EASTERN UNIVERSITY

COURSE TITLE : Artificial Intelligence Laboratory
COURSE CODE : CSE 432
PROJECT : SIMPLE CHAT APP

TYPE : Small Project
DATE : 14-NOV-2023

STUDENT NAME : Md. Kamruzzaman **ID**: 201400059 **SECTION** : 05(SUMMER)

SIMPLE CHAT APP
A PYTHON BASED CHAT APPLICATION

Contents List		Page
Chapter 1: Project Introduction		01
1.1	Introduction	
1.2	Objectives	
1.3	Outcome	
Chapter 2: Project Components		02-03
2.1	Chat Server	
2.2	Chat Client	
2.3	User Interface	
Chapter 3: Technology Stack		4
Chapter 4: Project Design		5
Chapter 5: Result and Conclusion		5
Chapter 6: Reference		6

SUBMITTED BY,
MD. Kamruzzaman
201400059 (CSE)

SUBMITTED TO,
Tazeen Tasnim
Assistant Professor
Faculty of E&T

1.1 Introduction

The Chat Application is a Python-based project that aims to create a simple and user-friendly chat application with features like customizable usernames, a graphical user interface (GUI), message sending and receiving, and a dynamic server IP configuration. The project consists of multiple components, including a Chat Server, Chat Client, and a User Interface built with Tkinter.

1.2 Objective

The primary objective of the project is to deliver a chat application with the following key features:

- Real-time communication
- User customization
- Server IP flexibility
- Robust error handling
- Integration with Chat Client
- Engaging user interface
- System commands for AI responses and weather information.

1.3 Expected Outcomes

The project aims to provide an engaging and customizable chat experience with a reliable infrastructure, facilitating seamless communication and interaction for users.

2.1 Chat Server

The Chat Server is responsible for managing client connections, handling messages, and providing a platform for users to interact. It offers the following features:

Socket Communication: The server creates a socket to communicate with connected clients.

Server Configuration: Parameters like header size, format, disconnect message, IP address, and port can be customized.

Graceful Shutdown: The server can be stopped gracefully when a SIGINT (Ctrl+C) signal is received.

Message Handling: It handles incoming messages from clients and processes specific commands like `/Jarvis` for AI responses and `/Weather` for weather information also added later on `/Guess` for a simple guessing game with leaderboard.

Dynamic IP Option: The server can run with either a dynamically determined or custom IP address, providing flexibility.

2.2 Chat Client

The Chat Client is the counterpart of the server, responsible for client-side communication. Its features include:

Socket Communication: The Chat Client class uses sockets to connect to the server.

Custom Server IP: Users can provide a custom server IP address.

Background Thread: A background thread is employed for receiving and processing incoming messages.

Message Callback: Users can define a custom function to handle incoming messages, allowing interaction with the user interface.

Message Sending: The client can send messages to the server, which are tagged with the sender's username.

Disconnect Functionality: The client can send a disconnect message to the server and close the socket connection.

Error Handling: The client script includes error handling for various scenarios, enhancing reliability.

2.3 User Interface

The User Interface is built using the Tkinter library, offering a visually appealing environment for users to interact with the chat application. Its features comprise:

Graphical User Interface (GUI): The application includes various elements, such as username input, message display, message input, and buttons for sending messages and disconnecting.

Username Submission: Users can submit their usernames, after which the input field and submit button become disabled.

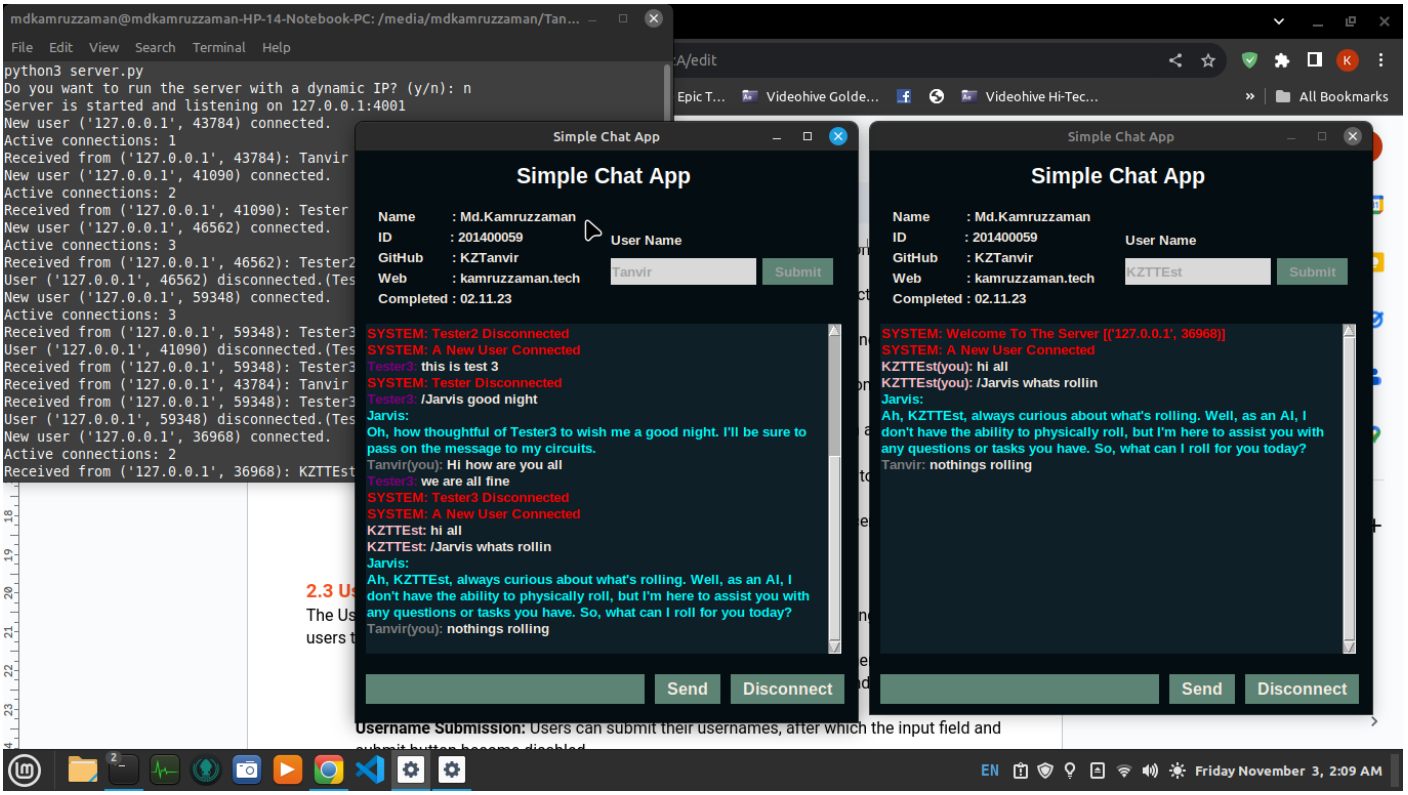
Message Display: Messages are displayed in a scrollable area, with different styles for user and system messages.

Message Formatting: Messages sent by the user include the sender's username and are tagged with a specific style.

Disconnect Button: Users can click the "Disconnect" button to gracefully exit the chat application.

Random User Color: Each user is assigned a random color for message differentiation.

Server Configuration: Users can enter the server's IP address when the application starts, making it adaptable for different servers.



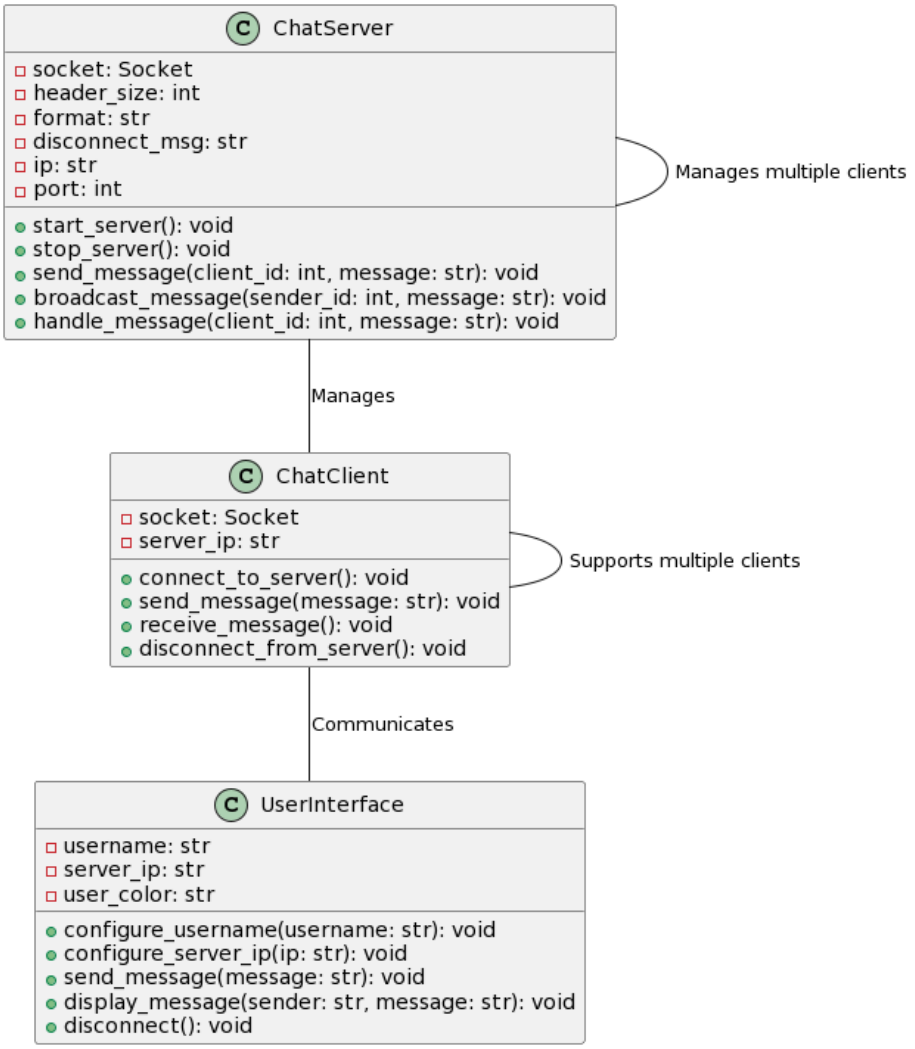
NOTE: Works on both localhost and internet if no custom host is given default is localhost.

The technology stack for the Chat Application includes the following components and technologies:

1. **Programming Language**:
   Python
2. **Server-Side**:
   Sockets for server-client communication
   Threading for handling multiple client connections
   Standard Python libraries for socket communication
3. **User Interface**:
   Tkinter for building the graphical user interface (GUI)
4. **Version Control**:
   Git and GitHub for version control and collaborative development
5. **Development Environment**:
   Integrated Development Environment (IDE) such as Visual Studio Code, PyCharm, or any preferred Python development environment

The UML(Unified Modeling Language) for the whole project is given below:

```
@startuml ChatApplication

class ChatServer {
 - socket: Socket
 - header_size: int
 - format: str
 - disconnect_msg: str
 - ip: str
 - port: int

 + start_server(): void
 + stop_server(): void
 + send_message(client_id: int, message: str): void
 + broadcast_message(sender_id: int, message: str): void
 + handle_message(client_id: int, message: str): void
}

class ChatClient {
 - socket: Socket
 - server_ip: str

 + connect_to_server(): void
 + send_message(message: str): void
 + receive_message(): void
 + disconnect_from_server(): void
}

class UserInterface {
 - username: str
 - server_ip: str
 - user_color: str

 + configure_username(username: str): void
 + configure_server_ip(ip: str): void
 + send_message(message: str): void
 + display_message(sender: str, message: str): void
 + disconnect(): void
}

ChatServer -- ChatClient: Manages
ChatClient -- UserInterface: Communicates
ChatServer -- ChatServer: Manages multiple clients
ChatClient -- ChatClient: Supports multiple clients
@enduml
```



Result

The Chat Application project delivered:

- A user-friendly chat platform
- A modular structure
- Robust communication
- Custom server configuration

Conclusion

The project offers an adaptable, interactive, and feature-rich chat platform, a foundation for customized applications.

References:

Stallings, W. (2014). Data and Computer Communications (8th ed.). Pearson.

Berry, P. (2016). Head First Python. O'Reilly Media.

Tkinter Documentation. (n.d.). Retrieved from [https://docs.python.org/3/library/tk.html]

OpenAI Playground. (n.d.). Retrieved from [https://playground.openai.com]

Accuweather. (n.d.). Retrieved from [https://developer.accuweather.com/apis] for API access.

Stack Overflow. (n.d.). How do you use threading on sockets for printing safely? Retrieved from https://stackoverflow.com/questions/53772738/how-do-you-use-threading-on-sockets-for-printing-safely

Special thanks to the following courses and lecturers for their valuable insights on socket communication and related topics:
- Amrito Biswas (Computer Networking Laboratory)
- Razia Sultana (Computer Networking)
- Tasniah Mohiuddin (OOP Laboratory)
- Tanzim Tamanna Shitu (Design Analysis Algorithm)
- Tazeen Tasneem (Data Structures Laboratory, Operating System)