**Team Name - Dal Chawal**   
1. Atharva Kalekar - <https://github.com/AtharvaKalekar/Campus-Sports-Ground-Booking-System-.git>

2. Samarth Devadiga - <https://github.com/SamarthD1/Campus_Sports_Ground_Booking_System-.git>

3. Ameya Sagwekar - <https://github.com/Ameya48/Campus-Sports-Ground-Booking-System-.git>

4. Rizwan Salmani - <https://github.com/Rizwan2611/Rizwan-Salmani>

5. Sarang Gole - <https://github.com/Saranggole9106/112_SARANG_GOLE>

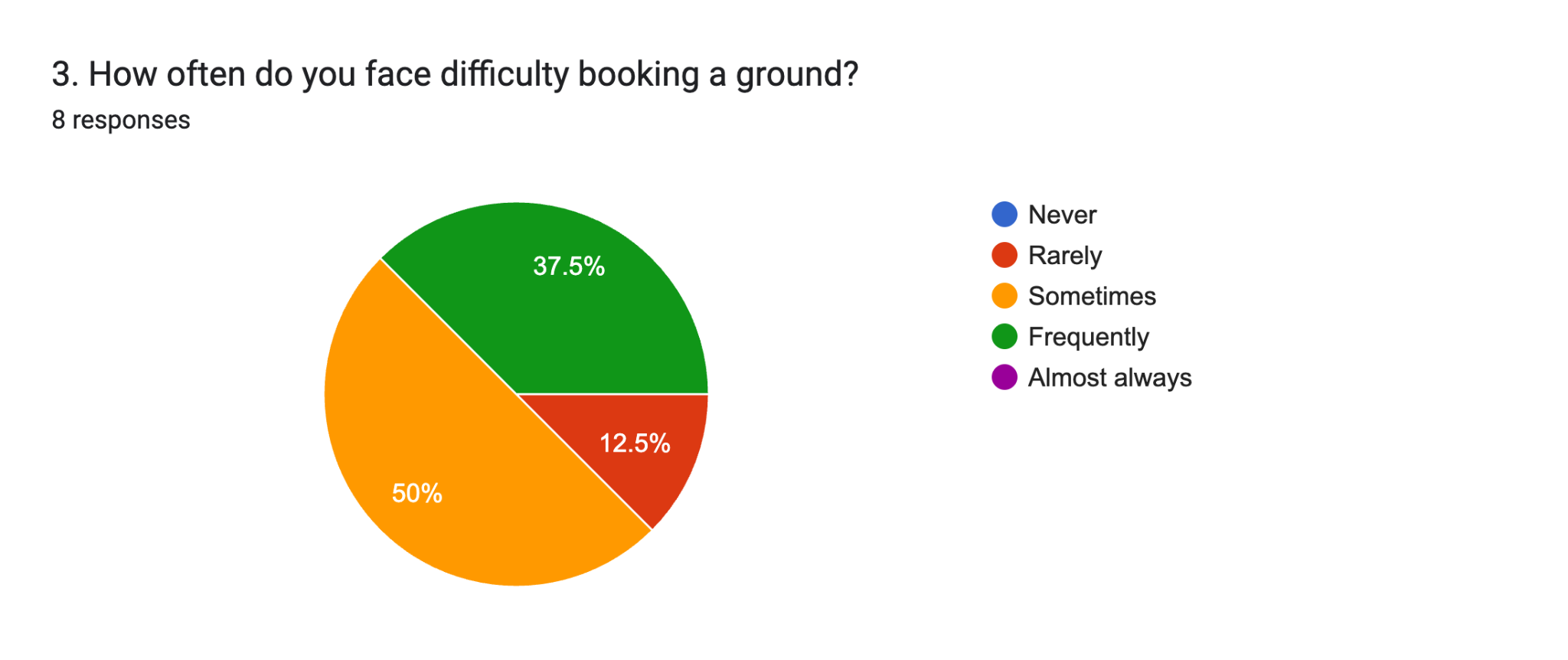
# **Campus Sports Ground Booking System Documentation**

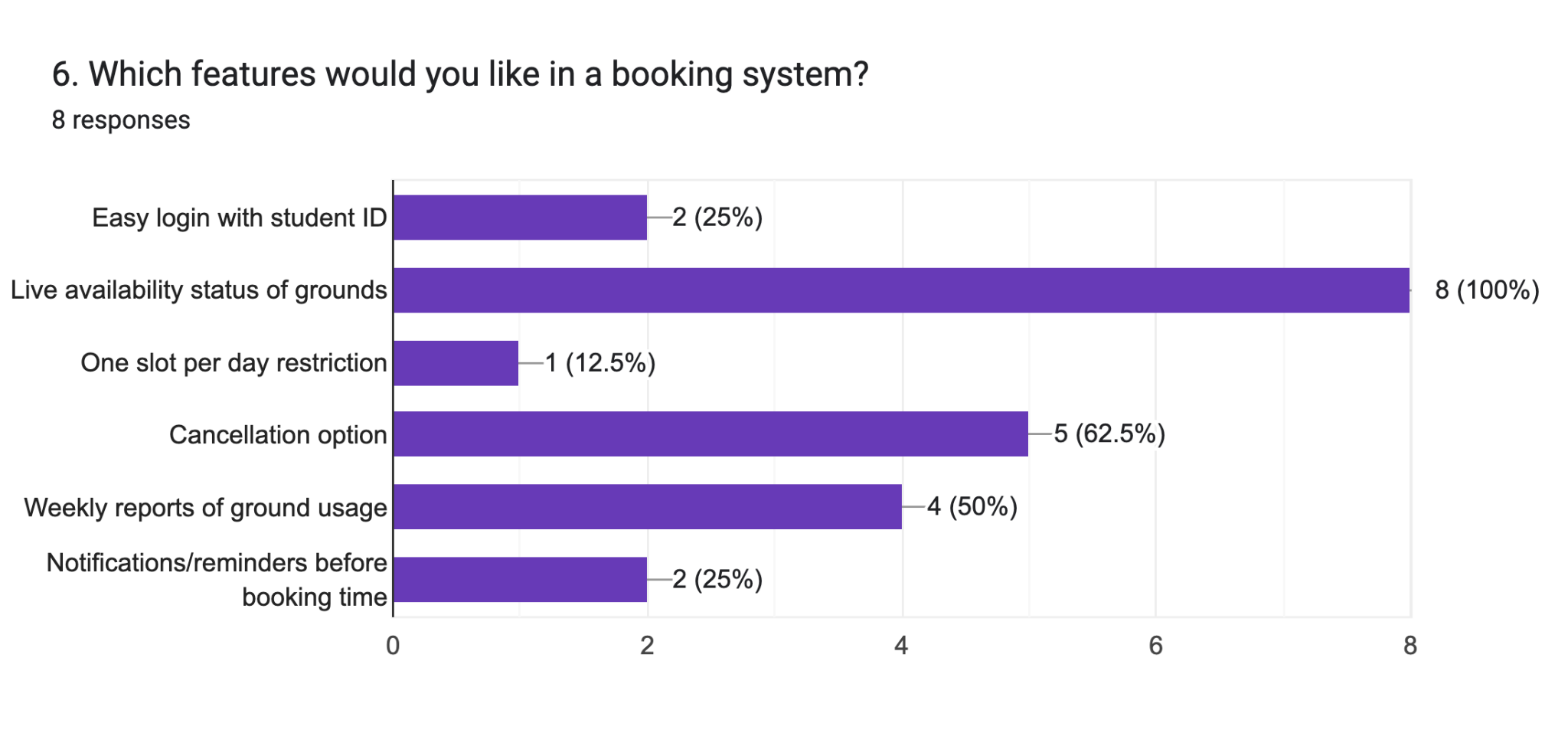
## **Introduction**

This documentation details the Campus Sports Ground Booking System project: its requirements, technology stack, backend and frontend design, deployment, research plans, and reflections.

### **Phase 1: Survey**

* **Goal**: Understand student & staff problems with booking.
* **Example Questions**:  
  + How do you currently book the ground? (manual register / message / other)
  + What’s your biggest issue? (double booking / no availability / fairness / etc.)
  + What problems do you face while trying to book or access a ground?
  + Would you prefer an online system?
  + Which features would you like in a booking system?
* **Expected Results**:  
  + Most students face unfair multiple bookings.
  + Everyone wants a transparent online portal.





### **Phase 2: Case Study**

* **Case Chosen**: Sports Ground Booking.
* **Interviews**:  
  + **Students** → “I try to play football but seniors book multiple slots.”
  + **Staff** → “We write names in registers; sometimes two students claim the same slot.”

## **1. Current Workflow**

At present, the booking of campus sports grounds is handled in a **manual and unstructured way**:

● **Manual Registers:** Students approach the Sports Office and write their names in a physical register for a time slot.

● **WhatsApp/Verbal Requests:** In some cases, bookings are done informally via WhatsApp groups or direct messages to the sports committee members.

● **No Centralized Record:** Records are either paper-based or scattered across different communication channels, making it difficult to track usage history.

## 

## **2. Identified Pain Points**

Through observation and discussion, the following issues were noted:

1. **Overlap Bookings:**Multiple students/groups end up booking the same slot for the same ground due to lack of a real-time system.
2. **No Transparency:**Students cannot see who has already booked a slot, leading to confusion and disputes.
3. **Misuse of Facility:**Some students book multiple slots in a single day, leaving less opportunity for others.
4. **Administrative Burden:**Sports committee staff spend considerable time manually updating and verifying registers.
5. **Lack of Notifications:**Students are not informed if their booking is confirmed, rejected, or cancelled.

## **3. Stakeholder Interviews**

### **A. Student Users (End-Users)**

● **Feedback 1 (Student A):***“I often go to the football ground only to find out someone else has also claimed the same slot. It feels unfair and confusing.”*

● **Feedback 2 (Student B):***“It would be great if there was an app where we could see available slots in real-time and just book them.”*

**Key Needs Identified:**

● Fair access (limit to one-slot-per-day).

● Transparency (see available slots before booking).

● Convenience (book online without visiting the office).

### **B. Sports Committee Staff (Administrators)**

● **Feedback 1 (Staff Member 1):***“We maintain a register, but students often argue if two names clash in the same time slot. It’s time-consuming for us to resolve.”*

● **Feedback 2 (Staff Member 2):***“Sometimes students book multiple slots in a single day. We have no effective way to stop this manually.”*

**Key Needs Identified:**

● Automation of booking validation.

● Reduced disputes.

● Easy monitoring through a dashboard.

## **4. Conclusion & Insights**

● **Students** want **fairness, transparency, and convenience.**

● **Staff** want **automation, reduced workload, and proper monitoring.**

● The **main problem** is the absence of a centralized, rule-based system.

● **Solution Direction:** A MongoDB-backed digital booking system enforcing **“one-slot-per-student-per-day”** with real-time updates and automated restrictions.

## **System Architecture**

Architecture Overview:

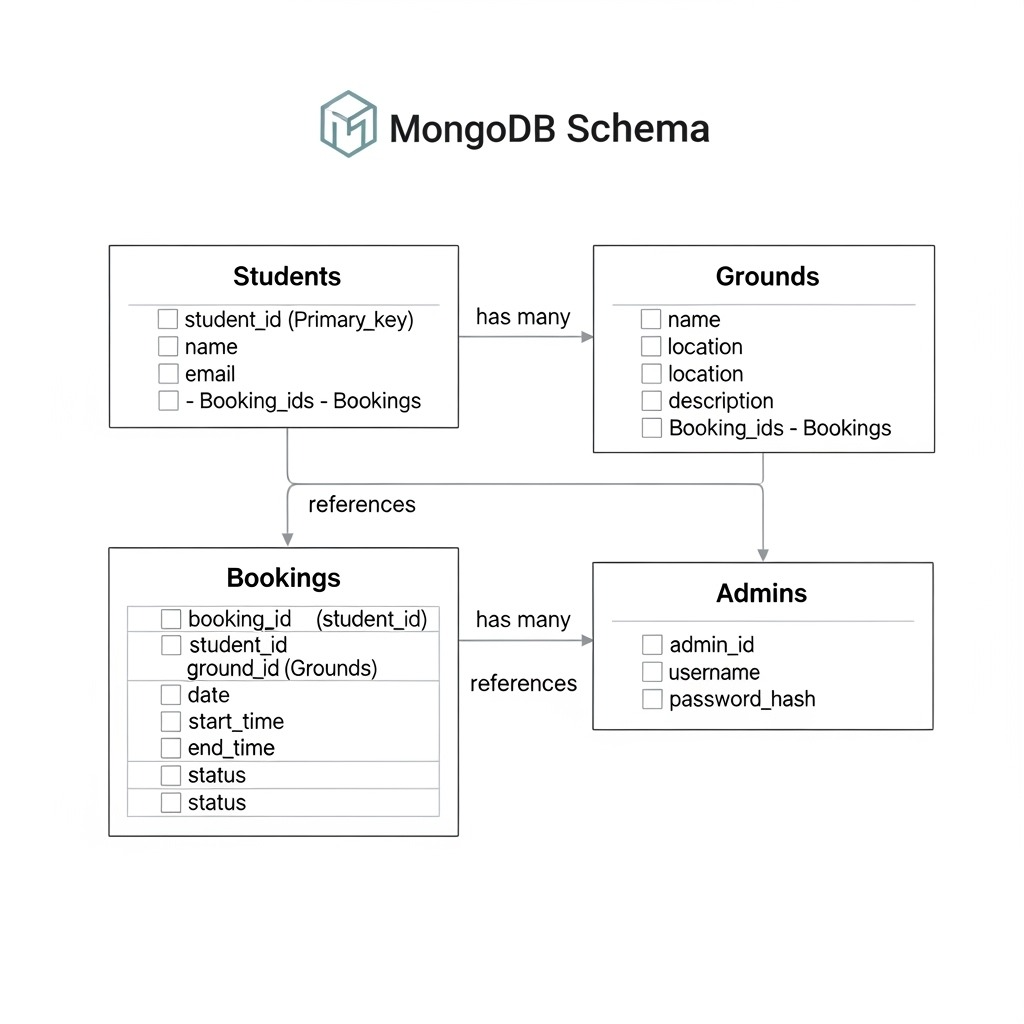
* Backend: Node.js + Express with MongoDB for persistence.
* Frontend: React (or HTML/CSS/JavaScript) with Firebase Authentication and Firebase Hosting.

### 

### 

### **Phase 3: MongoDB Schema**

* **Database**: MongoDB (flexible JSON-like).



**Students**

* \_id, name, email, rollNo, department  
  Example:

json

{

"\_id": ObjectId("507f1f77bcf86cd799439011"),

"name": "Atharva Kalekar",

"email": "atharva@123.gmail.com”,

"rollNo": "CS2021001",

"department": "Computer Science",

"firebaseUID": "firebase\_auth\_uid\_string",

"createdAt": ISODate("2024-01-15T10:30:00Z"),

"updatedAt": ISODate("2024-01-15T10:30:00Z")

}

**Grounds**

* \_id, name, location, slots[]  
  Example:

json

{

"\_id": ObjectId("507f1f77bcf86cd799439012"),

"name": "Football Ground",

"location": "North Campus Sports Complex",

"slots": [

"06:00-07:00",

"07:00-08:00",

"08:00-09:00",

"16:00-17:00",

"17:00-18:00",

"18:00-19:00",

"19:00-20:00"

],

"description": "Full-size football field with grass surface",

"capacity": 22,

"isActive": true,

"createdAt": ISODate("2024-01-15T10:30:00Z"),

"updatedAt": ISODate("2024-01-15T10:30:00Z")

}

**Bookings**

* \_id, studentId, groundId, date, slot, status  
  Example:

json

{

"\_id": ObjectId("507f1f77bcf86cd799439013"),

"studentId": "firebase\_auth\_uid\_string",

"groundId": ObjectId("507f1f77bcf86cd799439012"),

"date": ISODate("2024-01-20T00:00:00Z"),

"slot": "16:00-17:00",

"status": "confirmed",

"bookingReference": "BK1705312345ABC",

"createdAt": ISODate("2024-01-15T10:30:00Z"),

"updatedAt": ISODate("2024-01-15T10:30:00Z")

}

**Phase 4**

## Functional Requirements

1. User Authentication  
   Students sign up and log in using Firebase Authentication (via email/password or Google login).
   * Each logged-in student is mapped to a record in the Students collection in MongoDB.
2. Booking System  
   Students select a ground, date, and time slot.
   * Before confirming, the system checks MongoDB to ensure:
     + The slot is available.
     + The student hasn’t already booked for that date (restriction rule).
3. Real-Time Updates  
   * Bookings are synced using Firebase Realtime Database or Firestore.
   * If a slot gets filled, other students immediately see it as unavailable.
4. Dashboard  
   * Students → can view their upcoming bookings.
   * Admins → can view all bookings for all grounds and cancel/approve if needed.
5. Deployment  
   * Frontend + prototype deployed on Firebase Hosting.
   * MongoDB stores booking data permanently.

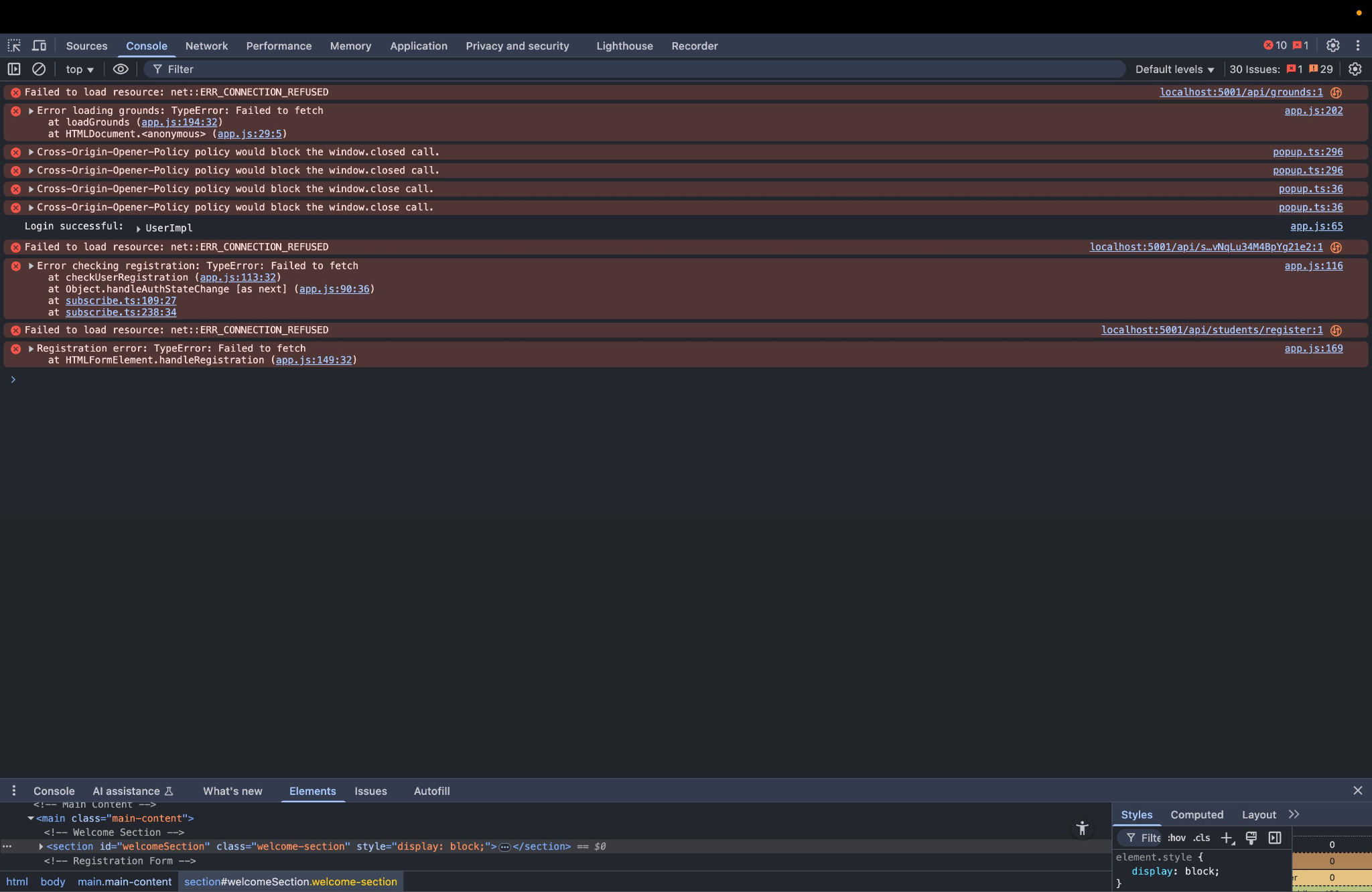
## **Roles**

* Authentication - Rizwan
* Database - Sarang
* JS Code - Ameya, Atharva, Samarth
* Schema - Atharva & Ameya
* Survey - Atharva
* Documentation - Ameya
* Case Study - Samarth & Rizwan

## 

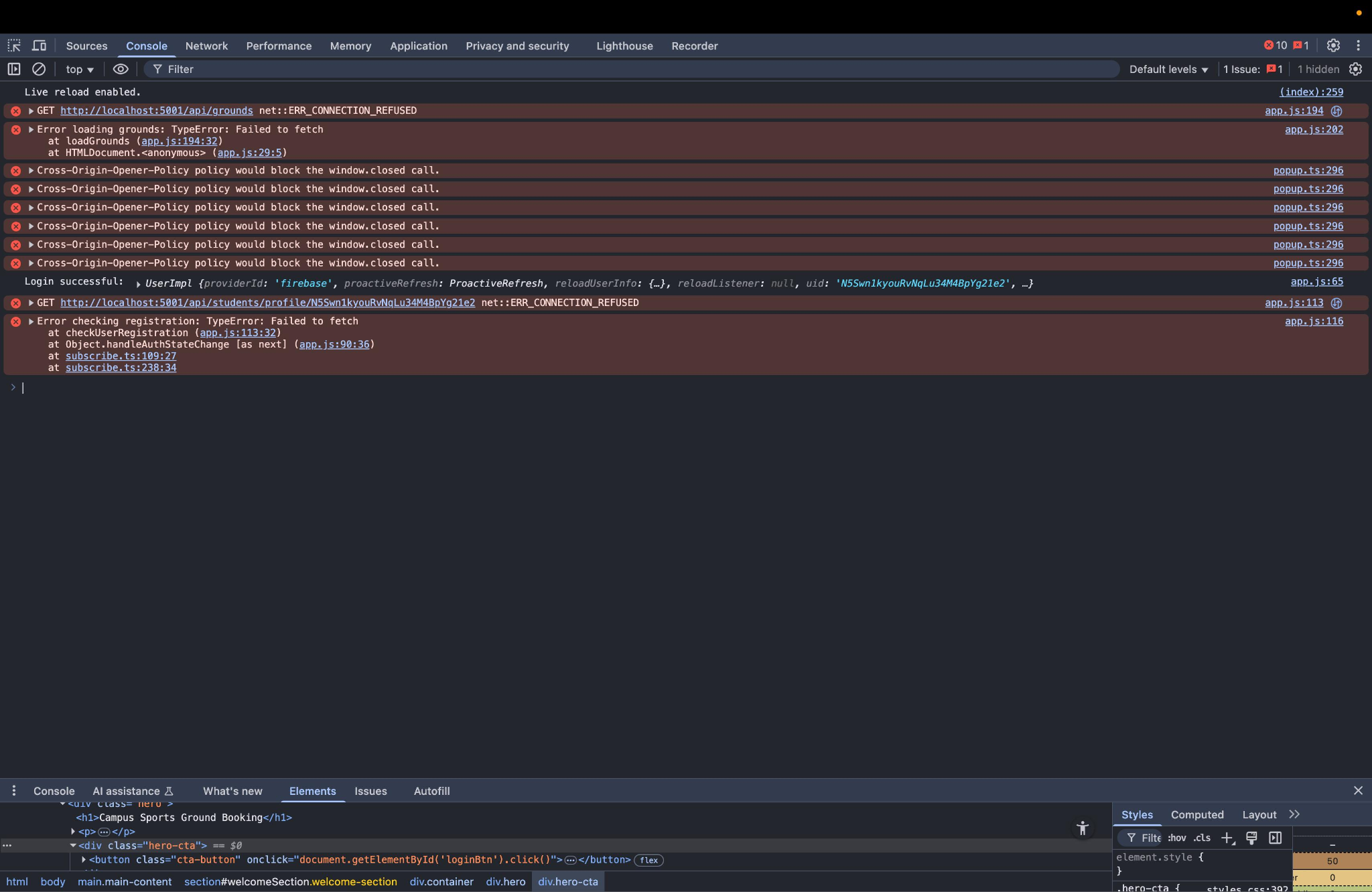
## 

## **Challenges / ERRORS faced**



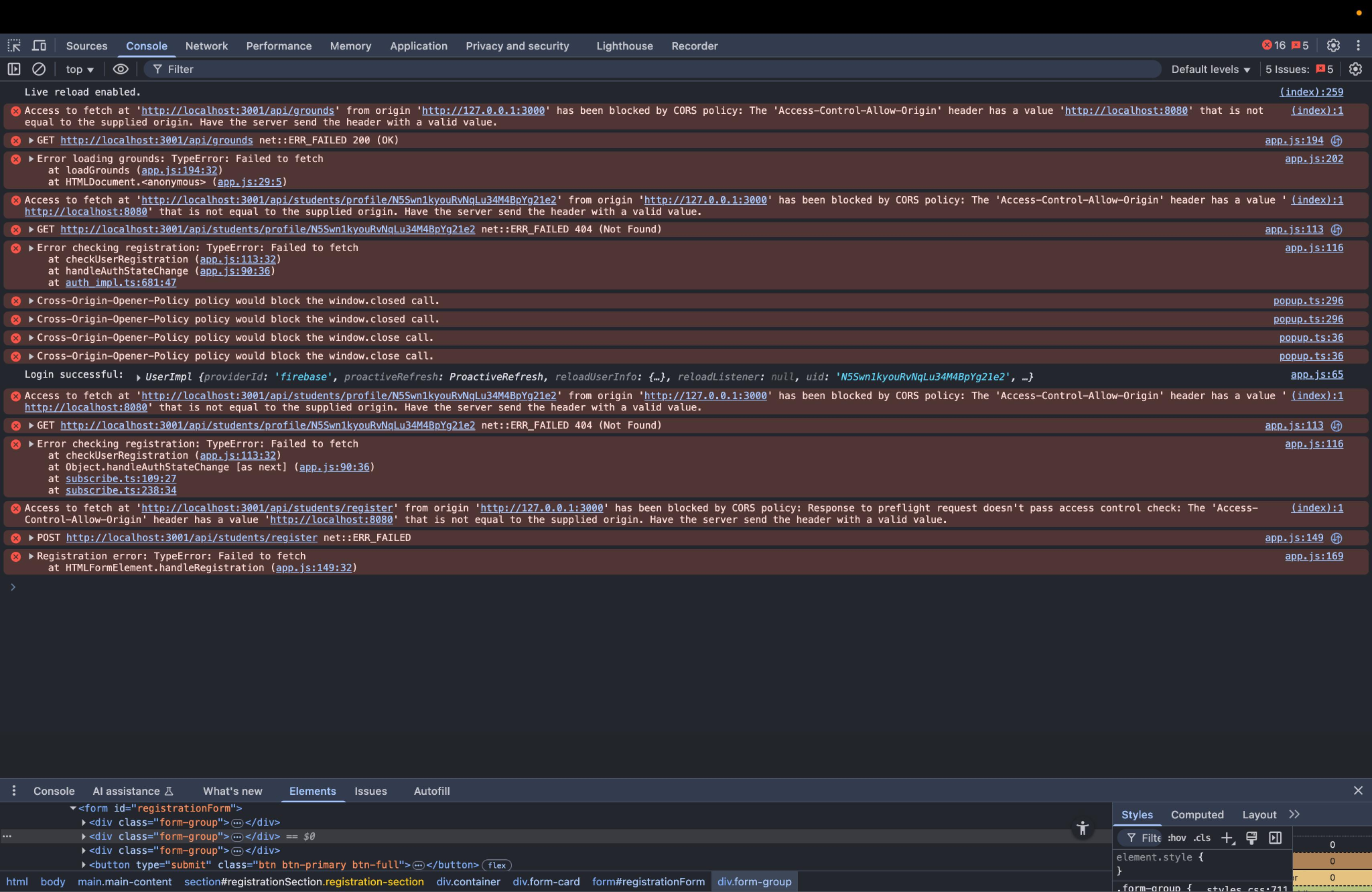
### **Errors**

1. **Failed to load resource: net::ERR\_CONNECTION\_REFUSED**
2. **TypeError: Failed to fetch**
3. **Cross-Origin-Opener-Policy warnings**



### **Error**

1. **GET http://localhost:5001/api/grounds net::ERR\_CONNECTION\_REFUSED**
2. **GET http://localhost:5001/api/students/profile/... net::ERR\_CONNECTION\_REFUSED**
3. **Cross-Origin-Opener-Policy warnings**



### **Errors**

1. **CORS Policy Error**
2. **404 (Not Found) Error**
3. **CORS Preflight Error (for POST)**

## **References**

* Google Docs API documentation
* Software project report examples