

**MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**

**BACHELOR OF ENGINEERING IN SOFTWARE ENGINEERING**

**ASSIGNMENT TITLE**: TASK MANAGEMENT SYSTEM

**SUBMITTED BY ASADULLALH MALIK(074)**

**NABEEL(38)**

**USMAN HASHMI(141)**

**SHAHZAD(156)**

# SUBMITTED TO:MR. ARSALAN

# DEPARTMENT OF SOFTWARE ENGINEERI

### Abstract

This project report presents the development of a versatile chatbot that integrates multiple APIs to provide users with information on movies, dictionary definitions, news, and weather updates. The primary objective of the project is to create a user-friendly chatbot capable of handling diverse queries efficiently. The chatbot leverages the Movies API to fetch detailed information about films, the Dictionary API to provide word definitions and synonyms, the News API to deliver the latest news articles, and the Weather API to offer current weather conditions and forecasts.

The system architecture is designed using a modular approach, ensuring scalability and ease of maintenance. The implementation involves using Python and Flask for the web framework, with each API integrated through HTTP requests. The chatbot's functionalities are tested through various test cases to ensure accurate and timely responses.

Challenges such as API rate limits were addressed by implementing caching mechanisms and optimizing API calls. The project demonstrates the feasibility and effectiveness of using APIs to enhance chatbot functionalities. Future improvements could include adding more APIs, enhancing the user interface, and incorporating machine learning for better query understanding.

**Project Report**

**Project title : Chatbot with Multiple APIs**

**1. Introduction**

The project involves developing a chatbot that integrates multiple APIs to provide users with information on movies, dictionary definitions, news, and weather updates. The main objective is to create a versatile and user-friendly chatbot that can handle various queries efficiently.

**2. APIs Used**

* **Movies API**: This API fetches information about movies, including details like title, genre, release date, and ratings. It helps users find information about their favorite movies.
* **Dictionary API**: This API provides definitions, synonyms, and antonyms for words. It enhances the chatbot's ability to assist users with vocabulary and language queries.
* **News API**: This API retrieves the latest news articles from various sources. It keeps users updated with current events and trending topics.
* **Weather API**: This API provides current weather conditions, forecasts, and alerts. It helps users stay informed about the weather in their location or any other specified area.

**3. System Architecture**

* **Overall Design**: The chatbot is designed using a modular architecture where each API is integrated as a separate module. This ensures scalability and ease of maintenance.
* **Components**: The main components include the user interface, API integration modules, and the core chatbot logic that processes user queries and fetches data from the APIs.

**4. Implementation**

* **Development Environment**: The project is developed using QT, with Flask for the web framework and various libraries for API integration.
* **Code Structure**: The codebase is organized into modules for each API, a main script for the chatbot logic, and templates for the user interface.
* **Integration**: Each API is integrated using HTTP requests. The responses are parsed and formatted before being presented to the user.

**5. Features**

* **User Interaction**: Users interact with the chatbot through a web interface where they can type their queries.
* **Functionalities**: The chatbot can provide movie details, dictionary definitions, latest news, and weather updates based on user queries.

**6. Testing**

* **Test Cases**: Various test cases were created to ensure the chatbot handles different types of queries correctly. For example, testing movie queries with different titles, checking dictionary definitions for various words, etc.
* **Results**: The chatbot successfully handled most queries, with accurate and timely responses from the integrated APIs.

**7. Challenges and Solutions**

* **Challenges**: One of the main challenges was handling API rate limits and ensuring the chatbot remained responsive.
* **Solutions**: Implemented caching mechanisms and optimized API calls to reduce the load and improve performance.

**8. Conclusion**

* **Summary**: The chatbot project successfully integrated multiple APIs to provide a wide range of information to users. It demonstrated the feasibility and effectiveness of using APIs to enhance chatbot functionalities.
* **Future Work**: Future improvements could include adding more APIs, enhancing the user interface, and implementing machine learning for better query understanding.

**9. Appendices**

* **Code Snippets**:
* # Example of integrating the Movies API
* import requests
* def get\_movie\_details(title):
* response = requests.get(f"https://api.example.com/movies?title={title}")
* return response.json()
* **References**: List of API documentation and other resources used.