Table of Contents

System Design ***2***

Instructor Interface **2**

Features 2

Student Interface **2**

Features 3

Data Management and Booking Logic **4**

Database Schema 4

Key Components 5

Preventing Double Booking and Conflict Handling 5

Technical Walkthrough ***6***

Architectural Choices **6**

Database Choice 6

Frontend 6

Backend 6

Database Schema and Endpoints **6**

User Experience Considerations **7**

Conflict Handling **7**

Other considerations 7

Design diagrams - **8**

Database Schema Diagram: 8

Sequence Diagram for student booking a class: 9

Sequence Diagram for instructor updating a class: 10

Sequence Diagram for locking mechanism: 10

WireFraming - **11**

Basic UI design- **12**

System Design

Instructor Interface

Instructor would need an user friendly interface having following capabilities

1. User friendly google calendar like view having their availability
2. Provide different view options e.g. Day, Week, Month, Year etc
3. Allow instructors to view bookings and update availability
4. Configure reminders

Features

1. Set Available Time Slots
    a. Instructors can add their availability time slots (say with 60min increments) on specific days
    b. Provide option for setting recurring time slots (weekly, biweekly, monthly) and single-day availability.
    c. Block slots as available or unavailable. E.g. Instructor may be available on all weekdays from 10 to 5 except for 1 to 2 (lunch break).

2. Update Availability
    a. Option for instructor to update or delete availability time slot.
    b. Note - if student has booked time from that slot, then notify the instructor. Question – if instructor still wants to delete the timeslot, then should we reschedule or cancel the booking and notify student?

3. View Upcoming Bookings
    a. Based on the view instructor can see details of the bookings.
        i. In weekly view due to space constraints we may can show only student name and timeslot.
        ii. In Day view, we can show more details like Student name, if the booking is recurring. Notes from the booking etc.
        iii. In today's schedule view we can show full details for appointments in rest of the day

Student Interface

Student needs an user friendly interface having following capabilities

1. User friendly google calendar like view having instructor's availability and another view for own bookings
2. Provide different view options e.g. Day, Week, Month, Year etc
3. Allow students to book timeslot, update/cancel bookings
4. Configure reminders

Features

1. View available time slots
    a. Students will see the instructor's available slots in calendar format in real-time. Booked slots will be grayed out without giving any additional information.
    b. It can be in different views like Day, Week, Month, Year etc
2. Book a time slot
    a. Students can click on an available time slot to select it. After providing required details (name, email, phone, duration, any notes for the instructor etc) user can confirm the booking
    b. User will get booking confirmation on email or SMS
3. View and Cancel Upcoming Bookings
    a. Students can see their upcoming bookings
    b. Students can cancel or reschedule, depending on the instructor's availability.

Data Management and Booking Logic

Database Schema

Using Firebase as the backend, we'll structure our data to ensure efficient retrieval and updates, avoiding data duplication and conflict. Here's an outline of the data schema:


```
{
  "students": {
    "studentID_123": {
      "name": "First Last",
      "email": "user@email.com",
      "password": “xxxxxx”,
      “education level": “Under Graduate"
    }
  },

  "instructors": {
    "instructorID_456": {
      "name": "Prof ABC”,
      "email": "user@email.com",
      "password": “xxxxxx”,
      "expertise": "Python, Artificial Intelligence, Cyber Security",
      "availability": {
        "2024-11-14": {
          "slots": [
            { "startTime": "10:00", "endTime": "11:00", "available": true },
            { "startTime": "11:00", "endTime": "12:00", "available": true }
          ]
        },
        "2024-11-15": {
          "slots": [
            { "startTime": "14:00", "endTime": "15:00", "available": true }
          ]
        }
      }
    }
  },


  "bookings": {
    "bookingID_789": {
      "studentID": "userID_123",
      "instructorID": "instructorID_456",
      "startTime": “2024-11-15T10:00:00",
     “date”: “2024-11-15”
      "status": "confirmed",  // or "cancelled"
```

```
    }
  }
}
```

Key Components

1. Users - stores details of students and instructors.
2. Instructors – availability data i.e. slots for each day and whether they are available
3. Bookings - stores the confirmed bookings, including the student, instructor and time slot.

Preventing Double Booking and Conflict Handling

To prevent double booking we will implement

1. Locking mechanism - When a student selects a time slot, that slot should be locked (marked as unavailable) for a short time while the student completes the booking. This prevents another student from booking the same slot while the first student is still in the process of booking.
   Note - This locking time can be configurable at a global level with default value 5 minute.
2. Real time updates - When an instructor or student modifies availability or bookings, Firebase's real time database can automatically notify all other users of the change, keeping them up-to-date on available slots.
3. Atomic Transactions - When updating availability or booking a slot, Firebase can ensure that the update is done atomically (i.e. it won’t allow two concurrent operations on the same slot at the same time).

Technical Walkthrough

Architectural Choices

Database Choice

Firebase Realtime Database offers easy integration with frontend frameworks like React, scalability with automatic synchronization, and built-in conflict handling for concurrent operations. It's a good fit for managing dynamic data like bookings and availability.

Frontend

React.js is a good choice for the frontend, as it provides a component-based architecture that can handle dynamic UIs, such as calendars and real-time updates for availability. React's state management (using Context API or libraries like Redux) will help manage the booking state across the app.

Backend

Node.js is well-suited for handling asynchronous requests, and with libraries like Express.js, we can quickly build the necessary API endpoints. Firebase Admin SDK will be used to interface with Firebase to perform CRUD operations.

Database Schema and Endpoints

Database schema is given in the Database Schema section above.

Here are the required API endpoints:

1. GET /availability/:instructorID - Returns a list of available time slots for a given instructor.
2. POST /book - Creates a new booking for a student. This will check for availability before confirming the booking.
3. PUT /update-availability - Allows an instructor to modify or delete availability.
4. GET /bookings/:studentID - Retrieves a list of all bookings for a student.
5. PUT /cancel-booking/:bookingID - Allows a student to cancel a booking.

User Experience Considerations

1. Easy Navigation - Provide clear UI elements for instructors and students to manage their schedules. For instance, clickable time slots in a calendar view, with color coding for available and booked slots.
2. Notifications - Send real-time push notifications (via Firebase Cloud Messaging) to both students and instructors when a booking is confirmed, cancelled, or rescheduled. Include reminders for upcoming sessions.
3. Smart Form Validation - Ensure the booking form is simple and intuitive with client-side validation (e.g., ensuring the student selects a valid time, or that the student doesn't double-book).

Conflict Handling

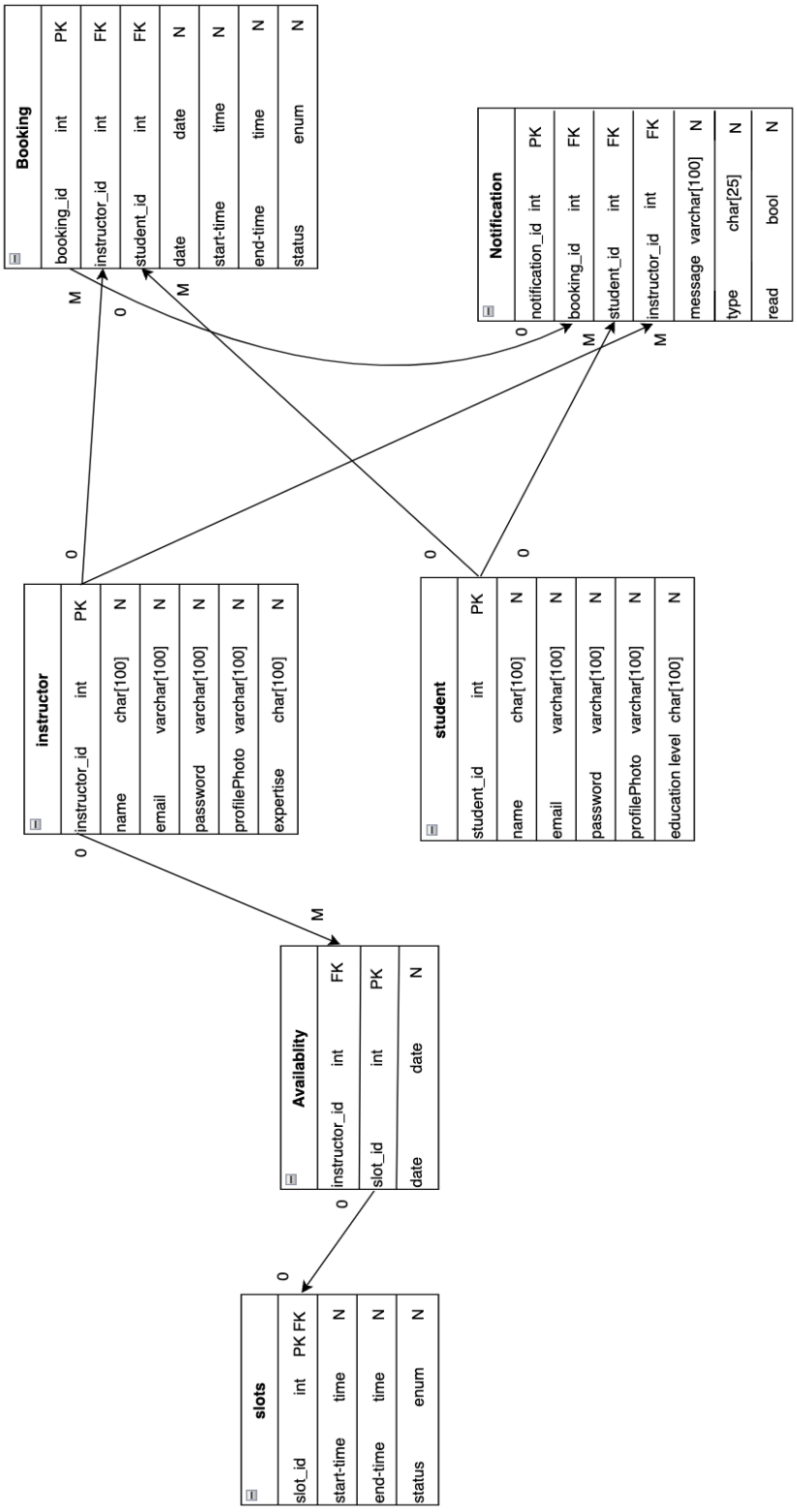
1. Real-Time Updates - Use Firebase's real-time capabilities to propagate availability changes instantly. If an instructor adds or removes time slots, all students and instructors are notified in real-time.
2. Retry Queues - If a student tries to book a slot that is no longer available (because another student beat them to it), provide them with an option to select a nearby time slot or join a waitlist.

Other considerations

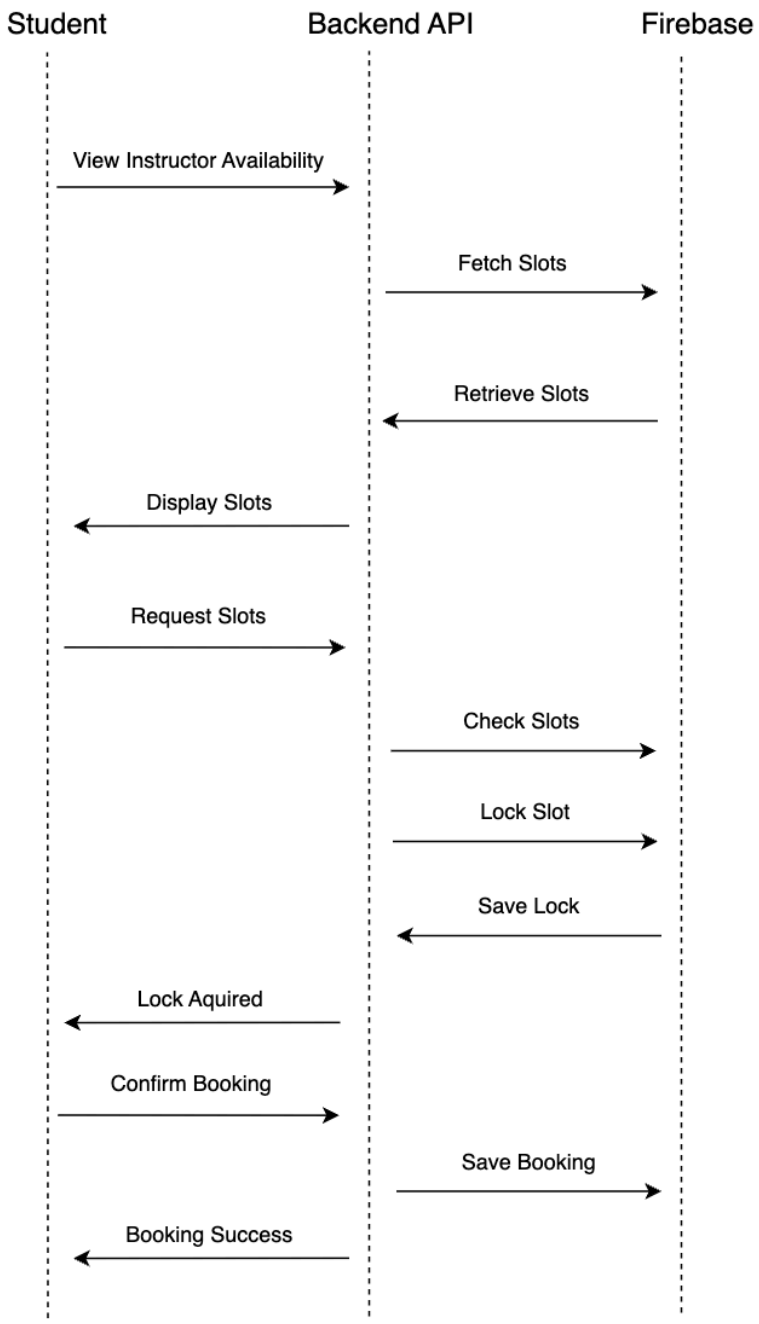
1. Spam protection
    a. Ensure user is registered at multiple
    b. Put constraints on number of active bookings one student can have
    c. Capta during login and booking
2. Data security and Privacy
    a. A student shouldn't be able to see booking details for instructor other than his/her own booking.
    b. A student shouldn't be able to see bookings of other students.
3. Logging
    a. Appropriate logs should be added with required levels e.g. info, debug, warning, error.
    b. Logs are maintained at centralized server and have purging
4. Auditing
    a. User registration/deletion
    b. Authentication events like login success/failures and logout
    c. Booking creation, update, cancellation and history
    d. Etc

Design diagrams -

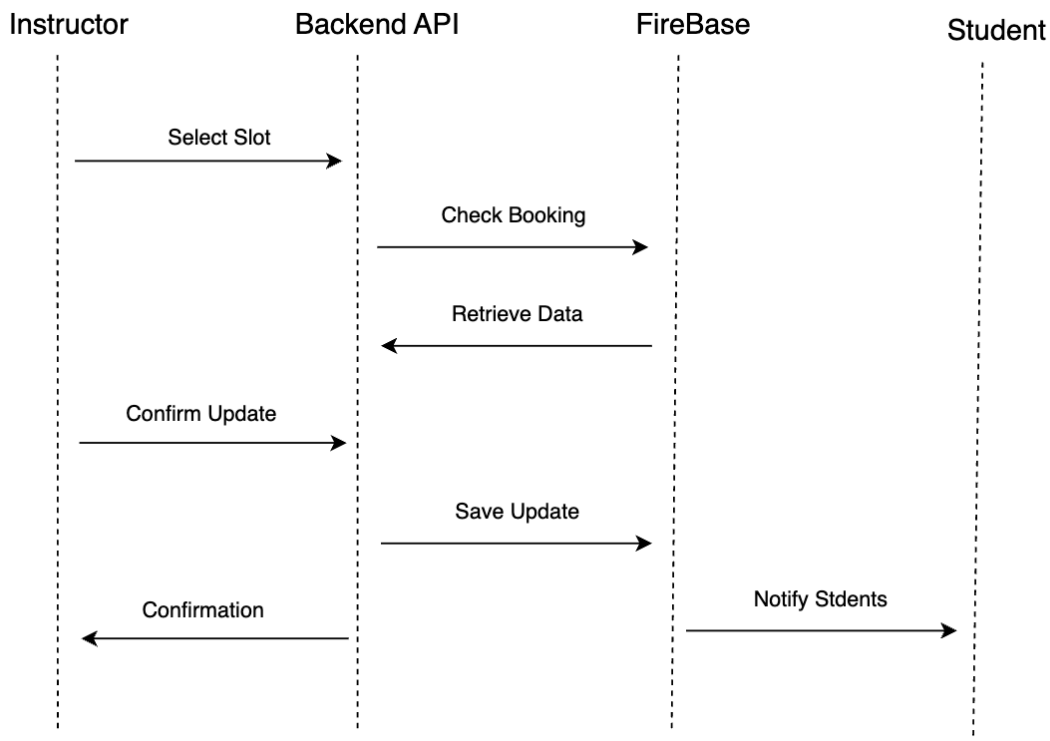
Database Schema Diagram:



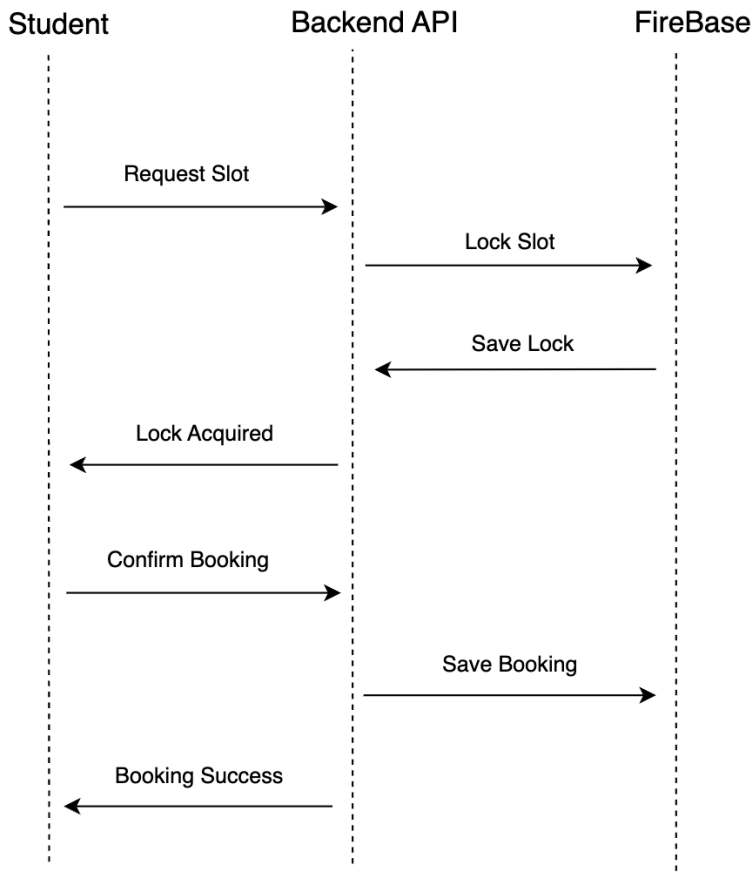
Sequence Diagram for student booking a class:



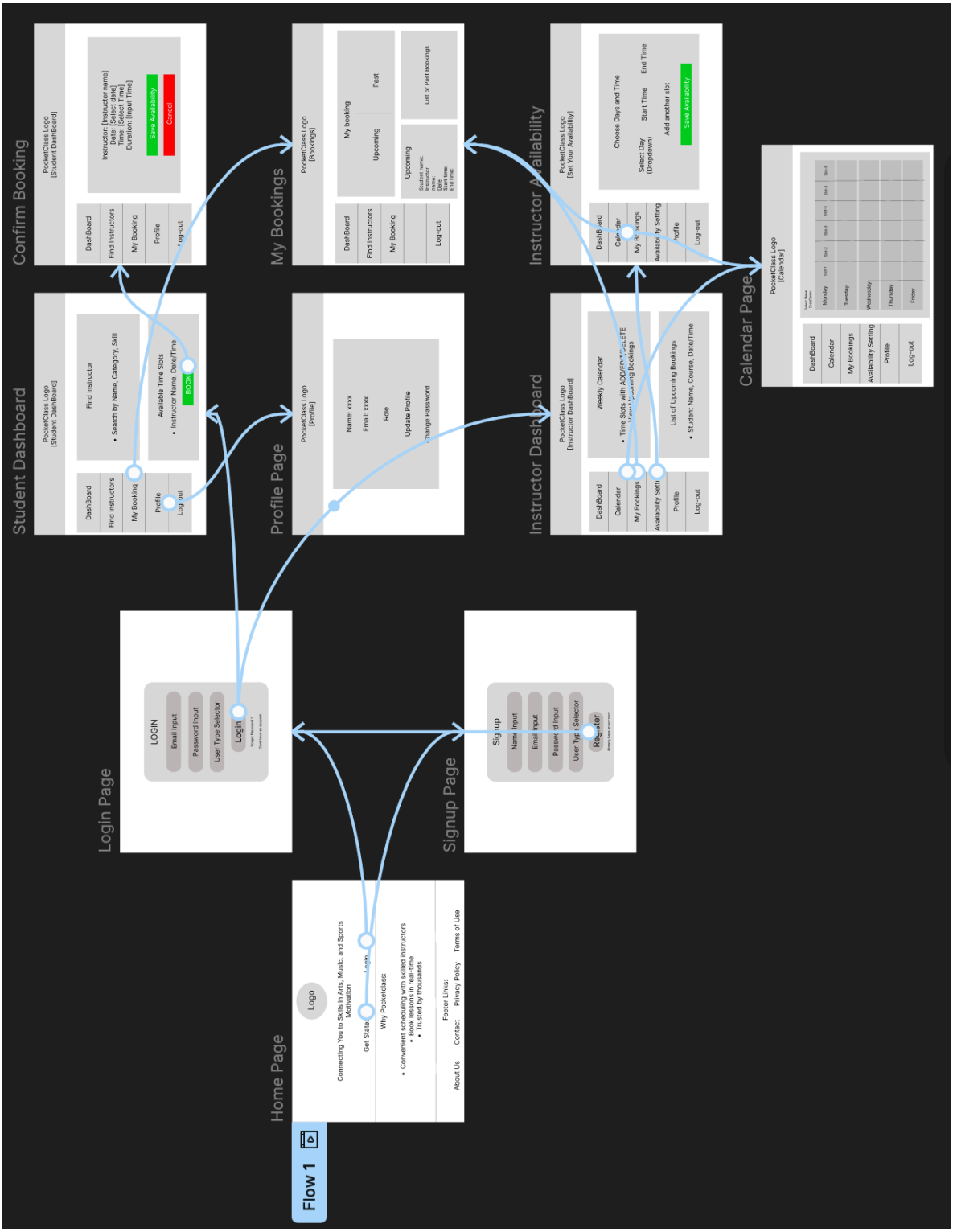
Sequence Diagram for instructor updating a class:



Sequence Diagram for locking mechanism:

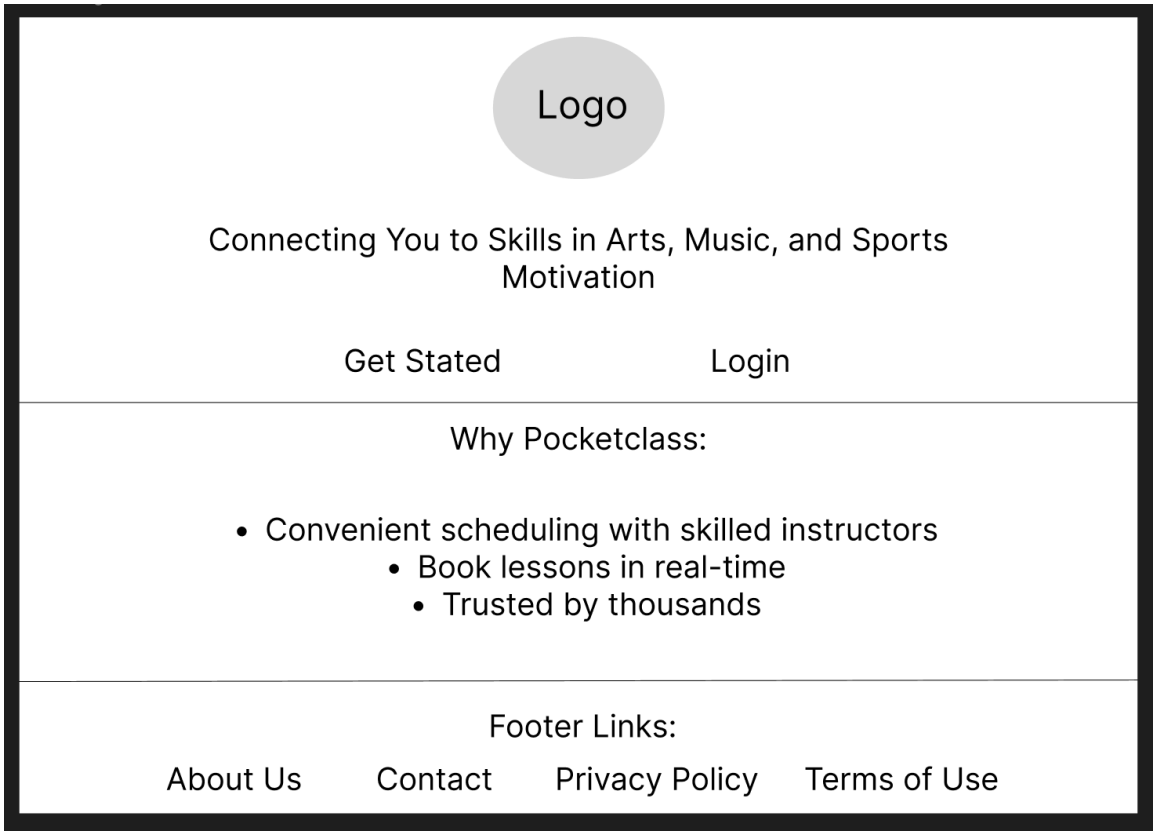


WireFraming -

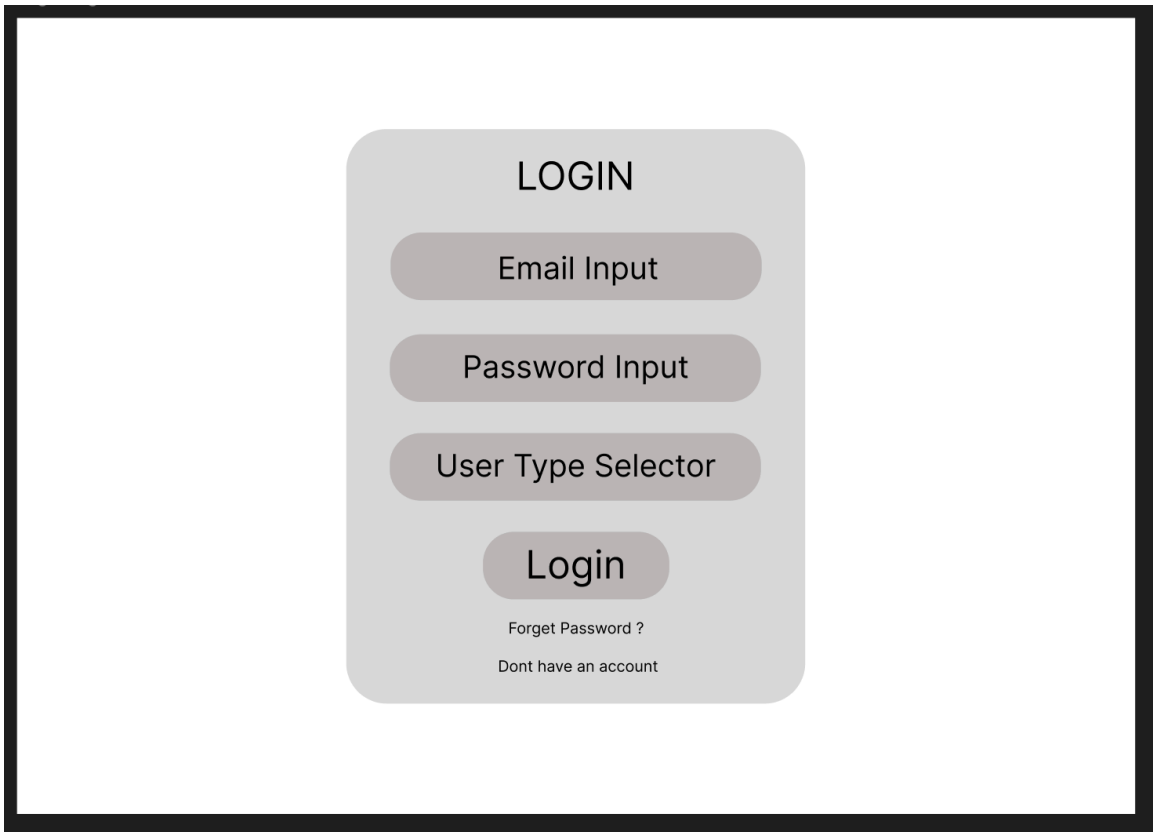


Basic UI design-

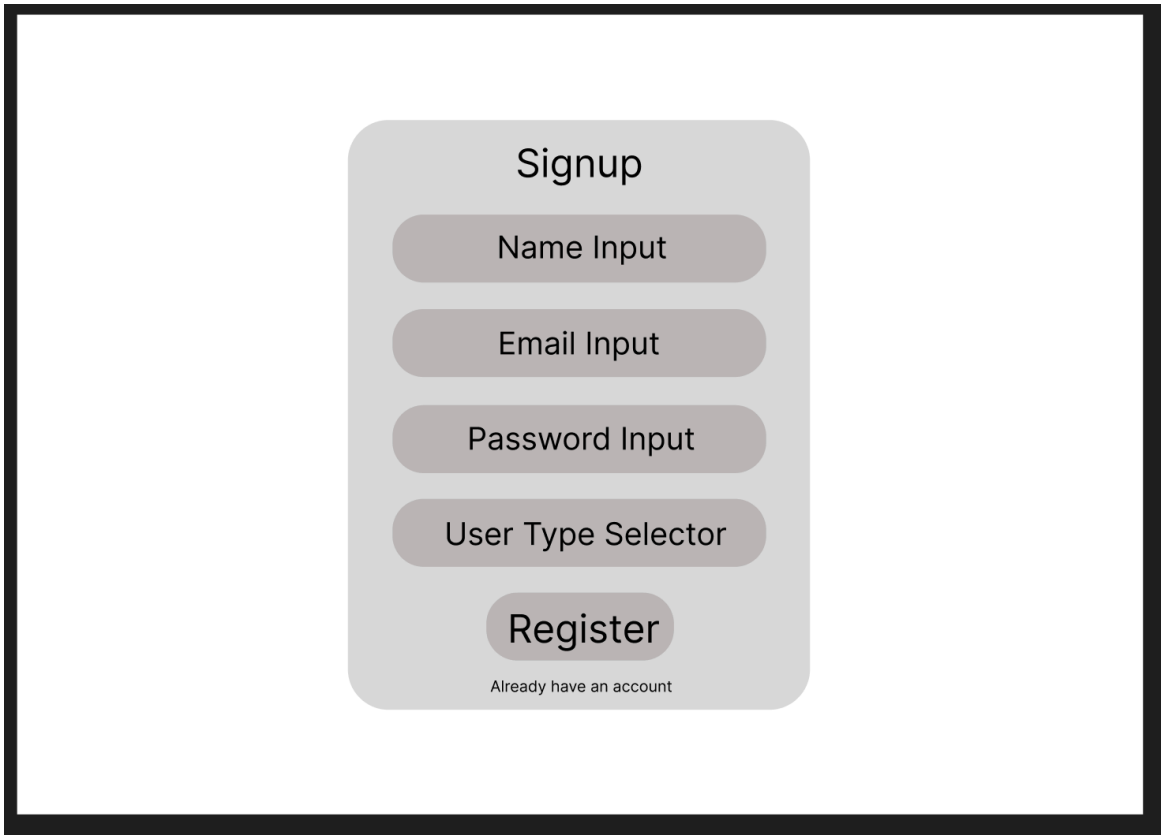
Home Page:



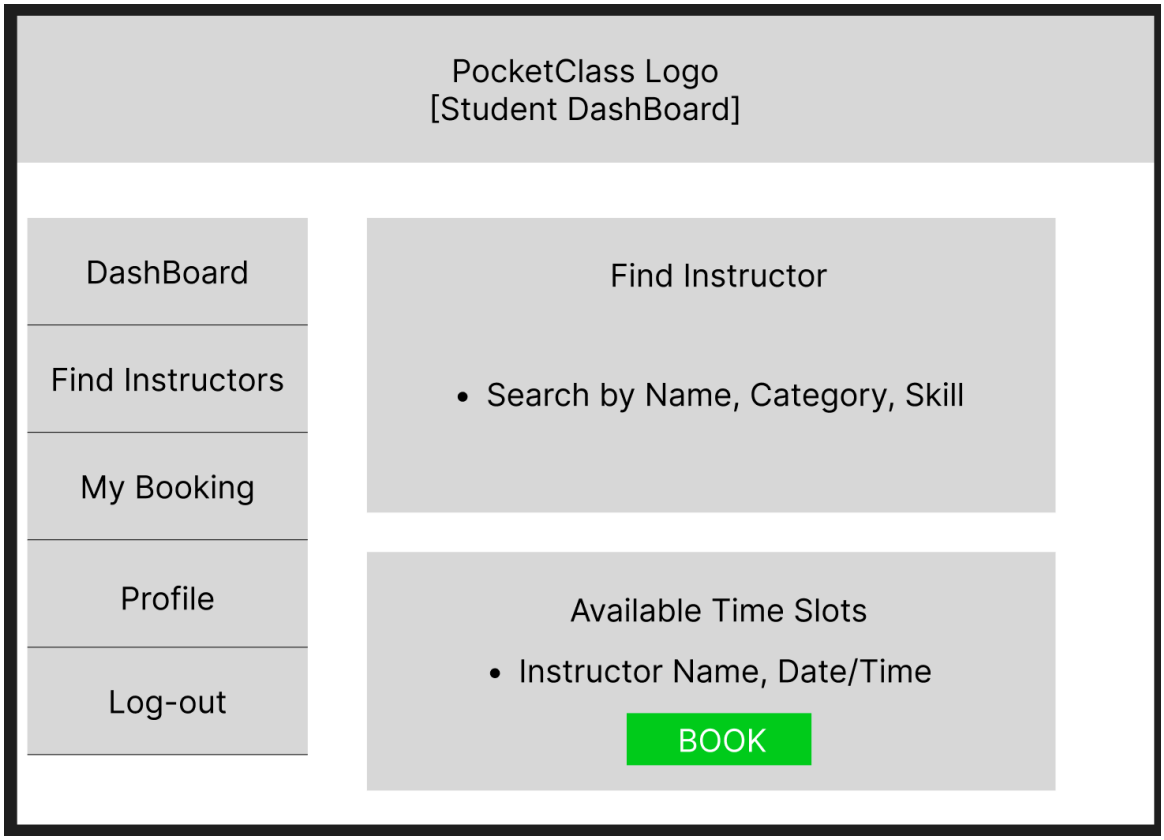
Login Page:



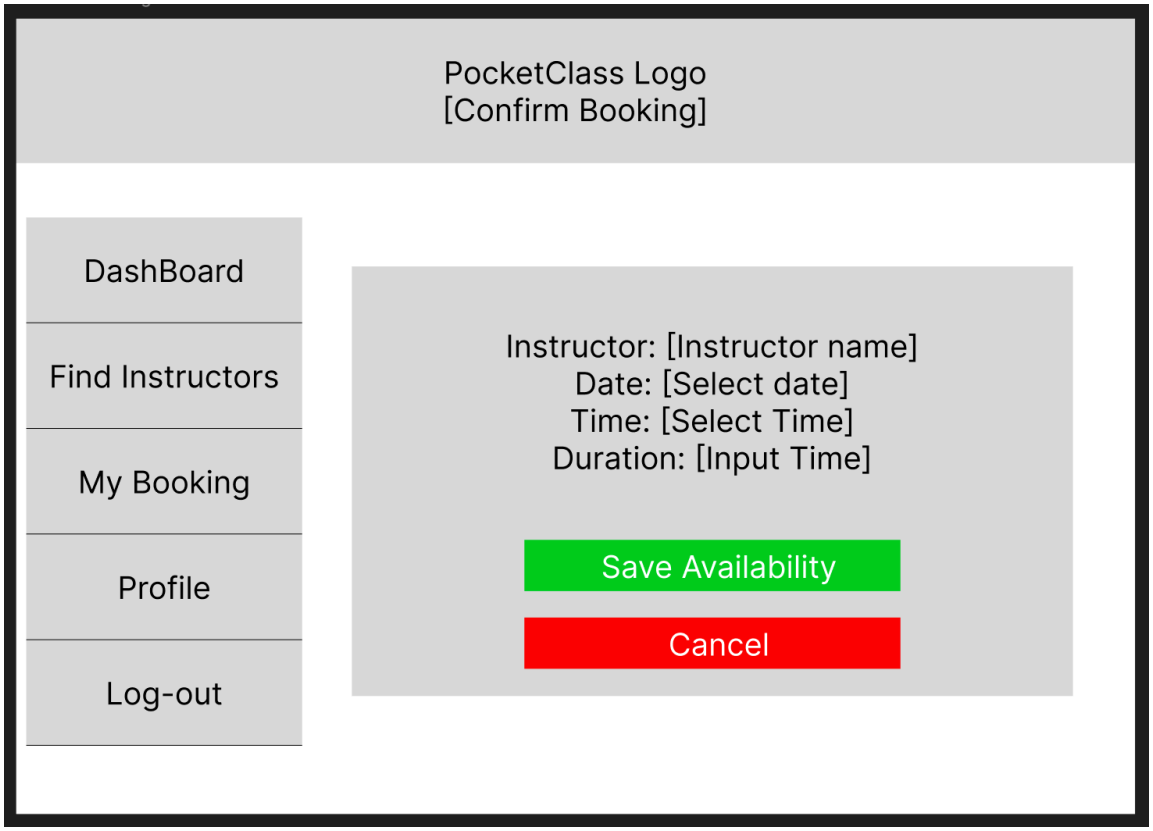
SignUp Page:



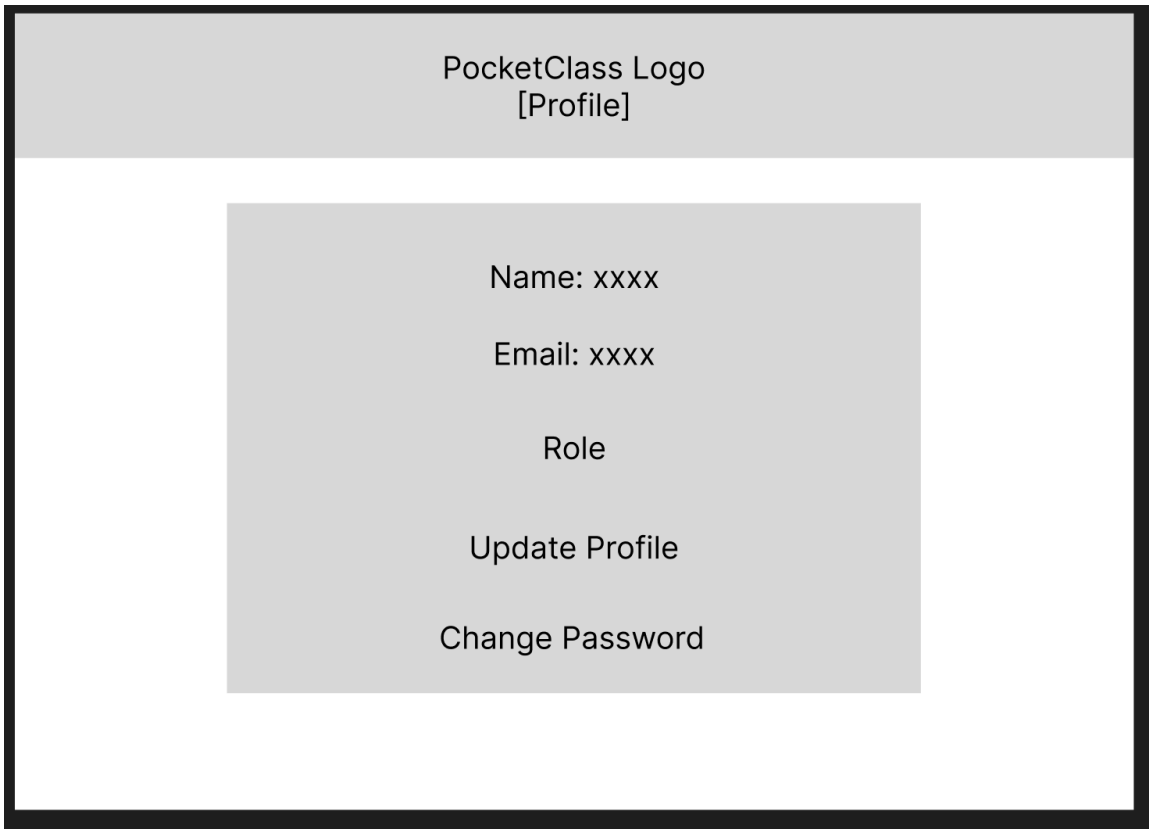
Student Dashboard:



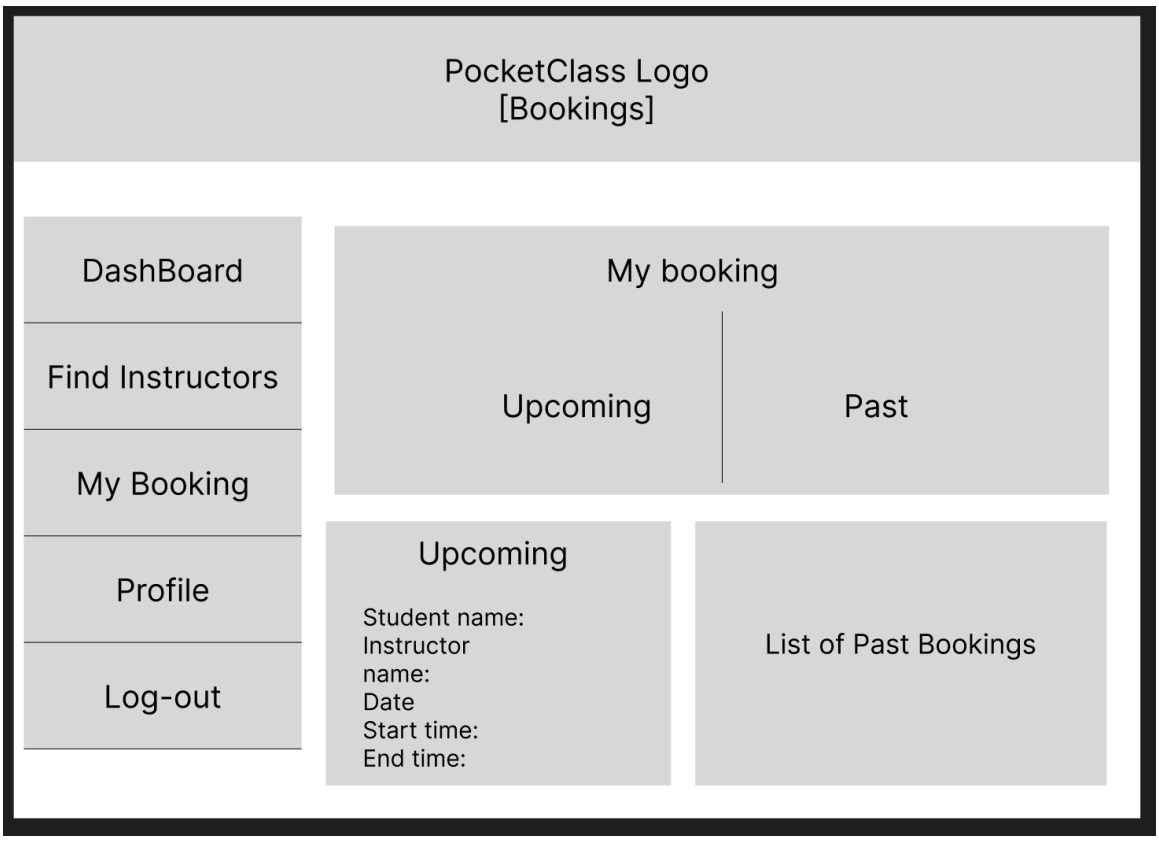
Confirm Booking:



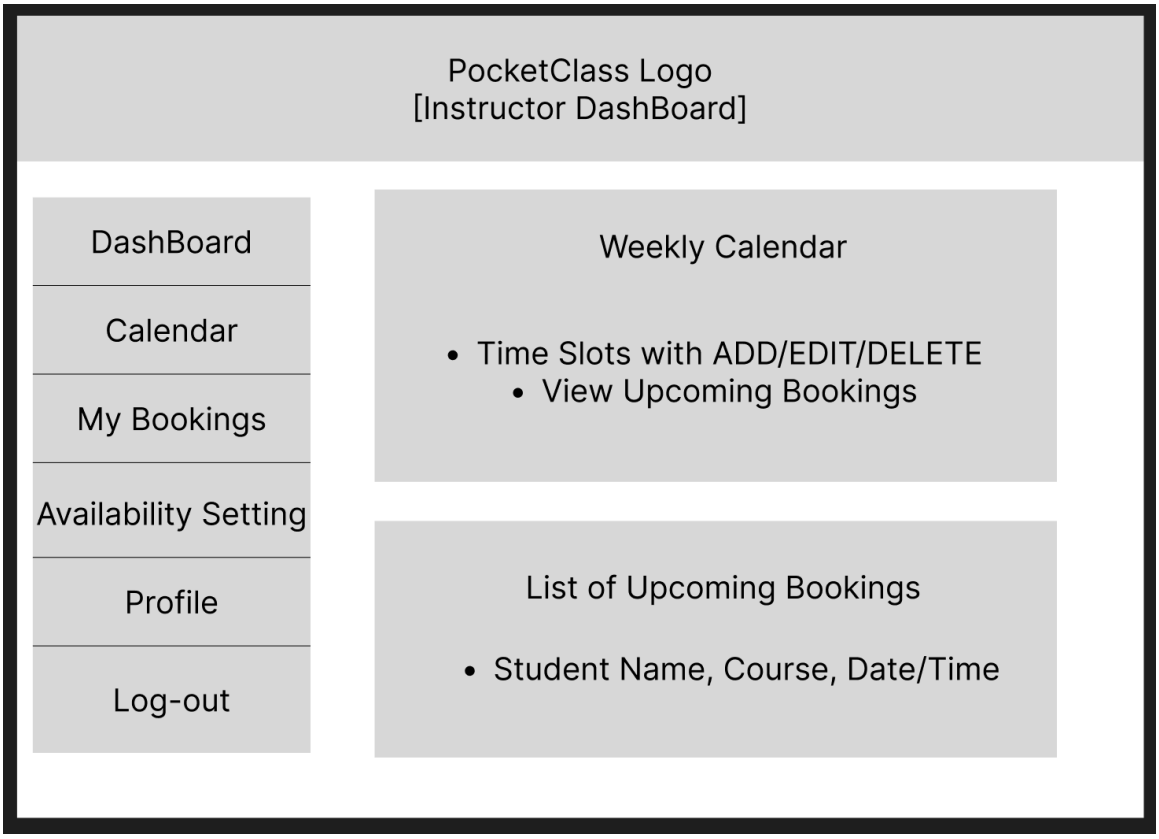
Profile Page:



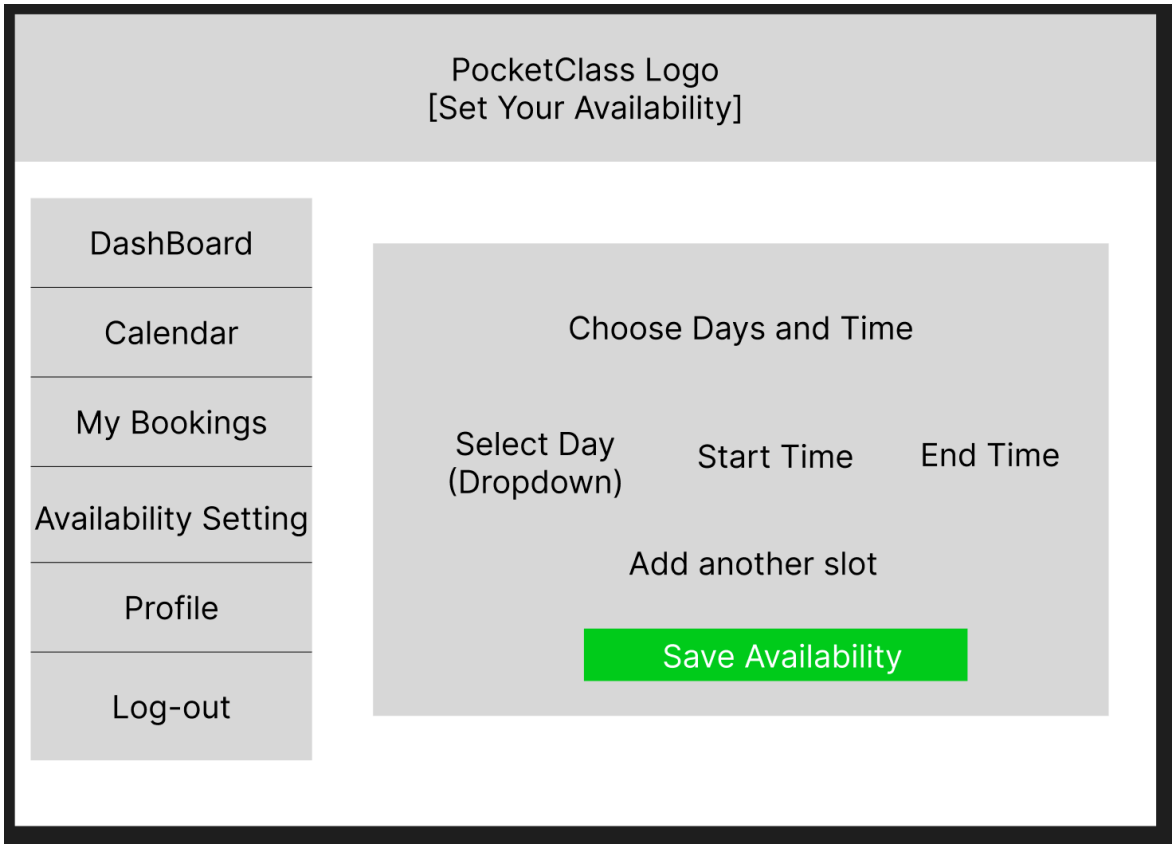
My Bookings Page:



Instructor Dashboard:



Instructor Availability Setting:



Instructor Calendar: