CS410 Smart Door Lock System – Code Manual

Team – Monark, Krishna, Shreyans, Chuan, Steven

1. Overview:

The Smart Door Lock System is a secure, IoT-enabled access control project developed as part of the CS410 course. It combines web-based remote control with hardware-based manual access. The system architecture consists of a React-based frontend dashboard for user interaction, Firebase for authentication and real-time data handling, a Node.js middleware to communicate with the Arduino microcontroller, and an Arduino Uno setup that includes an RFID module, capacitive touch sensor, and a servo motor for physically locking and unlocking a door.

The system is designed to offer both remote and local control of the door lock, log every interaction with timestamps, and notify users via email about every status change. It is ideal for smart homes, dorms, or office environments that require basic secure access control.

2. Project Structure:

The system is divided into three core components: frontend (React), backend (Node.js + Firebase), and hardware (Arduino Uno).

A. Frontend (React + Firebase):

- App.js: Manages routing and keeps track of the user's authentication state. It decides whether the user sees the login screen or the dashboard.
- Login.js: Presents a form where users enter their credentials. Uses Firebase Authentication to validate and sign in users.
- Home.js: This is the main dashboard. It includes buttons to lock/unlock the door, displays the current lock status, enables an auto-lock countdown, and renders a log of all previous door activities.
- firebase.js: Sets up the Firebase configuration for database and authentication services.
- App.css: Contains all custom styling used across the components for a consistent and responsive UI.

B. Backend (Node.js) :

- server.js: This middleware script acts as the bridge between Firebase and the Arduino device. It listens for any changes to the /door/status path in Firebase's real-time database. Upon detecting a change, it sends the corresponding command ("lock" or "unlock") to the Arduino over a serial connection. Additionally, it sends email notifications to the user using Nodemailer, configured with Gmail SMTP credentials.

C. Arduino Uno:

- The Arduino listens for serial input from the Node.js server. Upon receiving a "lock" or "unlock" command, it actuates a servo motor to rotate the latch accordingly.
- It also reads input from an RFID module and a capacitive touch sensor to allow manual unlocking.
- For debugging and monitoring, it sends responses back through Serial to confirm the action taken (e.g., “Door Locked”, “RFID Access Granted”).

3. System Workflow:

1. Authentication & Access: Users log into the web dashboard using their Firebase credentials through Login.js. If successful, the app loads the dashboard (Home.js).

2. Remote Access via Dashboard: Users can toggle the door lock using a button. This triggers toggleDoor() which updates /door/status in Firebase to either "locked" or "unlocked".

3. Backend Response: The Node.js server, continuously monitoring Firebase, detects the change in lock status. It sends the new command over the serial port to the Arduino and dispatches an email alert to the user.

4. Hardware Execution: The Arduino receives the command and rotates the servo motor to lock or unlock the door. Meanwhile, it listens for RFID card scans or touch sensor input to manually unlock the door without using the web interface.

5. Logging & Auto-Lock: Every action is timestamped and recorded in Firebase under /logs, including whether it was triggered remotely or manually. If the auto-lock option is enabled, a timer starts after each unlock event to automatically re-lock the door after the selected interval (5, 10, 15, or 30 seconds).

4. Key Features:

- Dual Access Control: Users can unlock the door via the web interface or manually using an RFID card or touch sensor.
- Activity Logging: All access events are logged in real-time with timestamps.
- Auto-Lock Countdown: Automatically locks the door after a preset duration.
- Email Notifications: Email alerts are sent on every lock/unlock action for enhanced security.

- Secure Authentication: Dashboard access is protected using Firebase Authentication.
- Real-Time Sync: Firebase ensures instant synchronization between the frontend and backend, allowing near-instant response to user actions.

5. Development & Coding Standards:

- The project follows best practices for clean, readable, and maintainable code across all components:
- Naming Conventions: CamelCase for variables/functions (toggleDoor, autoLockTimer), PascalCase for components (Home, Login).
- Modular Code: Code is broken into small, reusable components and functions to increase readability and testability.
- Comments & Documentation: Code is commented clearly to explain logic and complex operations, especially around Firebase triggers, serial communication, and component lifecycle.
- Security Considerations: Firebase database rules restrict write access to authenticated users. Email alerts add an extra layer of security awareness.

6. Tools & Libraries Used:

- Frontend: React.js, Firebase Auth, Firebase Realtime DB
- Backend: Node.js, Firebase Admin SDK, Nodemailer (SMTP)
- Hardware: Arduino Uno, Servo motor, MFRC522 RFID Reader, TTP223 Touch Sensor
- Communication: Serial over USB between Node.js and Arduino

End-User Manual

1. What Is This System?

This is a secure, IoT-based Smart Door Lock System that allows users to lock and unlock a door via:
- A web dashboard using Firebase.
- Physical input using an RFID card or capacitive touch sensor.

The system includes email notifications, auto-locking, and a log table to track door events.

2. Requirements

Hardware:
- Arduino Uno
- Servo motor (SG90 or similar)
- RFID reader
- RFID key card/tag
- Capacitive touch sensor
- Jumper wires
- Breadboard
- USB cable (for programming Arduino)

Software:
- VS Code (for frontend/backend)
- Arduino IDE
- Node.js (v18+)
- Firebase account (RTDB + Auth)
- Gmail account (with App Password enabled)

3. Setup Instructions

Hardware Setup:

1. Connect RFID:
   - SDA → D10
   - SCK → D13
   - MOSI → D11
   - MISO → D12
   - RST → D9
   - 3.3V and GND accordingly

2. Connect Touch Sensor:
   - VCC → D8
   - OUT → D7

- GND → GND

3. Servo:
   - Signal → D6
   - VCC → 5V
   - GND → GND

🖥️ Software Setup:

1. Frontend (React):

   cd smart-door-ui
   npm install
   npm start

2. Backend (Node.js server):

   npm install firebase-admin serialport nodemailer
   node server.js

3. Arduino Upload:
   - Open your .ino sketch in Arduino IDE
   - Select correct port (/dev/cu.usbmodemXXXX)
   - Upload the code
   - Keep Serial Monitor closed after upload (to free the port)

4. How to Use

1. Register/Login:
   - Open the app
   - Use valid credentials to log in

2. Lock/Unlock:
   - Click the central lock/unlock button
   - You'll get:
     - Email confirmation
     - Live status update
     - Entry added to history table

3. Auto-Lock:
   - Select timeout from dropdown (e.g., 10 sec)
   - After that time, the door auto-locks
   - You'll get notified and a new log entry appears

4. Manual Access (No Internet):
   - Touch the sensor or tap your authorized RFID card
   - Servo unlocks temporarily, then re-locks after a delay

5. Logout:
   - Click '🚪 Logout' to sign out

5. Notes

- Email alerts will be sent from your configured Gmail address using Nodemailer.
- Make sure Firebase Realtime Database rules allow authenticated reads/writes.
- If Serial port fails, close Arduino Serial Monitor or check correct port in server.js.