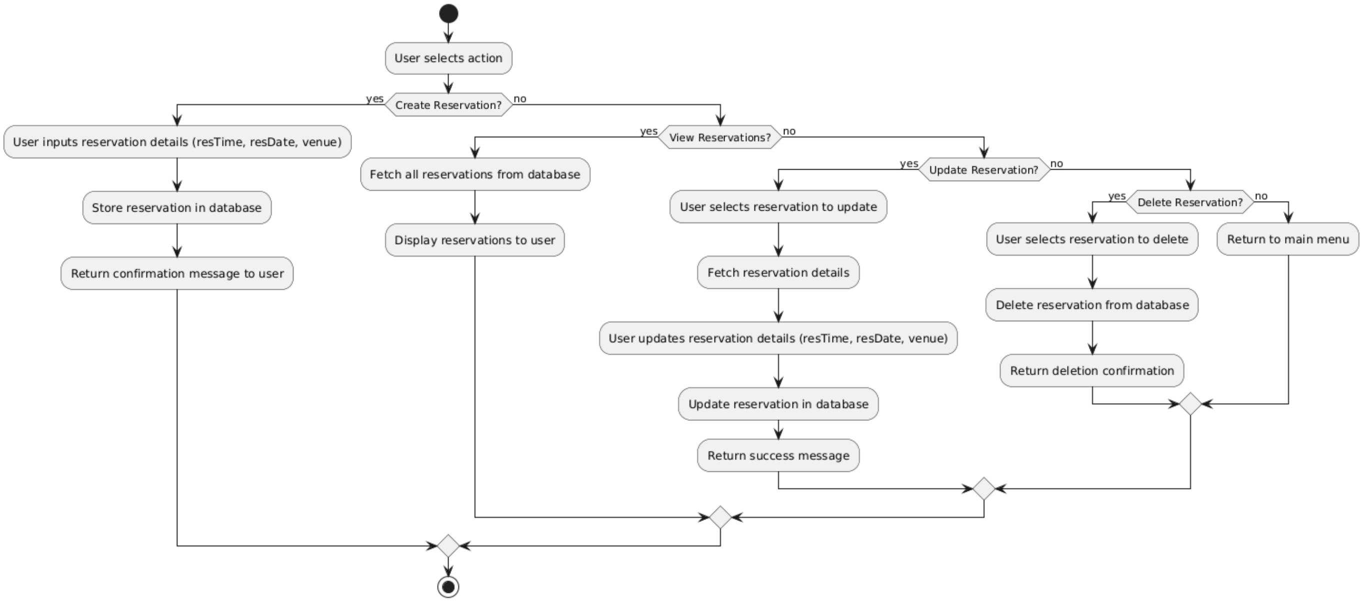


3 Activity Diagram

Description: The activity diagram shows the step-by-step workflow for managing dining reservations. It covers four main actions:

- **Create Reservation**: Users input reservation details (time, date, venue) and submit the form. The system stores the data in the database and sends a confirmation message.
- **View Reservations**: Users can retrieve and view their existing reservations from the database.
- **Update Reservation**: Users select an existing reservation, modify the details, and save the changes, which are updated in the database.
- **Delete Reservation**: Users can choose to delete an existing reservation, and the system will remove it from the database.

This diagram helps illustrate the decisions and steps users can take as they interact with the system.

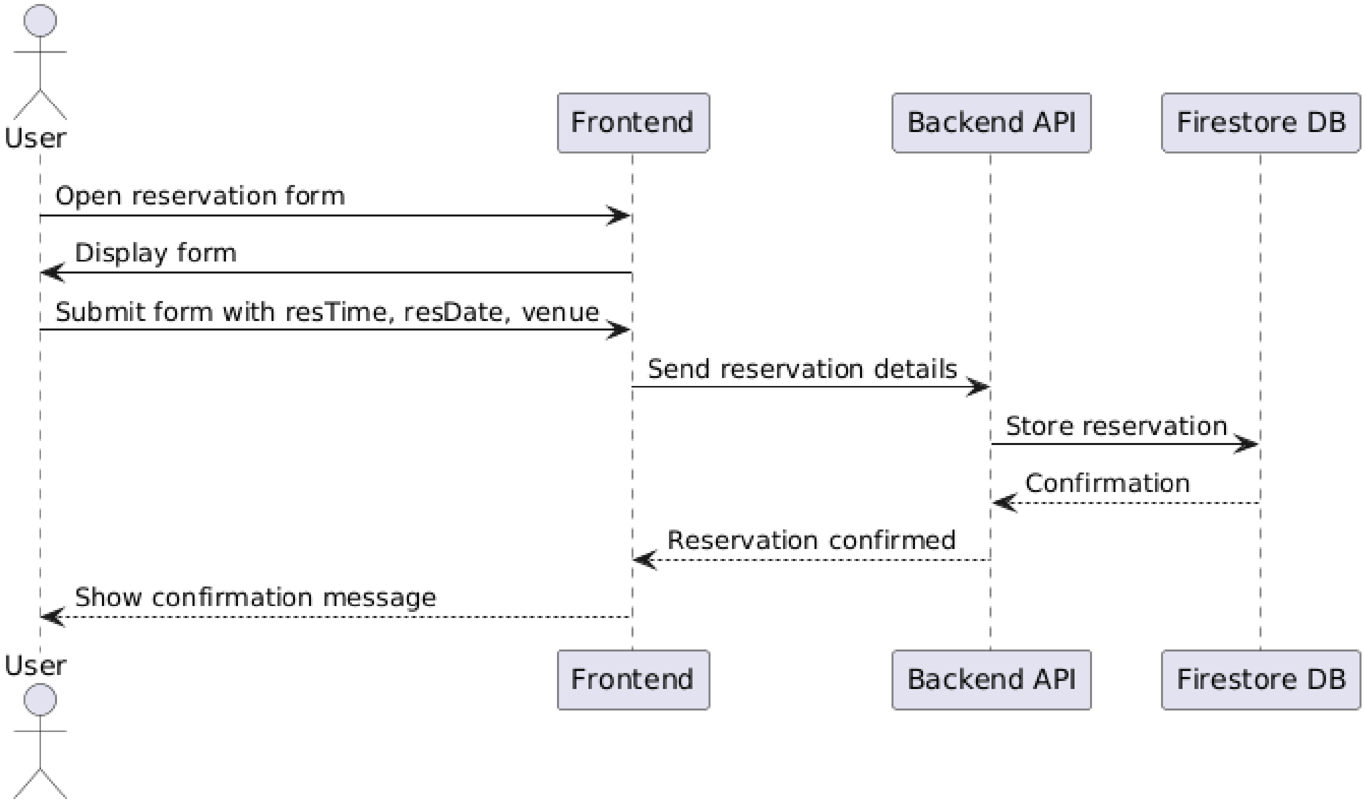

User selects action
yes
Create Reservation?
no
User inputs reservation details (resTime, resDate, venue)
Store reservation in database
Return confirmation message to user
yes
View Reservations?
no
Fetch all reservations from database
Display reservations to user
yes
Update Reservation?
no
User selects reservation to update
Fetch reservation details
User updates reservation details (resTime, resDate, venue)
Update reservation in database
Return success message
yes
Delete Reservation?
no
User selects reservation to delete
Delete reservation from database
Return deletion confirmation
Return to main menu

4 Sequence Diagram

Description: The sequence diagram illustrates the interaction flow between the user, the frontend interface, the backend API, and the database when a reservation is created:

- The user sends a request to create a reservation via the frontend.
- The frontend sends the request to the backend API, which processes the data.
- The backend API stores the reservation in the database and sends a confirmation back to the frontend.
- The frontend displays a confirmation message to the user.

This diagram emphasizes the sequence of messages and interactions required for a reservation to be successfully created.

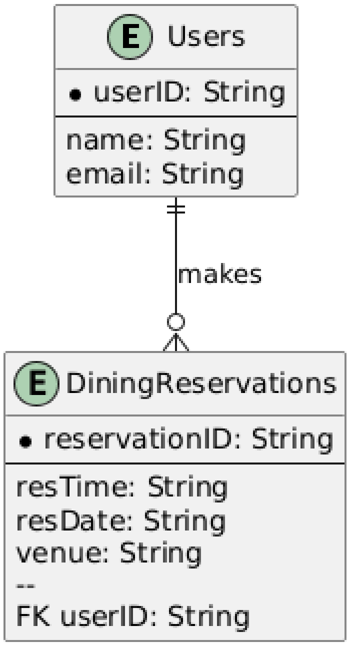


5 Entity-Relationship Diagram (ERD)

Description: The ERD shows the relationship between the ‘Users‘ and ‘DiningReservations‘ tables. Key points include:

- Each user (represented by the ‘Users‘ table) can make multiple reservations, leading to a one-to-many relationship between the ‘Users‘ and ‘DiningReservations‘ tables.
- The ‘userID‘ attribute in the ‘DiningReservations‘ table acts as a foreign key, linking each reservation back to the corresponding user in the ‘Users‘ table.

This diagram provides a clear representation of how data is structured and linked within the database.


E
Users
• userID: String
name: String
email: String
makes
E
DiningReservations
• reservationID: String
resTime: String
resDate: String
venue: String
--
FK userID: String