Project Report: Student-Teacher Appointment System

1. Project Overview

The Student-Teacher Appointment System is a full-stack(React + Firebase) web application that simplifies appointment booking and communication among students, teachers, and administrators. It includes secure authentication, role-based access control, and a messaging system—all built using modern web technologies.

Objective: To provide a seamless, role-specific platform for managing academic appointments, ensuring security, usability, and efficiency.

2. Live Demo

Access the application: https://student-teacher-appointment-eight.vercel.app/

Test Admin Credentials:

- Email: admin@gmail.com
- Password: 123456

3. Key Features

- Authentication with Firebase
- Role-Based Dashboards for Admins, Teachers, and Students
- Appointment Booking, Updating, and Cancellation
- Secure Messaging System
- Admin Controls: Approve users, assign roles
- Protected Routes using React Context
- Action Logging with timestamps and severity levels

4. Technologies Used

Technology - Purpose

- React (JSX) - Frontend framework
- Firebase - Authentication & Firestore database
- Vite - Build tool and dev server
- Tailwind CSS - UI design and responsive styling
- JavaScript (ES6+) - Logic and dynamic interactions

5. Project Structure

src/
├— components/ → UI components

```
├── context/      → Auth and role context providers
├── pages/        → Dashboards and login/signup screens
├── routes/       → ProtectedRoute components
├── services/     → Firebase logic (auth, db, messaging)
├── utils/        → Logger and helpers
├── App.jsx       → Main app component
└── main.jsx      → Entry point
```

6. Solution Design

LLD (Low-Level Design)

Document includes class diagrams, sequence diagrams, and context providers.

File Submitted: LLD_Solution_Design.pdf

Architecture Design

Wireframes for login, dashboards, appointment pages; system architecture includes Firebase Auth, Firestore, and role-based access via React Context.

Files Submitted: System_Architecture_Document.pdf, Wireframe_UI_Document.pdf

7. Code & Architecture Optimization

Code-Level Optimization

- Lazy loading routes to reduce bundle size
- Custom hooks for reusable logic
- Logger utility to track and debug events

Architecture-Level Optimization

- Separation of concerns
- Optimized Firestore queries
- Role resolution done once at login

8. Logging & Monitoring

Implemented via src/utils/logger.js with timestamped logs, severity levels, and tracking for authentication, appointments, and admin actions.

9. Test Cases

Test Case ID - Description - Input - Expected Output

- TC_01 - Student login - Valid email/password - Redirect to Student Dashboard
- TC_02 - Admin approval flow - New student signs up - Admin sees pending user
- TC_03 - Appointment creation - Valid time/date - Appointment saved and listed

- TC_04 - Messaging - Message sent to teacher - Message displayed in chat
- TC_05 - Access control - Student tries to access Admin route - Redirect to 403 page
- TC_06 - Logout - Click logout button - Redirect to login, session reset

10. Installation Instructions

npm install

npm run dev

Open http://localhost:3000 to view the app locally.

11. Access Control Summary

Admin: Manage users, view all appointments

Teacher: Manage their appointments, communicate with students

Student: Book, update, or cancel their own appointments

12. Limitations and Future Work

Current Limitations

- No real-time chat (currently uses Firestore polling)
- No calendar view integration

Planned Enhancements

- Google Calendar integration
- WebSocket-based messaging
- Notification system for appointment reminders

13. License