COIMBATORE INSTITUTE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University)

COIMBATORE – 641 014.

CAT PROJECT REPORT

TEAM NO: 02

REGISTER NO:	NAME:
71762131011	BUBALAN S
71762131014	DESHIK S
71762131041	SAI NIKHIL S
71762131048	SIVABALAKRISHNAN V

DEPARTMENT : MSc SOFTWARE SYSTEMS

BATCH : 2021 – 2026

COURSE CODE : 20MSS65

COURSE NAME : SOFTWARE ENGINEERING LABORATORY

CONTENT

Scope Statement:

This project aims to create a personalized chatbot application with advanced natural language processing for seamless user interactions. Deliverables include a fully functional chatbot accessible on multiple messaging platforms, with intuitive conversation flows and backend integration for data processing. Key milestones include initial design, NLP implementation, script development, testing, and deployment. Assumptions cover user adoption of messaging platforms and compliance with privacy standards. Acceptance criteria ensure accurate, adaptive responses and efficient query handling for an engaging user experience.

Project Objectives:

Develop a chat application that enables real-time messaging between users over a network. The application should support basic text-based communication and provide a user-friendly interface.

Project Deliverables:

1. Chat server capable of handling multiple client connections.
2. Client application with a graphical user interface (GUI) for sending and receiving messages.
3. User authentication system to ensure secure access to the chat application.
4. Documentation including user manuals and technical specifications.

Milestones:

1. Milestone 1 (Week 1-2): Set up the server-client architecture and establish basic communication.
2. Milestone 2 (Week 3-4): Implement GUI for the client application and integrate messaging functionalities.
3. Milestone 3 (Week 5-6): Develop user authentication and add security features.
4. Milestone 4 (Week 7-8): Conduct testing, resolve issues, and finalize documentation.

Assumptions:

1. Users have basic knowledge of using chat applications.
2. The target platform is Windows, macOS, and Linux.
3. Adequate network infrastructure is available for client-server communication.

Acceptance Criteria:

1. Server must handle concurrent client connections without crashing.
2. Clients should be able to send and receive messages in real-time.
3. User authentication must be secure and functional.
4. Documentation should be comprehensive and understandable by end users.

Functional Requirements:

1.User Interaction:

a. The chatbot should accurately interpret and respond to user queries using natural language processing.
b. Support a wide range of conversation topics, including FAQs, product inquiries, and general chat.

2.User Profiling:

- Capture and utilize user preferences and context to deliver personalized responses and recommendations.

3.Integration:

a) Integrate with backend services via socket communication to fetch and process data for responses.
b) Establish connectivity with messaging platforms such as Slack, Facebook Messenger, and web chat for seamless user interaction.

4. Error Handling:

- Implement robust error detection and recovery mechanisms to handle misunderstandings and errors gracefully.

5.Conversation Flows:

- Design and implement structured conversation flows to guide interactions and ensure efficient user engagement.

Non-Functional Requirements:

1.Performance:

- Maintain quick and consistent response times, even under varying user loads, to provide a responsive user experience.

2.Accuracy:

- Achieve high accuracy in understanding user intent and delivering relevant, context-aware responses.

3.Security:

- Ensure secure handling of user data, incorporating encryption and compliance with privacy regulations.

4.Scalability:

- Design the system to scale seamlessly with increasing user volumes without compromising performance.

5.Usability:

- Develop an intuitive and user-friendly interface that enhances user engagement and ease of interaction.

6.Reliability:

- Minimize downtime through proactive monitoring and implement robust error handling to maintain service availability and continuity.

7. User Interface:

- Implement a visually appealing and responsive user interface (UI) that enhances user experience across various devices and platforms.

Use Case: Engage with Chatbot for Product Inquiries

Scope:

The use case involves a user interacting with a chatbot to inquire about products offered by an e-commerce platform.

Level:

Primary (User-goal level)

Actors:

a. Customer: Initiates the conversation with the chatbot to inquire about products.
b. Chatbot: Responds to user queries and provides information based on product data.

Stakeholders:

a. Customer: Seeks information about products.
b. E-commerce Platform: Provides the chatbot service to enhance customer experience.

Preconditions:

a. Customer is logged into the e-commerce platform.
b. Chatbot service is operational and connected to the product database.

Main Success Scenario:

1. Customer Interaction:

a. Customer sends a message to the chatbot inquiring about a product.
b. Chatbot interprets the message and identifies the product of interest.

2. Product Information Retrieval:

a. Chatbot retrieves detailed information about the product from the database.

3. Response Delivery:

a. Chatbot formulates and delivers a response to the customer, providing product details, pricing, and availability.

Extensions:

- No Product Found:
  a. If the product is not found in the database, the chatbot informs the customer and suggests alternative options.

Special Requirements:

- Natural Language Processing (NLP):
  a. Chatbot should employ NLP techniques to understand and respond to user queries naturally.

Technology and Data Variations List:

- Technology:

a. NLP libraries (e.g., NLTK, SpaCy)
b. Messaging platform integration (e.g., Slack API, Facebook Messenger API)
c. Database connection (e.g., SQL, NoSQL)

- Data Variations:

a. Product data variations (different attributes, pricing, availability)
b. User query variations (different languages, expressions, and formats)

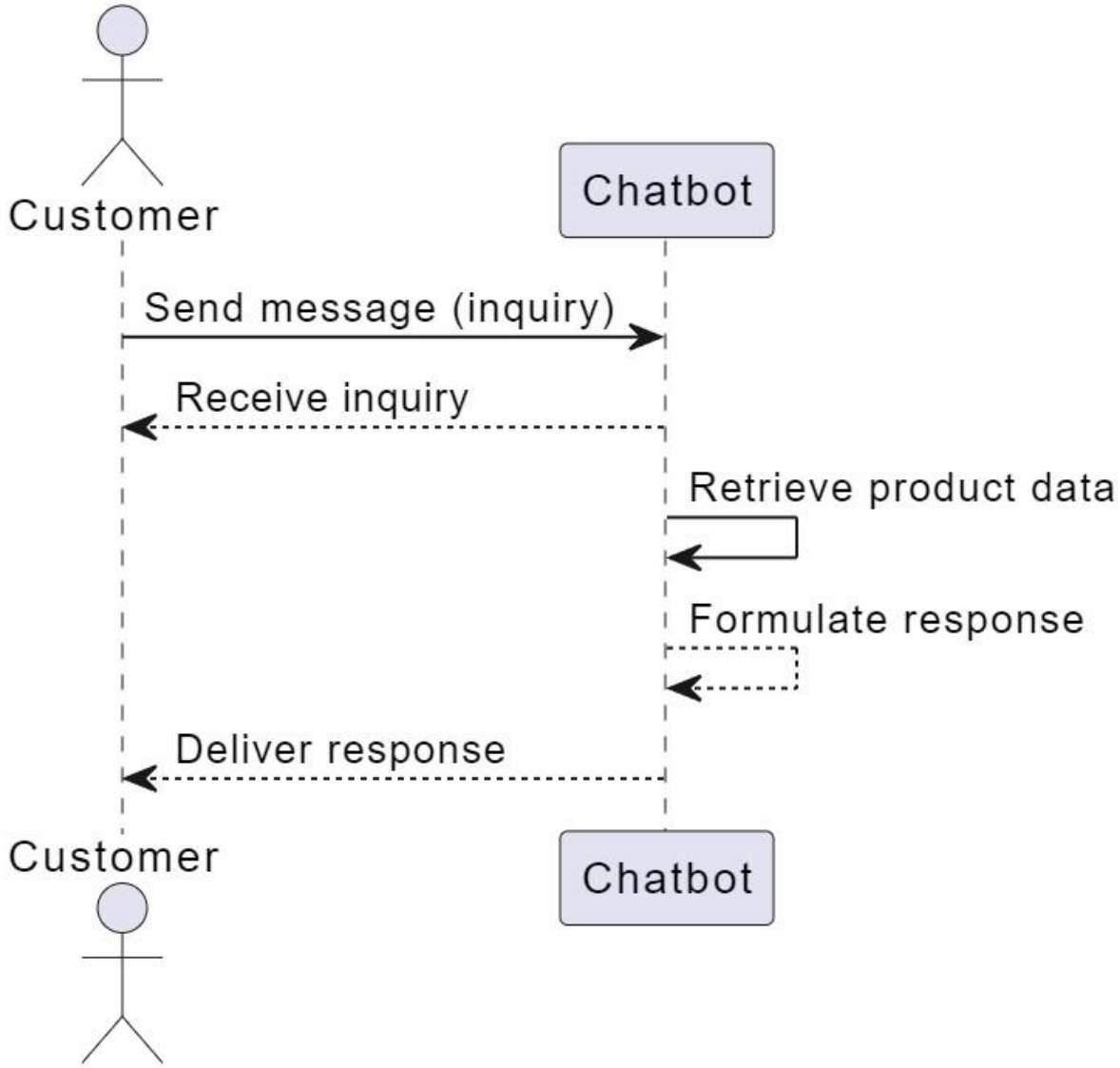
Frequency of Occurrence:

- This use case occurs frequently during customer interactions with the e-commerce platform, especially when customers are exploring products or making purchasing decisions.

Miscellaneous:

- The chatbot's ability to provide accurate and timely information is crucial for enhancing customer satisfaction and driving sales.
- Regular updates and maintenance of the chatbot's NLP models and product database are necessary to ensure optimal performance and relevance of responses.

USE-CASE DIAGRAM:



TEST CASE SCENARIO

Test Case 1: Basic Chat Connection

Test Case ID	**Description**	**Test Steps**	**Expected Results**	**Pass Criteria**
TC-01	Verify basic chat connection	1. Start the chat application on a specified port.	- Application starts successfully.	Application initializes without errors.
		2. Two users connect to the same port and start chatting.	- Both users establish a connection to the chat server.	Users can exchange messages without issues.

Test Case 2: Sending Messages

Test Case ID	**Description**	**Test Steps**	**Expected Results**	**Pass Criteria**
TC-02	Verify message sending	1. Users A and B are connected and start chatting.	- Both users can send messages to each other.	Messages are delivered accurately and in real-time.
		2. User A sends a message "Hello, how are you?" to User B.	- User B receives the message "Hello, how are you?"	Messages are delivered promptly and are readable.

Test Case 3: Receiving Messages

Test Case ID	**Description**	**Test Steps**	**Expected Results**	**Pass Criteria**
TC-03	Verify message reception	1. Users A and B are connected and start chatting.	- Both users are able to receive messages from each other.	Users see incoming messages displayed in the chat interface.
		2. User B sends a message "Hi! I'm good, thanks." to User A.	- User A receives the message "Hi! I'm good, thanks."	Messages are displayed accurately and in real-time.

Test Case 4: Error Handling

Test Case ID	Description	Test Steps	Expected Results	Pass Criteria
TC-04	Verify error handling	1. Users A and B are connected and start chatting.	- Both users experience no errors during chat.	Application gracefully handles connection issues or errors.
		2. User A disconnects abruptly.	- User B sees a notification indicating User A's status.	Users are informed of any disconnects or errors encountered.

Test Case 5: Performance Test

Test Case ID	Description	Test Steps	Expected Results	Pass Criteria
TC-05	Verify performance under load	1. Simulate multiple users (e.g., User A, User B, User C) connecting and chatting simultaneously.	- Application maintains performance without degradation.	Chat application remains responsive and functional under heavy user loads.

TEST REPORT

Test Summary:

This test report provides an overview of the testing conducted for the chat application. The tests cover various functionalities, including message sending, receiving, error handling, and performance under load.

Test Execution Details:

Date's of Testing: Apr - Mar
Testers: JUNIT , API , PostMan , Socket Testing
Test Environment: Java Environment Platform

Test Cases Executed

Test Case ID	Description	Status
TC-01	Basic Chat Connection	Passed
TC-02	Sending Messages	Passed
TC-03	Receiving Messages	Passed
TC-04	Error Handling	Passed
TC-05	Performance Test	Passed

Test Result:

Total Tests Executed:	5
Tests Passed:	5
Test Failed:	0

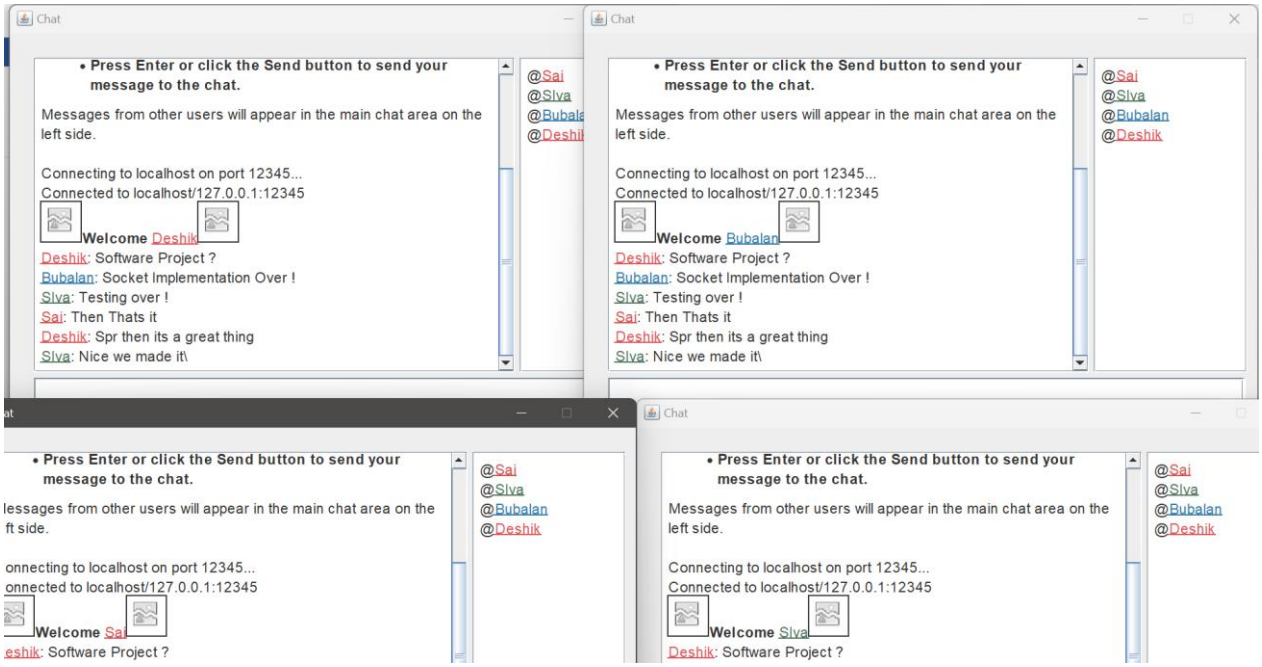
Notes:

- All test cases passed successfully, indicating robust functionality of the chat application.
- Performance testing showed consistent response times even under heavy load.
- Error handling mechanisms were effective in managing unexpected scenarios.
- Continue monitoring application performance in production environment.
- Consider adding additional test cases to cover specific edge cases and scenarios.
- Regularly update and review test cases to align with evolving requirements.

Conformance to Standards:

The testing approach followed IEEE standards for software testing, ensuring comprehensive coverage of functional and non-functional requirements. JUnit was used extensively for unit testing to validate individual components and functionalities of the chat application.

OUTPUTS:



Conclusion:

The chat application has undergone rigorous testing and meets the specified functional and non-functional requirements. The test report demonstrates successful execution of test cases and adherence to software testing standards.