



Smart Lock

A Cloud-Based Key

Group 5

*Nathaniel Laurente, Adam Wu, Neena Nguyen,
Jackson Kennedy*

May 26, 2025

1 Front Matter

1.1 Executive Summary

In an era where security and convenience are necessary, traditional lock-and-key systems are becoming outdated. Lost keys, forgotten access codes, and lack of remote control present serious challenges for homeowners and businesses alike. Our Auto Smart Lock project is a solution designed to enhance security while offering friendly user accessibility. Our goal is to develop a smart lock that integrates modern authentication methods with remote access capabilities. By leveraging an ESP32-C3 microcontroller in conjunction with Firebase cloud services, our smart lock enables users to lock and unlock their doors from anywhere, eliminating the risks associated with traditional keys.

At the core of our smart lock is an ESP32-C3 microcontroller, which connects to WiFi and Firebase cloud services to securely manage lock and unlock commands. When a user enters a PIN or sends a command from their phone, the system checks if the code is valid. If it is, the solenoid lock is triggered, unlocking the door. We've designed the lock to be fast, secure, and easy to install, with a 3D-printed casing to house the components.

Before finalizing the prototype, we are running key tests to ensure everything works smoothly. These tests check if the phone properly connects to the lock, if the lock responds quickly, if multiple users can access it without conflicts, and if temporary PINs expire when they should. Our next steps include assembling all parts, optimizing performance, and making sure the system is fully secure and reliable. By the end of Winter 2025, we aim to have a fully functional smart lock prototype that provides a safer, smarter, and more convenient way to secure homes.

1.2 Ethics Statement for Smart Lock Project

Our Smart Lock is designed to enhance security and convenience for users while upholding ethical standards in privacy, safety, and accessibility. We recognize our responsibility to develop and deploy technology that prioritizes ethical considerations in the following ways:

1.2.1 Privacy and Data Protection

We are committed to safeguarding user privacy by:

- Minimizing data collection to only essential information required for functionality.
- Implementing encryption and secure authentication methods to prevent unauthorized access.
- Ensuring that user data is never shared or sold to third parties without explicit consent.

1.2.2 Security and Reliability

To maintain the integrity of the smart lock system, we will:

- Use cybersecurity measures to prevent hacking and tampering.
- Regularly update software and firmware to address potential vulnerabilities.
- Ensure the lock functions reliably under various conditions to prevent accidental lockouts or failures.

1.2.3 User Safety and Accessibility

We aim to create a system that is safe and inclusive by:

- Designing intuitive user interfaces for easy access by individuals with different levels of technical proficiency.
- Ensuring compliance with accessibility standards for individuals with disabilities.

1.2.4 Ethical Use and Non-Discrimination

The smart lock must be used ethically and responsibly by:

- Preventing misuse that could lead to unauthorized surveillance or discrimination.
- Avoiding biases in authentication methods that may disadvantage certain user demographics.

1.2.5 Transparency and Accountability

To uphold ethical standards, we will:

- Clearly communicate to users how their data is handled and stored.
- Provide documentation on security features, risks, and best practices.
- Accept feedback and take responsibility for any ethical concerns that may arise during development and deployment.

By adhering to these ethical principles, we ensure that our Smart Lock product aligns with values of privacy, security, fairness, and social responsibility.

Table of Contents

1	Front Matter	1
1.1	Executive Summary	1
1.2	Ethics Statement for Smart Lock Project	1
1.2.1	Privacy and Data Protection	2
1.2.2	Security and Reliability	2
1.2.3	User Safety and Accessibility	2
1.2.4	Ethical Use and Non-Discrimination	3
1.2.5	Transparency and Accountability	3
2	Background	6
2.1	Problem Definition	7
2.1.1	Need & Goal Statements	7
2.1.2	Design Objectives	7

3	Concepts Considered	7
3.1	6-3-5 Method	7
3.2	Brainstorming	9
3.3	Morphological Charts	13
3.4	Mind Map	15
3.5	Concept Selection	16
	3.5.1 Weighting Factors	16
	3.5.2 Decision Table	16
3.6	Transitioning SmartLock from Prototype to Market-Ready Product	17
	3.6.1 Custom Hardware Design	17
	3.6.2 Backend Infrastructure Development	17
	3.6.3 Mobile/Web App Enhancement	18
	3.6.4 Compliance and Security Certification	18
3.7	Detail of Design	18
	3.7.1 System Overview	18
	3.7.2 Connectivity and Cloud Integration	19
	3.7.3 Hardware Design	20
	3.7.4 Life Cycle Assessment (LCA)	21
3.8	Battery Power Management and Notification System	23
	3.8.1 Battery Selection and Power Optimization	23
	3.8.2 Battery Monitoring and User Notification	23
	3.8.3 Safe and Secure Battery Replacement or Recharging	23
3.9	Ownership and Access Control	25
	3.9.1 Ownership Model	25
	3.9.2 Access Delegation and Permissions	25
	3.9.3 User Interface for Access Management	25
3.10	History and Status Logging	26
	3.10.1 History Log of User Actions	26
	3.10.2 Status Log for Lock Health Monitoring	26
	3.10.3 Action Confirmation and Notifications	26
3.11	Initial Installation and Setup Procedure	27
	3.11.1 Step 1: Unboxing and Physical Inspection	27
	3.11.2 Step 2: Mobile Application Download	27
	3.11.3 Step 3: Initial Bluetooth Pairing	27
	3.11.4 Step 4: WiFi Provisioning	28
	3.11.5 Final Notes and Engineer Considerations	28

4	Final Design Outline	29
4.1	Introduction	29
4.2	Cloud Infrastructure (AWS)	29
4.3	PIN Code Access	30
4.4	WiFi Connection	31
4.5	Many-to-Many Relationship: Locks and Users	32
4.6	Conclusion	32
5	Economic Analysis	32
5.1	Outstanding Issues	35
6	Appendices	35
6.1	Scope	35
6.1.1	Lock Hardware Functionality Test	36
6.1.2	Multi-User & Concurrency Test (Low Priority)	37
6.1.3	Firebase Mobile Connectivity Test	38
6.1.4	Lock Hardware Functionality Test	38
6.1.5	Multi-User & Concurrency Test	38
6.1.6	Firebase Mobile Connectivity Test	38
6.1.7	Lock Hardware Functionality Test	38
6.1.8	Multi-User & Concurrency Test	39
6.1.9	Safety Precautions	39
6.1.10	Data Collection Method	39
6.1.11	Observation of External Factors	40
6.2	Test Results	40
6.2.1	From Test Plan	40
6.2.2	Test Result Summary Table	41
7	Manufactured Product Testing	42

2 Background

Persona 1



Marcus Lee
Homeowner | 42 years old | IT Manager | Lives in a suburban neighborhood

Goals

- secure keyless entry for family
- enable remote monitoring when traveling
- integrate smart lock with his existing smart home system

Behaviors

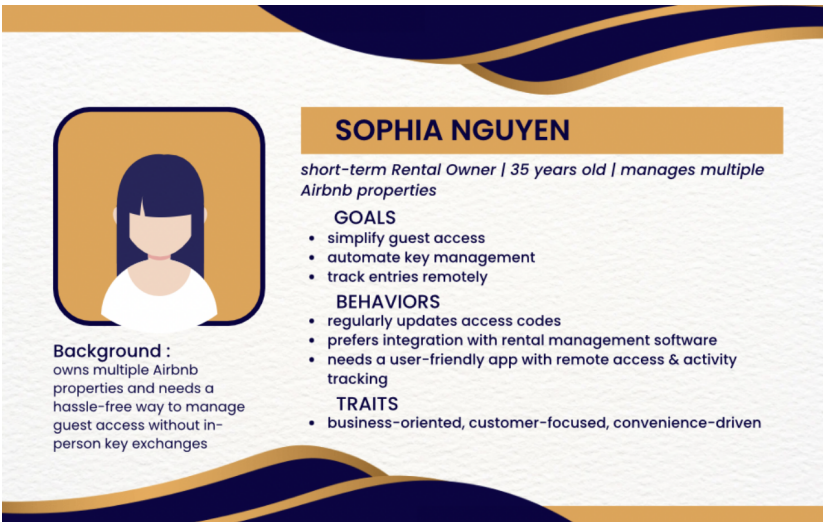
- uses smart home apps
- values reliability & ease of set up over complex features

Traits

- security-conscious, tech-savvy, efficiency-driven

The persona card for Marcus Lee features a stylized illustration of a man with dark hair, wearing a light blue shirt and a dark tie, enclosed within a blue hexagonal frame. The background of the card is white with blue and yellow geometric accents at the top and bottom.

Persona 2



SOPHIA NGUYEN
short-term Rental Owner | 35 years old | manages multiple Airbnb properties

Background :
owns multiple Airbnb properties and needs a hassle-free way to manage guest access without in-person key exchanges

GOALS

- simplify guest access
- automate key management
- track entries remotely

BEHAVIORS

- regularly updates access codes
- prefers integration with rental management software
- needs a user-friendly app with remote access & activity tracking

TRAITS

- business-oriented, customer-focused, convenience-driven

The persona card for Sophia Nguyen features a stylized illustration of a woman with dark hair, wearing a white top, enclosed within a rounded square frame with a yellow background. The card has a white background with blue and yellow wavy accents at the top and bottom.

2.1 Problem Definition

2.1.1 Need & Goal Statements

- Need Statement - Traditional lock-and-key systems are inconvenient and insecure, as users may forget, lose, or have their keys stolen. Additionally, they obviously lack remote access, making it difficult for someone to enter the building if someone lost/forgot their tiny key.
- Goal Statement - Our smart lock will address these issues by enabling remote locking and unlocking, getting rid of the need for physical keys, and ensuring secure authentication for authorized users.

2.1.2 Design Objectives

- Remote access & control: ESP32-C3 connection with Firebase
- Local authentication via keypad:
- Store & validate PINs securely using Firebase cloud
- Multi-user support
- solenoid lock integration with ESP32

3 Concepts Considered

3.1 6-3-5 Method

Hardware & Mechanics

- use a servo motor for deadbolt control
- hall effect sensor to detect if the door is open or closed
- implement linear actuator for silent locking
- maybe anti-backlash gears to reduce mechanical play
- consider designing a modular lock housing with 3D-printed parts
- have a backup battery to ensure operation during power outages

Smartphone Integration

- develop a smartphone app for remote locking/unlocking
- push notifications for lock activity (ex: “Door locked at 3:45 PM”)
- have biometric authentication (Face ID or fingerprint)
- Use Bluetooth for proximity-based auto-unlock
- NFC support for quick unlocking via phone tap
- include multi-user access control with time-based permissions

Security Features

- Implement AES-256 encryption for communication between the lock and phone
- maybe two-factor authentication for app access
- develop a tamper-detection alarm if the lock is forced
- lockdown mode for multiple failed attempts
- include a manual override mechanism in case of system failure
- utilize rolling codes for Bluetooth pairing to prevent hacking

Software & Control

- ESP32 microcontroller for Wi-Fi and Bluetooth control
- Develop a closed-loop system to verify if the lock is engaged properly
- make a real-time event log accessible via the app
- add a scheduling feature for auto-locking at specific times
- guest mode with temporary passcodes
- Integration with voice assistants like Alexa or Google Assistant

Advanced Features

- GPS stuff
- add geofencing to lock or unlock based on the user's location
- integrate with smart home platforms like Home Assistant
- use AI-based behavior analysis to suggest locking patterns
- add a camera with facial recognition for auto-unlock
- enable remote firmware updates
- learn database SQL
- use solar charging for battery-powered locks

3.2 Brainstorming

Adam Wu

- As a person who has amnesia, I would like to be able to find my keys anytime so that when I forget where I place them, I can find them.
 - Having a "find my" solution with a key.
- As a person who loves security, I would like to have the best lock for my house so that lock pickers are not able to pick my lock.
 - Making an "authentication" key that resets the key code within a set time, making it harder for hackers to unlock the door.
- As a person who is always last-minute out the door, I fear forgetting to lock the door when I close it.
 - Auto-locking door when a person closes the door.
- As a person who often forgets to bring their keys, I am scared of getting locked out.
 - Having a notification from the key to the phone that alerts: "keys are not close by to you."
- As a parent, I am scared of my kids forgetting their keys and locking themselves out of their room.

-
- Creating a “master key” that only parents/admins can use to unlock specific doors.
 - Concerned about key battery life.
 - Send a notification to the user when the key is on low battery.

Nathaniel Laurente

- Key has the ability to notify the user when too far away from the user’s phone/body.
- Key deactivates/won’t be able to open the door if too far away from the owner.
- “Tap to Pay” technology concept.
 - Unlocks the door like a credit card tap on a phone.
 - If too complex, explore alternative ways to unlock the door.
 - Eliminates the need for a physical key.
 - Prevents stolen keys from working if the user still has their phone.
- Secure deactivation of the key when too far from the user.
 - Possible solution: Use the user’s phone for deactivation.
- One-time password generator between lock and key to ensure only this exact key can enter the house.
- Backup way to get into the house if the user forgets/loses their key.
 - Pin access code.
 - App allows for 2FA authentication using a thumbprint and/or Face ID.
- Will the battery last long enough for multiple years?

Neena Nguyen

- Existing smart lock solutions:
 - Smart locks for dorm rooms using mobile apps, passcodes, and scanners.

-
- Who will use this lock?
 - People with memory issues (elderly, ADHD).
 - University students in dorm rooms.
 - Student ID scanner integration.
 - Parents with small children (child-proof locks).
 - Features for parental control.
 - Locks after a curfew time.
 - Prevents children from unlocking without parental approval.
 - Alerts parents when kids come home from school.
 - What kind of door lock will it be?
 - Facial recognition (requires camera and database knowledge).
 - Logs entry and exit timestamps.
 - Digital passcode through an app.
 - Auto-relocking mechanism after failed attempts.
 - Bluetooth detection for unlocking within a certain range.
 - Dual authentication (PIN + scan).
 - Optional security trigger after specific hours.
 - Alerts when the door is left unlocked for too long.
 - Auto-locking after prolonged unlocking.
 - Detection system to check if the key is on the person.
 - Prevents intruders from entering without a key.

Jackson Kennedy

- Normal keys can be lock-picked, but digital keys can be secured based on a communication protocol.
- Secure authentication methods.
 - PIN authentication with 2FA.
 - Optimal PIN length (e.g., 4-digit PIN has 1,048,576 combinations).
 - Brute force prevention strategies.

-
- Preventing communication protocol vulnerabilities.
 - What protocol should be used? (Bluetooth has vulnerabilities and short range.)
 - Cloud-based solutions rely on third-party vendor security.
 - What information needs to be transferred? (Video data, authentication signals?)
 - Lock activation logic.
 - How exactly will the lock know when to unlock? (Sending a 0 or 1 signal based on specific conditions?)
 - Security and alerting technologies.
 - Sensors to detect nearby people.
 - Hidden camera or biometric verification for identity confirmation.
 - Scheduling and timed access.
 - Physical locks do not have scheduling options.
 - Implement timed unlocking (e.g., unlock for 15 minutes for a babysitter).
 - Extra verification to prevent intruders from exploiting schedules.

3.3 Morphological Charts

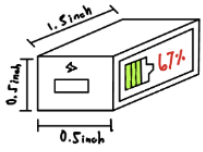

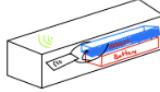




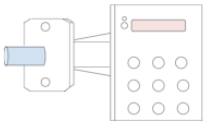
Physical Solutions				
	S1 - Nathaniel	S2 - Neena	S3 - Adam	S4 - Jackson
Key				
Lock		 <p>digital barcode scan + pin code input</p>		

Figure 1: Physical Solution Chart

Interface Solutions				
	S1 - Nathaniel	S2 - Neena	S3 - Adam	S4 - Jackson
User Input (app)	Code can be viewed in app and on lock display	-generates barcode to be scanned & allows user to set pin code -displays access logs (who & when lock was accessed) -tracks multiple keys/barcodes in use	Random generated codes every 30 min	Backup generated code/timing settings
Connection	Bluetooth	Bluetooth	NFC chip	NFC chip
Power	Key-charging station Lock-always plugged	Battery with power saving modes with charging bank	Using battery bank, or just a battery	Charger - USB-C or standardized port
Distance Tracker	GPS utilizing google maps API	Bluetooth low energy	Using some type of satellite tracker	Things like HC-SR04 or Wireless Senders/receivers
How to Unlock	“Tap to pay” tech Emergency view OTP from app	Utilizing app	Tap to unlock door	Tap key briefly to compare recurring random encrypted codes

Figure 2: Interface Solutions

3.4 Mind Map

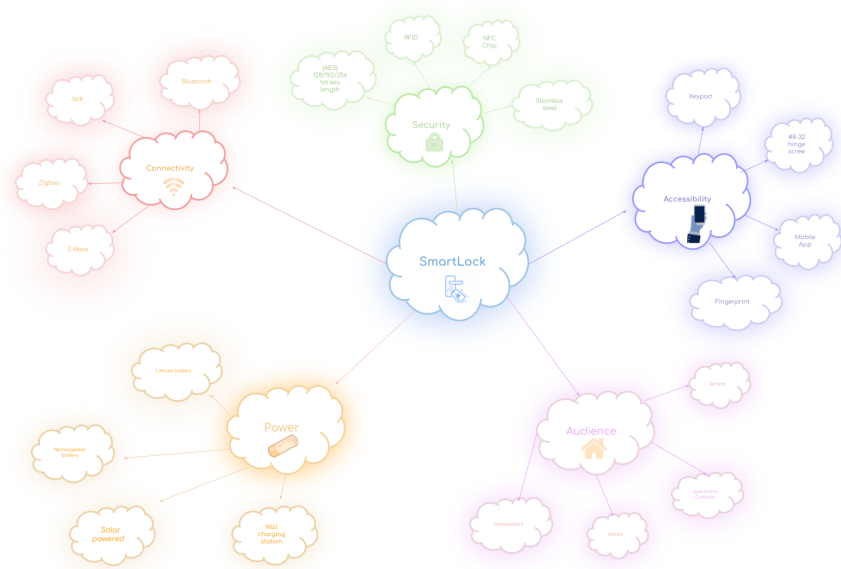


Figure 3: Mind Map

3.5 Concept Selection

3.5.1 Weighting Factors

Criteria	Weight (%)
Security	40
Cost	30
Power Consumption	20
User Convenience	10
Total	100

3.5.2 Decision Table

Criteria	Weight	Design 1 (Basic)	Design 2 (Mid-Range)	Design 3 (Advanced)
Security	40	6 (240)	8 (320)	10 (400)
Cost	30	10 (250)	6 (150)	3 (75)
Power Consumption	20	6 (90)	8 (120)	10 (150)
User Convenience	10	3 (30)	6 (60)	9 (90)
Total Score	100	680	720	780

Best Design: Design 3 (Advanced) with 780 points, prioritizing security, cost-efficiency, and power optimization.

3.6 Transitioning SmartLock from Prototype to Market-Ready Product

To move from a functional prototype to a marketable SmartLock product, several key engineering and business steps need to be addressed. This involves refining both hardware and software components, ensuring scalability and security standards.

3.6.1 Custom Hardware Design

The ESP32 microcontroller used in the prototype provides an platform for development and proposal, but a custom-designed microcontroller will offer improved performance, cost-efficiency, power-efficiency, and security.

- **PCB Design:** Develop a custom printed circuit board (PCB) integrating a microcontroller with essential peripherals (e.g., motor drivers, Wi-Fi/Bluetooth module, battery management).
- **Power Optimization:** Ensure low-power operation modes and efficient battery usage for real-world longevity.
- **Form Factor:** Design a compact form to fit lock, pin-pad, microcontroller dimensions and installation constraints.
- **Manufacturability:** Optimize for scalable manufacturing, component sourcing, manufacture, and assembly.

3.6.2 Backend Infrastructure Development

Firestore offers an accurate prototype, but a production system needs more control, security, and scalability.

- **Cloud-Based Database:** Develop a backend using platforms such as AWS, Azure solution with REST/GraphQL APIs.
- **Security Architecture:** Implement end-to-end encryption, secure authentication, and role-based access control.
- **Real-Time Communication:** Use MQTT or WebSockets for real-time lock state updates.

3.6.3 Mobile/Web App Enhancement

The current web interface can be transformed into a polished, user-friendly app for mainstream adoption.

- **Cross-Platform Support:** Build using Flutter, React Native, or Swift/Kotlin for Android and iOS platforms.
- **User Experience (UX):** Incorporate intuitive design, user onboarding, remote lock/unlock, and notification features.

3.6.4 Compliance and Security Certification

A production-ready smartlock must comply with legal standards and certifications.

- **Certifications:** Obtain CE, FCC, or other regional certifications for wireless and electronic devices.
- **Privacy Compliance:** Ensure GDPR/CCPA compliance with respect to user data handling and storage.

3.7 Detail of Design

The Smart Lock prototype consists of both hardware and software components, designed to provide secure, remote access control with user authentication and cloud connectivity.

3.7.1 System Overview

Our system integrates a physical keypad-based lock with Wi-Fi-enabled remote access through a mobile application. Users can authenticate using a PIN or biometric authentication (fingerprint or facial recognition), and manage access remotely.

- Users log in or create an account to access the smart lock.
- The main control screen provides lock/unlock buttons.
- Users can view and manage active PIN codes for secure access.
- The system communicates with a cloud database (Firestore) for real-time authentication and remote control.

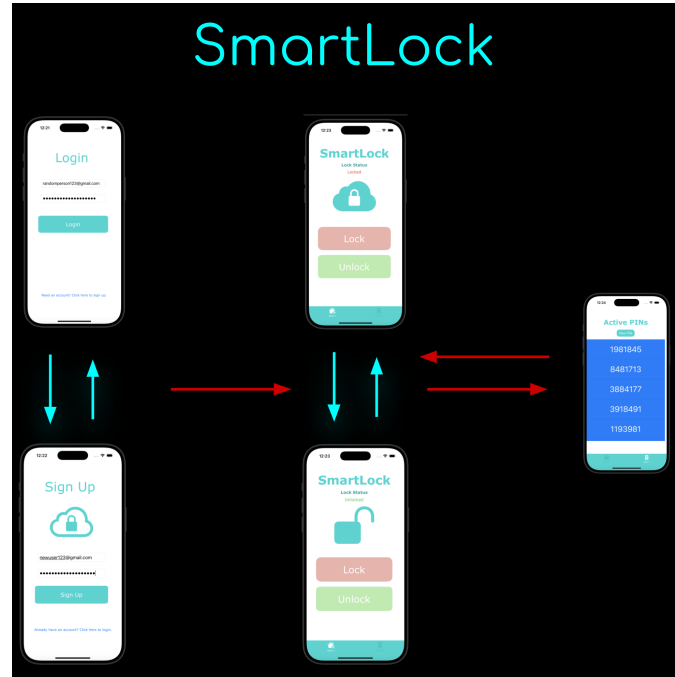


Figure 4: Mobile App Flowchart for Smart Lock System

3.7.2 Connectivity and Cloud Integration

The Smart Lock is cloud-connected, enabling users to manage access remotely via a mobile app. The system:

- Authenticates PIN entries locally for quick access.
- Processes biometric authentication for increased security.
- Syncs PIN codes with Firebase, allowing updates in real-time.
- Sends lock/unlock status to the cloud, ensuring remote monitoring.
- Logs user access, creating an audit trail for security.

Our cloud integration will have a many-to-many relationship with the mobile app and the lock. The mobile app will send requests to the cloud, so that there can be many users with multiple locks. Furthermore it allows the user to have control over which locks they want to unlock. To have this many-to-many relationship, we will create two different table collections. The first collection will be the user collection, which will have the user ID, email, and password. The second collection will be the lock collection, which will have the lock ID, lock name, and lock state. The user collection will have

a reference to the lock collection, so that we can have multiple locks for each user. This will allow us to have a many-to-many relationship between the user and the lock. Furthermore, we are planning to use a join table to connect the two collections. The join table will have the user ID and lock ID, so that we can have multiple locks for each user and multiple users for each lock. This will allow us to have a many-to-many relationship between the user and the lock. Additionally, to add more information about each lock we can create a history collection, which will have the lock ID, user ID, and timestamp. This will allow us to keep track of which user unlocked which lock and when. This will be useful for auditing purposes and for keeping track of who has access to which locks. It will also allow us to see if there are any unauthorized access attempts to the locks. The history collection will be linked to the lock collection and the user collection, so that we can easily access the information about each lock and each user. This will allow us to have a complete picture of the access control system and will help us to identify any potential security issues. One last collection we can have is a status collection, which will have the lock ID and the status of the lock. This will allow us to keep track of the status of each lock and to see if there are any issues with the locks. The status collection will be linked to the lock collection, so that we can easily access the information about each lock. This will allow us to have a complete picture of the access control system and will help us to identify any potential security issues. We will have a diagram shown below to show the relationship between the collections and how they are connected to each other.

3.7.3 Hardware Design

The Smart Lock hardware consists of a 3D printed casing, a keypad for PIN entry, a biometric scanner, a solenoid lock mechanism, and an ESP32-C3 microcontroller.

Here is the isometric view of the Smart Lock prototype. The top section will potentially include a biometric scanner, for fingerprint or facial recognition authentication, and a numeric keypad for PIN entry.

Key components:

- **Biometric Scanner:** Supports fingerprint or facial recognition for enhanced security
- **Numeric Keypad:** Users enter PIN codes as an alternative authentication method

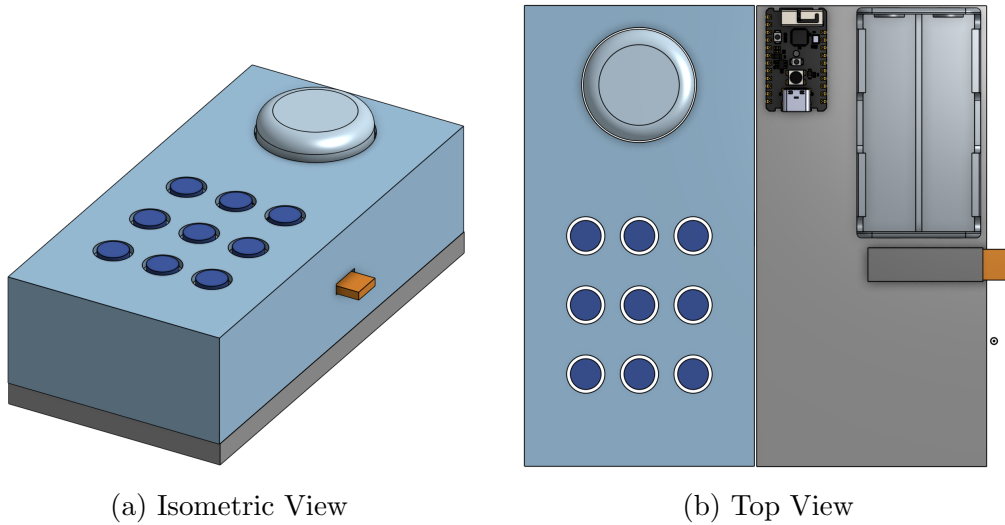


Figure 5: First lock design

- **Solenoid Lock:** Engages or disengages upon successful authentication
- **Microcontroller (ESP32-C3):** Handles user input, authentication, and connectivity
- **Rechargeable Battery:** Powers the system, ensuring reliability even in case of power outages

This design ensures compactness while maintaining modular repairability, allowing components to be easily accessed or replaced.

3.7.4 Life Cycle Assessment (LCA)

The life cycle of the Smart Lock was analyzed to ensure sustainability and environmental responsibility. Key aspects include:

Eco-Friendly Manufacturing

- Uses locally sourced materials to minimize transportation emissions
- Modular design allows for easy repair and component replacement, reducing e-waste
- Prioritizes recyclable materials in construction

Energy Efficiency

- Operates on a rechargeable lithium battery to extend lifespan
- Supports an optional solar charging module to reduce reliance on disposable batteries

Smarter Chemistry

- Lead-free soldering eliminates hazardous materials.
- Arsenic- and mercury-free components ensure environmental safety.
- PVC-free wiring minimizes harmful plastic waste.

End of Life Considerations

- Designed for easy disassembly, allowing parts to be recovered for recycling.
- Encourages users to recycle old locks through e-waste partnerships.

Commitment to Sustainability

- Uses stainless steel casing with the option for recycled metal integration.
- Packaging minimizes non-recyclable plastics, favoring biodegradable materials



Figure 6: Life Cycle Assessment of the Smart Lock

By integrating secure hardware, cloud-based software, and sustainable practices, the Smart Lock aims to offer a modern, environmentally responsible solution to traditional key-based access control.

3.8 Battery Power Management and Notification System

While the prototype operates on a continuous power source, a commercial SmartLock will use a secure, long-lasting battery to ensure usability during power outages and in locations without fixed power. Proper battery management involves monitoring charge levels, notifying users, and providing a safe and secure method for recharging or replacing the battery.

3.8.1 Battery Selection and Power Optimization

Selecting the right battery type and minimizing power consumption are critical for long-term operation.

- **Battery Type:** Use high-capacity rechargeable lithium-ion or lithium-polymer batteries capable of lasting several months on a single charge.
- **Low Power Modes:** Implement deep sleep and wake-on-interrupt features in the firmware to minimize power draw during idle periods.

3.8.2 Battery Monitoring and User Notification

Monitoring battery status and notifying users to ensure maintenance and avoid unexpected lockouts.

- **Voltage Monitoring Circuit:** Integrate a voltage divider or dedicated battery management gauge to continuously monitor battery levels.
- **Battery State Logging:** Periodically upload battery status to the cloud alongside status logs.
- **User Notifications:** Send low battery alerts via push notifications, email, or in-app messages when battery drops below a critical threshold (e.g., 30%).

3.8.3 Safe and Secure Battery Replacement or Recharging

Users must be able to service the battery safely without compromising lock security or usability.

- **Secure Battery Access:** Design an internal battery compartment accessible only when the lock is in an unlocked state or with a secure override mechanism.

-
- **Rechargeable Design:** Provide a USB-C or magnetic charging port for easy recharging without disassembling the lock.
 - **Tamper Detection:** Include log entries if the battery compartment is accessed, and notify the owner.
 - **Backup Power Support:** Consider integrating a backup battery to retain state data during battery swap, or integrate a way to reset lock data upon battery replacement or recharge.

3.9 Ownership and Access Control

Managing ownership and granting user access are critical to ensuring that SmartLock devices remain secure while allowing for flexible sharing. This system must allow primary users to delegate lock access to other users, while ensuring that the delegated users have the appropriate permissions.

3.9.1 Ownership Model

Each SmartLock must have a primary owner who holds full control over the device and its permissions.

- **Lock Registration:** When a new lock is initialized, the registering user becomes the owner.
- **Ownership Table:** Store ownership information in the database alongside lock records (e.g., `lock_id`, `owner_id`).

3.9.2 Access Delegation and Permissions

Primary owners can grant access to other users, allowing them to interact with specific locks under defined conditions.

- **Access Table:** Maintain a table or collection with fields such as `lock_id`, `user_id`, `access_level` (view, control), `granted_by`, and `expiration` if necessary.
- **Access Rights Enforcement:** Backend logic and security rules should enforce permissions before allowing user actions.
- **Revocation and Expiration:** Include features for the owner to revoke access at any time or set expiration timers for temporary access.

3.9.3 User Interface for Access Management

The frontend application should allow users to manage and review access settings.

- **Access Dashboard:** Display which users currently have access to a given lock and their permission levels.
- **Invite Flow:** Enable owners to invite users via email or user ID, specifying which lock(s) they can access.

3.10 History and Status Logging

Logging improves security, aids in diagnostics, and enhances transparency for users. The SmartLock system should log both user actions and technical status changes.

3.10.1 History Log of User Actions

Tracking user interactions with locks helps in audits, diagnostics, and usage insights.

- **Log Entry Fields:** Include `timestamp`, `user_id`, `lock_id`, and `action` (e.g., lock, unlock, access granted).
- **Database:** Automatically write to the history log whenever a user successfully executes an action.
- **User Access to Logs:** Allow users to view logs for locks they own or have access to.

3.10.2 Status Log for Lock Health Monitoring

Status logs are essential for monitoring device reliability and connectivity.

- **Microcontroller Event Logging:** Program the microcontroller to push logs to Cloud structure or a backend data on significant events (e.g., Wi-Fi reconnect, battery low).
- **Technical Dashboard:** Visualize connection stability and firmware state for each lock.

3.10.3 Action Confirmation and Notifications

Users must be notified when actions are completed to reinforce trust and responsiveness.

- **Action Acknowledgement:** ESP32 should confirm action execution back to Cloud database (e.g., “locked: true”).
- **Notification Service:** Use real-time updates to the mobile app or web interface.
- **Failure Alerts:** Alert users if an action fails, with details such as connectivity issues or mechanical faults.

3.11 Initial Installation and Setup Procedure

The following outlines the essential steps required to ensure a easy setup and fast experience for end users upon receiving the SmartLock. This section will guide engineers in implementing a user-friendly lock with secure technical integration. The procedure includes steps from unboxing to successful WiFi connection via Bluetooth communication.

3.11.1 Step 1: Unboxing and Physical Inspection

Upon arrival of the SmartLock device, users are expected to perform basic inspection of the package contents.

- Ensure all components are present:
 - SmartLock main unit
 - Mounting bracket and screws
 - Quick-start guide with QR Code at Front Page (printed)
- Verify the unit is free of visible defects or damage.
- Charge the unit if not pre-charged (optional: LED will indicate low battery if charging is required).

3.11.2 Step 2: Mobile Application Download

Before any configuration can begin, users must download the official SmartLock companion app. The app serves as the bridge between the mobile device and the lock, handling both Bluetooth communication and network provisioning.

- Available on iOS (App Store) and Android (Google Play Store).
- App name: **SmartLock Home**.
- Upon opening the app, users must grant Bluetooth and location permissions as required by the mobile OS.

3.11.3 Step 3: Initial Bluetooth Pairing

Once the app is installed, users are prompted to begin the pairing process via Bluetooth. This enables the mobile device to establish a secure connection with the SmartLock unit.

-
- The SmartLock should be placed in pairing mode. This is triggered automatically on first boot, indicated by a flashing blue LED.
 - The app will scan for nearby devices. The SmartLock will be listed as **SmartLock-XXXX**, where **XXXX** represents the last 4 digits of the device's MAC address.
 - Once selected, the user will initiate pairing. This may include a confirmation prompt or PIN verification depending on device configuration.
 - A secure Bluetooth connection will be established upon successful pairing.

3.11.4 Step 4: WiFi Provisioning

After a secure Bluetooth channel has been established, the next phase involves transferring the user's WiFi credentials to the SmartLock for permanent network connectivity.

- The mobile app will prompt the user to select a WiFi Network.
- Users will enter the SSID and password of their home network.
- Credentials are encrypted and transmitted to the SmartLock via Bluetooth.
- The SmartLock attempts to connect to the provided network. Connection success or failure will be relayed to the app.
- Upon successful connection, the LED will turn solid green for 5 seconds, and then turn off to conserve power.

3.11.5 Final Notes and Engineer Considerations

- Ensure Bluetooth Low Energy (BLE) is used for communication to optimize power efficiency.
- All user-facing steps should include error handling and clear guidance for troubleshooting.
- Logs of pairing attempts, WiFi status, and user interactions should be stored locally for support diagnostics.

This setup process is designed to offer an intuitive user experience while providing the flexibility and security required for a modern smart home device. Engineers should maintain this structure while accounting for evolving mobile OS standards and networking best practices.

4 Final Design Outline

4.1 Introduction

This document outlines the final design of the Smartlock system, detailing its major components and their interactions. The system leverages cloud infrastructure, secure authentication, wireless connectivity, and a robust data model to provide secure access.

4.2 Cloud Infrastructure (AWS)

The Smartlock system utilizes Amazon Web Services (AWS) as its cloud service. We will use AWS for the following:

- Each SmartLock owner will have a unique AWS account and will have access to any other locks purchased through their personal cloud service.
- No personal information will be stored on the cloud. The only information stored will be the lock(s) ID(s), the owner ID, the PIN codes, a history log, and other user IDs that the owner authorizes. Nothing else will be stored on the cloud.

AWS makes HTTPS easy and secure to use. First, you get a website certificate through AWS Certificate Manager, which proves your site is who it says it is and automatically renews itself. You then attach that certificate to our app. When someone visits our app, their iOS device checks the certificate and, if it's valid, both sides agree on a secret key to encrypt all data sent back and forth. Behind the scenes, AWS uses its identity controls to keep your certificates safe and logs everything so you can see what's happening. This way, you can trust that your login data and pin codes are secure.

Benefits of AWS Compared to Other Cloud Infrastructures For Smartlock

AWS offers several advantages over other cloud solutions such as PostgreSQL or Google Firestore:

- **Comprehensive Service Integration:** AWS provides a wide range of services (compute, storage, authentication, IoT, etc.) that work seamlessly together.

-
- **Scalability and Reliability:** AWS infrastructure is designed for automatic scaling to handle varying workloads, and in our case multiple users for multiple locks.
 - **Managed Security:** Built-in https security features.
 - **Automated Certificate Management:** AWS Certificate Manager simplifies HTTPS setup and renewal, reducing operational overhead.
 - **Unified Monitoring and Logging:** Built-in tools for monitoring and logging data from app or smartlock.
 - **Flexible Pricing Models:** Pay-as-you-go and reserved pricing options to optimize costs.

4.3 PIN Code Access

Each smartlock supports PIN code access for users. The PIN code system details these features.

One-Time PIN Codes

- One-time Pin Codes be generated on the app and sent to the cloud.
- Transmitted securely to the lock device through AWS IoT services.
- The lock verifies the code against the cloud database.
- One-time PIN codes are valid for a single use and expire after a set time period (e.g., 5-60 minutes).
- After the code is used, it is blacklisted in the cloud, app, and lock and cannot be reused.
- After code is used, time of use is logged in the cloud database.

Emergency PIN Codes

- Access to emergency PIN codes is restricted to the owner and authorized users, and will have an extra layer of security to access these pins.
- The codes are generated on app and sent to the cloud upon Wifi connection.

-
- Extra emergency PIN codes can be generated by the owner and authorized users
 - After an emergency PIN code is used, that pin will be blacklisted from the app, cloud, and lock.
 - All emergency PIN codes are refreshed upon WiFi reconnection.
 - Pins are validated locally to allow or deny access. Pin access is can be used even without internet connectivity to ensure that any user can still get into their own home.

PIN codes can be assigned, revoked, or updated remotely by authorized users through the cloud interface. After multiple failed attempts, the lock will notify the user and enter a temporary lockout state to prevent brute-force attacks. The cloud and app logs all PIN code activities, including successful and failed attempts.

4.4 WiFi Connection

Smartlocks connect to the internet via WiFi, through a WiFi accesspoint model. This allows for the user to smoothly connect their smartlock to their home network.

Smartlock WiFi Features

- SmartLock needs WiFi in order to initially connect to the cloud.
- WiFi SSID and password are not stored in the lock, cloud, or app ensuring security.
- After connecting smartlock to WiFi through accesspoint mode and entering your smartlock's personal API key, the lock will be operational.

The WiFi module is configured during device setup and supports secure protocols (e.g., WPA2) to protect network traffic. If a user want's to change their WiFi network, they can do so through the same process as the initial setup. The lock will enter accesspoint mode, and the user can connect to the lock's WiFi network to change the SSID and password.

4.5 Many-to-Many Relationship: Locks and Users

The system supports a many-to-many relationship between users and locks:

- Each user can have access to multiple locks
- Each lock can be accessed by multiple users
- Access rights are managed in the cloud database, with mapping tables linking users and locks
- Permissions can be updated dynamically, supporting temporary or permanent access

This flexible model allows for efficient management of access in environments such as offices, apartments, or shared facilities.

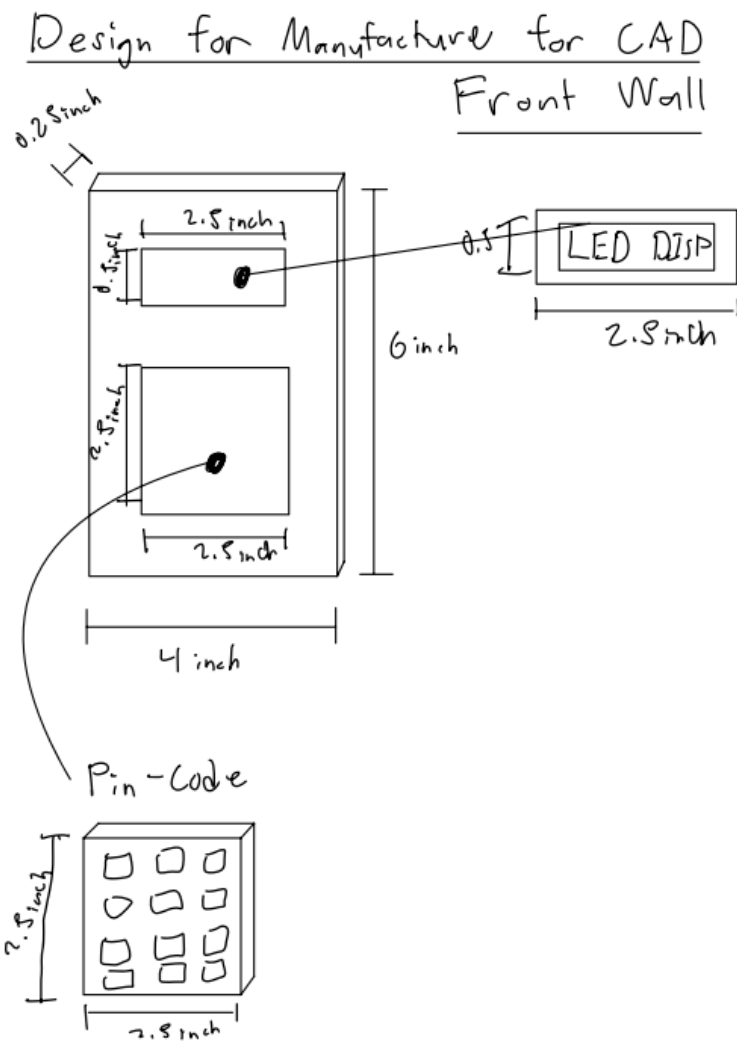
4.6 Conclusion

The Smartlock system integrates cloud services, secure authentication, wireless connectivity, and a flexible data model to deliver a robust and scalable access control solution.

5 Economic Analysis

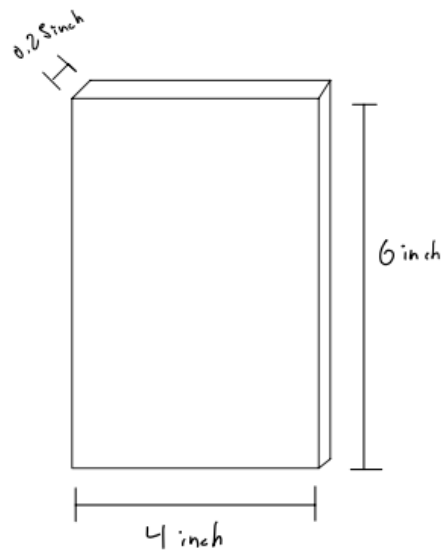
Below we have the images for DFM, DFA, and Life Cycle Assessment.

DFM & DFA

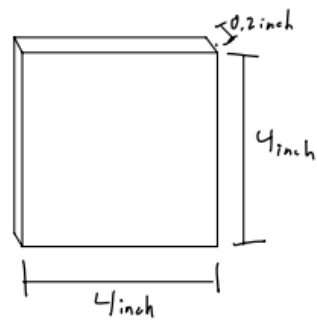


Design for Manufacture for CAD

Side/Back Walls



Top/Bottom Walls



5.1 Outstanding Issues

Hardware/Physical

- Need to find solenoid lock compatible with ESP32-C3 for testing
- Need to make sure generated and customized PINs work with custom hardware authorization/verification within physical keypad
- Need to find battery pack with lifespan of 1.5yrs+
- Need to find keypad that can be wired securely to ESP32-C3 in lock without ruining integrity of lock

Software/Online

- Programming correct working PIN interface storage with secure random generation
- ESP32-C3 pin-keypad communication and input signal processing

6 Appendices

6.1 Scope

Firestore Mobile Connectivity Test

Test Goals and Purpose

- Ensure that an IOS device can securely connect and push data to the Firestore servers.
- What happens when there may be bad network connection
- Set a certain time frame for the pin to be qualified for

Parameters

- 1/0 LOCK_STATE necessary for defining lock-state, 1 is unlocked, 0 is locked.
- 7 digit QUALIFIED_PIN list on an allow list for physical keypad on SmartLock.
- Make sure what data we send from phone is on firestore
- Provide time frame as input for start and expiration time for pin code

Expectations of Test

- There will be no issue with pushing a simple piece of data for LOCK_STATE to the firestore servers, clicking lock will properly push a 0 and clicking unlock will store a 1. Additionally, the pin list composed of generated and custom user pins will also have no trouble being communicated with Firestore servers to later be fetched by the physical device as an additional mechanism for locking and unlocking.
- Testing the time frame of the pin code, ensure only the time when valid to use the pin. Otherwise other times should be invalid.

6.1.1 Lock Hardware Functionality Test

Test Goals and Purpose

- Ensure that an ESP32-C3 device can securely connect and retrieve data from the Firebase Firestore servers.
- Ensure that the ESP32-C3 board can correctly interact with hardware to enable mechanisms for locking/unlocking.
- Ensure that invalid codes should not release bolt, while valid codes should.

Parameters

- 1/0 LOCK_STATE necessary for defining lock-state, 0 is locked, 1 is unlocked. Retrieved from firebase.
- 7 digit QUALIFIED_PIN list on an allow list for physical keypad on SmartLock.

Expectations of Test

- The locking mechanism will respond appropriately after retrieving from Firestore, and the bolt will trigger when the parameter is set to 0 and likewise release given a 1 value. Additionally, the bolt will release given a valid input from the QUALIFIED_PIN list and reject other non-valid input pins.
- Expecting the lock to not unlock after a code is expired, only unlock when it is still valid in a time frame.

6.1.2 Multi-User & Concurrency Test (Low Priority)

Test Goals and Purpose

- Ensure that multiple users can simultaneously interact with the smart lock through the mobile app and keypad without causing system conflicts.
- Verify that Firebase can handle concurrent updates to the lock state and PIN list without failure.
- Ensure the lock responds appropriately when multiple unlock requests are received from different users.

Parameters

- Simultaneous requests from different mobile devices attempting to unlock/lock the door.
- Simultaneous PIN entries from different users on the physical keypad and mobile app.

Expectations of Test

- Multiple users attempting to unlock the door at the same time should not cause conflicts or undefined behavior.
- The lock should process and execute the most recent valid request, ensuring no delays or duplicate actions.
- System logs should correctly track each user's request, ensuring accountability and reliable data collection for user feedback.

Administrative Details

6.1.3 Firebase Mobile Connectivity Test

Date & Location: Mar 12, 2025 in class

Conducting Test: Jackson Kennedy, Adam Wu, Nathaniel Laurente, Neena Nguyen, and Professor Harrison.

6.1.4 Lock Hardware Functionality Test

Date & Location: Mar 31, 2025 in class

Conducting Test: Jackson Kennedy, Adam Wu, Nathaniel Laurente, Neena Nguyen, and Professor Harrison.

6.1.5 Multi-User & Concurrency Test

Date & Location: April 18, 2025 in class

Conducting Test: Jackson Kennedy, Adam Wu, Nathaniel Laurente, Neena Nguyen, and Professor Harrison.

Design of Experiment

6.1.6 Firebase Mobile Connectivity Test

Testing method type: Functional Testing, does the simple job of sending unlock or lock to firebase server, also sending pins.

Test apparatus: Firebase Cloud service

Independent Variables: Lock Status, Acceptable Pins

Dependent Variables: N/A

Number of factors:

Sampling Procedures: 2 samples of observing Firestore for 1/0 LOCK_STATE. 2 randomly generated PINs and 1 custom-user pin.

6.1.7 Lock Hardware Functionality Test

Testing method type: When phone sends data for unlock or lock onto firebase, the lock should quickly retrieve information and respond accordingly. The purpose of this is to make sure the lock receives the data fast enough so that people waiting outside the lock does not wait too long for door to unlock. This should also be similar for the pin code case.

Test apparatus: Utilize ESP32-C3 alongside a Solenoid door lock.

Independent Variables: Make sure that the ESP32-C3 can read a simple "Hello world" on the server for basic testing.

Dependent Variables: Everytime we are generating a new code from phone to firebase, the ESP32-C3 should also be able to retrieve those codes as valid codes immediately.

Number of factors: N/A factors considered in this moment

Sampling Procedures: Samples obtained from servers for LOCK_STATE, only 0 or 1. Then we have samples for pin codes as well from Firebase.

6.1.8 Multi-User & Concurrency Test

Testing method type: Multiple phones will send an unlock signal from phone at nearly the exact same time and see if any undefined behavior occurs. Send a lock signal and an unlock signal at the same time to test if we can prioritize actions that came first, or lock the app after an action from one device has been made.

Test apparatus: Mobile Device along with ESP32-C3.

Independent Variables: Make sure that the ESP32-C3 can read a simple "Hello world" on the server for basic testing.

Dependent Variables: ESP32-C3 behavior after multiple devices send a signal at the same time or immediately one after the other.

Number of factors: 2 factors to be considered

Sampling Procedures: Samples obtained from servers for LOCK_STATE, only 0 or 1. Determine if multiple signals causes undefined or expected behavior.

Testing Procedures

6.1.9 Safety Precautions

Safety precautions for our project includes not having fast and forceful locks that can hurt individuals.

6.1.10 Data Collection Method

Collect data from the phone providing the custom pins onto firebase, as well as collecting data for locking or unlocking. We will also note down which users are grouped together to be able to use the same pin for uncocking door.

6.1.11 Observation of External Factors

Some external factors that could make our product not function properly could include:

- Bad network connection from phone or from ESP32-C3
- Firebase not loading properly

6.2 Test Results

6.2.1 From Test Plan

We will be testing our Firebase mobile connectivity, Esp32 hardware functionality, and multi-user and concurrency. For each of the tests, I will provide step by step instructions.

In the first test for Firebase mobile connectivity, we will test the connection between the mobile app and the firestore database. We will first build the mobile app and connect the firebase API keys to the app. Then with the mobile app, we will test the connection to the firestore database by sending a value to the database document. We can check if this succeeds by checking the firebase console and seeing if the value is updated.

In the second test for the ESP32 hardware functionality, we will test the connection between the ESP32 and wifi. Once connected to wifi, we will test the connection to the firestore database by using the API keys provided by firebase. Then we will create a document in the firestore database and use the ESP32 to fetch the document data. We can check if this succeeds by checking the fetched data is the same as the data in the firestore database. Also from the hardware functionality test, we are going to test the functionality of the solenoid lock, making sure that we can send a signal to unlock and lock the solenoid lock. One more test that we will need to take into consideration is the keypad functionality. We will test the functionality of the keypad by making sure that we can read the input from the keypad and send the input to the ESP32.

In the third test for multi-user and concurrency, we will test the connection between multiple mobile apps and the firestore database. We will first build the mobile app and connect the firebase API keys to the app. Then with the mobile app, we will test the connection to the firestore database by sending a value to the database document. We can check if this succeeds by checking the firebase console and seeing if the value is updated. We will then test the connection between multiple mobile apps and the firestore database by sending values to the database document from multiple mobile apps. We

can check if this succeeds by checking the firebase console and seeing if the values are updated.

6.2.2 Test Result Summary Table

Objective (Target)	Result	Met?	Discussion
Firebase Connectivity	Appropriate 1/0	N/A	Half this test done - explain how lock/unlock work but haven't tested PINs yet.
Lock Hardware Functionality	N/A - <i>Mar 31, 2025</i>	N/A	While this test has not been conducted yet, we suspect we will pass this test as the Firebase - Firestore lock/unlock was already successful, and the appropriate data was well-received by the ESP32-C3. All that remains is to ensure that the solenoid lock responds properly to an input signal, as well as compares and responds accordingly to valid/invalid codes.
Concurrency	N/A - <i>April 18, 2025</i>	N/A	While this test has not been conducted yet, we suspect that we will pass this test. Firebase should be able to handle multiple calls close to each other, as well as transmit this signal in the appropriate order. The Firestore database updated extremely quickly in real-time and shouldn't have trouble with concurrent calls. The PIN entries from different users also shouldn't cause conflict, and the door should unlock/lock as intended.

7 Manufactured Product Testing

Common Scenarios (Tests 1–5)

1. Valid PIN Entry Test

Test Goals and Purpose

- Check that entering a correct 4-digit PIN immediately unlocks the door.
- Make sure the system records the success.

How We Test It

- Enter a valid PIN from the list.
- Observe if LOCK_STATE is set to 1 in Firestore.

Expectations of Test

- Bolt pulls back within 0.5 s of entering PIN.
- Firestore shows LOCK_STATE = 1.
- No errors in the logs.

2. Invalid PIN Lockout Test

Test Goals and Purpose

- Ensure three wrong PINs in 30 s locks out further tries.
- Check that an alarm or light indicates lockout.

How We Test It

- Enter three different invalid PINs quickly.
- Check LOCKOUT_COUNT and look for alarm.

Expectations of Test

- No PIN entries accepted for 60 s.
- Alarm or LED shows lockout.
- Firestore logs all failed attempts and the lockout event.

3. Emergency PIN Test

Test Goals and Purpose

- Confirm that a special emergency PIN still works after lockout.
- Make sure normal lockout resets after emergency use.

How We Test It

- Lock out with wrong PINs, then enter the emergency PIN.
- Check Firestore for an EMERGENCY_UNLOCK record.

Expectations of Test

- Door unlocks right away.
- Firestore logs the emergency unlock.
- Lockout counter resets.

4. OTP Entry Test

Test Goals and Purpose

- Verify that a one-time code from the app unlocks the door.
- Make sure the code is sent and checked securely.

How We Test It

- Request an OTP thats valid for 5 min.
- Enter it on the keypad.

Expectations of Test

- Firestore stores the OTP and its expiry time.
- Valid OTP unlocks the door.
- Invalid or expired OTP is rejected.

5. OTP Expiry Test

Test Goals and Purpose

- Check that entering the OTP after it expires does not unlock.
- Show a message saying “OTP expired.”

How We Test It

- Wait until after 5 min expiry.
- Enter the same OTP on the keypad.

Expectations of Test

- Door stays locked.
- Firestore logs “OTP_EXPIRED.”
- App shows “OTP expired.”

Less Common Scenarios (Tests 31–35)

31. Multi-factor Auth Proximity Test

Test Goals and Purpose

- Make sure PIN + Bluetooth both work together for unlock.
- See what happens if Bluetooth disconnects during entry.

How We Test It

- Enter a valid PIN while moving phone out of Bluetooth range.
- Check logs for separate PIN and BLE events.

Expectations of Test

- Door only opens if both PIN and BLE are OK.
- Firestore shows which factor failed if it doesn't open.

32. Time-zone Mismatch Test

Test Goals and Purpose

- Test OTP and scheduled lock when phone and lock have different clocks.
- Make sure times sync up correctly.

How We Test It

- Set phone to PST, lock to UTC.
- Generate OTP and try it.

Expectations of Test

- OTP still works within the correct window.
- Auto-lock happens at the right local time.

33. DST Transition Auto-lock Test

Test Goals and Purpose

- Check auto-lock around daylight-saving time changes.

How We Test It

- Schedule auto-lock at 2 AM on DST change day.

Expectations of Test

- Only one auto-lock event, no skips or doubles.

34. Admin vs. Guest Role Change Test

Test Goals and Purpose

- Make sure changing a PINs role (admin/guest) updates immediately.

How We Test It

- Change PIN role in Firestore and try using both roles.

Expectations of Test

- Lock enforces the new role right away.

35. BLE Range Boundary Test

Test Goals and Purpose

- See if Bluetooth unlock works at the edge of its range (5 m).

How We Test It

- Move phone from 1 m to 6 m away in 0.5 m steps.

Expectations of Test

- Unlocks when signal is strong enough, fails when its too weak.

Rare Scenarios (Tests 61–65)

61. Cosmic Ray Bit-Flip Test

Test Goals and Purpose

- Simulate random bit errors in memory to check stability.

How We Test It

- Inject bit-flips in RAM/flash every 10 s.

Expectations of Test

- System catches or corrects errors, stays stable.

62. Lightning Surge Test

Test Goals and Purpose

- Test surge protection by sending a voltage spike.

How We Test It

- Apply a 1 kV pulse on the 12 V input for 1 μ s.

Expectations of Test

- No damage, system recovers normally.

63. Seismic Vibration Test

Test Goals and Purpose

- Make sure the lock survives shaking and vibration.

How We Test It

- Run a 5–50 Hz vibration sweep at 0.5 g for 10 min.

Expectations of Test

- No loose parts, sensors stay stable, lock/unlock still works.

64. Extreme Cold Operation Test

Test Goals and Purpose

- Verify lock works at minus 40 °C.

How We Test It

- Put the lock in a minus 40 °C chamber for 2 h and do 10 cycles.

Expectations of Test

- Bolt moves on time, no cracking, battery holds up.

65. Extreme Heat Operation Test

Test Goals and Purpose

- Check lock at +85 °C over 2 h.

How We Test It

- Run lock/unlock every 5 min in the heat chamber.

Expectations of Test

- MCU stays safe, no shutdowns, data intact.