FINAL PROJECT REPORT

STUDENT TEACHER BOOKING MANAGEMENT SYSTEM

(Advance Project)

TECHONLOGY USED:

HTML, CSS, JAVASCRIPT, FIREBASE (Authentication & Fire-store)

Internship Company

UNIFIED MENTOR PRIVATE LIMITED

Duration

25-01-2025 to 25-07-2025

SUBMITTED BY:

NAME: **Astha Dhiman**

UNID**: UMIP275397**

SUBMITTED TO:

Unified mentor

Pvt. Ltd.

INTRODUCTION

The "Student-Teacher Booking Management System" is a real-time web-based application developed during the internship at Unified Mentor Pvt. Ltd. It is designed to bridge the communication gap between students and teachers by allowing students to schedule appointments with teachers and enabling teachers to manage those requests efficiently. This system also includes an admin panel to monitor all users and appointments.

Project Objectives

- To create a platform that allows student to book appointments with teachers based on availability.
- To implement role-based access control for students, teacher, and admin.
- To provide real-time appointment updates using Firebase.
- To facilitate seamless appointment management with approval/rejected features.
- To centralize user and appointment data under a secure and scalable database.

Project Scope

The project supports three distinct user roles:

- Student: Can register/Login, view teacher lists, send appointments requests, view status (pending, approved, rejected), and cancel requests.
- Teacher: Can register/Login, view all incoming appointments requests, approve/reject them, and monitor request history.
- Admin: Has full visibility over the system, including all users and appointments logs. Can also remove (except self).

LOGIC OF THE PROJECT

1. Architecture overview:

The system is built using a front-end stack (HTML, CSS, JavaScript) and uses Firebase as the backend. Firebase Authentication handles role-based access, and Fire-store stores all user and appointments data. The application is role-driven, and each user role sees different pages and permissions.

2. Module Design:
   - Authentication Module: Manages registration/login and role-based redirection.
   - Student Module: Allows viewing teachers, booking appointments, and tracking statuses.
   - Teacher Module: Allows viewing appointments requests, approving/rejecting them, and seeing student info.
   - Admin Module: Centralized dashboard to monitor appointments and users.

3. Database Schema (Fire-Store);

A. Collection: users
   -Fields: uid, name, email, role(student/teacher/admin), department (if teacher), subject, message.

B. Collection: appointments
   -Fields: appointment-Id, teacher-Id, student-Id, status(pending/approved/rejected), date, message.

4. Data Flow;
   - Student registers and logs in → redirected to student dashboard.
   - Student books an appointment → Fire-store creates an appointment record.
   - Teacher logs in → views appointments addressed to them.
   - Teacher approves/rejects → Fire-store updates the document.
   - Admin monitors all collections → reads logs and appointments flow.

5. File Structure and Role Segregation;

Each user has a separate JS file handling its logic:

- student.js: Handles appointment booking, viewing status.

- teacher.js: Handles request viewing and status updating.

- admin.js: Views all users and appointments.

6. Fire-base Call Behavior:

Instead of traditional REST APIs, the system uses Fire-base SDK methods:

-firebase.auth().createUserWithEmailAndPassword()

-firebase.firestore().collection('appointments').add()

- firebase.firestore().doc().update()

System Modules and Features

✅ Authentication Module

- Firebase Authentication for login and registration.
- Role-based redirected and validation (Student/Teacher/Admin).

👩‍🎓 Student Module

- Book appointment with teacher.
- View appointment history and live status.
- Cancel pending requests.

👩‍🏫 Teacher Module

- View student appointment requests.
- Approve or reject with one click.
- View appointment subject, department, and student details.

👨‍💼 Admin Module

- View total student and teacher count.
- Monitor all appointments.
- Delete users (except the currently logged-in-admin).
- View detailed logs and reports.

Technology Stack

- Frontend: HTML, CSS, JavaScript
- Backend: Firebase (Google’s Backend-as-a-Service)
  - Firebase Authentication
  - Firebase Fire-store (Cloud NoSQL Database)

Folder Structure

```
/project-root
|
├── index.html
├── login.html
├── register.html
├── student.html
├── teacher.html
├── admin.html
├── book-appointment.html
├── view-status.html
├── view-appointments.html
├── all-appointments.html
├── all-users.html
├── appointments-log.html
|
├── css/
│   └── style.css
|
├── js/
│   ├── app.js
│   ├── student.js
│   ├── teacher.js
│   ├── admin.js
│   ├── view-status.js
│   ├── view-appointments.js
│   ├── appointments-log.js
│   ├── all-appointments.js
│   └── all-users.js
|
└── firebase/
 └── firebase-config.js
```

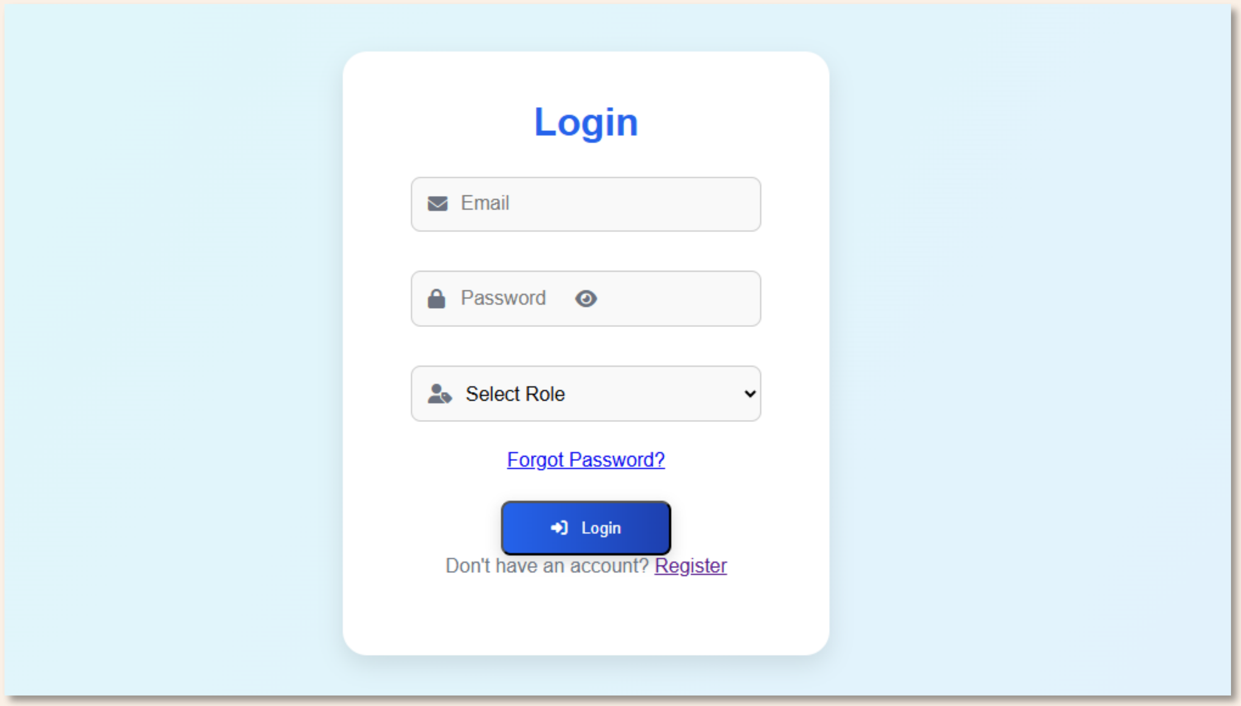
Implementation Overview

Each user has a unique dashboard and permissions. Firebase Auth manages the role during login and registration. Fire-store store user profiles and appointment record.

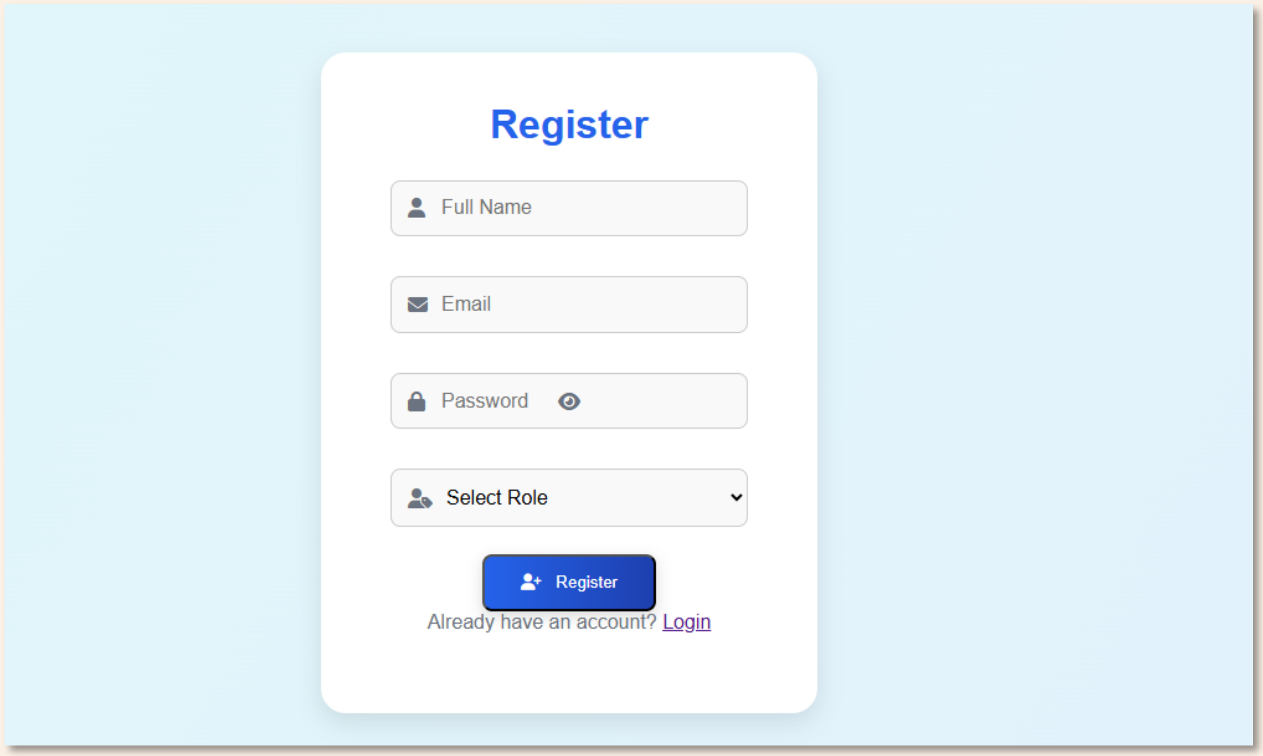
- Appointments are stored with details: teacher-Id, student-Id, date, message, and status.
- When a student books, a record is created.
- Teachers access appointments filtered by their ID.
- Admin access all record for monitoring.
- Status updates (approve/reject) happen in real time.

Screenshots

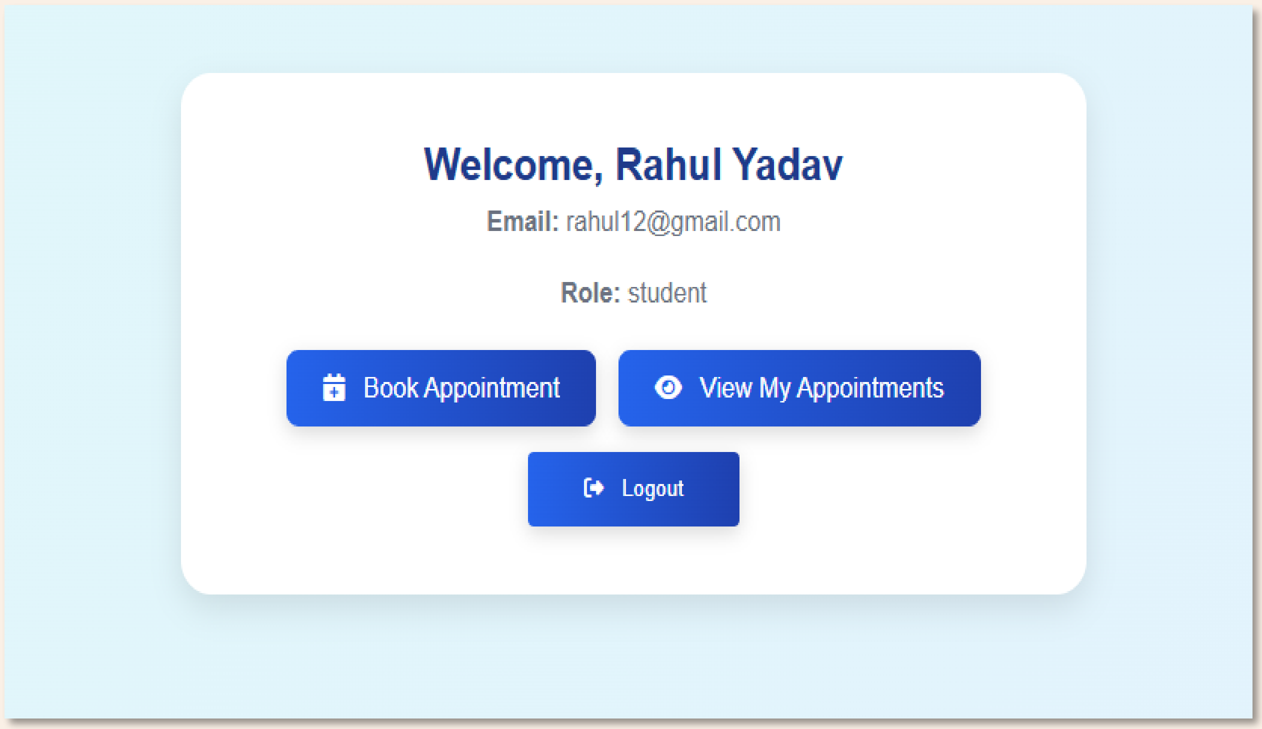
- Login Page



- Register Page



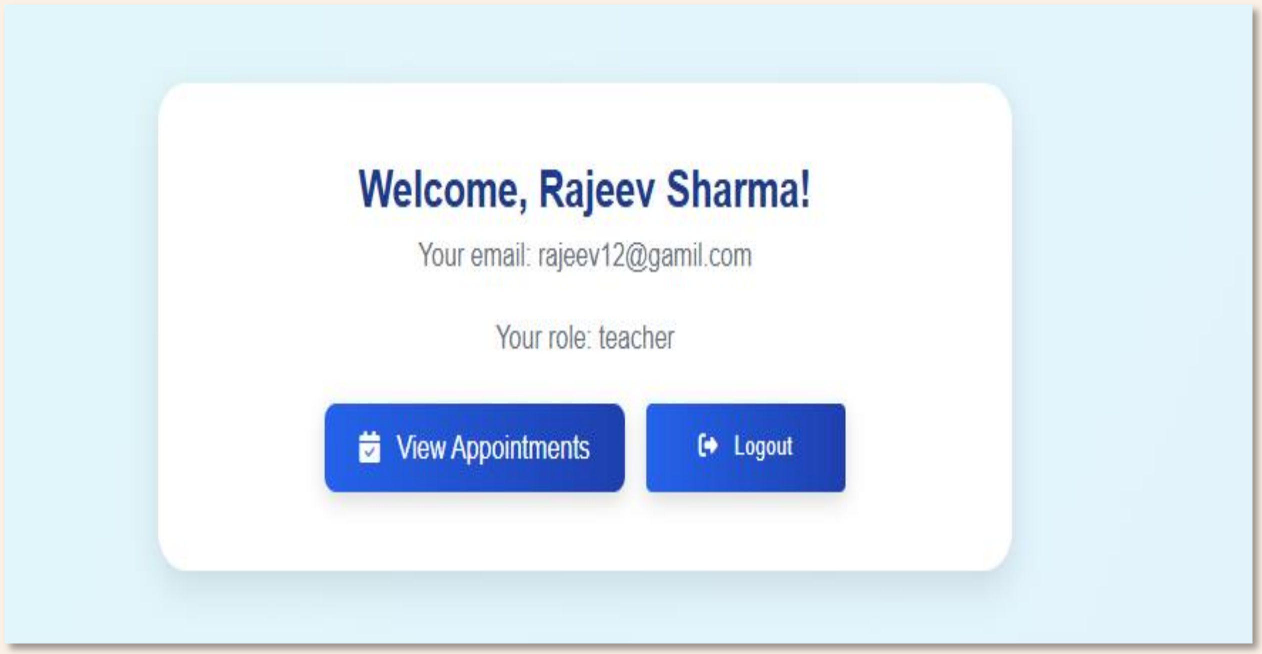
- Student Dashboard


Welcome, Rahul Yadav

Email: rahul12@gmail.com

Role: student

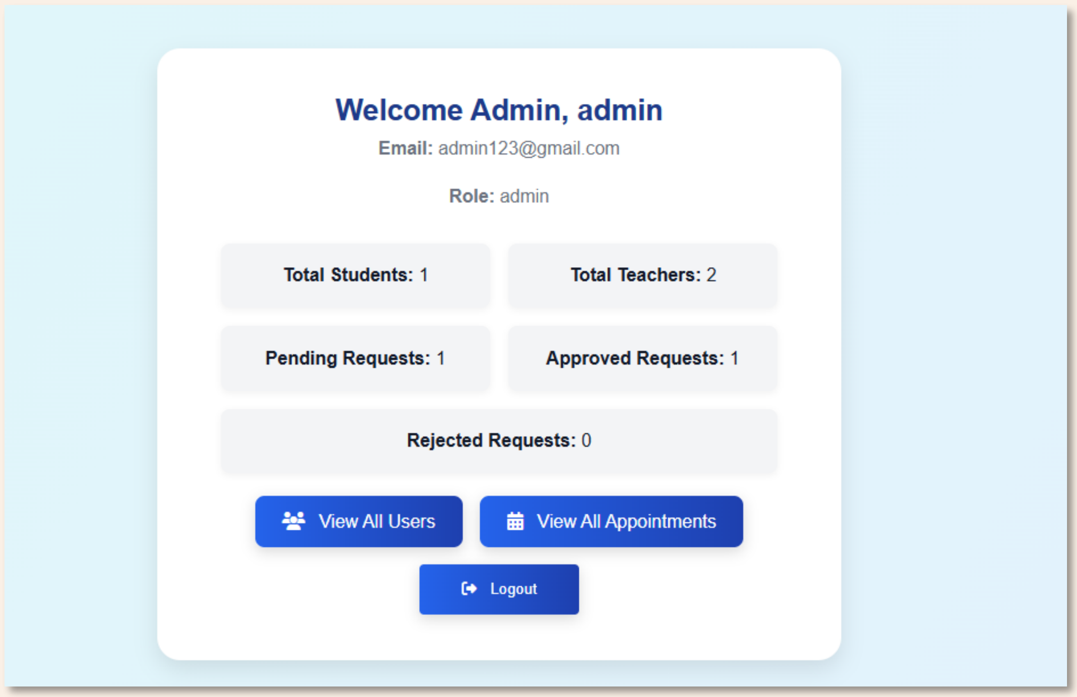
- Teacher Dashboard


Welcome, Rajeev Sharma!

Your email: rajeev12@gamil.com

Your role: teacher

- Admin Dashboard



Welcome Admin, admin

Email: admin123@gmail.com

Role: admin

Total Students: 1

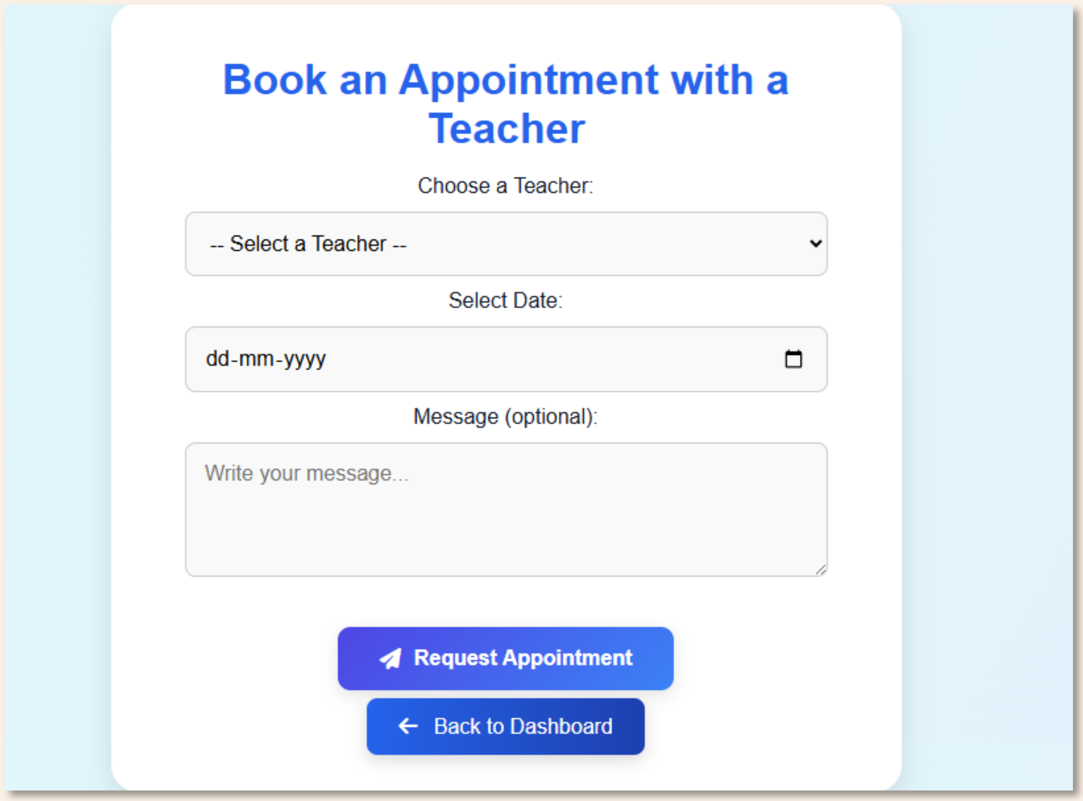
Total Teachers: 2

Pending Requests: 1

Approved Requests: 1

Rejected Requests: 0

- Book Appointment Page



Book an Appointment with a Teacher

Choose a Teacher:

-- Select a Teacher --

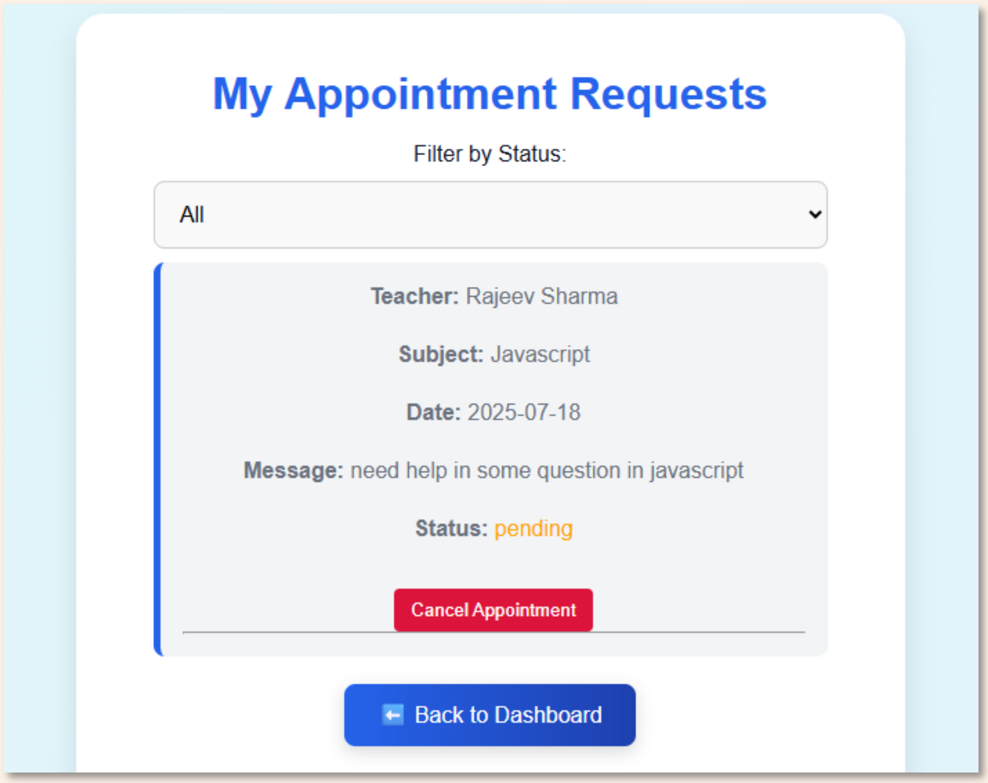
Select Date:

dd-mm-yyyy

Message (optional):

Write your message...

- View Appointment Status Page (Student View) & (Teacher View)



My Appointment Requests

Filter by Status:

All

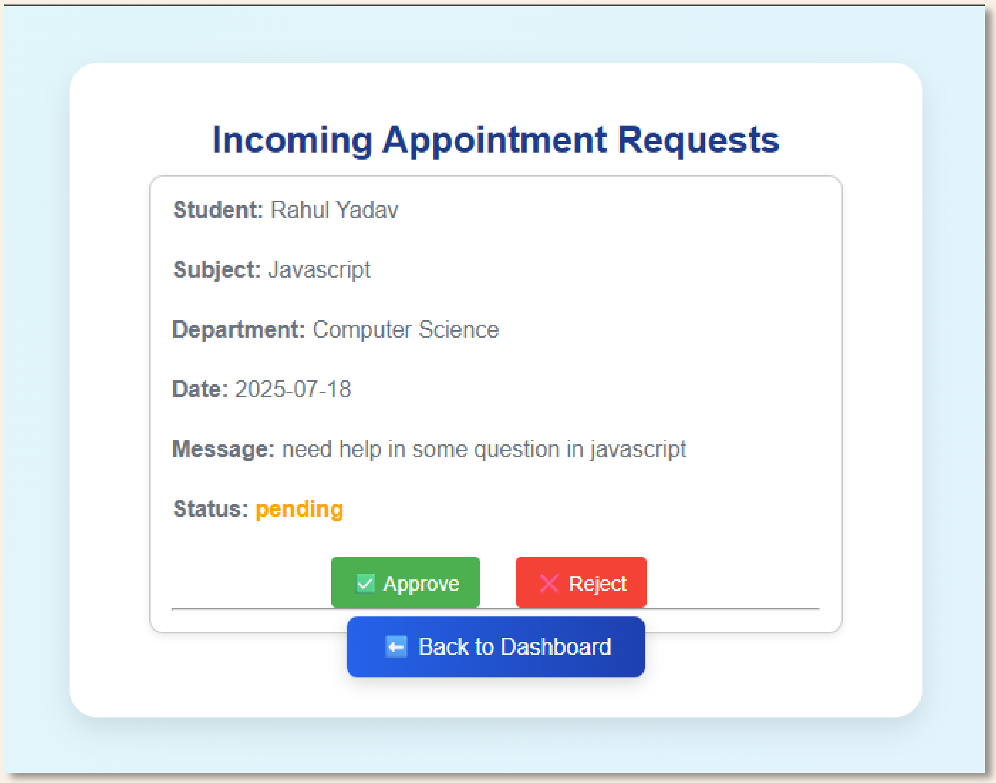
Teacher: Rajeev Sharma

Subject: Javascript

Date: 2025-07-18

Message: need help in some question in javascript

Status: pending



Incoming Appointment Requests

Student: Rahul Yadav

Subject: Javascript

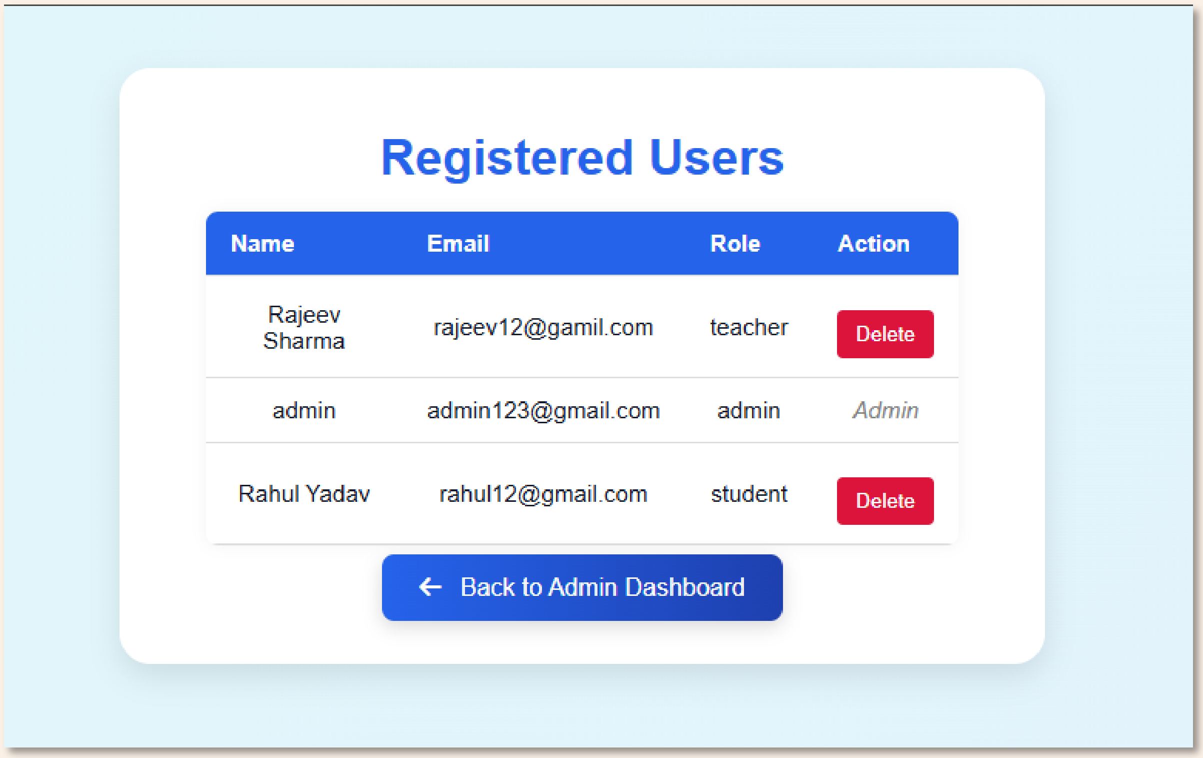
Department: Computer Science

Date: 2025-07-18

Message: need help in some question in javascript

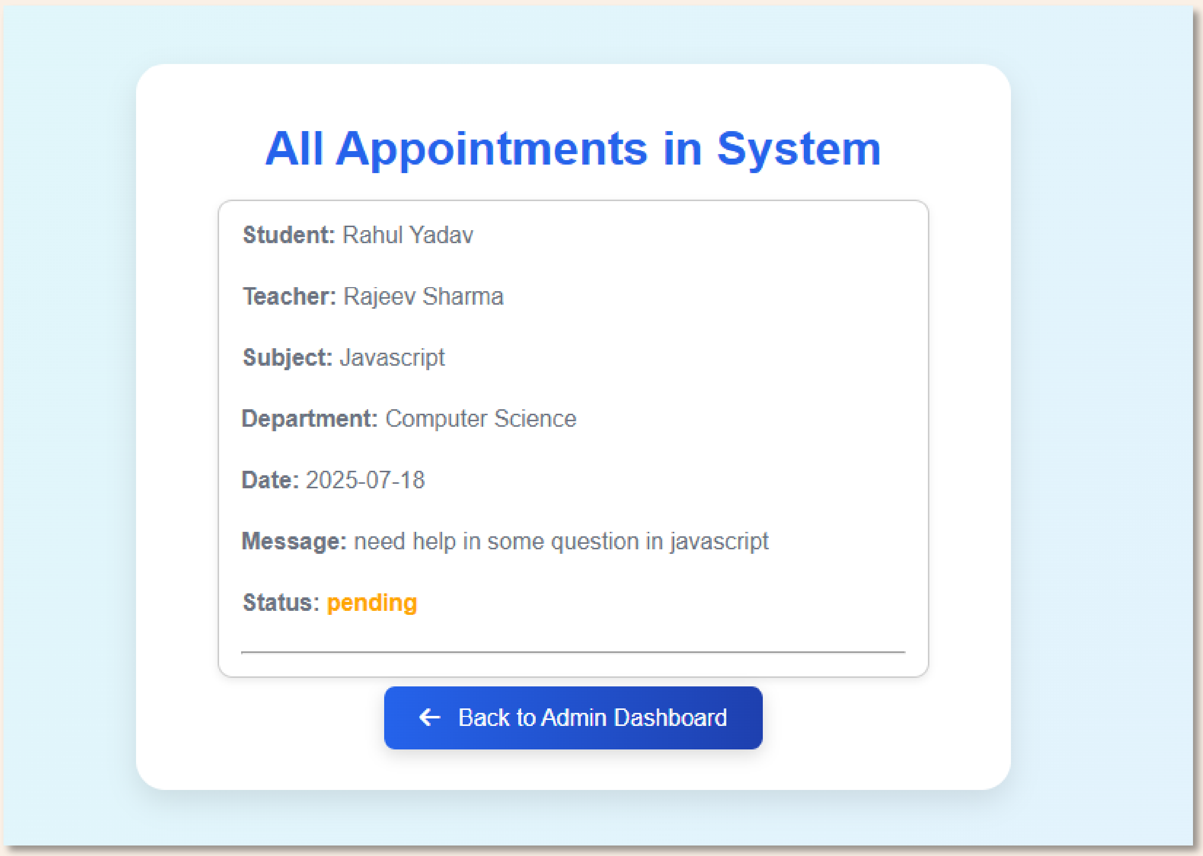
Status: pending

- Logs and System-Wide Appointment Page



Registered Users

Name	Email	Role	Action
Rajeev Sharma	rajeev12@gamil.com	teacher	Delete
admin	admin123@gmail.com	admin	*Admin*
Rahul Yadav	rahul12@gmail.com	student	Delete



All Appointments in System

Student: Rahul Yadav

Teacher: Rajeev Sharma

Subject: Javascript

Department: Computer Science

Date: 2025-07-18

Message: need help in some question in javascript

Status: **pending**

Challenges Faced

- Handling dynamic role-based redirection securely.
- Ensuring real-time Fire-store syncing without conflicts.
- Managing secure access to Firebase API.
- Debugging role-specific UI element and access restrictions.

Learning Outcomes

- Gained experience with Firebase Authentication and Fire-store.
- Improved understanding of front-end and back-end integration.
- Practiced full-stack development in a real-world internship setting.
- Learned best-practices in file structuring and modular code.

Security & Deployment Notes

- Firebase API keys are placed in a separated config. File.
- Role checks are performed on every page load.
- Public release version should use environment variables or .env file to mask sensitive data.

Future Enhancements

- Add notification support for appointment status updates.
- Implements search/filter and pagination for appointments.
- Enable appointment export to CSV or PDF (for admin).
- Setup emails alerts for approved/rejected appointments.

- Responsive and mobile-first UI improvements.

Conclusion

The “Student-Teacher Booking Management System” fulfills its purpose of streamlining academic appointments between students and teachers. It demonstrates an understanding of real-time web apps, user management, and secure role-based systems. The experienced gained form this internship has contributed significantly to practical learning and problem-solving skills.