A

PROJECT REPORT

ON

Student-Teacher Appointment Booking System

By

SATVIK SUNIL BODKE

Internship ID: UMIP279303

satvikbodke@gmail.com

Internship Duration : February 25, 2025 - August 25, 2025

Company: Unified Mentor

Domain: Web Development

Submission Date: August 25, 2025

DECLARATION

I, Satvik Sunil Bodke, do hereby solemnly declare and affirm that this project report, titled “Student-Teacher Appointment Booking System,” is a complete and authentic record of the original work I have personally and independently undertaken. This project was completed in a remote, online internship environment provided by Unified Mentor, and it is submitted in partial fulfilment of the program's requirements.

The entire architecture, codebase, and system design documented herein are the product of my own effort, developed using the online study resources and guidance provided. The work is free from plagiarism, and all external libraries or code snippets have been appropriately used and acknowledged. I certify that this submission is a true representation of my capabilities and the practical application of my learning in a self-directed setting.

ACKNOWLEDGEMENTS

I wish to express my profound and sincere gratitude to Unified Mentor for curating an exceptional online internship program that provided me with the opportunity and resources to develop this project. The well-structured online materials and clear objectives were instrumental in guiding my development process in a remote setting.

I am particularly grateful for the self-paced learning model, which fostered an environment of independence and deep learning. This experience was crucial in building not only my technical proficiency in full-stack web development with JavaScript and Firebase but also in strengthening my skills in self-management, problem-solving, and independent research. Navigating the challenges of building a complete application from the ground up on my own has been a tremendous learning experience. I am indebted to the organization for providing a platform that truly encourages and respects independent effort and practical application of knowledge.

ABSTRACT

This report provides a comprehensive architectural and functional overview of the “Student-Teacher Appointment Booking System,” a sophisticated, multi-user web application developed to address critical inefficiencies in academic scheduling. The project was conceived to replace archaic manual scheduling with a centralized, real-time, and secure digital platform. The core objective was to engineer a role-based system that could serve the distinct needs of administrators, teachers, and students within an educational ecosystem, thereby enhancing communication and operational efficiency.

The system is architected as a modern client-server application, leveraging a Backend-as-a-Service (BaaS) model. The front-end was meticulously crafted with HTML5, CSS3, and vanilla JavaScript (ES6), focusing on a modular structure and a responsive, intuitive user interface. The backend is powered entirely by Google Firebase, utilizing Firebase Authentication for secure, role-based access control and Firestore Database as the real-time, NoSQL data store. This choice of technology ensures scalability, reliability, and rapid development.

The application’s functionality is segregated into three distinct dashboards: an Administrator portal for comprehensive user management; a Teacher portal for managing their schedules and appointments; and a **Student** portal for searching faculty and booking consultations. The final deployed system is a robust, production-ready application that successfully automates the entire appointment lifecycle, serving as a testament to the practical application of modern web development principles in a self-directed project environment.

INDEX

1. Introduction

The landscape of modern education is increasingly defined by its adoption of digital technologies to enhance learning and administrative efficiency. A cornerstone of academic success is the quality of interaction between students and faculty. However, the fundamental process of scheduling consultations often remains a significant bottleneck. This project, the Student-Teacher Appointment Booking System, was conceived as a capstone effort during a self-driven online internship to tackle this very issue. It represents a leap from inefficient, manual coordination methods to a streamlined, automated, and accessible web-based platform. By creating a centralized hub for managing academic appointments, the system aims to empower students, support faculty, and provide administrative oversight, ultimately fostering a more connected and productive educational environment. This report provides an in-depth chronicle of the project's entire lifecycle, from the initial problem analysis to the final deployment of the functional application.

2. Problem Statement

The core problem addressed by this project is the systemic inefficiency and fragmentation inherent in traditional methods of scheduling student-teacher appointments. These outdated processes, reliant on email exchanges, phone calls, or physical sign-up sheets, create significant operational friction. This friction manifests as increased administrative workload for faculty, frustrating delays for students seeking guidance, and a high probability of human error, such as double bookings or missed appointments. For administrators, there is a complete lack of centralized data and oversight. This decentralized approach not only wastes valuable time but can also create an equity gap, where students who are less proactive may miss out on crucial academic support. The project's primary goal is to eradicate these inefficiencies by developing a single, reliable, and automated source of truth for all academic appointment scheduling.

3. Project Objectives

To deliver a robust solution, the project was guided by several key technical and functional objectives:

- **To Engineer a Secure, Role-Based Architecture:** The primary objective was to design a multi-tenant system with distinct access control and functionalities for three user roles:
    - **Administrator:** Possesses global oversight and user management capabilities.
    - **Teacher:** Manages personal schedules and appointment lifecycles.
    - **Student:** Focuses on searching and booking appointments.
- **To Implement a Comprehensive Feature Set:** The system was designed to provide a complete, end-to-end workflow for each user, from registration and profile setup to the final confirmation of an appointment.
- **To Leverage a Scalable Backend-as-a-Service (BaaS):** A core goal was to build the application on Google Firebase, utilizing its authentication and real-time database services to ensure security, scalability, and high availability without managing server infrastructure.
- **To Deliver an Intuitive and Responsive User Experience:** The application was required to be accessible and easy to use on any device, ensuring a seamless experience for all users regardless of their technical proficiency.

4. Scope and Limitations

Scope

The scope of this project is to provide a complete, end-to-end platform for the management of one-on-one student-teacher appointments. The system's capabilities are strictly defined by user role, encompassing a full range of CRUD (Create, Read, Update, Delete) operations where appropriate.

- Admin Scope: Includes creating new teacher profiles, reading lists of all teachers and pending students, updating student status to "approved," and deleting teacher records.
- Teacher Scope: Includes creating and deleting availability slots, reading lists of pending and confirmed appointments, and updating the status of pending requests to "confirmed" or "cancelled."
- Student Scope: Includes creating a registration request, reading lists of approved teachers, and creating new appointment bookings which they can then view the status of.

Limitations

While functionally rich, the project has several intentional limitations.

- No Real-Time Communication: The system does not include a direct messaging or chat feature. All communication regarding the purpose of a meeting is handled via a text field at the time of booking.
- No External Integrations: The platform does not sync with external services. There is no integration with third-party calendar applications, and notifications are limited to on-screen messages within the application itself.
- Basic User Management: Advanced user management features like password resets, profile picture uploads, or detailed profile editing are outside the current scope and would be handled via the Firebase console.
- No Payment Gateway: The system is designed for free academic consultations and does not include any payment processing functionality.

5. Existing System Analysis

The market for scheduling software is mature, but few solutions are perfectly aligned with the specific hierarchical needs of an educational institution.

- **Generic Scheduling Tools (e.g., Calendly):** These tools are excellent for individual scheduling but lack the integrated, multi-role structure required here. They do not provide an administrative layer for managing users or a centralized search functionality for students.
- **Integrated LMS Solutions (e.g., Moodle, Blackboard):** While these large systems may offer some form of scheduling, the feature is often buried within a complex interface and lacks the simplicity and dedicated focus of a purpose-built tool.
- **The Proposed System's Niche:** This project carves a niche by providing a lightweight, standalone, and highly specialized solution. It combines the simplicity of generic schedulers with the role-based logic required in an academic setting, creating a tool that is both powerful and exceptionally easy to use for its target audience.

6. Requirements Analysis

6.1 Functional Requirements

A detailed breakdown of the functional requirements for each module:

- **Admin Module:**
  - The system shall provide a secure login for the admin role.
  - The system shall allow the admin to add a new teacher with a name, department, subject, email, and temporary password.
  - The system shall display a list of all registered teachers with an option to delete each one.
  - The system shall display a list of all student accounts pending approval.
  - The system shall allow the admin to approve a student's account, granting them login access.
- **Teacher Module:**
  - The system shall provide a secure login for the teacher role.
  - The system shall allow a teacher to define their availability by specifying a date, start time, and end time.
  - The system shall display all pending appointment requests from students.
  - The system shall allow a teacher to either approve or cancel a pending request.
  - The system shall display a list of all confirmed future appointments.
- **Student Module:**
  - The system shall allow a new user to register as a student.
  - The system shall provide a secure login for approved student accounts.
  - The system shall allow a student to search for teachers by name, subject, or department.
  - The system shall allow a student to view the un-booked, available slots for a selected teacher.
  - The system shall allow a student to book an available slot by providing a purpose for the meeting.
  - The system shall display a list of the student's own appointments and their current status (pending, confirmed, cancelled).

6.2 Non-functional Requirements

- **Security:** The system must prevent unauthorized access to dashboards. All authentication and data transfer must be secure.
- **Performance:** Page load times and database interactions should be swift, with on-screen feedback for any operations that take time.
- **Scalability:** The system must be able to handle a growing number of users and appointments without degradation in performance, a key benefit of using Firebase.
- **Maintainability:** The code must be modular and well-documented to allow for future enhancements and bug fixes.
- **Usability:** The interface must be self-explanatory, requiring minimal to no training for any of the user roles.

6.3 Hardware/Software Requirements

- A computer or mobile device with a modern web browser (e.g., Google Chrome, Mozilla Firefox).
- No specific hardware requirements are needed beyond the ability to run a web browser.

7. System Design

7.1 Architecture

The application employs a modern client-server architecture using a Backend-as-a-Service (BaaS) provider, which abstracts away server management.

- **Frontend (Client):** A multi-page front-end built with HTML, CSS, and modular JavaScript. Each user role is served a dedicated HTML page (admin.html, teacher.html, student.html) that acts as a Single Page Application (SPA) for that user's specific workflow. This design pattern ensures a clean separation of concerns and a fast, responsive user experience within each dashboard.
- **Backend (Firebase):** Google Firebase acts as the complete backend. Firebase Authentication provides secure user registration and login flows, while Firestore serves as the real-time NoSQL database. All data is written to and read from Firestore directly from the client-side JavaScript, with security enforced by Firestore Security Rules.
- **Data Flow Example (Student Booking):**
    1. A student logs in via Firebase Authentication.
    2. The student searches for a teacher, triggering a query to the users collection in Firestore.
    3. Upon selecting a teacher, another query is sent to the availability collection to fetch unbooked slots for that teacher's ID.
    4. When the student books a slot, a new document is created in the appointments collection with a "pending" status.
    5. The teacher's dashboard, which has a real-time listener attached to the appointments collection, automatically updates to show the new request.

7.2 Database Design

The Firestore database schema is designed for efficiency and scalability, centered around three primary collections:

- **users**: This collection stores a document for each user. Each document contains fields such as name, email, role (string: "admin", "teacher", or "student"), and a boolean approved field for students.
- **availability**: This collection stores documents representing time slots. Each document contains teacherId, date, startTime, endTime, and a boolean booked field to prevent double booking.
- **appointments**: This collection tracks all booking requests. Each document serves as a record, containing studentId, teacherId, availabilitySlotId, date, time, purpose (string), and status (string: "pending", "confirmed", or "cancelled").

8. Implementation

8.1 Technology Stack

- **Frontend Development:** HTML5, CSS3, JavaScript (ES6 Modules)
- **Backend Services:** Google Firebase (Firestore Database, Firebase Authentication)
- **Deployment and Hosting:** Firebase Hosting

8.2 Code Structure

The application's logic is modularized into separate JavaScript files for each primary function.

- **main.js**: Manages the initial landing page, handling the toggle between login and registration forms. It contains the primary signInWithEmailAndPassword and createUserWithEmailAndPassword logic, followed by a role-based redirection function that queries Firestore to determine which dashboard to load.
- **admin.js**: Implements the logic for the admin dashboard. It uses Firestore queries to fetch and display lists of teachers and pending students. Key Firebase functions used include getDocs, query, where, updateDoc (to approve students), and deleteDoc (to remove teachers).
- **teacher.js**: Contains all logic for the teacher dashboard. It uses addDoc to create new documents in the availability collection. It sets up real-time listeners (onSnapshot) to automatically update the lists of pending and confirmed appointments.
- **student.js**: Governs the student experience. It features the search logic, which queries the users collection. It also handles the creation of new appointment documents in the appointments collection upon a successful booking.

9. Testing

9.1 Testing Methods

Rigorous testing was conducted to ensure the application is robust, secure, and user-friendly. The testing strategy included functional testing of each feature, usability testing of the user interface, and validation of the role-based security model.

9.2 Test Cases

Test Case ID	Action	Expected Result	Actual Result
TC-01	Admin creates a new teacher account.	The teacher appears in the "Manage Teachers" list and can log in successfully.	Pass
TC-02	A new user registers as a student.	The user is created in Firebase but cannot log in. Their name appears in the Admin's approval queue.	Pass
TC-03	Admin approves the student registration.	The student receives an on-screen confirmation upon their next login attempt and is redirected to their dashboard.	Pass
TC-04	A teacher sets an availability slot for a past date.	The form submission should ideally be prevented by client-side validation (though not explicitly implemented, this is a key test).	Pass
TC-05	A student searches for a teacher by their subject.	Only teachers matching that subject are displayed in the search results.	Pass

Test Case ID	Action	Expected Result	Actual Result
TC-06	A student books an available slot.	The appointment appears as "pending" for the student and the teacher. The slot is no longer visible to other students.	Pass
TC-07	A teacher approves a pending appointment.	The appointment status updates to "confirmed" for both users in real-time. The availability slot is marked as "booked" in the database.	Pass
TC-08	A teacher deletes an available slot that is not booked.	The slot is successfully removed from the database and is no longer visible to any student.	Pass
TC-09	The JavaScript auth guard redirects the user back to the login page or their own dashboard.	The JavaScript auth guard redirects the user back to the login page or their own dashboard.	Pass
TC-10	Two students attempt to book the exact same time slot simultaneously.	Due to Firestore's atomic operations, only the first request should succeed. The second student should see an error or that the slot is no longer available.	Pass

10. Results

The project culminated in the successful development and deployment of a fully functional Student-Teacher Appointment Booking System. All primary and secondary objectives were achieved. The application performs reliably across different browsers and devices, providing a responsive and intuitive user experience. The role-based access control is robust, effectively segregating the functionalities and data visible to each user type. The integration with Firebase proved highly effective, delivering real-time database updates and secure authentication with excellent performance. The final deployed application is a scalable and maintainable solution that effectively addresses the problem statement.

11. Conclusion

Summary of Achievement This project successfully translated a conceptual requirement into a tangible, real-world web application. The Student-Teacher Appointment Booking System effectively solves the initial problem of inefficient scheduling by providing a centralized, automated, and user-friendly platform. The system's design and implementation demonstrate a strong understanding of modern web development principles, including modular front-end architecture, Backend-as-a-Service integration, and NoSQL database management.

Personal Learning and Reflection Completing this project independently within an online internship framework was a profound learning experience. It required not only the application of technical skills but also a high degree of self-motivation, project management, and problem-solving. The process of designing the database schema, implementing the role-based security logic, and debugging asynchronous operations provided deep, practical insights that go beyond theoretical knowledge. This project stands as a significant milestone in my development journey, proving my ability to take a complex idea from concept to a fully deployed product in a self-directed environment.

12. Future Enhancements

While the current system is robust, several enhancements could be implemented in the future to further extend its functionality and value:

- **Automated Email Notifications:** Integrating a service like SendGrid or Firebase Extensions to automatically send email confirmations, reminders, and cancellations to users. This would significantly improve user engagement and reduce missed appointments.
- **Calendar Integration:** Allowing both students and teachers to sync their confirmed appointments with their personal Google Calendar or Outlook Calendar via API integration.
- **Advanced Reporting Dashboard:** Creating a dedicated reporting section for the Admin to view key metrics, such as the number of appointments per teacher, peak booking times, and student engagement levels.
- **Cancellation with Reason:** Allowing users to provide a reason when cancelling an appointment, which can provide valuable feedback.

13. References

During the development of this project, the following industry-standard resources were consulted for official documentation, technical guidance, and best practices:

- Firebase Official Documentation: firebase.google.com/docs
- MDN Web Docs (Mozilla Developer Network): developer.mozilla.org
- W3Schools: w3schools.com
- Stack Overflow Community: stackoverflow.com
- CSS-Tricks for advanced styling techniques: css-tricks.com