RFID-Based Secure Chatroom Final Report

1. Project Introduction (Summary):

This project is an RFID-based secure chatroom system. Users scan a preloaded RFID tag with their mobile device (we used an iPhone 14) to gain access into the chatroom. Once they are in, they will be prompted to enter a password that is unique to them. After the correct password is entered, the user is granted entry into the chatroom and can send and receive encrypted messages until they exit the chatroom. Overall, this system provides secure communication with AES-encrypted messages, ensuring privacy and controlled access.

2. Tools:

- Glitch.com
- GitHub
- VS Code
- SQLite3
- Node.js
- NFC Tools
- Languages: JavaScript, HTML, Python, SQL
- Hardware: N-tag 215, iPhone 14

The main framework we are using is Node.js. This is a run-time environment that is a lightweight framework. Before we developed our chatroom, we were working with STM/Backend, we tried to fork this to meet our needs but ran into issues with cryptography. We tried deploying the project on our local machine but ran into technical issues. We realized by using Glitch.com that this managed to resolve many of our issues.

3. Use Cases:

Use Case 1 (Fully Dressed): Access Chatroom

Use Case Name:	*Access Chatroom*
Scope:	*RFID Chatroom*
Level:	*User goal*
Primary Actor:	*Chatroom Client*
Stakeholders and Interests:	- Chatroom Client: Wants to quickly and securely access system. - System Administrators: Wants to receive notification of system access request.
Preconditions:	- Chatroom Client has credentials within database. - Chatroom Client has RFID tag and passphrase. - Chatroom Client has a smart device with internet access and RFID reader.
Success Guarantee:	- Chatroom is shown and becomes available to send messages. - System access request is generated and sent to the System Administrator.
Main Success Scenario:	1. Chatroom Client wants to access chatroom with RFID tag. 2. Chatroom Client scans RFID tag with smart device. 3. RFID Tag prompts client to input passphrase to access Chatroom Log-In page. 4. Chatroom Client enters passphrase. 5. RFID Tag verifies passphrase and if valid, will populate the reader interface with a link. 6. Chatroom Client will click the link to display Chatroom Log-In page. 7. Chatroom Log-In page prompts client to enter password with the account while the username is automatically populated from the RFID Tag. 8. Chatroom Client enters password. 9. Chatroom Log-In page verifies account credentials and if valid, the server creates a thread for the client and opens the chatroom interface. 10. Chatroom Client can send and receive messages from other clients.
Extensions:	4a. At any time, System fails: 1. System shuts down server. 2. System disables link to access Log-In page. 5a. RFID Tag detects incorrect passphrase. 1. Reader interface displays error. 2. Return to Step 3. 7a. Chatroom Log-In page does not detect username. 1. Display error page. 2. Generate error report and send it to the System Administrator. 3. Return to Step 1. 9a. Chatroom detects incorrect password. 1. Reader interface displays error. 2. If the number of incorrect password detections is more than 3. 2.1. Generate invalid user report and send it to the System Administrator. 2.2. Close Chatroom Log-In page. 2.3. Return to Step 1. 3. If the number of incorrect password detections is less than 3.

	3.1. Return to Step 7.
Special Requirements:	- Text must be designed for touchscreen UI and standard desktop. - User verification must have a response within 10 seconds.
Technology and Data Variations List:	- Passphrase entered by touchscreen or keyboard. - Password entered by touchscreen or keyboard. - Time stamps will be in local device time.
Frequency of Occurrence:	Could be nearly continuous as it occurs every time anyone wants to access the system.
Open Issues:	None

Use Case 2 (Fully Dressed): Send Message

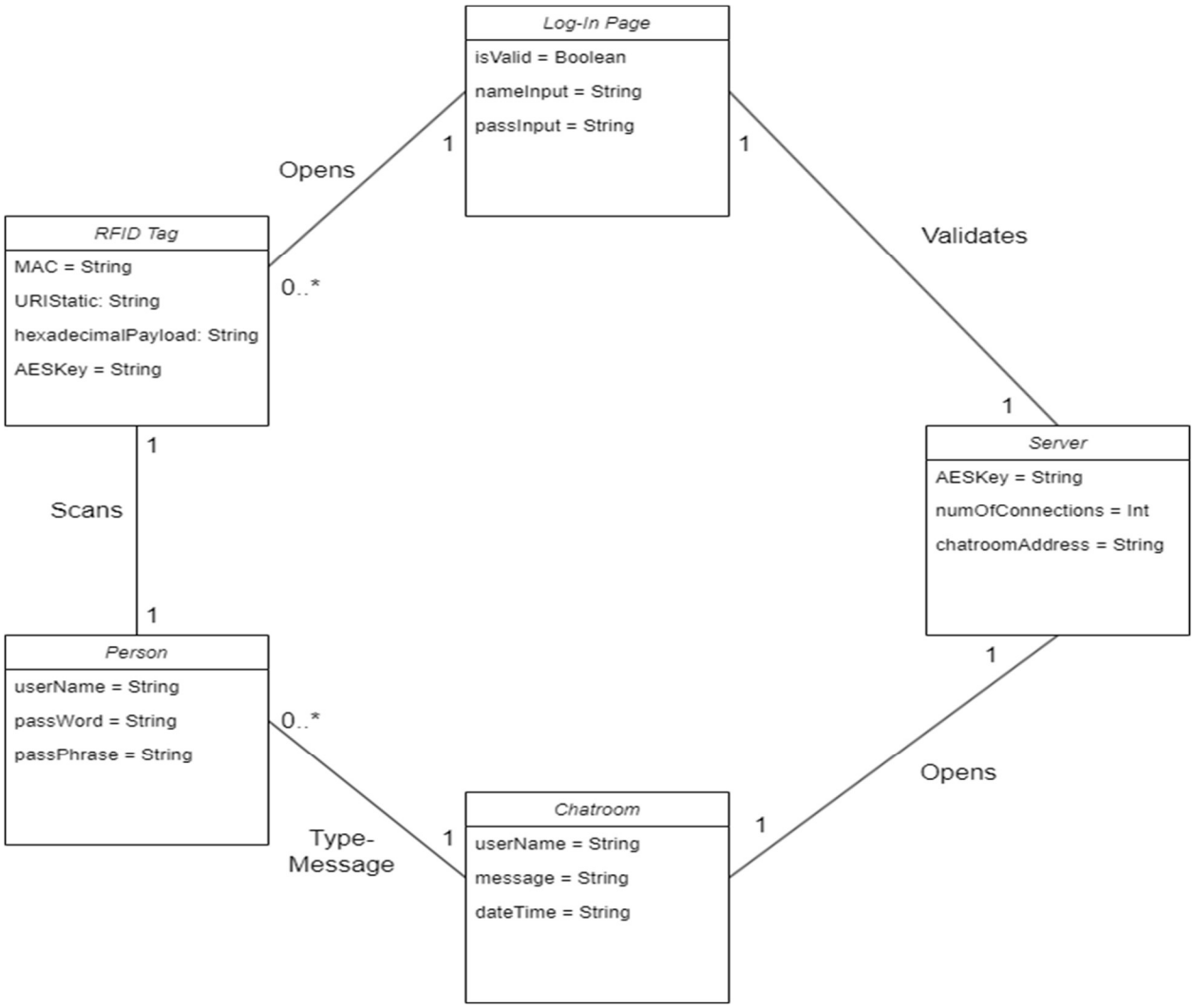
Use Case Name:	*Send Message*
Scope:	*RFID Chatroom*
Level:	*User goal*
Primary Actor:	*Chatroom Client*
Stakeholders and Interests:	- Chatroom Client: Wants to be able to send and receive messages
Preconditions:	- Chatroom Client has gained access to chatroom with RFID tag and password
Success Guarantee:	- Chatroom is shown and becomes available to send messages. - System access request is generated and sent to the System Administrator.
Main Success Scenario:	1. Chatroom Client inputs message into send box. 2. Chatroom Client hits send button or presses enter. 3. Unencrypted messages appear for own user. 4. Message is encrypted using AES key. 5. Encrypted messages are sent to the server. 6. Server broadcasts message to all other users.
Extensions:	2a. At any point user sends message that is too long, it will show *Error message* in box and user must re-enter message before sending
Special Requirements:	- Message sent must be received in a way that is able to be read by other users. - Message is encrypted until it reaches its endpoint.
Technology List:	- Text must be entered by touchscreen or keyboard
Frequency of Occurrence:	Could be nearly continuous as it occurs every time anyone wants to send a message.
Open Issues:	None

Use Case 3 (Fully Dressed): Close Chatroom

Use Case Name:	*Close Chatroom*
Scope:	*RFID Chatroom*
Level:	*User goal*
Primary Actor:	*Chatroom Client*
Stakeholders and Interests:	- Chatroom Client: Wants to be able to close chatroom
Preconditions:	- Chatroom Client has gained access to chatroom with RFID tag and password
Success Guarantee:	- Chatroom Client presses the close window button within the Chatroom page. - The server closes the connection for the client. - Displays exit message for all other threads and closes the interface for the client.
Main Success Scenario:	1. Chatroom Client exits presses close on window. 2. Server closes thread. 3. Local variable holding AES key is destroyed. 4. Remove user session and user from active lists. 5. Server sends message to everyone else in the chatroom that the user has left the chatroom.
Extensions:	1a. At any point the user loses internet access, they will be logged out of the chatroom. 1b. At any point the user refreshes page, they will be logged out of the chatroom. 1c. At any point the user leaves page, they will be logged out of the chatroom.
Special Requirements:	- Consider web browsing on mobile devices.
Technology List:	None
Frequency of Occurrence:	User is only able to exit the chatroom once per login.
Open Issues:	None

4. Domain Model

Domain Model 1: Overview of System



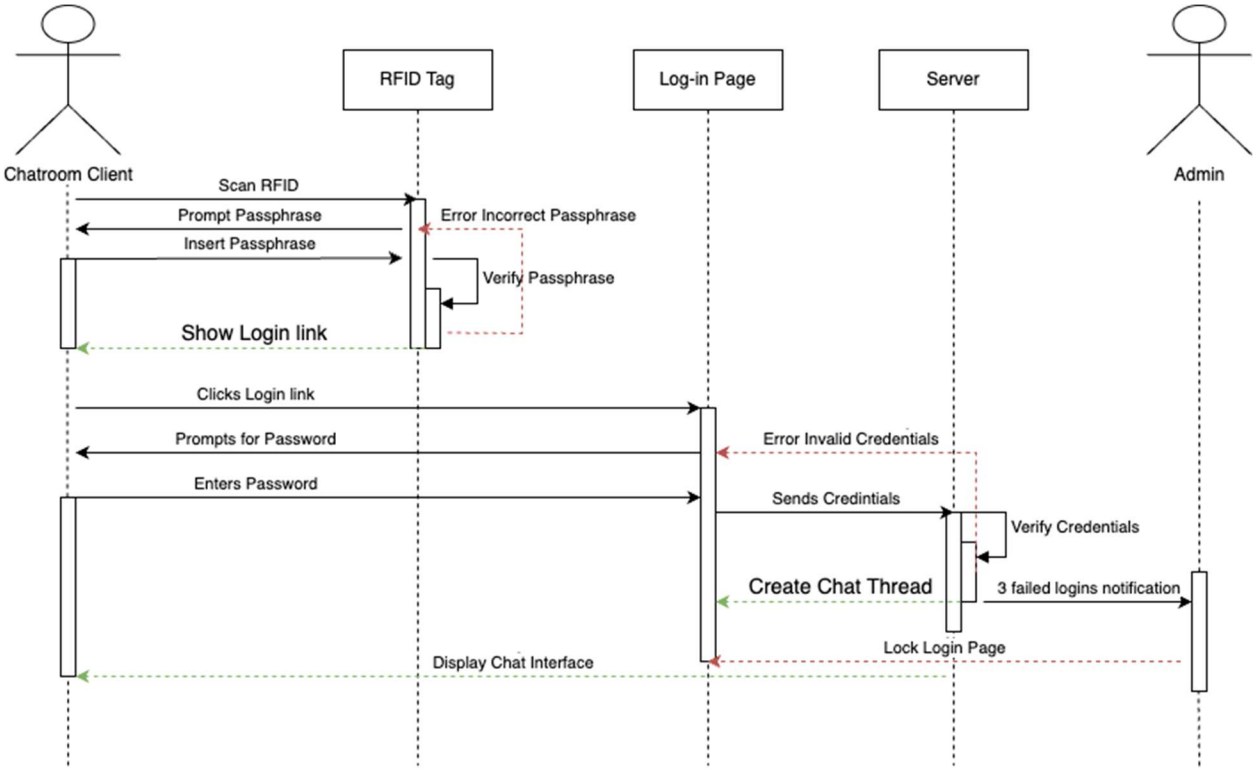
Our domain model has improved in both functionality and structure since it was first developed in class. Initially, the classes were *Person*, *RFID Tag, Server*, *Chatroom*, and *Database*. However, the Database is not meant to be shown in this model as it is a software artifact. Another change was the addition of the *Login Page,* which is an essential part of the program. One of the major aspects we missed when creating a domain model in class was the associations between the classes. As we progressed, we realized the need to clarify the interactions between these classes, by specifying how they connect. Improving attributes, particularly in terms of authentication and encryption, also took place. Important steps included

the addition of attributes such as “AESKey” and methods to handle encryption and decryption of messages.

Ultimately, our domain model now better supports the product's vision of secure, authenticated communication, with each class working together to ensure user verification and real-time secure message transmission in the chatroom.

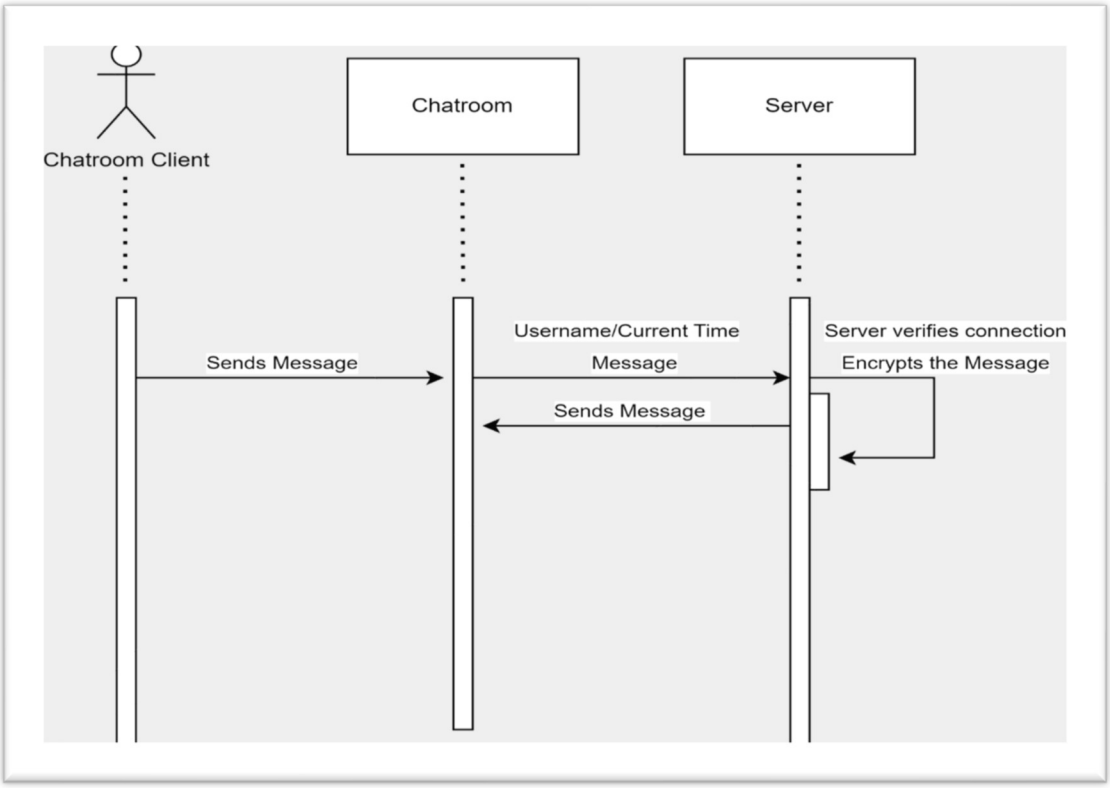
5. Sequence Diagrams

Sequence Diagram 1: Access Chatroom



This sequence diagram shows the process of accessing a chatroom using an RFID system. The Chatroom Client scans the RFID with their device and is prompted with a link to the Chatroom Log-In Page. The client then enters their password, and the Server verifies the credentials. If valid, the client gains access to the chatroom and can start sending messages. If the passphrase or password is incorrect, the system displays an error. This ensures the process is secure while providing a clear flow for valid and invalid cases.

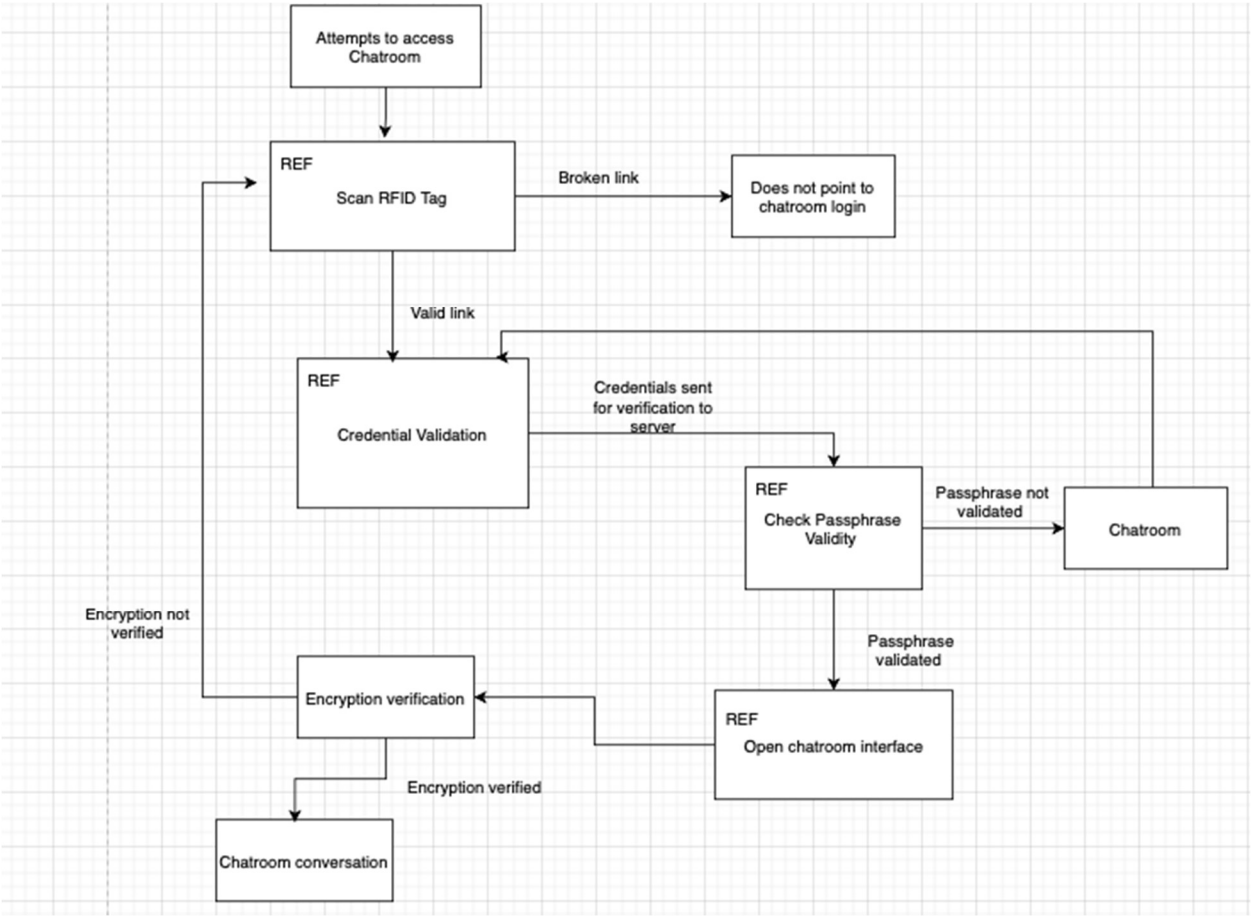
Sequence Diagram 2: Send Message



This sequence diagram illustrates the interaction involved in the "Send Message" process between the Chatroom Client, Chatroom, and Server. The process begins with the Chatroom Client sending a message to the Server. The Server verifies the client's connection, retrieves the username and current time, encrypts the message using an AES key, and sends the encrypted message to other connected threads (represented by the Chatroom). Each step in the interaction is shown with synchronous message passing, ensuring the message is validated, encrypted, and distributed correctly before the process is complete.

Key design decisions include using solid arrows to represent the synchronous nature of the communication, particularly for critical tasks like connection verification and encryption, as these need to be completed before the message is shared with other users. The diagram emphasizes the secure and sequential handling of the message to ensure proper delivery in the chatroom system.

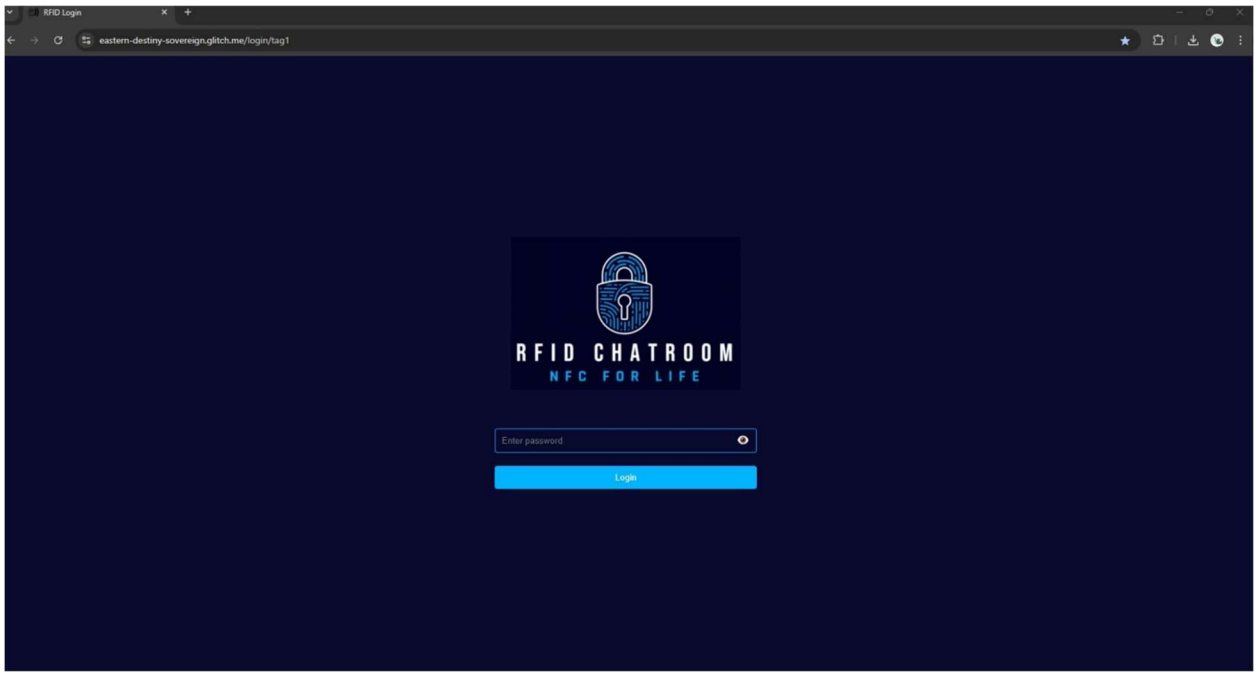
Interaction Overview Diagram



The interaction begins with the Chatroom Client scanning their RFID Tag. The diagram shows how the RFID Tag takes the user to a log-in page where the client enters a password as seen in credential verification, which is then sent to the server for verification. If the passphrase is valid, the chatroom interface is opened. A test is then performed to verify if the encryption mechanism is working correctly. If the encryption mechanism test passed, then the client can communicate in the chatroom. If the encryption mechanism test is not passed, then the client must start the login process from the “scan RFID tag” step. If the client is not a bad actor, they should still have the RFID tag to tap and their unique password.

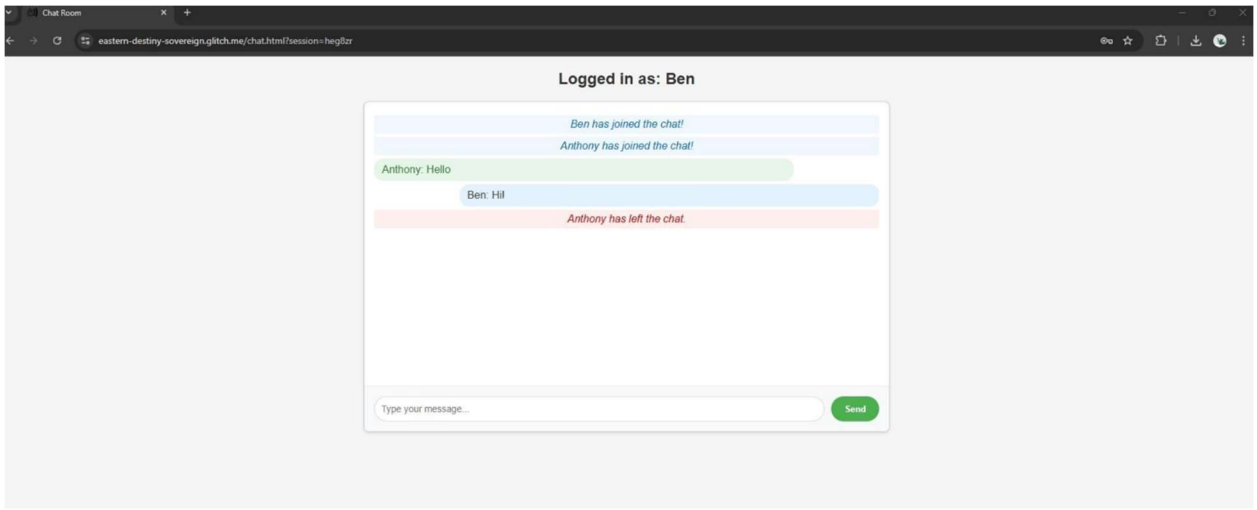
6. Design Prototype (Wireframes and Screenshots):

Chatroom Login Screen



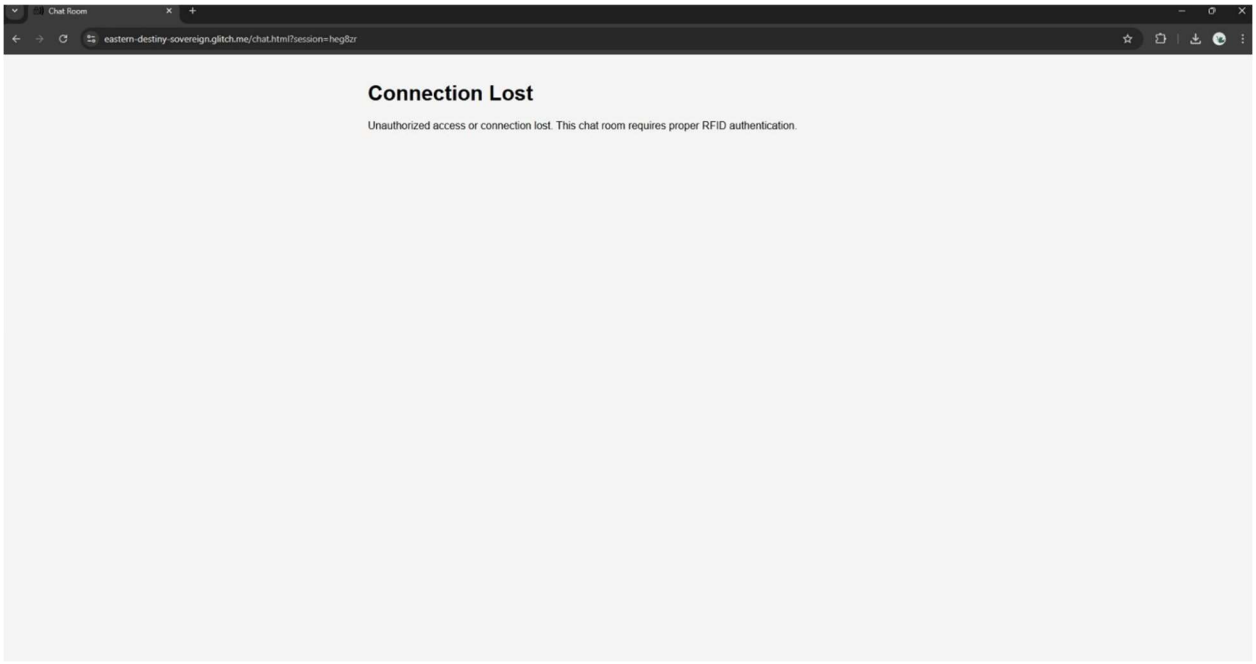
This is the login screen that the user gains entry to upon correctly utilizing his RFID tag in conjunction with his smart device.

Active Chatroom Screen



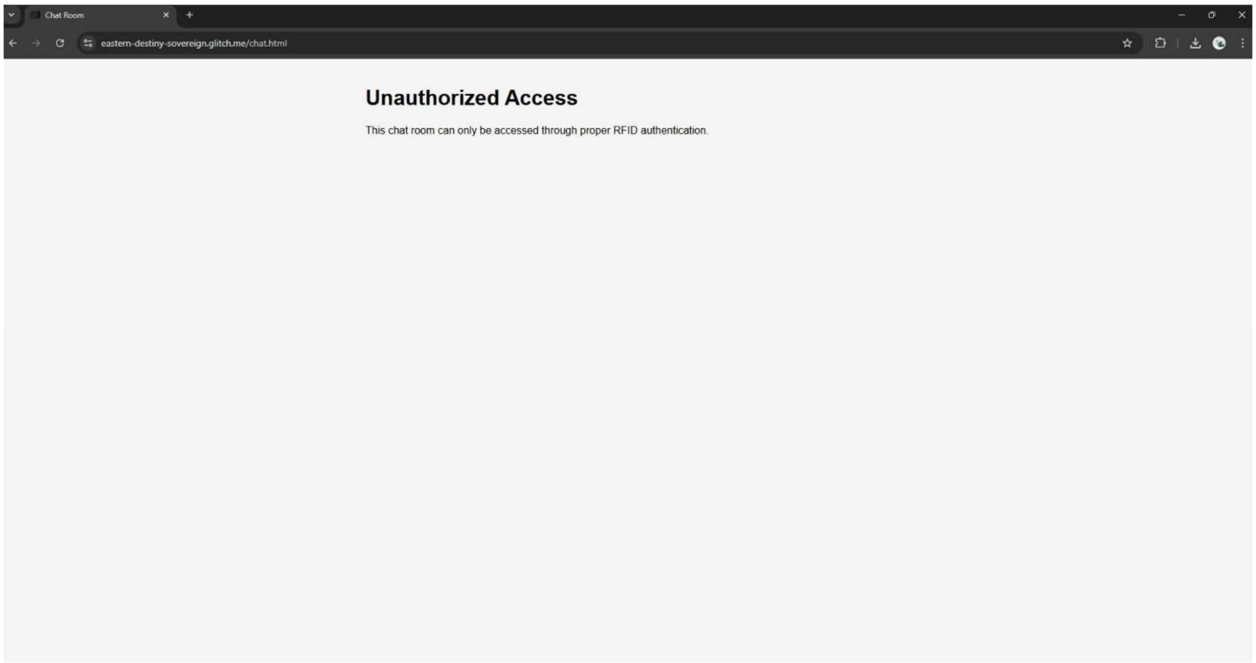
This image shows an active chatroom with two users who have successfully entered the system. It also shows the messages that both users have exchanged within the chatroom, as well as one of the users successfully exiting the chatroom.

Connection Lost Screen



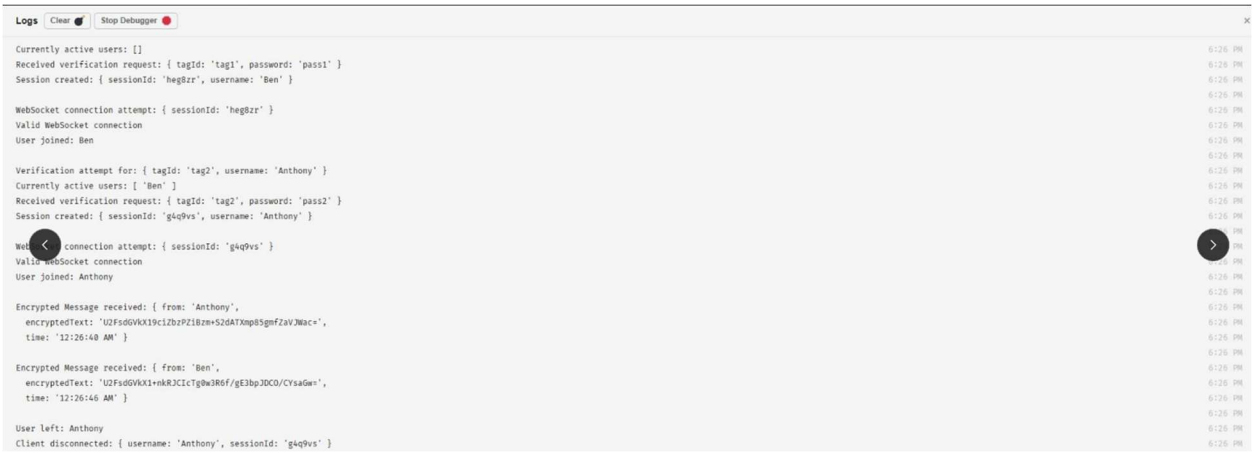
This image shows what the user will see if they are disconnected from the chatroom under any of the situations explained in Use Case 3.

Unauthorized Access Screen



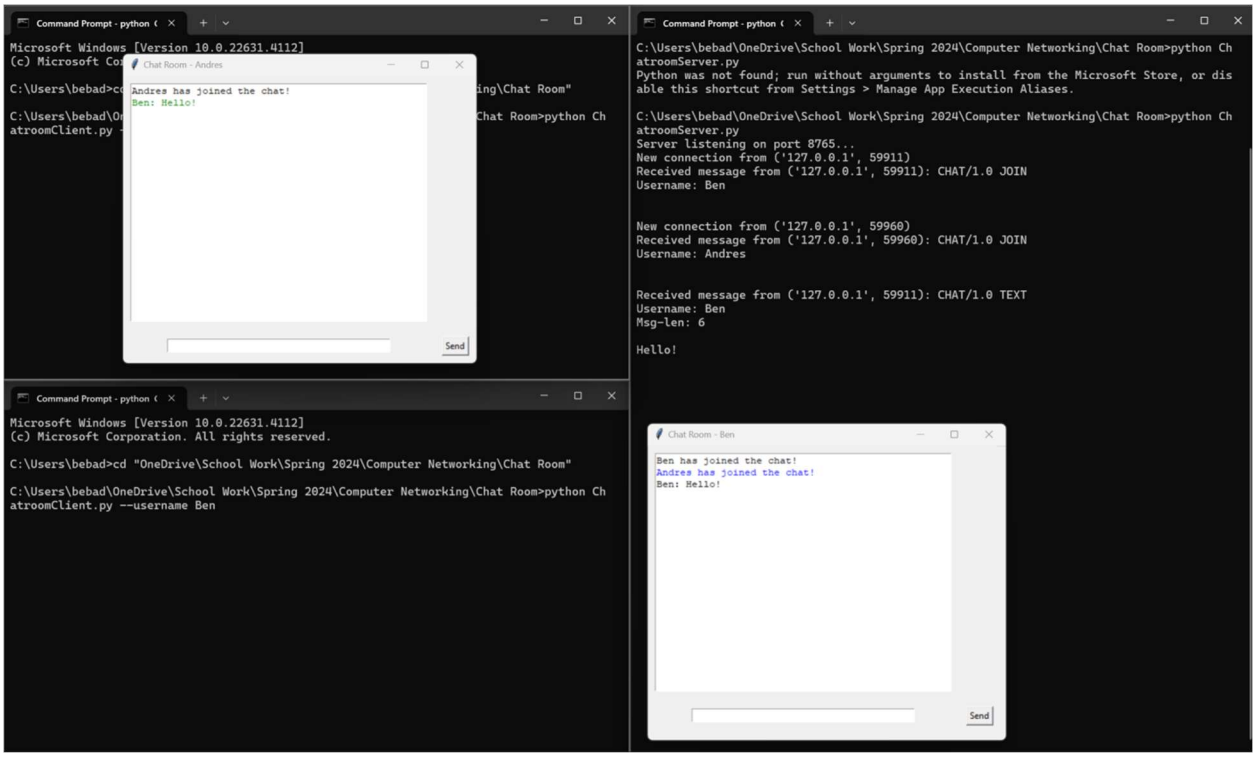
This image shows what an unauthorized user will see if they attempt to gain entry into the RFID chatroom without authorized access or valid credentials.

Terminal Log for Server



This image is a log of the terminal that shows information such as currently active users, verification attempts and encrypted message statuses.

Wireframe



This is our initial visual blueprint of our chatroom and represents the skeletal structure of a user interface in our chatroom design.

7. Future Work:

We would like to implement a feature that allows for unique RFID Tag specific urls to be made automatically. The current implementation of the solution requires RFID specific links to be created manually. While this may not seem like a big problem, it can compound it not addressed. The current implementation allows for up to 3 users to enter the chatroom and converse. But what if instead of 3 members there are 100? The time it would take to credential and craft each one of the URLs will be noticed. A solution would be to make a program that asked for a username and user credentials. The program will take the information and craft a unique URL. The URL will then be converted to hexadecimal so that we can leverage the cryptographic features the NTAG 215 allows for.

The second improvement is purchasing a bulk reader writer such that multiple tags can be written at a single time. This is useful because it would allow us to quickly produce multiple credentialed tags for a single user. This, as the first improvement, will be a timesaver making our solution easier to use and more efficient.

Around a week prior to writing this report we had “message encryption” in the future work section. But we were ultimately able to figure it out. An improvement we could make is figuring out how to extract the information from the tag using a iPhone 14 without having to build a ISO application. For example, how can we extract a key from the tag without the iPhone displaying that the keys were extracted upon a tap.

8. Glossary Table

Glossary Index	
Extracting Crypto Keys	Retrieving a set of securely wrapped software codes referred to as ‘keys’
RFID Technology	Radio Frequency Identification. This technology utilizes radio waves that detect people or objects
Deploying a Server on a Local Network	Process of deploying our software to one dedicated server
Scanning an RFID tag with iPhone	Place the NFC antenna area of your phone close to the RFID/NFC tag in order to begin the process of access
AES-256	An Advanced encryption standard 256bits
SHA-256	Secure Hash Algorithm 256Bits
Local Host	The computer that is hosting on the local network
Private Key	A key that is used to decrypt data and or create digital signatures. This key is not to be shared with the public.
Public Key	A key that is used to encrypt data. This key is to be shared with the public.
MAC	Message Authentication Code.
NTAG 424	Near Field Communication tags developed by NXP Semiconductors. Designed for secure, dynamic applications.
cryptography	The process of transforming data into an unreadable format
CMAC	Cypher-Based Message Authentication Code
SDM-Backend	GitHub repository contains an application for secure NDEF messaging.
Glitch.com	online platform for creating, editing, and sharing web applications and projects.

9. References:

[1] *GitHub · build and ship software on a single, Collaborative Platform*. GitHub. (n.d.). https://github.com/

[2] *NFC PVC card - NTAG215*. GoToTags Store. (n.d.). https://store.gototags.com/nfc-pvc-card-ntag215/

[3] *Node.js - run JavaScript everywhere*. Node.js -. (n.d.). https://nodejs.org/en

[4] *Sqlite Home Page*. SQLite Home Page. (n.d.). https://www.sqlite.org/

[5] *The Friendly Community where everyone builds the web*. Glitch. (n.d.). https://glitch.com/

[6] *W3schools.com*. W3Schools Online Web Tutorials. (n.d.-a). https://www.w3schools.com/js/DEFAULT.asp

[7] *W3schools.com*. W3Schools Online Web Tutorials. (n.d.-a). https://www.w3schools.com/html/

[8] Wuerthele, M. (2022, September 19). *Learn about iphone 14 and iphone 14 plus*. AppleInsider. https://appleinsider.com/inside/iphone-14