CSE7101- Capstone Project
Review-4

Secure AI Chat System (End-to-End Encryption + AI Moderation)

Batch Number: COM_58

Roll Number :
20221COM0136

Name:
R. Charu Swathi Sree

Under the Supervision of,

Dr.Ruhin Kouser R
Assistant Professor – Senior Scale
School of Computer Science and Engineering
Presidency University

Name of the Program: **B.tech. Computer Engineering (AI&ML)**
Name of the HoD: **Dr.Pallavi R**
Name of the Program Project Coordinator: **Dr.Debasmita Mishra**
Name of the School Project Coordinators: **Dr. Sampath A K , Dr. Geetha A**

Content

- Source Code (100%)
- Implementation (100%)
- Completed Report
- Completed Research Paper

Source Code: Backend (db.py)

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker, declarative_base


DATABASE_URL = "sqlite:///./users.db"


engine = create_engine(
    DATABASE_URL, connect_args={"check_same_thread": False}
)


SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)


Base = declarative_base()

```

Source Code: Backend (main.py)

```
from fastapi import FastAPI, WebSocket, WebSocketDisconnect
from fastapi.middleware.cors import CORSMiddleware
from pydantic import BaseModel
import json


app = FastAPI()


# ---------------- CORS ----------------
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_methods=["*"],
    allow_headers=["*"],
)


# ---------------- MODERATION ----------------
BAD_WORDS = {
    "kill", "hate", "abuse", "stupid", "dumb", "idiot", "moron"
}


class Message(BaseModel):
    text: str


@app.post("/moderate")
def moderate(msg: Message):
    text = msg.text.lower()
    for word in BAD_WORDS:
        if word in text:
            return {"status": "BLOCKED"}
    return {"status": "SAFE"}
```

Source Code: Backend (main.py)

```
# ---------------- WEBSOCKET ----------------
connections = []


@app.websocket("/ws/chat")
async def chat_socket(ws: WebSocket):
    await ws.accept()
    connections.append(ws)
    print("🟢 Client connected")


    try:
        while True:
            data = await ws.receive_text()
            for c in connections:
                await c.send_text(data)
    except WebSocketDisconnect:
        connections.remove(ws)
        print("🔴 Client disconnected")


# ---------------- ROOT ----------------
@app.get("/")
def root():
    return {"status": "Backend running"}
```

Source Code: Backend (models.py)

```
from sqlalchemy import Column, Integer, String
from app.db import Base


class User(Base):
    __tablename__ = "users"


    id = Column(Integer, primary_key=True, index=True)
    username = Column(String, unique=True, index=True)
    password = Column(String)
```

Source Code: Frontend (admin.html)

```
<!DOCTYPE html>
<html>
<head>
<title>Admin Dashboard</title>
<style>
body { font-family: Arial; padding: 20px; }
.card { background:#f5f5f5; padding:20px; margin:10px; border-radius:10px; }
</style>
</head>
<body>

<h2>📊 Moderation Analytics</h2>

<div class="card" id="stats"></div>

<script>
async function loadStats() {
  const res = await fetch("http://127.0.0.1:8000/stats");
  const data = await res.json();

  document.getElementById("stats").innerHTML = `
    <p>Total Messages: <b>${data.total_messages}</b></p>
    <p>Blocked Messages: <b>${data.blocked_messages}</b></p>
  `;
}

loadStats();
</script>
</body>
</html>
```

Source Code: Frontend (crypto.js)

```
let currentUser = "alice";
let socket;


// ---------------- USER SWITCH ----------------
function setUser(user) {
  currentUser = user;
  connectWebSocket();
  document.getElementById("chatBox").innerHTML = "";
  document.getElementById("status").innerText =
    "Status: Logged in as " + user + " (open second tab for other user)";
}


// ---------------- WEBSOCKET ----------------
function connectWebSocket() {
  if (socket) socket.close();


  socket = new WebSocket(
    `ws://127.0.0.1:8000/ws/chat?user=${currentUser}`
  );


  socket.onopen = () => {
    document.getElementById("status").innerText = "Status: Connected";
  };


  socket.onmessage = (event) => {
    const data = JSON.parse(event.data);


    if (data.sender !== currentUser) {
      addMessage(`${data.sender}: ${atob(data.ciphertext)}`, false);
    }
  };
```

Source Code: Frontend (crypto.js)

```
  socket.onclose = () => {
    document.getElementById("status").innerText = "Status: Disconnected";
  };
}

// ---------------- KEY GENERATION (DEMO) ----------------
async function generateAndUploadKeys() {
  console.log("RSA keys generated (demo)");
  alert("RSA keys generated & uploaded (demo)");
}

// ---------------- SEND MESSAGE ----------------
async function sendEncrypted() {
  const text = document.getElementById("messageInput").value.trim();
  if (!text) return;

  // AI moderation check
  const res = await fetch("http://127.0.0.1:8000/moderate", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ text })
  });
  const result = await res.json();

  if (result.status === "UNSAFE") {
    alert("Blocked: " + result.reason);
    await fetch("http://127.0.0.1:8000/stats/update?blocked=true", { method: "POST" });
    return;
  }

  await fetch("http://127.0.0.1:8000/stats/update?blocked=false", { method: "POST" });

  const payload = {
    sender: currentUser,
    ciphertext: btoa(text),
    timestamp: new Date().toISOString()
  };
```

Source Code: Frontend (crypto.js)

```
  // Show encrypted payload
  document.getElementById("payloadBox").innerText =
    JSON.stringify(payload, null, 2);


  socket.send(JSON.stringify(payload));
  addMessage(`${currentUser}: ${text}`, true);


  document.getElementById("messageInput").value = "";
}


// ---------------- UI MESSAGE ----------------
function addMessage(text, isMe) {
  const msg = document.createElement("div");
  msg.className = "message " + (isMe ? "me" : "other");
  msg.innerText = text;
  document.getElementById("chatBox").appendChild(msg);
  msg.scrollIntoView();
}


// Auto start
connectWebSocket();
```

Source Code: Frontend (index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Secure AI Chat</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />

  <style>
    body {
      margin: 0;
      font-family: Arial, sans-serif;
      background: #efeae2;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
    }

    .chat-container {
      width: 420px;
      height: 90vh;
      background: #f0f0f0;
      display: flex;
      flex-direction: column;
      border-radius: 10px;
      box-shadow: 0 10px 30px rgba(0,0,0,0.2);
      overflow: hidden;
    }
```

Source Code: Frontend (index.html)

```
.header {
  background: #075e54;
  color: white;
  padding: 12px;
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.messages {
  flex: 1;
  padding: 15px;
  overflow-y: auto;
  background: #efeae2;
}

.message {
  max-width: 75%;
  padding: 10px;
  margin-bottom: 10px;
  border-radius: 10px;
  font-size: 14px;
  line-height: 1.4;
}

.me {
  background: #dcf8c6;
  margin-left: auto;
}

.other {
  background: white;
  margin-right: auto;
}
```

Source Code: Frontend (index.html)

```
.system {
  background: #ffe7a3;
  text-align: center;
  margin: 10px auto;
  border-radius: 8px;
  font-size: 13px;
}

.input-area {
  display: flex;
  padding: 10px;
  background: #f0f0f0;
  border-top: 1px solid #ccc;
}

.input-area input {
  flex: 1;
  padding: 10px;
  border-radius: 20px;
  border: 1px solid #ccc;
  outline: none;
}

.input-area button {
  margin-left: 10px;
  padding: 10px 16px;
  background: #075e54;
  color: white;
  border: none;
  border-radius: 20px;
  cursor: pointer;
}

.payload {
  background: black;
  color: #00ff00;
  font-size: 12px;
  padding: 10px;
  height: 120px;
  overflow-y: auto;
}
```

Source Code: Frontend (index.html)

```
  </style>
</head>

<body>

<div class="chat-container">
  <div class="header">
    <strong>Secure AI Chat</strong>
    <select id="userSelect" onchange="switchUser()">
      <option value="alice">Alice</option>
      <option value="bob">Bob</option>
    </select>
  </div>

  <div id="messages" class="messages"></div>

  <div class="input-area">
    <input id="messageInput" placeholder="Type a message..." />
    <button onclick="sendMessage()">Send</button>
  </div>

  <div class="payload" id="payloadBox">
    Encrypted payload will appear here...
  </div>
</div>

<script>
  let currentUser = "alice";
  const ws = new WebSocket("ws://127.0.0.1:8000/ws/chat");

  ws.onmessage = (event) => {
    const data = JSON.parse(event.data);
    addMessage(data.sender, data.text);
  };

  function switchUser() {
    currentUser = document.getElementById("userSelect").value;
  }
```

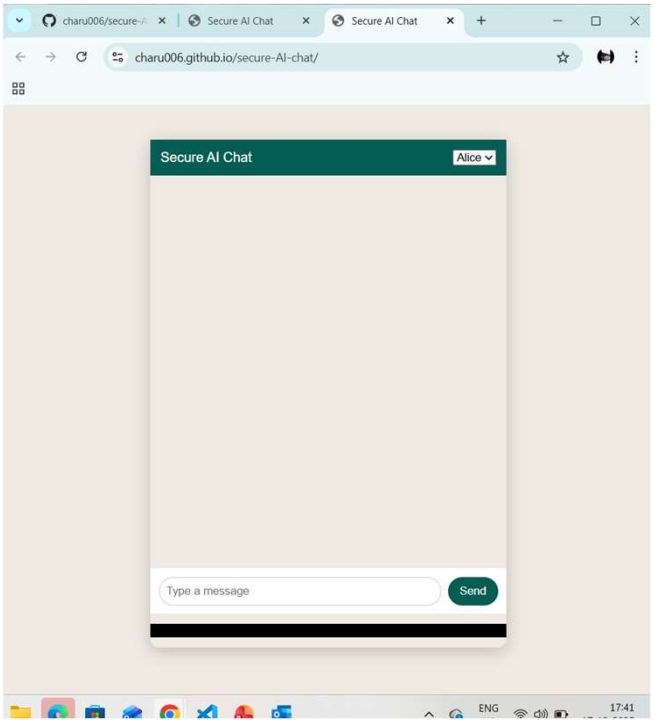
Source Code: Frontend (index.html)

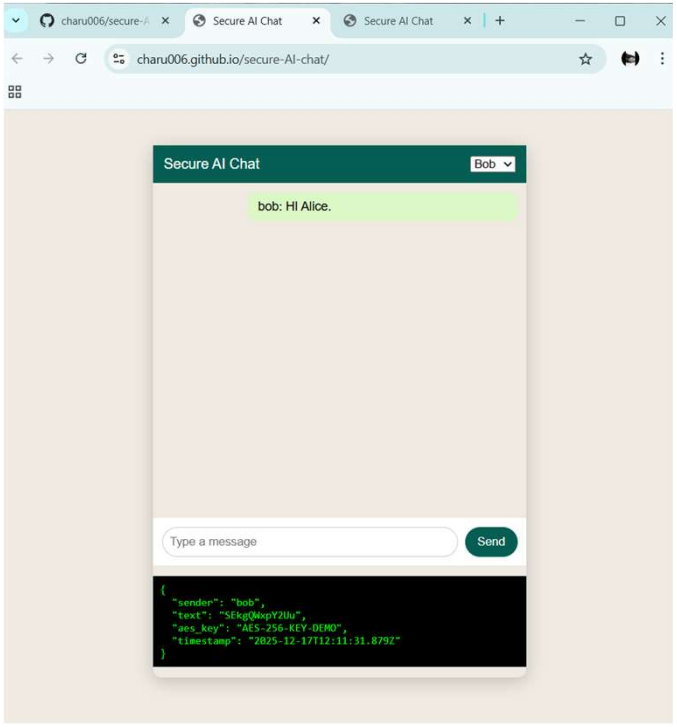
```
async function sendMessage() {
  const input = document.getElementById("messageInput");
  const text = input.value.trim();
  if (!text) return;

  // Moderation check
  const response = await fetch("http://127.0.0.1:8000/moderate", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ text })
  });

  const result = await response.json();

  if (result.status === "UNSAFE") {
    addSystemMessage("Message blocked by safety system");
    input.value = "";
    return;
  }

  // Encryption demo payload
  const encryptedPayload = {
    sender: currentUser,
    text: btoa(text),
    aes_key: "AES-256-KEY-DEMO",
    timestamp: new Date().toISOString()
  };

  document.getElementById("payloadBox").innerText =
    JSON.stringify(encryptedPayload, null, 2);

  ws.send(JSON.stringify({
    sender: currentUser,
    text: text
  }));

  input.value = "";
}
```

Source Code: Frontend (index.html)

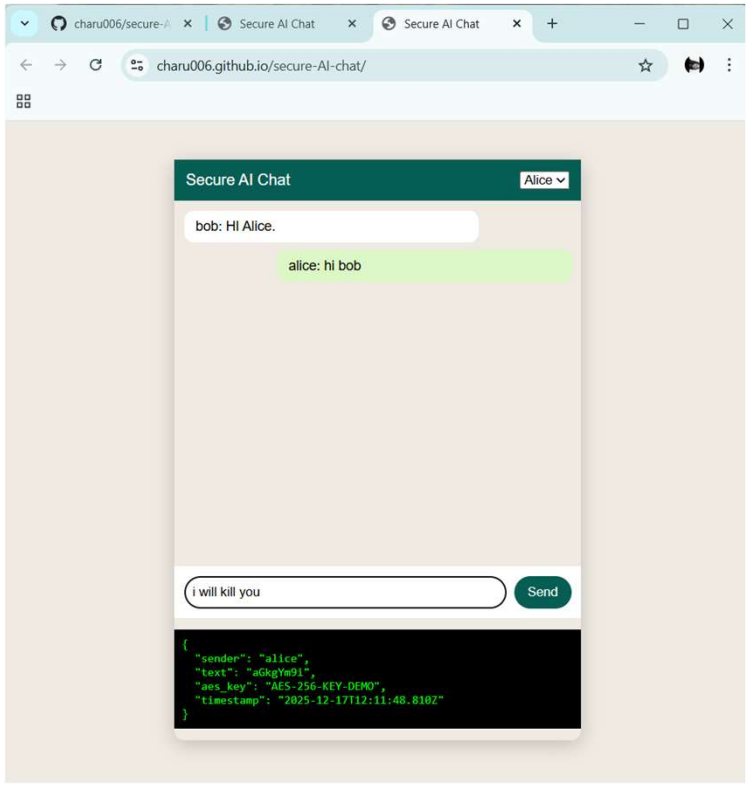
```
  function addMessage(sender, text) {
    const msgBox = document.getElementById("messages");
    const div = document.createElement("div");
    div.className = "message " + (sender === currentUser ? "me" : "other");
    div.innerText = sender + ": " + text;
    msgBox.appendChild(div);
    msgBox.scrollTop = msgBox.scrollHeight;
  }


  function addSystemMessage(text) {
    const msgBox = document.getElementById("messages");
    const div = document.createElement("div");
    div.className = "message system";
    div.innerText = text;
    msgBox.appendChild(div);
  }
</script>


</body>
</html>
```

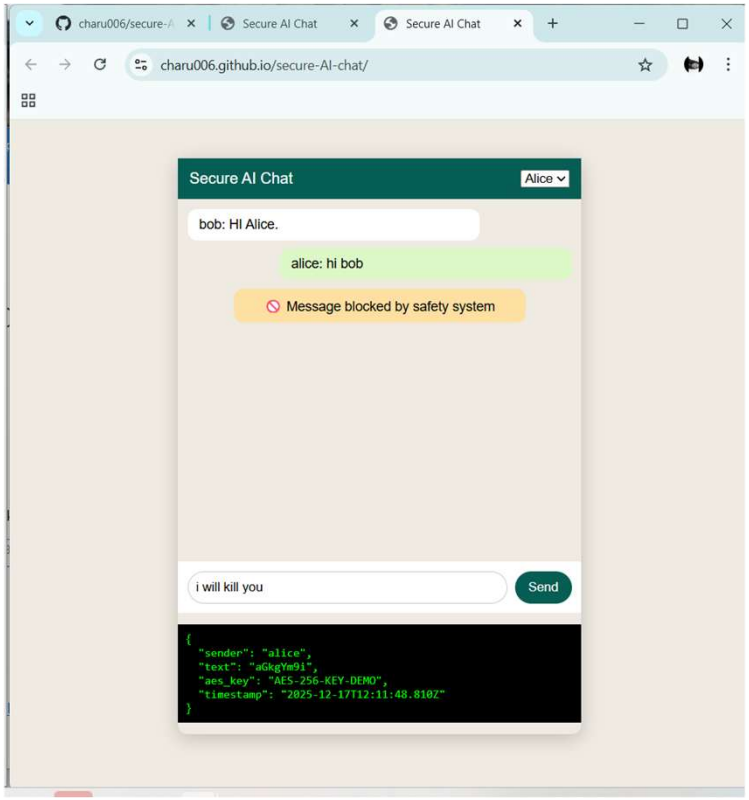
Implementation





Implementation






THANK YOU