ABSTRACT

The Secure Group Chat System utilizing the ESP8266 is a hardware project designed to offer a secure and efficient platform for group communication within a local private network. By harnessing the capabilities of the ESP8266 microcontroller and WiFi module, this project establishes a robust local network infrastructure, enabling users to connect and engage in conversations securely.

The system boasts a user-friendly web interface accessible via any device equipped with a web browser, ensuring widespread accessibility and convenience for participants. Through this interface, users can effortlessly join the network, select personalized usernames, send messages, and monitor incoming messages from other participants in real-time.

Key features of the Secure Group Chat System include:

Secure Communication: Leveraging advanced encryption techniques, the system guarantees that messages transmitted between users remain shielded from unauthorized access, ensuring the confidentiality and integrity of communications.

User Authentication: To foster accountability and deter anonymous participation, users are mandated to authenticate themselves using unique usernames, thereby enhancing the overall security posture of the system.

Message Persistence: Enhancing user experience and facilitating continuity in discussions, the system temporarily stores messages exchanged within the group chat, allowing users to revisit past conversations and maintain context over time.

Responsive Web Interface: The web interface is meticulously crafted to be responsive, seamlessly adapting to diverse screen sizes and device types. This responsiveness ensures an optimal user experience across various platforms, fostering user engagement and satisfaction.

These features, the Secure Group Chat System not only addresses the fundamental need for secure group communication but also prioritizes user convenience and accessibility, ultimately culminating in a comprehensive solution tailored to the demands of modern collaborative environments.

1.INTRODUCTION

The Secure Group Chat System implemented using ESP8266WiFi and ESP8266WebServer libraries presents an innovative solution for facilitating secure and collaborative communication within a localized network environment. With the pervasive use of wireless networks and the growing need for secure communication channels, this project addresses the demand for a reliable and efficient platform for group discussions while ensuring data privacy and integrity.

By leveraging the capabilities of the ESP8266 module, which serves as an access point (AP), users can connect to the local network created by the system using their Wi-Fi-enabled devices. This eliminates the dependence on external servers or internet connectivity, making the solution ideal for environments where internet access may be limited or security concerns are paramount.

The system features an intuitive web-based interface accessible through standard web browsers, allowing users to participate in group conversations seamlessly. Through the integration of HTML, CSS, and JavaScript, the interface provides interactive elements for inputting usernames, composing messages, and viewing chat history. Additionally, the system incorporates functionalities for message submission, conversation clearing, and page refreshing, enhancing user experience and convenience.

Key components of the Secure Group Chat System include message storage, request handling, and user interaction mechanisms. Messages exchanged during the chat session are stored locally in an array, ensuring continuity and persistence across interactions. Request handlers are implemented to manage incoming HTTP requests, enabling actions such as sending messages, refreshing the page, and clearing conversation history. With its robust features and user-friendly interface, the Secure Group Chat System offers a versatile communication platform suitable for various applications and environments. Whether deployed in educational institutions, corporate settings, or community gatherings, the system empowers users to engage in secure and collaborative discussions, fostering productivity, knowledge sharing, and community building.

1.1 Overview Of The Project

1.1.1 Background

The project is developed in response to the growing need for secure and efficient communication solutions within local network environments. With the proliferation of IoT devices and interconnected systems, there is a demand for chat systems that prioritize data privacy, accessibility, and ease of use. The project leverages the ESP8266 platform to create a self-contained chat system that operates within a localized network, eliminating the dependence on external servers and internet connectivity.

1.1.2 Objective

The primary objective of the project is to design and implement a Secure Group Chat System using ESP8266, providing users with a reliable platform for real-time communication within a confined network environment. Key objectives include:

Ensuring data privacy and integrity through secure communication protocols.

Creating an intuitive web-based interface accessible from any Wi-Fi-enabled device.

Facilitating user authentication to identify participants and maintain accountability.

Implementing functionalities for message storage, retrieval, and interaction.

Enhancing user experience through interactive elements and seamless navigation.

1.1.3 Scope

The project scope encompasses the development of both hardware and software components required to deploy the Secure Group Chat System. This includes:

Configuring the ESP8266 module to act as an access point (AP) and host the chat server.

Developing the web-based user interface for composing messages, viewing chat history, and managing conversations.

Implementing server-side logic to handle message transmission, storage, and retrieval.

Integrating user authentication mechanisms to validate participant identities and ensure secure access.

Testing the system for functionality, reliability, and performance within a local network environment.

1.1.4 Technical Feasibility

The project's technical feasibility is supported by the capabilities of the ESP8266 platform, which offers built-in Wi-Fi connectivity and web server functionalities. Additionally, the availability of ESP8266 libraries such as ESP8266WiFi and ESP8266WebServer provides developers with tools to rapidly prototype and deploy network-based applications. With the appropriate hardware setup and software development skills, implementing a Secure Group Chat System using ESP8266 is both technically viable and economically feasible.\

2.1 Module Descriptions

2.1.1 WiFi Configuration Module

This module configures the ESP8266 microcontroller as an access point for the local network. It sets up the SSID (Service Set Identifier) and password for the network, allowing devices to connect to it. Utilizes the WiFi.softAP() function provided by the ESP8266WiFi library to configure the microcontroller as an access point.

2.2.2 Web Server Module

This module handles HTTP requests and serves web pages for user interaction. It listens for incoming requests and responds with the appropriate HTML/CSS/JavaScript files to render the user interface. Uses the ESP8266WebServer library to create an instance of a web server (ESP8266WebServer server(80)), defines routes for different endpoints, and handles requests using callback functions.

2.2.3 Message Handling Module

This module manages the processing and storage of user messages. It extracts message content and user details from HTTP requests, stores the messages in an array (MessageData[]), and updates the message count. Implemented in the handleSubmit() function, which checks for message and username arguments in the HTTP request, constructs the message string, and stores it in the array.

2.2.4 User Interface Module

This module generates the user interface for the web application. It dynamically generates HTML/CSS code to create an interactive interface for users to input messages, view conversations, and perform actions like clearing the conversation and refreshing the page. Utilizes the HTML_DESIGN() function to construct HTML/CSS code with placeholders for message data, username input, message input, and buttons for sending messages, refreshing the conversation, and clearing the conversation.

2.SYSTEM ANALYSIS

2.1 Existing System

The existing system lacks a dedicated group communication platform within a local private network. In typical scenarios, users rely on generic communication tools like email or instant messaging applications, which may not be tailored for secure group conversations within closed network environments. This absence of a specialized system can lead to inefficiencies and security concerns, as users may resort to ad-hoc solutions that lack robust security measures and features necessary for effective group communication.

2.2.1 Disadvantages of Existing System

Lack of Security Measures: Traditional communication methods may lack adequate security measures, making conversations susceptible to interception or unauthorized access.

Absence of User Authentication: Without proper user authentication mechanisms, it can be challenging to verify the identity of participants, leading to potential security breaches or misuse of the system.

Limited Message Persistence: Most existing communication tools do not offer built-in message persistence features within a local network context, making it difficult for users to access past conversations or maintain continuity in discussions. Inefficient Communication: Ad-hoc communication solutions within local networks often lack features necessary for efficient group communication, such as real-time updates, message threading, or multimedia support. This can result in fragmented communication experiences and hinder productivity.

2.2 Proposed System

2.2.1 Advantages of Proposed System

The proposed Secure Group Chat System utilizing the ESP8266 offers several key advantages over the existing system: Enhanced Security Protocols: By leveraging encryption techniques, the system ensures that messages exchanged between users remain confidential and protected from unauthorized access. User Authentication and Accountability: Implementing user authentication mechanisms ensures that each participant is accountable for their messages, reducing the risk of malicious activities and fostering a trustworthy communication environment.

Message Persistence and History: The system stores messages temporarily, allowing users to access past conversations and maintain continuity in discussions. This feature enhances collaboration and knowledge sharing among participants.

Responsive and User-Friendly Interface: The web interface of the system is designed to be responsive, providing seamless access to group chat functionalities across various devices and screen sizes. Intuitive design elements and clear navigation enhance user experience and promote adoption.

Scalability and Flexibility: The system architecture is designed to be scalable, accommodating a growing number of users and supporting future feature enhancements. Flexibility in deployment options allows for easy integration with existing network infrastructures.

3.SYSTEM SPECIFICATION

3.1 Software Specification

The software components of the Secure Group Chat System include:

ESP8266 Module: The ESP8266 module is a crucial hardware component serving as the core of the Secure Group Chat System. It provides Wi-Fi connectivity and hosts the chat server, enabling communication between users. Computer or Mobile Devices: Users access the chat system using their computer or mobile devices, which act as clients. These devices connect to the ESP8266 module over Wi-Fi and interact with the web-based chat interface.

Arduino IDE: The Arduino Integrated Development Environment (IDE) is used for programming the ESP8266 microcontroller, facilitating the development of the firmware for the chat system.

ESP8266WiFi Library: This library provides the necessary functions to enable the ESP8266 module to connect to a local private network, acting as an access point for the chat system.

ESP8266WebServer Library: The ESP8266WebServer library is utilized to handle HTTP requests and serve web pages, enabling the implementation of a user-friendly web interface for the chat system.

HTML, CSS, and JavaScript: These web technologies are used to design and implement the user interface of the chat system. HTML is employed for structuring the content, CSS for styling elements, and JavaScript for client-side interactivity and validation.

3.2 Hardware Specification

The hardware components of the Secure Group Chat System consist of:

Server-Side Logic: The server-side logic is developed using the Arduino IDE and ESP8266 libraries. It handles various functionalities such as message transmission, storage, and retrieval. This logic ensures seamless communication between users and manages the chat system's operations. Web-Based User Interface: The web-based user interface is designed using HTML, CSS, and JavaScript. It provides users with a platform to interact with the chat system, compose messages, view chat history, and manage conversations. The interface is accessible through standard web browsers on users' devices.

ESP8266 Module: The ESP8266 microcontroller serves as the core hardware component, providing Wi-Fi connectivity and processing capabilities to the chat system.

Wi-Fi Antenna: A Wi-Fi antenna is integrated into the ESP8266 module to facilitate wireless communication with other devices within the local private network.

Power Supply: The system can be powered using a USB cable connected to a computer or a dedicated power source capable of providing the required voltage and current to the ESP8266 module.

3.3 About the Software

The software aspect of the Secure Group Chat System encompasses the firmware running on the ESP8266 microcontroller and the web interface accessible to users. The firmware is responsible for handling network connections, message processing, and serving web pages to users. It is programmed using the Arduino IDE and relies on libraries such as ESP8266WiFi and ESP8266WebServer for network functionality.

The web interface is designed to be intuitive and user-friendly, allowing participants to join the chat system, select a username, send messages, and view messages from other users in real-time. HTML, CSS, and JavaScript are employed to create an interactive and responsive interface that adapts to various screen sizes and devices.

3.3.1 Key Functionalities:

User Registration and Authentication: The system allows users to register and authenticate themselves before accessing the chat system. This ensures secure participation and prevents unauthorized access.

Real-Time Messaging: Users can exchange messages in real-time within the group chat environment. This feature enables instant communication and fosters collaboration among users.

Message History: The system maintains a record of past messages and conversations, allowing users to view and reference them as needed. This feature enhances continuity and context within the chat system.

Message Encryption: To ensure message security and user privacy, the system implements encryption algorithms for message transmission. This encryption mechanism protects messages from unauthorized access and interception.

Clear Conversation: Users have the option to clear the conversation history, removing all messages from the chat interface. This feature enhances user privacy and facilitates data management.

Refresh Chat: Users can refresh the chat interface to view the latest messages and updates. This functionality ensures that users have access to real-time information and maintains the system's responsiveness.

3.3.2 Requirements:

Hardware Requirements: The system requires an ESP8266 module for hosting the chat server and computer or mobile devices with web browsing capabilities for user access.

Software Requirements: Development tools such as the Arduino IDE, ESP8266 libraries, and web development technologies including HTML, CSS, and JavaScript are needed to implement the system.

Networking Requirements: A local Wi-Fi network is necessary for communication between the ESP8266 module and users' devices.

Security Requirements: Implementation of encryption protocols, user authentication mechanisms, and secure data transmission protocols is essential to ensure message confidentiality and integrity.

3.3.3 Architecture Components:

ESP8266 Server: The ESP8266 module serves as the server hosting the chat application. It manages message transmission, storage, and retrieval, ensuring seamless communication between users.

Web-Based User Interface: Users interact with the chat system through the web-based user interface. This interface enables users to compose messages, view chat history, and manage conversations in real-time.

Database (Optional): While not mandatory, a database component can be incorporated into the system architecture to store user credentials, chat history, and other relevant information. This database enhances data management and persistence within the system.

3.3.4Feasibility Factors:

Technical Feasibility: The system's technical feasibility is supported by the capabilities of the ESP8266 platform and available software libraries. These resources enable rapid development and implementation of the chat system, ensuring its functionality and performance.

Economic Feasibility: The project's economic feasibility depends on the affordability of hardware components and the availability of open-source software tools. Cost-effective solutions enable the development and deployment of the system within budget constraints.

Operational Feasibility: Operational feasibility is determined by user feedback, ease of deployment, and maintenance requirements. The system's practicality and usability are essential considerations, ensuring its successful adoption and operation in real-world scenarios.

3.SYSTEM DESIGN

The design of the Secure Group Chat System encompasses both architectural and detailed design aspects, ensuring the system's reliability, scalability, and maintainability. This section outlines the key components and design considerations involved in developing and implementing the system.

4.1 Input Design

The input design of the Secure Group Chat System focuses on providing a user-friendly interface for users to input data and interact with the system. Key aspects of the input design include:

Username Input: Users are prompted to enter a username when accessing the chat system. This input field ensures that each user is uniquely identified within the chat environment.

Message Input: A text field is provided for users to type and send messages to the chat. This input allows participants to contribute to ongoing conversations and engage with other users in real-time.

Buttons: Buttons are incorporated into the user interface to enable actions such as sending messages, refreshing the chat window, and clearing the conversation history. These buttons enhance the usability of the system by providing intuitive controls for common tasks.

Form Submission: Input fields and buttons are enclosed within HTML forms, allowing users to submit their inputs easily. JavaScript functions are used to handle form submission events and perform appropriate actions, such as sending messages or refreshing the chat window.

4.1.1 ESP8266 Server

Message Handling: The server component is responsible for handling incoming messages from clients, storing them temporarily, and broadcasting them to other connected clients.

Authentication: User authentication mechanisms are implemented to verify the identity of users before granting access to the chat system. This ensures secure participation and prevents unauthorized access.

Message Encryption: Encryption algorithms are integrated into the server logic to encrypt outgoing messages, ensuring message security and protecting user privacy during transmission.

4.1.2 Web-Based User Interface

User Registration: The web interface includes user registration functionality, allowing new users to create accounts and establish credentials for accessing the chat system.

Message Composition: Users can compose messages using the web interface, which provides a text input field for entering messages and a send button to initiate message transmission.

Chat History Display: The interface displays the chat history, showing past messages and conversations in chronological order. Users can scroll through the chat history to view older messages and reference previous discussions.

User Interaction: Various interactive elements, such as buttons for clearing conversation history and refreshing the chat interface, are included to enhance user experience and facilitate interaction with the system.

4.2 Database Design

The Secure Group Chat System does not rely on a traditional database for storing messages. Instead, it utilizes an array data structure within the firmware running on the ESP8266 microcontroller to store and manage message data.

Field	Type	Null	Description
MessageID	INT(11)	Yes	Unique identifier for each message
Username	VARCHAR(50)	Yes	Username of the message sender
Message	TEXT(11)	Yes	Content of the message
Timestamp	TIMESTAMP	Yes	Timestamp indicating when the message was sent

In this table structure:

MessageID: An auto-incrementing integer field serving as the primary key for the table.

Username: A variable-length string field storing the username of the message sender.

Message: A text field to store the content of the message.

Timestamp: A timestamp field indicating the date and time when the message was sent.

4.2.1 Database (Optional)

Data Storage: If a database component is incorporated into the system, it is responsible for storing user credentials, chat history, and other relevant information. The database ensures data persistence and enables efficient retrieval of stored information.

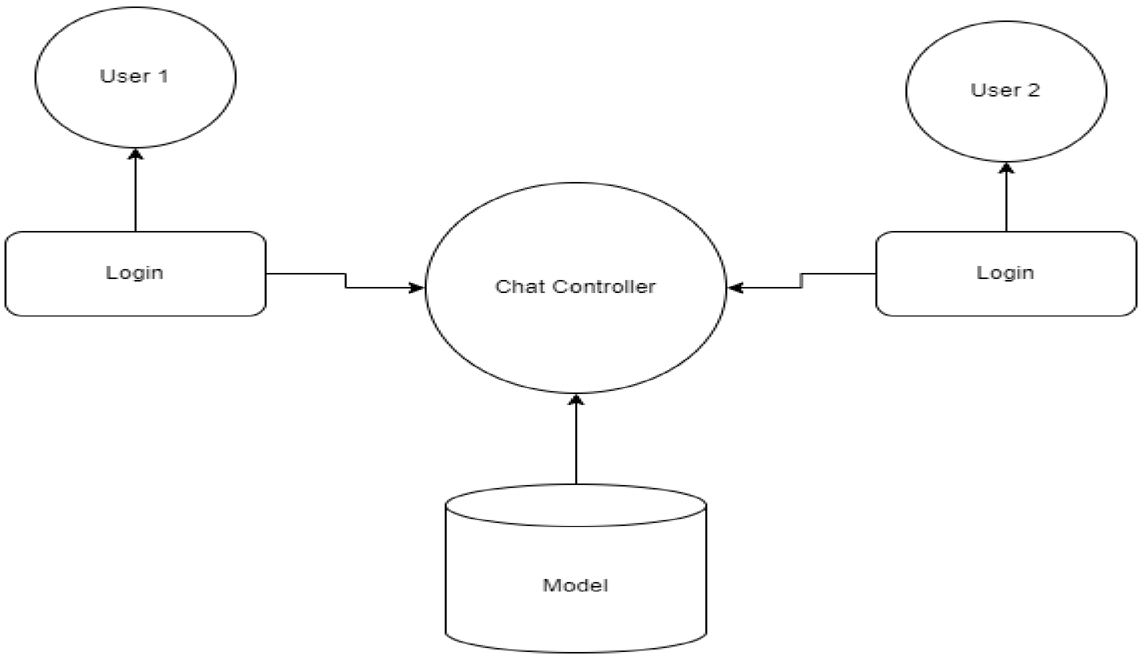
Data Management: The database component includes functionalities for managing data, such as adding, updating, and deleting records. These operations maintain the integrity and consistency of stored data within the system.

4.2.2 Security

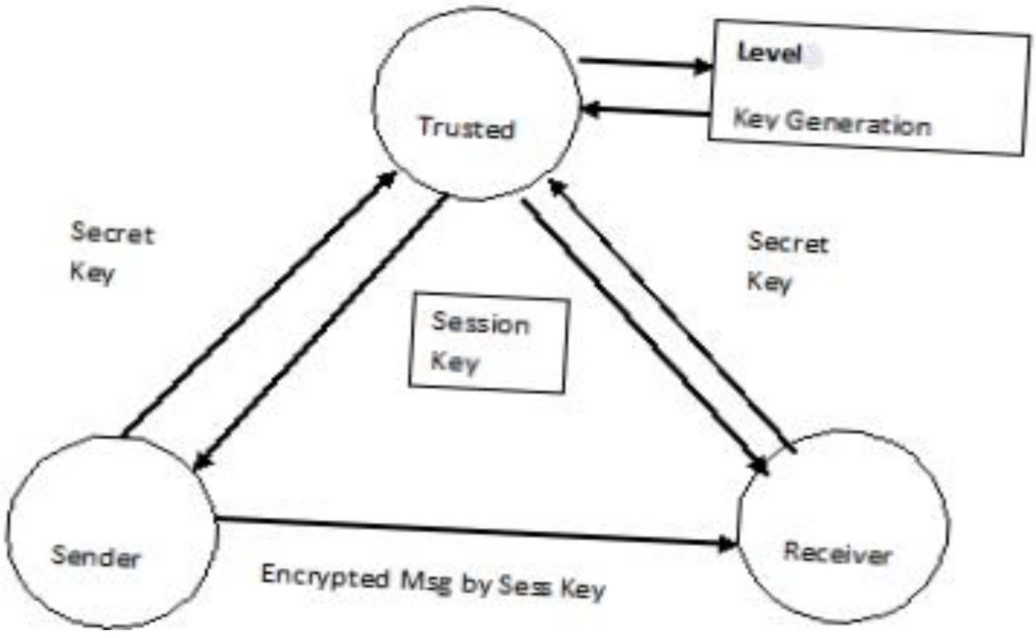
Message Encryption: Strong encryption algorithms are employed to encrypt messages during transmission, ensuring message confidentiality and preventing unauthorized access.

User Authentication: Robust authentication mechanisms are implemented to verify the identity of users and prevent unauthorized access to the chat system.

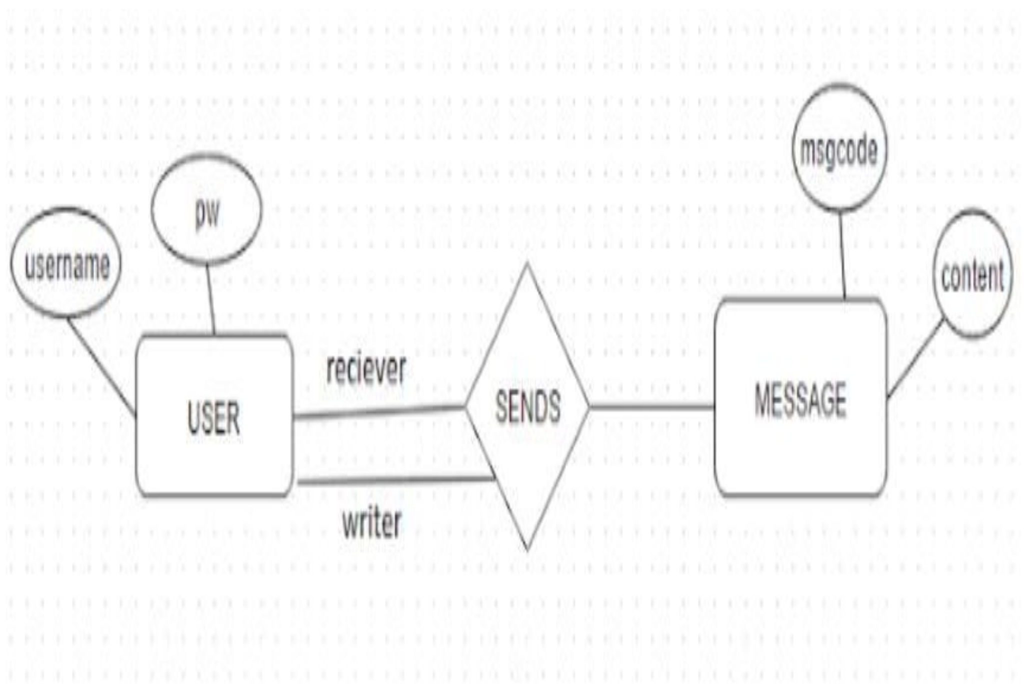
4.3 System flow Diagram



4.4 Data flow Diagram



4.5 Entity Relationship Diagram



4.6 Output Design

Output design is a crucial aspect of system design, determining how information is presented to users and stakeholders. In the context of the Secure Group Chat System utilizing the ESP8266, the output design focuses on delivering an intuitive and informative user interface for participants accessing the chat system. Here are key considerations for the output design:

Message Display: Messages exchanged within the chat system are prominently displayed to users in a readable format. Each message should be clearly visible, with appropriate formatting to distinguish between messages from different users.

Username Visibility: Usernames associated with each message should be clearly displayed alongside the corresponding message. This enhances communication transparency and allows participants to identify message authors easily.

Real-Time Updates: The output design ensures that new messages are dynamically added to the chat interface in real-time, providing participants with instant access to the latest conversations. This feature enhances the interactive nature of the chat system.

5.SYSTEM TESTING AND IMPLEMENTATION

5.1 System Testing

System testing is a critical phase in the development lifecycle of the Secure Group Chat System. This phase involves verifying and validating the functionality, performance, and security aspects of the system to ensure that it meets the specified requirements. Here are the key aspects of system testing:

Functional Testing: This involves testing the individual features and functionalities of the chat system to ensure that they operate as intended. Functional testing includes verifying user authentication, message sending and receiving, message persistence, and error handling.

Integration Testing: Integration testing focuses on testing the interaction between different components of the system. This includes testing the integration between the ESP8266 microcontroller, web server, and user interface to ensure seamless communication and data flow.

Performance Testing: Performance testing evaluates the responsiveness and scalability of the system under various load conditions. This includes testing the system's ability to handle multiple concurrent users, large message volumes, and network latency.

Security Testing: Security testing is essential to identify and mitigate potential vulnerabilities in the system. This includes testing for encryption strength, secure authentication mechanisms, and protection against common security threats such as cross-site scripting (XSS) and injection attacks.

Usability Testing: Usability testing assesses the user-friendliness and intuitiveness of the chat system's interface. This involves gathering feedback from users to identify any usability issues and making iterative improvements to enhance the user experience.

5.2 System Implementation

System implementation involves deploying the Secure Group Chat System in a real-world environment, making it accessible to users within the local private network. The implementation phase includes the following steps:

Hardware Setup: Set up the necessary hardware components, including the ESP8266 microcontroller and any additional peripherals required for network communication.

Software Installation: Install the required software components, including the Arduino IDE for programming the ESP8266 microcontroller and any libraries or dependencies needed for the chat system.

Configuration: Configure the ESP8266 microcontroller to act as an access point, providing network connectivity for users to access the chat system. Set up the web server and initialize the chat system with appropriate settings.

Monitoring and Maintenance: Continuously monitor the deployed system for any issues or performance bottlenecks. Implement regular maintenance procedures to address any software updates, security patches, or hardware maintenance requirements.

6.CONCLUSION

In conclusion, the Secure Group Chat System represents a comprehensive solution for facilitating secure and efficient communication within a group or community. Through the integration of hardware and software components, including the ESP8266 module and web-based interface, the system offers a user-friendly platform for real-time messaging while ensuring data privacy and security.

Throughout the development and implementation process, various aspects were considered to ensure the system's effectiveness, usability, and reliability. From system design and testing to implementation and operational considerations, each phase was meticulously planned and executed to meet the project's objectives. The testing phase involved rigorous testing methodologies to validate the system's functionality, performance, and security. Usability testing sessions provided valuable feedback on the user interface and overall user experience, leading to iterative improvements and enhancements. During implementation, security configurations were implemented to safeguard against potential threats and vulnerabilities, while scalability considerations ensured the system's adaptability to future growth and expansion.

Continuous monitoring and maintenance are essential for ensuring the system's ongoing performance, security, and compliance with regulatory requirements. User training and feedback mechanisms play a crucial role in enhancing user adoption and satisfaction, driving continuous improvement and innovation. In conclusion, the Secure Group Chat System represents a robust and reliable solution for secure group communication, offering users a seamless and intuitive platform for exchanging messages while maintaining data privacy and confidentiality. Through careful planning, testing, and ongoing maintenance, the system can meet the evolving needs of its users and provide a valuable communication tool for various applications and industries.

7.FUTURE ENHANCEMENTS

In the future, the Secure Group Chat System could undergo several enhancements to further improve its functionality and user experience. Firstly, expanding its compatibility to support various devices and platforms like desktops, tablets, and smartphones would increase accessibility. Integrating multimedia messaging capabilities would allow users to share images, videos, and audio files seamlessly, enhancing communication. Advanced security features such as end-to-end encryption and two-factor authentication could be implemented to bolster data privacy and protection. Additionally, introducing user management tools for registration, profile management, and access control would offer administrators greater control over user accounts. Integration with external services like cloud storage providers and productivity tools would enable enhanced collaboration. The incorporation of AI technologies could bring intelligent features like chatbots, automated moderation, and personalized recommendations. Offline messaging support would ensure users can stay connected even when temporarily disconnected from the network. Accessibility features such as screen reader support and keyboard navigation would enhance inclusivity. Continuously optimizing performance and fostering a vibrant user community through public chat rooms and forums would further enrich the user experience. Overall, these future enhancements would transform the Secure Group Chat System into a versatile, secure, and user-friendly communication platform, catering to a wide range of user needs and preferences while staying abreast of emerging trends in communication technology.

8.BIBLIOGRAPHY

1. Artificial Intelligence: Foundations of Computational Agents" by David L. Poole and Alan K. Mackworth.

2. Andreas M. Antonopoulos: A well-known author and speaker on blockchain and cryptocurrencies, his books like "Mastering Bitcoin" offer insights into cryptography and secure communication protocols relevant to IoT.

3. A Modern Approach" by Stuart Russell and Peter Norvig: This comprehensive textbook provides an in-depth exploration of artificial intelligence concepts, including machine learning, natural language processing, and intelligent agents.

4. An Introduction" by Richard S. Sutton and Andrew G. Barto: This seminal work in reinforcement learning provides a comprehensive overview of the field, covering topics such as value functions, policy gradients, and deep reinforcement learning.

5. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurélien Géron: Focusing on practical applications, this book offers hands-on experience with popular machine learning libraries and frameworks, including Scikit-Learn, Keras, and TensorFlow.

Online Resources

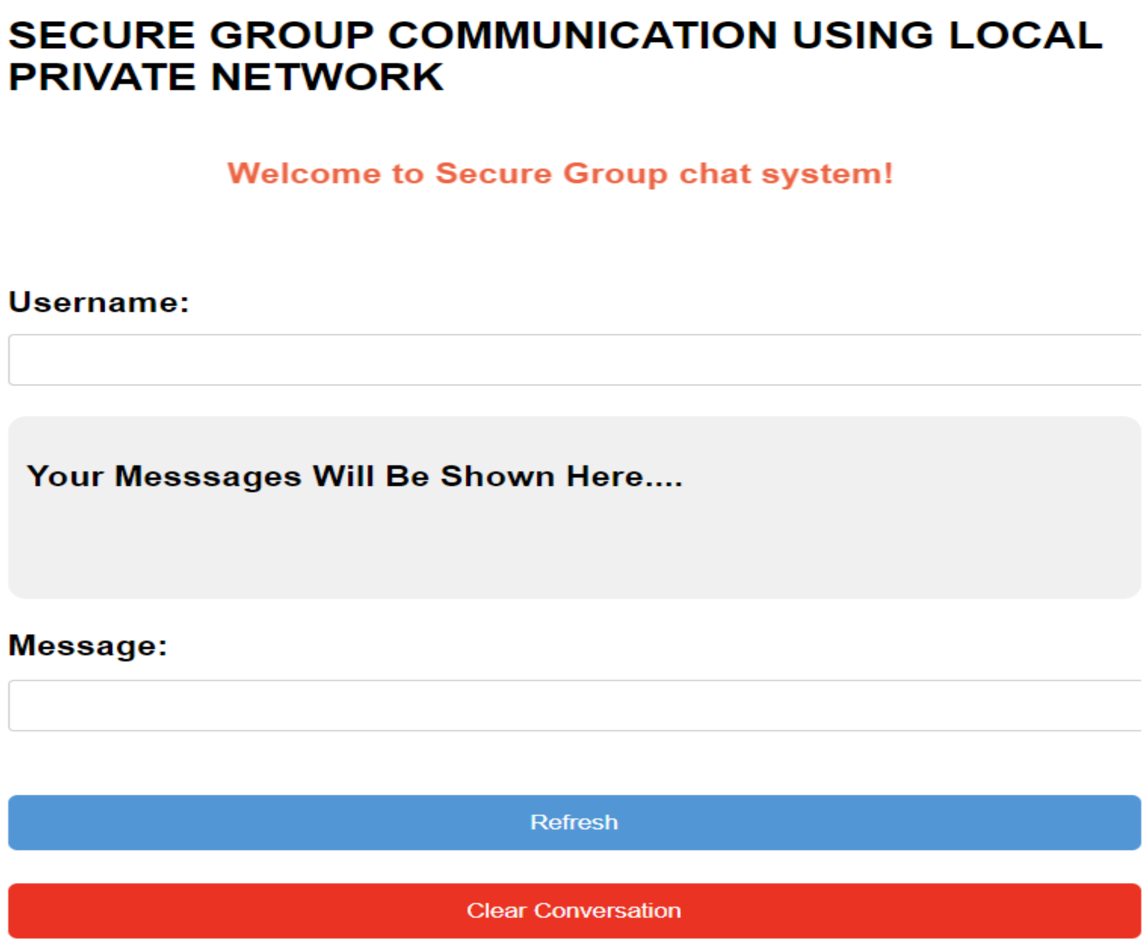
1. Official Arduino Documentation: https://www.arduino.cc/
2. ESP8266WiFiDocumentation: https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html
3. ESP8266WebServerDocumentation: https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WebServer/README.md
4. Project Research: https://www.geeksforgeeks.org/machine-learning-projects/
5. Project Guidance: https://randomnerdtutorials.com/projects-esp8266/

9.APPENDIX

9.2 Source Code

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>

int messagecount = 0;
const char* ssid = "WEB CHAT SYSTEM ";
String txtusername;
const char* pass = "12345678";
String html = "";
ESP8266WebServer server(80);
String MessageData[500];

void HTML_DESIGN(String username) {
  html = "<!DOCTYPE html>\
<html lang=\"en\">\
<head>\
<meta charset=\"UTF-8\">\
<meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">\
<title>SECURE GROUP CHAT SYSTEM</title>\
<style>\
    body {\
        font-family: Arial, sans-serif;\
        background-color: #f2f2f2;\
        margin: 0;\
        padding: 20px;\
    }\
    .container {\
        max-width: 600px;\
        margin: auto;\
        background-color: #fff;\
        border-radius: 5px;\
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);\
        padding: 20px;\
    }\
    label {\
        display: block;\
        margin-bottom: 10px;\
        font-weight: bold;\
    }\
    input[type=\"text\"] {\
        width: calc(100% - 10px);\
        padding: 8px;\
        margin-bottom: 20px;\
        border: 1px solid #ccc;\
        border-radius: 3px;\
    }\
    input[type=\"submit\"] {\
        background-color: #4CAF50;\
        color: white;\
        padding: 10px 20px;\
        border: none;\
        border-radius: 3px;\
        cursor: pointer;\
    }\
    input[type=\"submit\"]:hover {\
        background-color: #45a049;\
    }\
    /* Responsive layout */\
    @media screen and (max-width: 600px) {\
        .container {\
            width: 90%;\
            padding: 10px;\
        }\
        input[type=\"text\"],\
        input[type=\"submit\"] {\
            width: 100%;\
        }\
    }\
    /* Marquee styling */\
    .marquee {\
        width: 100%;\
        overflow: hidden;\
        white-space: nowrap;\
    }\
    .marquee span {\
        display: inline-block;\
        padding-left: 100%;\
        animation: marquee 20s linear infinite;\
    }\
    .blue-button {\
    background-color: #3498db;\
    color: white;\
    width: 100%;\
    padding: 10px 20px;\
    border: none;\
    border-radius: 5px;\
    cursor: pointer;\
}\
\
.blue-button:hover {\
   background-color: #2980b9;\
}\
.red-button {\
   background-color: red;\
   color: white;\
   width: 100%;\
   padding: 10px 20px;\
   border: none;\
   border-radius: 5px;\
   cursor: pointer;\
}\
\
.red-button:hover {\
   background-color: darkred;\
}\
      .message-area {\
       background-color: #f0f0f0;\
       border-radius: 10px;\
       padding: 10px;\
       margin-bottom: 20px;\
   }\
   @keyframes marquee {\
      0% { transform: translate(0, 0); }\
      100% { transform: translate(-100%, 0); }\
   }\
</style>\
\
<script>\
   function send() {\
     alert(\"Message Sent Successfully...\");\
   }\
    function clear1() {\
     alert(\"Conversation Cleared Successfully...\");\
   }\
    function refresh() {\
     alert(\"Refershed Successfully...\");\
   }\
</script>\
\
</head>\
<body>\
\
<div class=\"container\">\
   <h2>SECURE GROUP COMMUNICATION USING LOCAL PRIVATE NETWORK</h2>\
   <div class=\"marquee\">\
    <h3> <span style=\"color: #FF5733;\">Welcome to our chat system!
</span><h3>\
   <br>\
   <br>\
    <form action=\"/send-message\" method=\"get\">\
      <label for=\"username\">Username:</label>\
      <input type=\"text\" id=\"username\"  name=\"username\" value='" + username +
"' required>\
      <div class=\"message-area\">\
        <p>Your Messsages Will Be Shown Here....</p>\
        <br>\ ";
  for (int i = 0; i <= messagecount; i++)
  {
    html = html + "<p>" + MessageData[i] + "</p>\ ";
  }
  html = html + " </div>\
      <label for=\"message\">Message:</label>\
      <input type=\"text\" id=\"message\" name=\"message\" required>\
      <input type=\"submit\" value=\"Send Message\" onclick=\"send()\" >\
     </form>\
     <br>\
   </form>\
      <form action=\"/refresh\" method=\"get\">\
      <button class=\"blue-button\" type=\"submit\" onclick=\"refresh()\"
>Refresh</button>\
   </form>\
   <br>\
      <form action=\"/Clear-Conversation\" method=\"get\">\
      <button class=\"red-button\" type=\"submit\" onclick=\"clear1()\" >Clear
Conversation</button>\
   </form>\
</div>\
\
</body>\
</html>\ ";
}

void handleRoot() {
  HTML_DESIGN("");
  server.send(200, "text/html", html);
}

void handleSubmit() {

  if (server.hasArg("message") && server.hasArg("username")) {
    String txtmessage = server.arg("message");
    txtusername = server.arg("username");
    String message = txtusername + " : " + txtmessage;
    MessageData[messagecount] = message;
    Serial.println(message);
    messagecount++;
  }
  HTML_DESIGN(txtusername);
  server.send(200, "text/html", html);

}

void handlerefresh() {

  HTML_DESIGN(txtusername);
  server.send(200, "text/html", html);

}
void handleclear() {
  clearArray(MessageData, messagecount);
  HTML_DESIGN("");
  server.send(200, "text/html", html);

}

void clearArray(String array[], int size) {
  for (int i = 0; i < size; i++) {
    array[i] = ""; // Clear each String object
  }
}

void setup() {
  Serial.begin(115200);

  // Set ESP8266 as access point
  WiFi.softAP(ssid, pass);

  Serial.println("");
  Serial.println("ESP8266 AP Configured");
  Serial.print("IP Address: ");
  Serial.println(WiFi.softAPIP());

  server.on("/", handleRoot);
  server.on("/send-message", handleSubmit);
  server.on("/refresh", handlerefresh);
  server.on("/Clear-Conversation", handleclear);
  server.begin();
  Serial.println("HTTP server started");
}

void loop() {
  server.handleClient();
}
```

9.2 Screen shots



SECURE GROUP COMMUNICATION USING LOCAL PRIVATE NETWORK

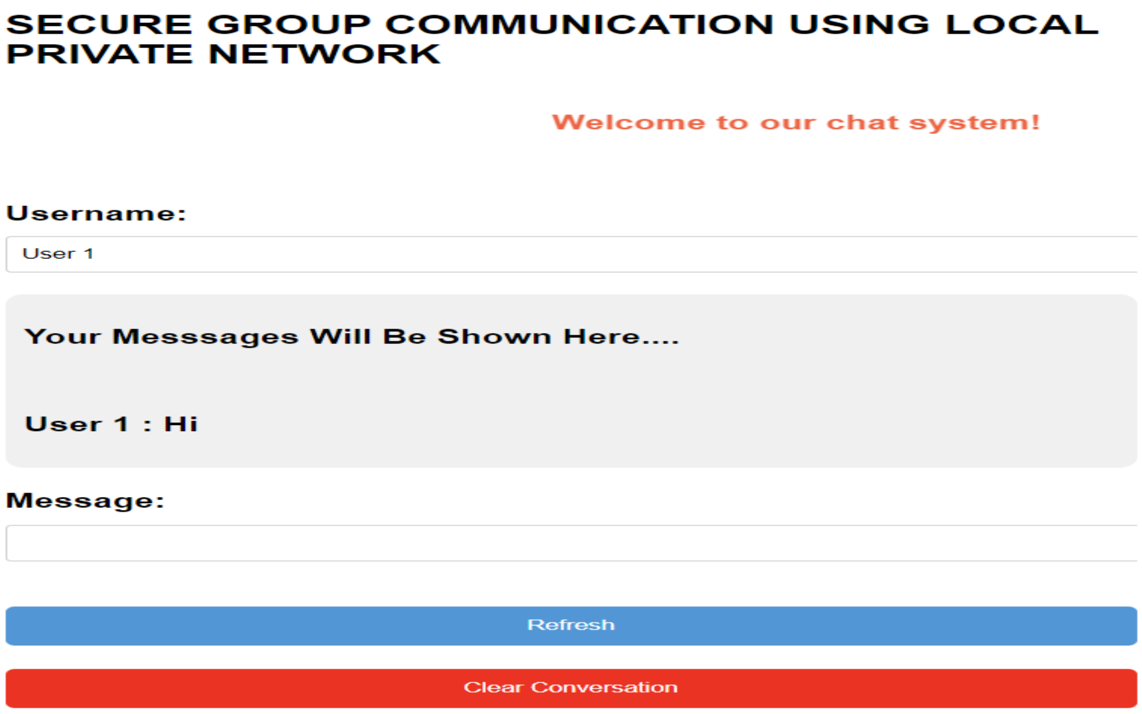
Welcome to Secure Group chat system!

Username:

Your Messsages Will Be Shown Here....

Message:

FIG 1 – STARTING PAGE


SECURE GROUP COMMUNICATION USING LOCAL PRIVATE NETWORK

Welcome to our chat system!

Username:

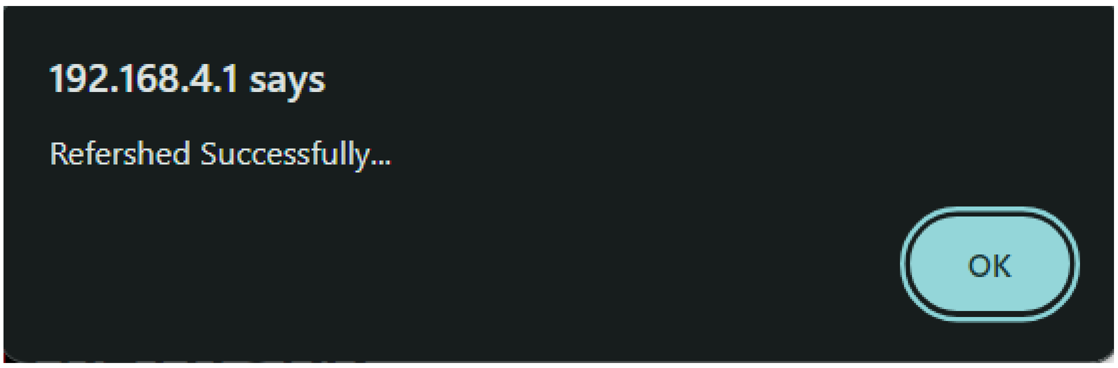
User 1

Your Messsages Will Be Shown Here....

User 1 : Hi

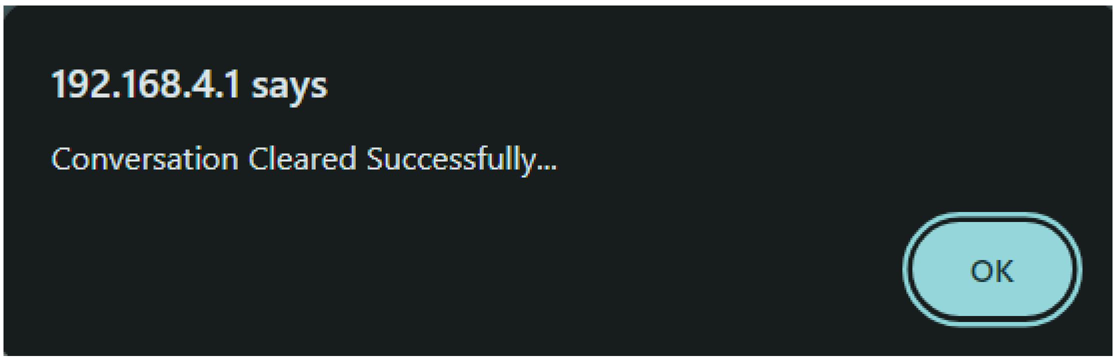
Message:

FIG 2 – USER CHAT


192.168.4.1 says

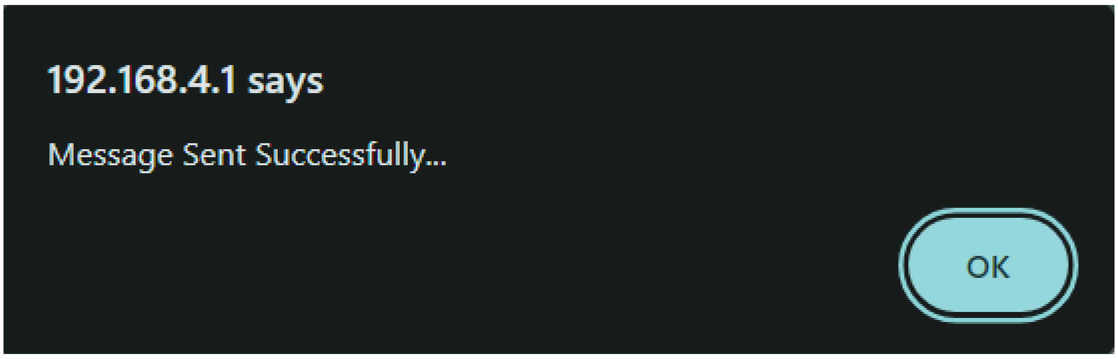
Refershed Successfully...

FIG 3 – PAGE REFERSHED


192.168.4.1 says

Conversation Cleared Successfully...

FIG 4 – CONVERSATION CLEARED


192.168.4.1 says

Message Sent Successfully...

FIG 5 – MESSAGE SENT