Bus Booking Systems

Introduction

The Bus Booking System presented here is a comprehensive solution designed to streamline the bus reservation process, catering to the needs of both administrators and users.

Bus Booking System aims to deliver a reliable, responsive, and scalable platform for a superior booking experience. With a focus on user satisfaction, cost-effectiveness, and performance, this system represents a significant advancement in the realm of online bus reservations.

Roles Description -

User Features:

1. Browse Buses:
    a. Explore available buses based on source and destination.
    b. Prioritize buses located near the user's location for convenience.
2. Check Seat Availability:
    a. View the number of available seats for a selected bus.
    b. Ensure real-time information on seat availability.
3. Seat Booking:
    a. Reserve a seat on a preferred bus.
    b. Receive a distinctive seat number upon successful booking.
    c. Prevent concurrent booking of the same seat.
4. Cancel Booking:
    a. Effortlessly cancel a previously booked seat.
    b. Update current occupancy for accurate seat availability.
5. ETA and Distance:
    a. Display estimated time of arrival (ETA) and distance for each bus.
    b. Make informed decisions based on travel time and distance.

Admin Features -

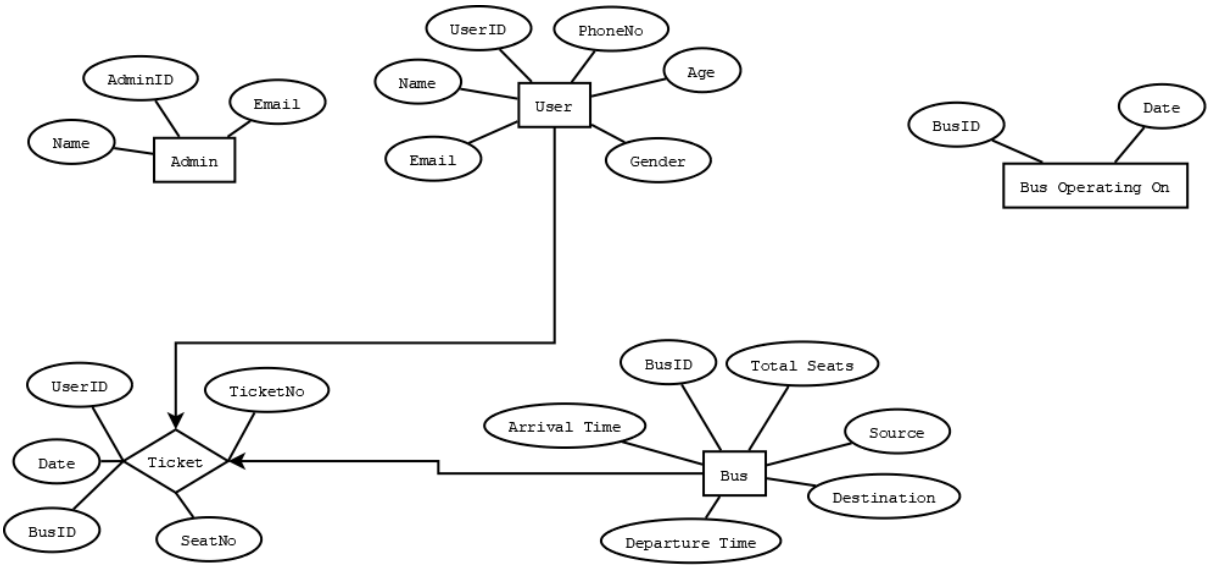
1. Manage Bus Details:
    a. Add, update, and delete information related to buses.
    b. Maintain accurate records of bus details, including name, total seats, occupancy, and available days.
2. Upload Seat Plans:
    a. Upload bus seat plans with seat numbers and designs.
    b. Enhance user experience with visual representations of bus layouts.
3. Bus Route Creation:
    a. Define bus routes by specifying source and destination.
    b. Facilitate efficient organization of bus schedules.
4. Real-time Monitoring:
    a. Monitor and manage current occupancy to ensure accurate seat availability.
    b. Receive alerts for potential issues or discrepancies.

Features

1. Authentication - We can use google API to sign in for different roles for both user and admins. And after that accordingly the session will be created and the user will be directed to the concerned web page.
2. Concurrent Control - Semaphores is used to avoid the conflicts when two users try book same seat in the bus.

Design

1. Database Schema -



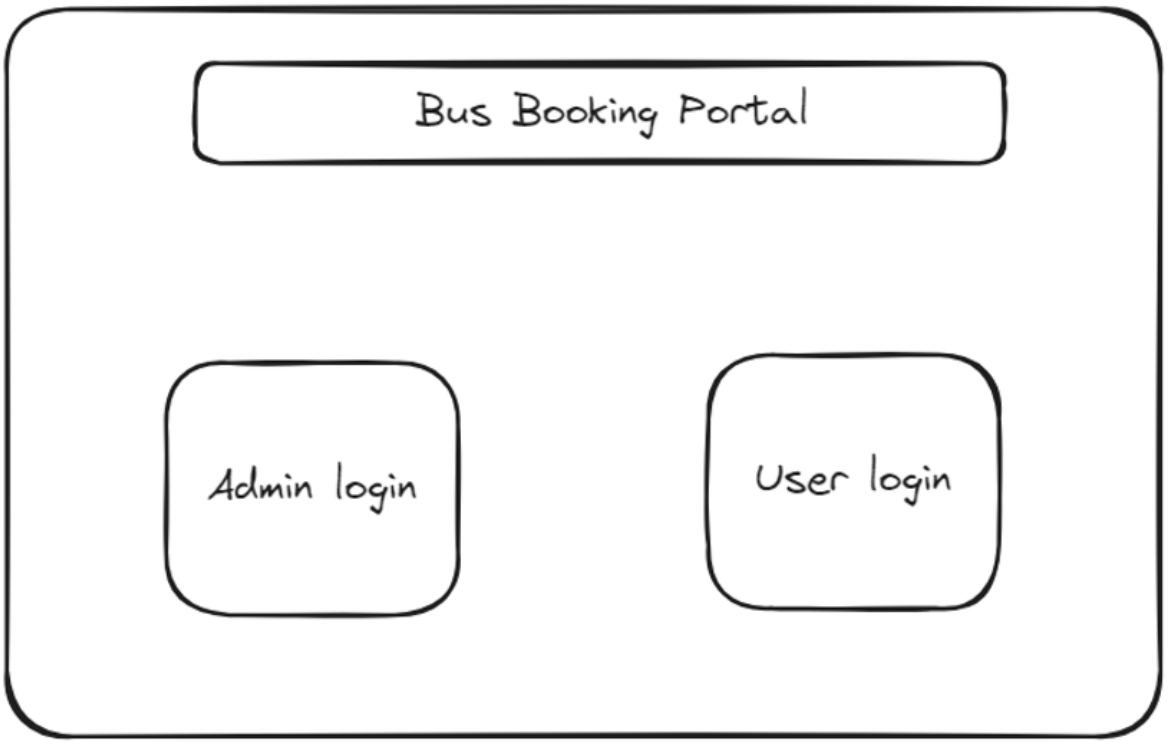
2. OOPs - Following classes will be implemented
    a. Class User -
    Functions
        i. Getter & setter functions
        ii. Book Ticket
        iii. Add User
        iv. Get Bookings
        v. Cancel Booking
    b. Class Admin -
    Functions
        i. Getter & Setter functions
        ii. Get Buses
        iii. Add Bus
        iv. Update Bus
        v. Delete Bus
        vi. Get All Bookings
    c. Ticket -
    Functions
        i. Getter & Setter functions
        ii. Add Ticket
        iii. Delete Ticket

    d. Bus -
        i. Getter & Setter functions
3. Caching -
    a. Use a key-value store to map unique identifiers to cached data and cache bus details using BusID as the key.
    b. Cache frequently queried data such as:
        i. Bus details (BusID, BusName, TotalSeats, CurrentOccupancy, AvailableDays, etc.).
        ii. Seat availability for specific buses.
        iii. Recently retrieved user-specific data.

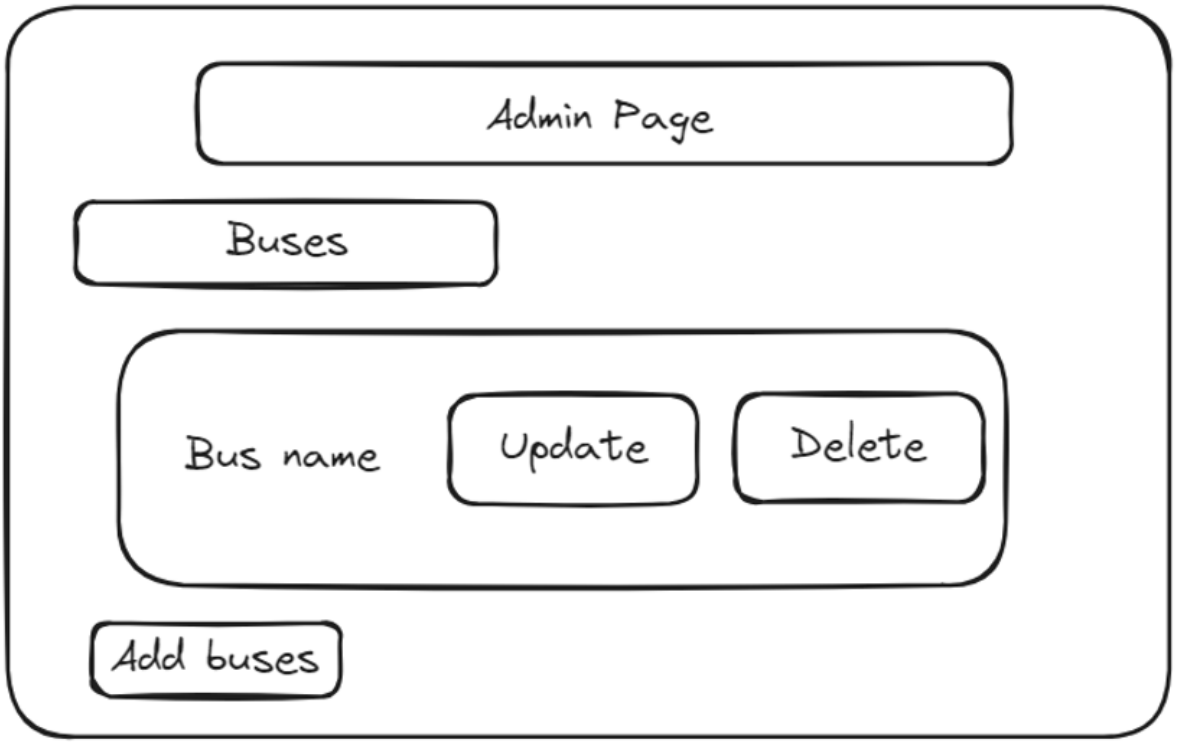
Low Level Design - Cost Estimation & Datastructure,

Testing & Error Handling -

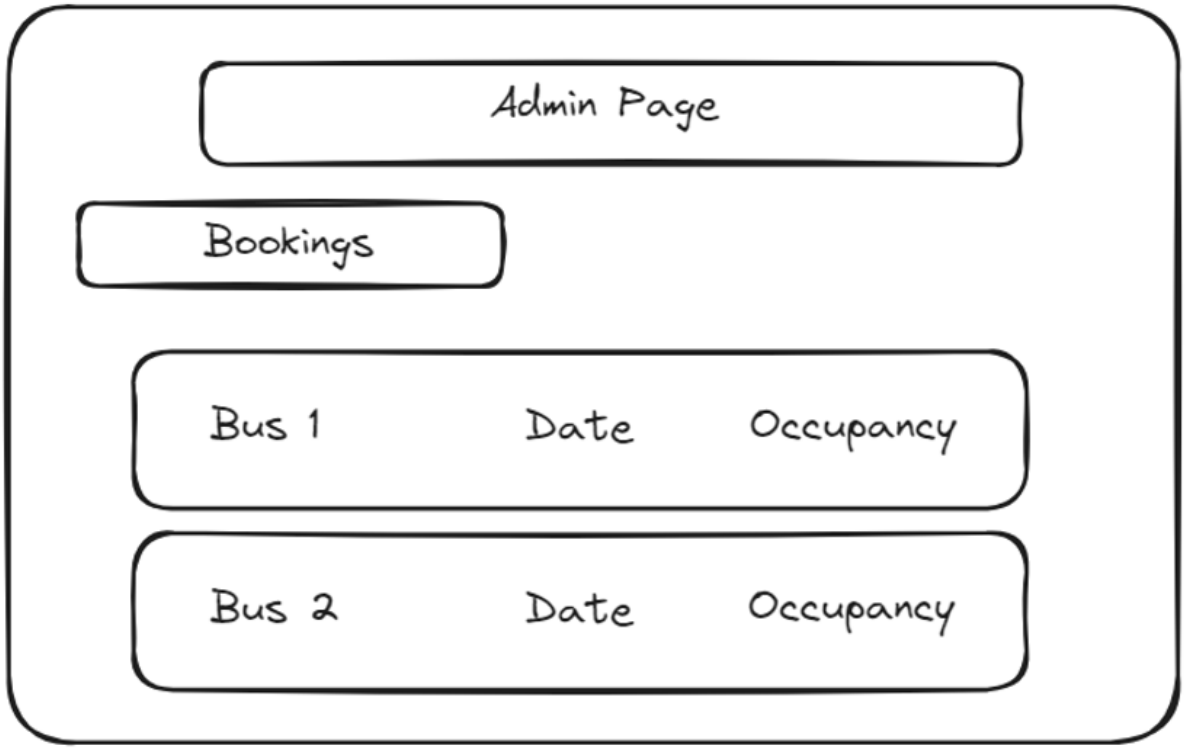
API Design



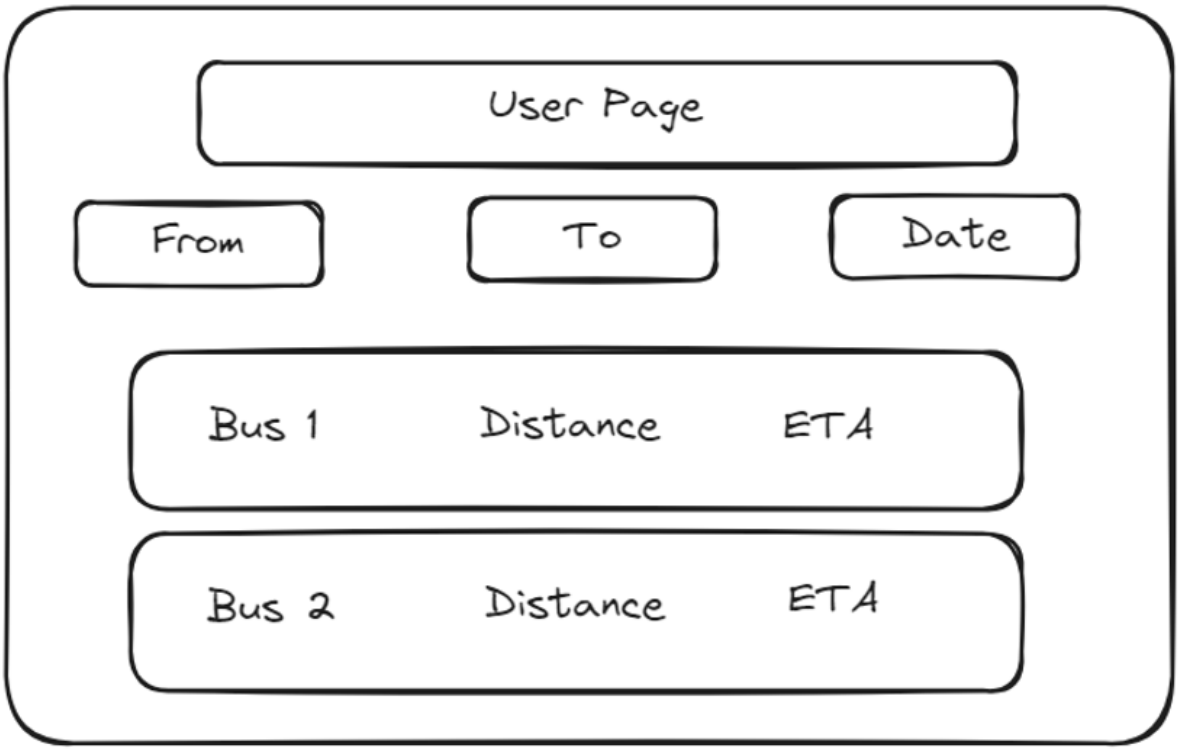
GET: /login/admin
GET:/ login/user



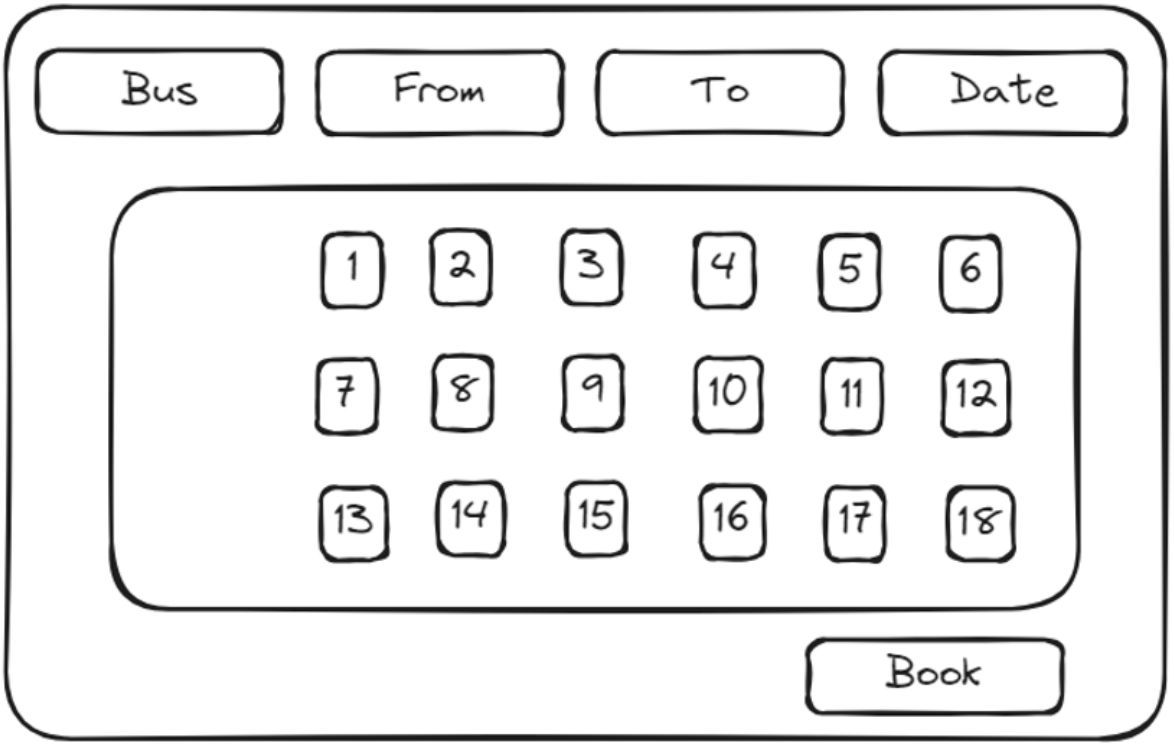
GET: /admin/show_buses
POST: /admin/update/{bus_id}
POST: /admin/delete/{bus_id}
POST: /admin/insert/{bus_id}



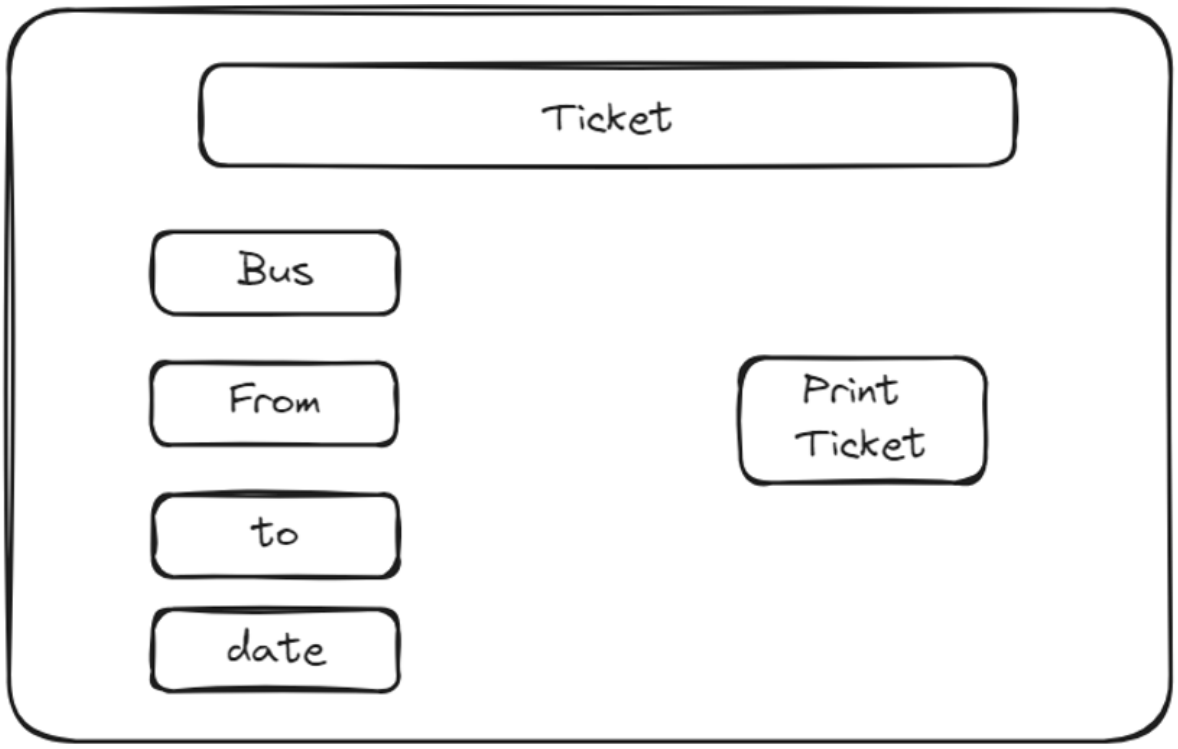
GET: /admin/bookings/show_all



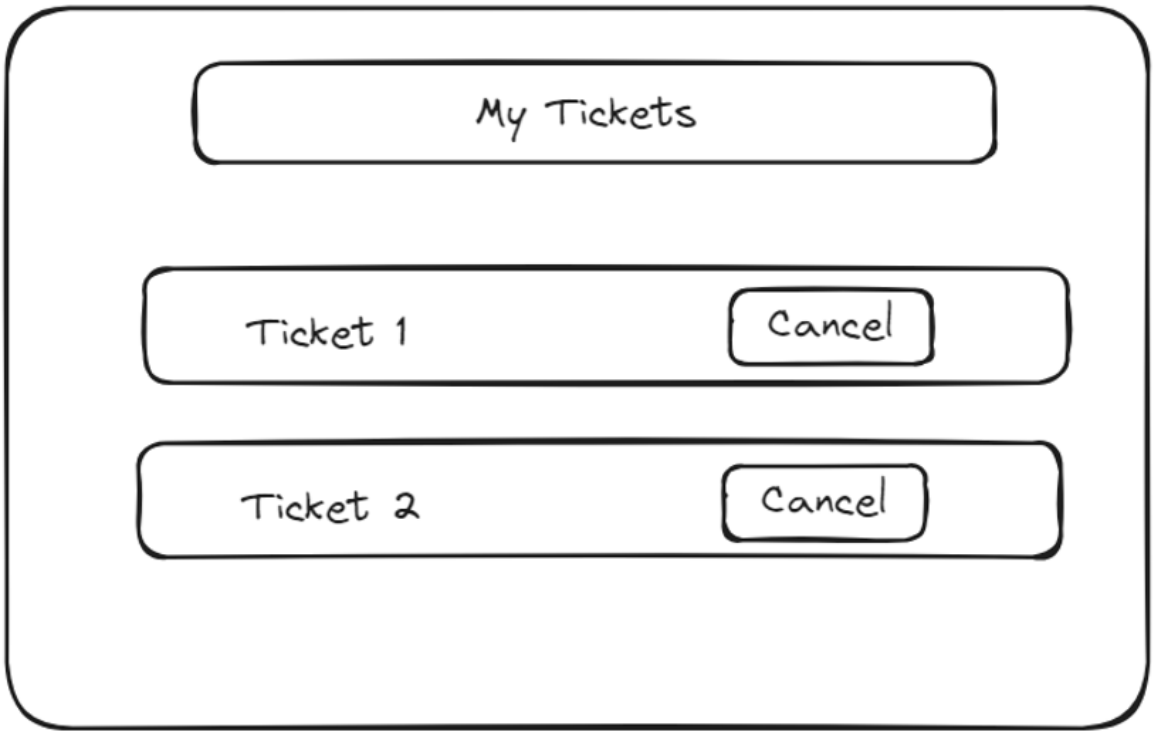
GET: /user/get_routes/{from}/{to}/{date}



POST: /user/book/{bus_id}
POST Body: {from,to,date,selected_seat}



GET: /user/show_ticket/{ticket_id}



GET: /user/show_all_tickets
POST: /user/delete_ticket/{ticket_id}