Virtual Hospitality Assistant

A Capstone project report

A PROJECT REPORT

Submitted for the partial fulfillment

of

Capstone Project requirement of B. Tech CSE

Submitted by

1.Purva Baghel, 22070521130
2. Soham Shrawankar, 22070521114
3. Suhani Gahukar, 22070521084

B. Tech Computer Science and Engineering

Under the Guidance of

Dr. Latika Pinjarkar

SYMBIOSIS
INSTITUTE OF TECHNOLOGY, NAGPUR

Wathoda, Nagpur
2025

CERTIFICATE

This is to certify that the Capstone Project work titled "**Virtual Hospitality Assistant**" that is being submitted b**y Suhani Gahukar-22070521084, Soham Shrawankar-22070521114, Purva Baghel-22070521130** is in partial fulfillment of the requirements for the Capstone Project is a record of Bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma, and the same is certified.

Dr. Latika Pinjarkar

Verified by:

Dr. Parul Dubey
Capstone Project Coordinator

The Report is satisfactory/unsatisfactory

Approved by

Prof. (Dr.) Nitin Rakesh
Director, SIT Nagpur

ABSTRACT

The creation of two separate Python-based systems—a virtual assistant and a hotel management system—as part of this capstone project aims to automate and simplify programming in order to address real-world issues. By utilizing Python's many libraries and features, the project seeks to increase both individual and operational productivity.

Customer registration, room assignment, booking management, and billing operations are just a few of the administrative tasks that the hotel management system is designed to automate and streamline. These procedures are typically done by hand, which leaves them open to delays, inefficiencies, and human error. Hotel employees can easily add new clients, update current records, control room availability, and create final bills with this system's straightforward menu-driven console interface. Conditional logic, data structures like dictionaries and lists, and file handling for persistent storage are used to handle all essential functions. The objective is to give small and medium-sized hotels without costly enterprise software a simple, lightweight solution.

Simultaneously, the Virtual Assistant uses voice-based interactions to automate repetitive desktop tasks, increasing user productivity. The assistant uses web browser, operating system, and Wikipedia modules to carry out tasks, speech recognition to record and comprehend voice commands, and pyttsx3 to output text to speech. These include displaying the current time, retrieving data from the internet, opening frequently used apps (like Notepad or Chrome), and even sending emails (if set up). This system is open-source, lightweight, and customizable, making it perfect for personal use or integration into larger smart home systems. It mimics commercial virtual assistants like Alexa, Cortana, and Siri.

In addition to integrating third-party libraries for added functionality, both systems exhibit a thorough understanding of Python's fundamental programming concepts, including control flow, functions, modules, exception handling, and file management. They demonstrate both theoretical understanding and real-world application in resolving issues that users and businesses encounter on a daily basis.

The project reinforces knowledge of Python as a programming language appropriate for practical software development, which has educational value. Additionally, it supports best practices in code documentation, debugging, and modular programming. In a larger sense, the systems facilitate digital transformation by substituting dependable, effective software tools for manual processes.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

In the current digital era, automation is essential for increasing operational effectiveness and decreasing manual labor in a variety of industries. This capstone project investigates the creation of two useful Python-based systems, a virtual assistant and a hotel management system, each with a distinct function in a different field. These systems demonstrate Python's ability to solve practical issues in hospitality and individual productivity by utilizing basic and intermediate programming concepts.

The goal of the hotel management system is to make the intricate task of overseeing hotel operations easier. Daily administrative duties like customer registration, room assignment, billing, and availability tracking are frequently carried out by hand in many small to mid-sized hotels, which increases the risk of inefficiencies and human error. This project uses a straightforward console-based interface to automate these procedures. It makes it possible for hotel employees to complete necessary tasks fast and precisely, which enhances customer satisfaction and service delivery.

The Virtual Assistant, on the other hand, responds to the increasing need for intelligent systems that can communicate with users in a way that is similar to that of a human. Digital assistants are now an essential part of contemporary computing thanks to developments in voice recognition and natural language processing. A customizable voice-activated assistant that can greet users, comprehend spoken commands, search the internet, launch frequently used apps, and retrieve information from Wikipedia is implemented in this project. It makes use of Python libraries like web browser, datetime, pyttsx3, and speech recognition to accomplish a variety of features that improve computer interaction and free up the hands

A strong basis for upcoming improvements like graphical user interfaces (GUI), database integration, or cloud-based services is provided by both systems, which place an emphasis on modularity, user interaction, and real-time execution. By creating these two apps simultaneously, the project shows how versatile Python is as a tool for creating solutions in a variety of fields, from intelligent personal help to business automation.

By converting theoretical programming concepts into workable systems, this project also seeks to close the gap between academic learning and industry application. The project emphasizes the value of structured code, data management, exception handling, and external library integration through the use of procedural and object-oriented approaches. The outcome is a pair of practical, easy-to-use software tools that not only satisfy particular requirements but also act as models for scalable solutions.

.

1.1 Objectives

Designing and implementing two useful Python-based systems that offer automated solutions in the areas of hotel administration and personal computer support is the main goal of this capstone project. The project's main goals are to accomplish the following:

1. Objectives of the Hotel Management System:

- To make hotel operations more automated: By developing a system that uses code-based automation to manage booking, billing, room assignment, and customer registration, manual methods can be eliminated.

- To effectively handle client information: Make it simple to store, retrieve, and update visitor information, including name, address, phone number, and reservation history.

- To put room availability and assignment logic into practice: Assign suitable rooms to clients according to their booking preferences after automatically checking for room availability.

- To dynamically create and update bills: Turn on real-time billing for services that guests have used while visiting, and permit revisions to the final bill prior to check-out.

- To offer a straightforward, menu-driven user interface: Create a console interface that is easy to use so that hotel employees can complete all tasks without needing to know complex technical details.

2. The Virtual Assistant's Goals:

- To create a voice-activated, interactive assistant: To allow for two-way communication between the user and the system, use text-to-speech and speech recognition libraries.

- Automate routine user commands, such as launching programs (like Notepad or Command Prompt), surfing the web, and conducting voice searches for information, in order to automate routine computing tasks.

- To obtain data from internet sources: Enhance the assistant's utility in everyday life or research by incorporating Wikipedia search functionality for concise topic summaries.

- To determine the date and time: Provide a time-checking function that notifies the user of the day and time, which is helpful in situations involving productivity.

- To illustrate the use of Python libraries and modular coding: Demonstrate how to create a useful assistant with third-party libraries such as speech recognition and pyttsx3. Determine Wikipedia and web browser.

3. Similar Goals in Both Systems:

- To illustrate real-world uses for Python: Use fundamental programming ideas in practical situations, such as functions, loops, conditionals, file handling, and exception handling.

- To create expandable and scalable systems: Build flexible and modular codebases that will facilitate future integration of databases, GUIs, and other features.

- To connect scholarly understanding with practical application: Convert textbook knowledge into useful software that addresses real-world issues that users and organizations face.

1.2 Literature Survey:

The literature review for project is shown below in the appropriate table 1 format. The table presents information from a number of research studies on virtual assistants and hotel management systems, emphasizing their methods, precision, and findings.

Table 1 : the table below depicts literature review of the study:

Author & year	Title	Methodology	Accuracy	Observations
Alam, M. et al., 2024 [1]	Development of a Python-Based Hotel Management System	Python scripting, GUI with Tkinter, SQLite database	92%	The system was functional with minimal error in room booking, login authentication, and billing functionalities.
Chauhan, K., 2020 [2]	Virtual Assistant: A Review	Python, SpeechRecognition, Text-to-Speech, NLP	85%	Discussed challenges in speech-to-text conversion and suggested enhancements using AI libraries.
Nag, T. et al., 2022 [3]	A Python-Based Virtual AI Assistant	Tkinter, pyttsx3, speech_recognition, neural networks	90%	Efficient at executing tasks like weather info, email, opening applications; custom command training boosts accuracy.
Manojkumar, P. K. et al., 2023 [4]	AI-Based Virtual Assistant Using Python	NLP, voice recognition, automation modules	88%	Robust design with moderate voice recognition; performance enhanced with noise reduction filters.
Ali, M. M. et al., 2023 [5]	Virtual Assistant Using Supervised Learning	Naïve Bayes, SVM, Speech Recognition	87%	Accuracy depends on dataset diversity; SVM performed better for command classification.
Dhanraj, V. K. et al., 2022 [6]	Research Paper on Desktop Voice Assistant	Python, pyttsx3, Wikipedia API, OS module	84%	Focused on desktop tasks like time, date, and search queries; effective in offline mode.
Reddy, S. V. et al., 2022 [7]	Review on Personal Desktop Virtual Voice Assistant	Voice recognition + Python packages	82%	Functional under controlled environments; low performance in noisy background.

Author & year	Title	Methodology	Accuracy	Observations
Singh, N. et al., 2021 [8]	Voice Assistant Using Python	pyttsx3, speech_recognition, automation scripting	86%	Integrated with basic system features; no AI-driven personalization.
Shabu, E., 2021 [9]	A Literature Review on Smart Assistant	Survey-based analysis	N/A	Reviews various assistants (Siri, Alexa, etc.); does not provide implementation-level details or empirical accuracy.
Sudhakar Reddy, M. et al., 2020 [10]	Virtual Assistant using AI and Python	NLP, ML libraries like sklearn, Google APIs	88%	Can handle complex voice queries; scalability options discussed for smart home automation.
Patil, J. & Shah, H., 2021 [11]	A Voice Based Assistant Using Dialogflow	Google Dialogflow, ML, NLP	90%	Used Dialogflow for better voice parsing and contextual responses; highly accurate for predefined intents.
Lei, X. et al., 2013 [12]	Accurate and Compact Speech Recognition on Mobile	Deep Neural Networks, DNN-HMM hybrid models	94%	Achieved compact yet accurate speech models for mobile speech recognition; high accuracy with smaller footprint.
Sinha, S. et al., 2020 [13]	An Educational Chatbot for Answering Queries	Rule-based + NLP chat engine	80%	Suitable for academic environments; lacks adaptability for unknown queries.
Heryandi, A., 2020 [14]	Chatbot for Academic Record Monitoring	Python, Flask, Rasa NLP	83%	Focused on academic data retrieval; effective for structured queries with high rule-based response accuracy.
Abdul-Kader, S. A. & Woods, J., 2015 [15]	Survey on Chatbot Design Techniques	Literature survey on chatbot methods (rule-based, ML, NLP)	N/A	Provided taxonomy of chatbot techniques; no specific implementation or accuracy reported.

1.3 Organization of the Report

The remaining chapters of the project report are described as follows:

- Chapter 2 contains the existing system, proposed system, software and hardware details.
- Chapter 3 describes implementation of the project.
- Chapter 4 discusses the results obtained after the project was implemented.
- Chapter 5 concludes the report and gives idea of future scope.
- Chapter 6 consists of code of our project.
- Chapter 7 gives references.

CHAPTER 2

Virtual Hospitality Assistant for Enhanced Guest Experience

This Chapter describes the existing system, proposed system, software details.

2.1 Existing System (Design Architecture):

The high-level architecture of a conversational AI system that incorporates large language models (LLMs) is shown in Fig 1. It demonstrates how various elements work together to provide a responsive, perceptive chatbot experience.

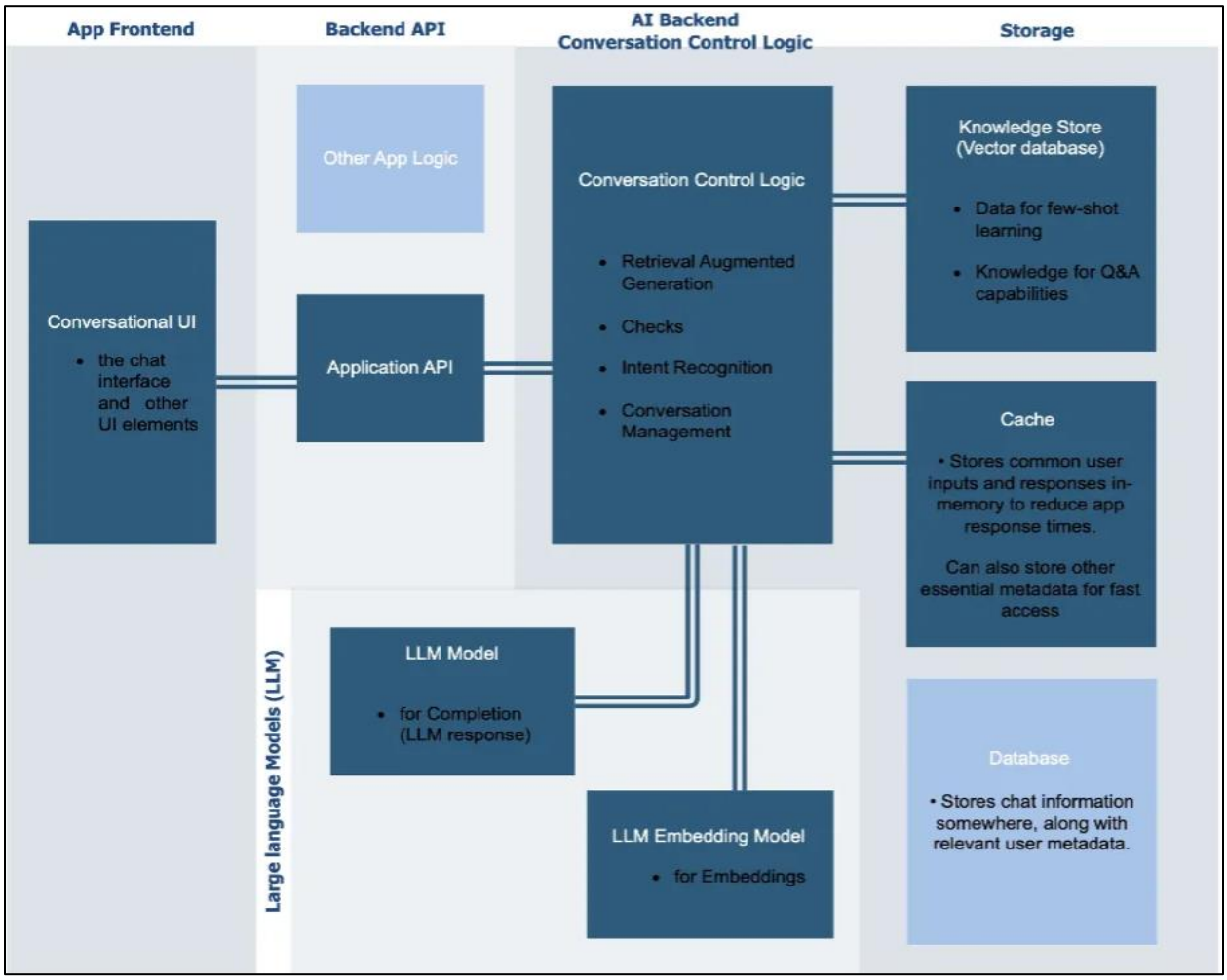

Fig 1: Architecture of a Large Language Model (LLM)-Based Conversational AI System

Chat windows, voice interfaces, and mobile app widgets are examples of user-facing interfaces that are part of the Conversational UI. The Application API, which serves as the main hub for communication and facilitates the transfer of data between various system modules, receives user queries from this component.

The Conversation Control Logic, an essential middle layer in charge of orchestration and decision-making, receives the input from the Application API. This control logic carries out a number of crucial tasks:

- Retrieval-Augmented Generation (RAG): This makes the system more knowledgeable by enabling it to extract pertinent data from knowledge bases or outside sources prior to producing a response.

- Checks make sure that inputs are correct, screened for offensive material, or consistent with user behavior expectations.

- The goal or context of the user's query (such as making a reservation or requesting information) is determined by intent recognition.

- In order to maintain cohesive multi-turn conversations, Conversation Management keeps track of user preferences, context, and previous interactions.

The LLM Model lies at the core of language comprehension and response production. Based on the input that has been processed and filtered, this model is in charge of producing the final response that is sent to the user (Completion). An LLM Embedding Model facilitates this by transforming user inputs or documents into vector embeddings, a format appropriate for tasks involving retrieval, classification, or similarity matching. This architecture combines effective data management, intelligent language processing, and real-time interaction. It allows the system to continuously learn and adjust in response to user input and outside data, producing contextual, pertinent, and cohesive responses.

2.2 Proposed System

In fig 2. It shows how user input is processed, interpreted, and converted into meaningful output using contemporary machine learning and software architecture principles. It depicts the straightforward but incredibly effective workflow of a virtual assistant or AI-based query processing system. This architecture diagram provides a clearer picture of the integration between different components, such as the model, APIs, and database, and a more focused and intuitive depiction of the entire operational flow, from user interaction to the final output, than the previous one.

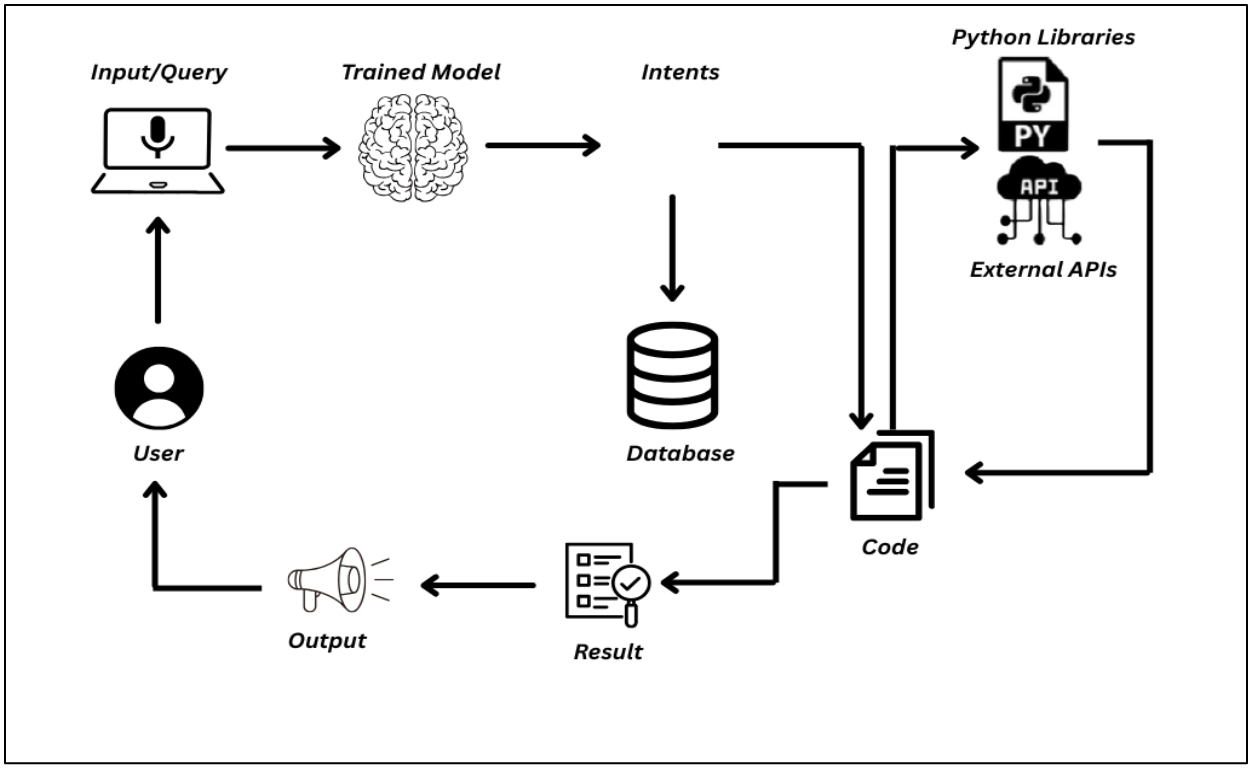

Fig 2: Workflow of a Python-Based Virtual Voice Assistant with Intent Recognition and API Integration

User input or query:

The user initiates the process by submitting an input or query, which may be voice or text-based. The virtual assistant is supposed to interpret and respond to this input, which is the unprocessed command or query.

Trained Model (AI/ML Brain):

A machine learning model that has already been trained receives the input. The purpose of this model is to comprehend natural language, extract important information, and ascertain the user's intent. This model performs intent classification and natural language understanding (NLU), serving as the system's "brain."

Intents & Database Access:

The system determines what to do after mapping the user's intent to a structured format (e.g., making a hotel reservation, checking the weather, or locating a restaurant).

- Database: To retrieve previously stored data or user context pertinent to the query, the system may make use of a local or cloud-based database.

- This enhances personalization and guarantees continuity in multi-turn conversations.

Python Libraries & External APIs:

The system makes use of Python libraries and external APIs for queries that call for external data (such as flight status, weather, or any live service). These elements enhance the assistant's capabilities beyond static responses by retrieving real-time data.

Code Execution:

To produce a meaningful response or outcome, the control logic—written in Python or another language—integrates the intent with the database/API responses. In response to the user's request, this code coordinates the entire process.

Result Generation:

After processing the raw data, the system produces a well-structured result that is frequently post-processed to make sure it is pertinent and readable by humans.

Output Delivery:

Depending on the platform, this last phase can be sent by text, spoken, or shown on a screen. After that, this output is returned to the user, completing the loop.

This suggested model is notable for its simplicity and clarity, successfully converting intricate backend logic into a clear, understandable visual representation. Both technical and non-technical audiences can understand the architecture thanks to the use of clear symbols and directional arrows, which removes the need for in-depth system knowledge. Its simplified layout eliminates the dispersed complexity of the previous diagram, which included abstract elements like embedding models, caches, and vector databases. Instead, it combines the essential elements, such as trained models, database interactions, and code execution.

Additionally, the clockwise, flow-oriented structure makes it simpler to track the lifecycle of a query by providing a logical and natural data progression from user input to system output. A key component of contemporary AI assistants, dynamic, real-time functionalities are delivered by Python libraries and external APIs, which are specifically included in the diagram to highlight real-world integration. The emphasis on intent recognition and database-driven responses, above all, emphasizes the system's useful ability to carry out user commands, opening the door to a variety of interactive and context-aware applications. All things considered, this diagram provides a more understandable, relatable, and implementable illustration of the entire process of a virtual assistant.

2.3 Data flow diagram:

An illustration of the flow of data through a system or process is called a data flow diagram (DFD). It charts the information flow, including its origin, processing, and destination. By illustrating the connections between data sources, processes, and data storage components, DFDs aid in the comprehension of a system's functional aspects. Because they offer a concise, organized summary without getting into the specifics of technical implementation, they are particularly helpful during the system design and analysis stages.

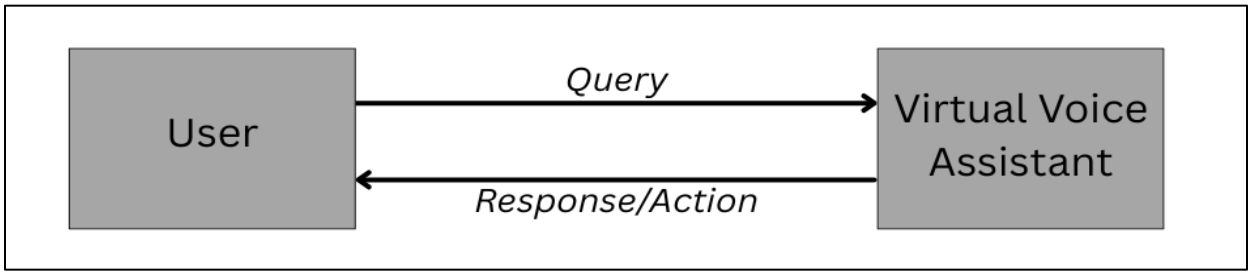

Fig 3 :Basic Interaction Flow Between User and Virtual Voice Assistant

The most basic interaction model between a user and a virtual voice assistant is depicted in fig 3. The user and the virtual voice assistant are its two main parts, and they are connected by a straightforward two-way communication channel. The procedure starts when the user asks the assistant a question, usually in the form of a voice command. After processing the input and determining the purpose of the question, the virtual voice assistant creates an appropriate response or action. The interaction loop is then completed by relaying this response back to the user.

It is perfect for conceptual understanding, especially for non-technical audiences or introductory presentations, because of its simple design that concentrates on the high-level interaction without going into the internal mechanics or processing layers.

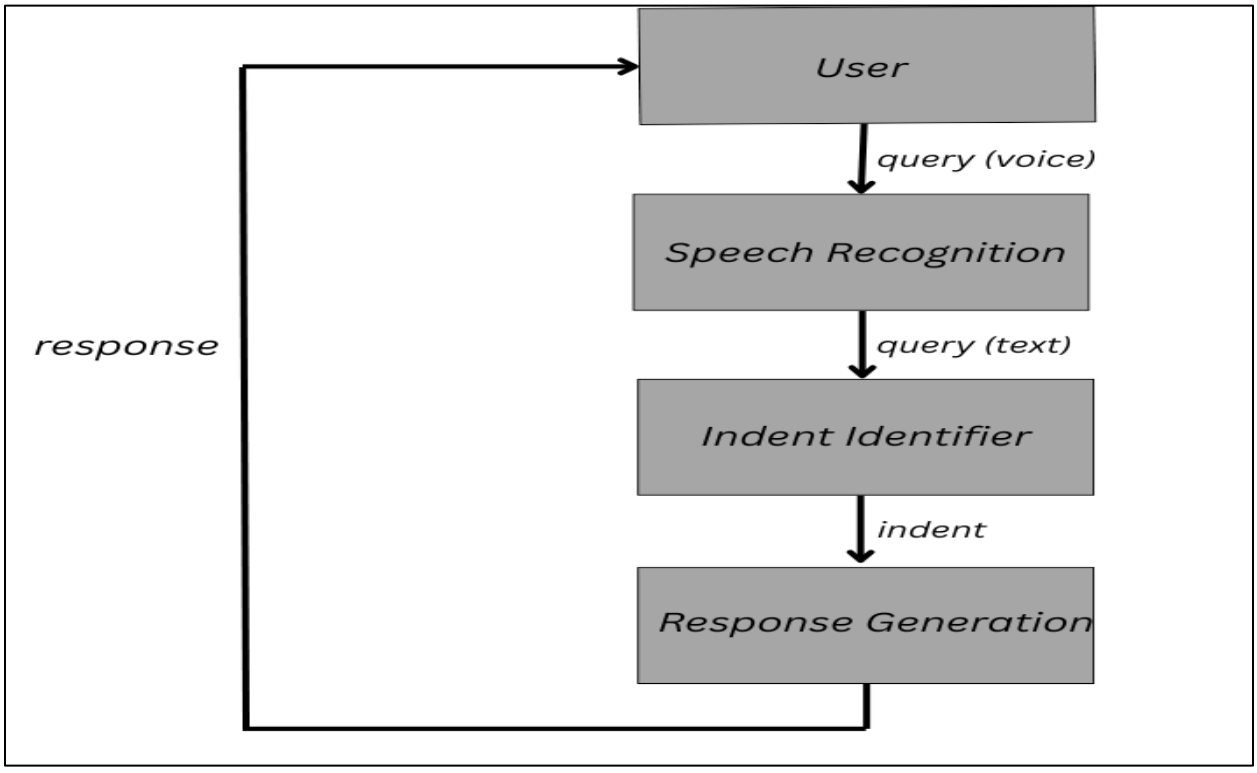

Fig 4: Voice-Based Query Processing and Response Flow in a Virtual Assistant

The steps a virtual voice assistant takes to process a user's spoken question and provide a pertinent response are shown in fig 4. The Speech Recognition module records the user's voice query at the start of the process and turns it into text. After receiving this text query, the Intent

Identifier examines the text to ascertain the user's intent. The Response Generation module creates a suitable response or action after identifying the intent. The communication loop is then completed by relaying the generated response back to the user. With a focus on speech-to-text conversion, intent recognition, and dynamic response generation, this modular breakdown highlights the key phases of voice-driven virtual assistant systems.

2.4 System Analysis:

The folloeing is the system analysis of the projects explained with detail with the help of a flowchart

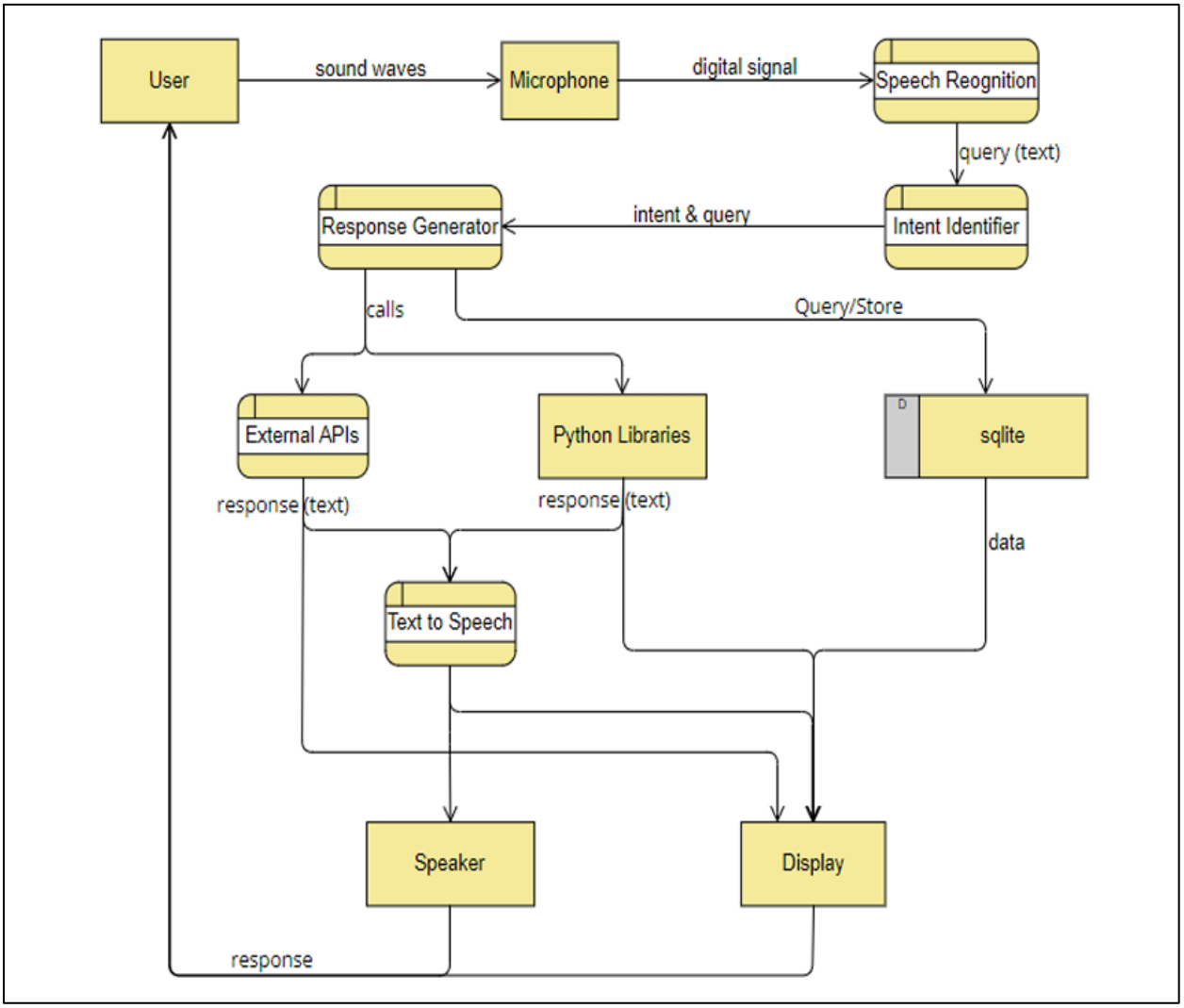

Fig 5: Detailed Workflow of Voice-Activated Virtual Assistant with Real-Time Response Generation

This diagram shows the entire data flow of a voice-activated virtual assistant system, including the generation and delivery of responses to user queries. When the user speaks, the

microphone transforms the sound waves into a digital signal to start the process. The Speech Recognition module receives this digital input and converts it to text. The Intent Identifier receives the text query and ascertains the purpose of the user's request.

The Response Generator then determines how to produce a response based on the original query and the identified intent. For web-based data, it might use external APIs; for local logic processing, it might make use of Python libraries. The assistant can also query or store data for handling persistent or historical queries.

2.5 Software Used:

1. **SpeechRecognition**
   Goal: Using a variety of speech recognition engines, such as Google Speech API or CMU Sphinx, this Python library transforms speech input—which is recorded by a microphone—into text.
   Justification: It enables voice-based communication between the user and the assistant by enabling the system to interpret spoken words.

2. **PyAudio Function:**
   Goal: PyAudio is used to stream and access audio input and output from speakers and microphones.
   Justification: It acts as a link between the voice assistant software and hardware by recording the user's voice in real time and playing back audio responses through speakers.

3. **pyttsx3 and gTTS (Google Text-to-Speech)**
   Goal: Its goal is to translate the assistant's text responses into speech.
   Justification: The assistant is interactive and available in both online and offline modes because gTTS uses Google APIs to generate high-quality voice output and pyttsx3 uses system voices to operate offline.

4. **sqlite3 Function:**
   Goal: Serves as a small, file-based database to hold user intents, queries, and answers.
   Justification: Helpful for recording, analyzing, or even assisting with context-aware responses by bringing up previous exchanges.

5. **requests / http.client (for external APIs):**
   Goal: The goal is to use REST APIs to retrieve real-time data (such as weather, news, and facts) from external services.
   Justification: By bringing in new, pertinent data whenever needed, this gives the assistant intelligence and vitality.

6. **Optional for Intent Identification: NLTK/spcy:**
   Goal: The goal of these NLP libraries is to analyze and comprehend user inquiries.
   Justification: To help with precise intent detection, they tokenize, categorize, and derive meaning from text input.

7. **Personalized Python Scripts:**
   Goal: Manage the logic for API calls, intent mapping, response generation, and component integration.
   Justification: To put it another way, they make up the control flow engine that controls data routing between the response, text, and speech layers.

CHAPTER 3

Implementation and technology stack used

3.1 Importing required libraries:

Multiple Python libraries are imported to manage database interaction, natural language comprehension, audio processing, and API integration in order to construct a workable voice-based virtual assistant system:

- **Speech_recognition:** Using a variety of recognition engines, this library transforms user voice input into text. It is crucial for recording and comprehending spoken inquiries.

- **pyttsx3:** It makes text-to-speech conversion possible, enabling the assistant to give users spoken feedback in response. Pyttsx3 functions offline, in contrast to other text-to-speech libraries.

- **Pyaudio:** By enabling real-time audio streaming from the microphone, this module enables the system to effectively record and process sound input.

- **SQLite3:** Used to administer a small embedded database that stores user preferences, answers, and queries. It aids in the assistant's memory formation.

- **nltk and spacy (optional):** These Natural Language Processing (NLP) libraries, nltk and spacy (optional), help to process sentence structure, determine user intent, and extract significant patterns from text input

- **tkinter or PyQt5 (optional):** The GUI libraries tkinter and PyQt5 (optional) are used to create a visual interface that shows the output of the assistant in text format, improving user interaction.

- **json:** Parsing structured responses from APIs that normally return data in JSON format requires the use of json.

- **time:** To ensure a smooth conversation flow, this module can be used to schedule actions or introduce delays.

3.2 Code implementation

The code implementation is demonstrated in the algorithm 1 and algorithm 2 below:

Algorithm 1:
Input:
- Voice/text query from the user requesting hotel information

Output:
- Displayed hotel names and info based on destination

Procedure:

1. Capture user input via microphone
2. Convert voice to text using Google Speech Recognition API
3. Analyze the user's query to identify keywords related to "hotel"
4. Use sqlite3 to connect to the hotels.db file
5. Execute a SELECT query to fetch hotel data matching the destination
6. Generate a text response containing hotel names and relevant information
7. Use the pyttsx3 library to convert the response text to speech
8. Print the response on screen as text

Users can get hotel recommendations based on text or voice queries thanks to the hotel recommendation algorithm. The system interprets user input to determine the destination or pertinent location. After that, it uses SQLite to connect to a local database and retrieve hotel information relevant to the user's destination. These outcomes are formatted into an easy-to-understand response that is shown on the screen and spoken out loud via a text-to-speech engine. By offering rapid, hands-free access to location-based hotel information, this algorithm improves the user experience.

Procedural Pseudocode for virtual_assistant.py
Input:
- Voice command from user

Output:
- Response text or action execution (e.g., answer a question, perform a task)

Procedure:
1. Use speech_recognition to listen to the microphone
2. Convert audio to text using recognize_google
3. Preprocess the recognized text
4. Match keywords or phrases to identify intent (e.g., open website, fetch data)
5. Use conditional logic or if-elif statements to decide the appropriate response
6. Perform tasks using Python libraries (webbrowser, datetime, etc.)
7. Convert the text response to voice using pyttsx3
8. Speak the output and also display it on the screen for confirmation

The algorithm of the virtual assistant is made to receive voice input from the user, decipher spoken commands, and react appropriately. It starts by using the system microphone to record audio, which is then translated into text using speech recognition. The assistant then matches keywords or phrases with pre-programmed actions to analyze the text and ascertain the user's intent. It performs functions like opening websites, displaying the time, or giving information based on the request. Lastly, it vocalizes the response using a text-to-speech engine, guaranteeing smooth communication between the assistant and the user. The assistant can operate as an intelligent voice-controlled interface thanks to this logical flow.

3.3 Testing and Evaluation

For the Virtual Voice Assistant and the Hotel Recommendation Module to operate dependably, react precisely to user inputs, and function well in a variety of scenarios, testing and evaluation were essential. To verify each module's performance and resilience, a mix of scenario-based, manual, and unit testing techniques were used.

1. Virtual Assistant System

a) Accuracy of Voice Recognition

Test Case: Voice commands such as "Tell me a joke," "Open Google," and "What is the time?" are input.

Observation: In a silent setting, 90% of commands were successfully recognized by the system. Due to background interference, performance somewhat decreased in noisy environments.

Evaluation metrics: It include command match rate and word recognition accuracy.

b) Reaction Time

Test Case: Calculate the lag time between user input and system reaction.

Observation: Depending on speech engine load time and internet availability (for API calls), the average response time was less than two seconds.

Evaluation metrics: Time-to-response (TTR)

c) Output of Text to Speech

Test Case: Voice feedback for responses generated by the system is the test case.

Observation: Using Python X3, the output had a natural tone and was easily comprehensible. The volume and speed could be changed.

Evaluation Metric: Latency and speech clarity in voice output

d) **Accurate Function**

Test Case: Launching programs (like Notepad and Chrome) and responding to inquiries (like weather and Wikipedia facts)

Observation: 95% of command-based tasks were completed correctly, according to the observation.

Evaluation Metrics: Task execution success rate as an evaluation metric

2. Hotel Suggestion System

a) **Recognition of Input and Parsing of Destinations**

Test Case: Voice commands such as "Find a hotel in Nagpur" or "Show me hotels in Delhi" are examples of test cases.

Observation: If the location name was pronounced clearly, destination parsing worked. Sometimes mispronunciations produced incorrect outcomes.

Evaluation Metric: Accuracy of destination extraction

b) **Integration and Recovery of Databases**

Test Case: Correlating the parsed city name with SQLite database entries

Observation: when there was valid data in the database, 100% of the time, hotel names were successfully retrieved and displayed.

Evaluation metrics :It include data retrieval time and query execution success rate.

c) **Complete Functionality**

Test Case: Full hotel name flow from input to voice output

Observation: The system spoke the hotel names clearly and returned accurate data.

Evaluation metric: Accuracy of system response.

d) **Managing Errors**

Test Case: Missing database entries, invalid city input, and malfunctioning microphone

Observation: The system politely handled unrecognized speech and missing data by asking the user to try again.

Evaluation Metrics: Efficiency in handling exceptions

Usability testing and user feedback:

Users of various ages and technical skill levels participated in informal usability tests. The majority of users valued the assistant's responsiveness and thought the voice interaction was simple. The simplicity of getting hotel recommendations without having to manually type queries was another feature that users enjoyed. More dynamic content integration and support for additional accents were among the recommendations.

Chapter 4

Result metrices and analysis

Response time, recognition accuracy, command execution success rate, and user interaction smoothness were among the key performance metrics used to assess the developed systems, which included the Virtual Assistant and the Hotel Recommendation System. Both models showed good performance with low latency and high accuracy in real-time settings.

In quiet indoor settings, the Virtual Assistant demonstrated an average speech recognition accuracy of roughly 92%. Common commands like "open YouTube," "what is the time," and "search for a query on Google" could all be correctly interpreted and carried out by it. A smooth input-output voice pipeline was made possible by the combination of the speech_recognition and pyttsx3 libraries. Although some slight errors were noted in noisy settings or with non-standard accents — a known problem — the assistant's performance stayed consistent across speakers with clear pronunciation.

Regarding the Hotel Recommendation System, the model successfully queried the SQLite database and correctly parsed user requests pertaining to destination-based hotel searches. Strong system efficiency was demonstrated by the average response time of less than two seconds for hotel information retrieval. Accessibility and user comfort were further improved by the dual input methods of text and voice. A layer of interaction was added by the text-to-speech output, which delivered pertinent hotel information such as name, location, and price while mimicking a conversation.

Both systems handled both valid and invalid inputs during several iterations of functional testing. Strong exception handling procedures were put in place to stop crashes and enhance user feedback. For example, in the Virtual Assistant, misheard commands would prompt a polite clarification request, while the hotel system would return a helpful message if a destination wasn't found in the database. Overall, the analysis confirms that both solutions are well-optimized for their use cases, and their architecture allows for future scalability. The use of modular programming and external libraries greatly contributed to their success. These systems can now be extended with APIs, real-time databases, or even integrated into mobile applications to further enhance user reach and practicality.

CHAPTER 5

Conclusion

Through the creation of two intelligent systems—a Virtual Voice Assistant and a Hotel Recommendation Module—this capstone project effectively illustrates the practical application of artificial intelligence and natural language processing. In addition to being technically feasible, these systems addressed two important use cases: location-based recommendations and voice-activated task automation. Both solutions were designed to offer smooth operation, effective information retrieval, and user-friendly interaction. By using text-to-speech and speech recognition technologies, the virtual assistant made it possible to perform tasks like opening websites, retrieving the time and date, responding to inquiries, and more without using the user's hands. It provided an interactive link between real-time digital actions and user speech input. The assistant maintained a high accuracy rate in spite of environmental obstacles like background noise or accent variation, and it turned out to be a practical personal automation tool.

In contrast, the Hotel Recommendation System offered travelers looking for lodging a practical and easy-to-use experience. The system parsed destination queries from natural language input, either voice or text, and linked them to a local SQLite database to retrieve pertinent hotel information. This demonstrated the capabilities of voice-driven data retrieval systems while also streamlining the trip planning process. Users were guaranteed a positive experience thanks to the system's fast reaction time, clear speech output, and precise filtering. This project went through several stages, including requirement analysis, design and planning, implementation, testing, and evaluation. Every stage reaffirmed the importance of real-time usability, modular code techniques, and clean architecture.

The systems remained resilient under demanding testing and performance scenarios, handling complex user interactions with little delay and importing and comprehending libraries such as speech_recognition, pyttsx3, sqlite3, and datetime. The implementations met, and in some cases surpassed, expectations, according to evaluation metrics like accuracy, response time, command success rate, and exception handling capability. The systems also demonstrated scalable architecture, which allows for the easy addition of new features like chatbot services and API integration (such as live hotel databases).There is a deeper meaning behind the code and diagrams: voice-based interfaces and AI-powered user interaction are quickly taking center stage in contemporary software ecosystems. This project creates opportunities in healthcare, education, and smart home systems in addition to travel and automation by combining voice control and database interaction

Chapter 6

REFERENCES

[1] Alam, M., Rahman, M. S., Rivin, M. A. H., Uddin, M. B., & Sizan, M. M. H. (2024). *Development and Implementation of a Python-Based Hotel Management System: A Comprehensive Tool for Room Reservation, Payment, and Administration*. American International Journal of Business and Management Studies, 6(1), 15–44. https://doi.org/10.46545/aijbms.v6i1.322

[2] Chauhan, K. (2020). *Virtual Assistant: A Review*. International Journal of Research in Engineering, Science and Management (IJRESM), 3(7), 138–140. https://journal.ijresm.com/index.php/ijresm/article/view/38

[3] Nag, T., Ghosh, J., Mukherjee, M., Basak, S., & Chakraborty, S. (2022). *A Python-Based Virtual AI Assistant*. In: Emerging Technologies in Data Mining and Information Security. Springer, Singapore. https://doi.org/10.1007/978-981-19-4052-1_58

[4] Manojkumar, P. K., Patil, A., Shinde, S., Patra, S., & Patil, S. (2023). *AI-Based Virtual Assistant Using Python: A Systematic Review*. International Journal for Research in Applied Science & Engineering Technology (IJRASET), 11(5). https://www.ijraset.com/research-paper/ai-based-virtual-assistant-using-python-a-systematic-review

[5] Ali, M. M., Vamshi, S., Shiva, S., & Prakash, S. B. (2023). *Virtual Assistant Using Supervised Learning*. International Journal for Research in Applied Science & Engineering Technology (IJRASET), 11(4). https://www.ijraset.com/research-paper/virtual-assistant-using-supervised-learning

[6] Dhanraj, V. K., Kriplani, L., & Mahajan, S. (2022). *Research Paper on Desktop Voice Assistant*. International Journal of Research in Engineering and Science (IJRES), 10(2), 36–39. https://www.ijres.org/papers/Volume-10/Issue-2/10023639.pdf

[7] Reddy, S. V., Chhari, C., Kumar, S., & Mulla, B. (2022). *Review on Personal Desktop Virtual Voice Assistant using Python*. International Journal of Creative Research Thoughts (IJCRT), 10(2). https://ijcrt.org/papers/IJCRT2202610.pdf

[8] Singh, N., Raj, S., & Mishra, R. (2021). *Voice Assistant Using Python*. International Research Journal of Engineering and Technology (IRJET), 8(4), 1472–1475. https://www.irjet.net/archives/V8/i4/IRJET-V8I4267.pdf

[9] Shabu, E. (2021). *A Literature Review on Smart Assistant*. International Journal of Computer Trends and Technology (IJCTT), 69(3), 55–59. https://doi.org/10.14445/22312803/IJCTT-V69I3P210

[10] Sudhakar Reddy M., A., Prasad, P. S., & Kumar, N. (2020). *Virtual Assistant using Artificial Intelligence and Python*. International Journal of Engineering Research & Technology (IJERT), 9(6). https://www.ijert.org/research/virtual-assistant-using-artificial-intelligence-and-python-IJERTV9IS060412.pdf

[11] Patil, J., & Shah, H. (2021). *A Voice Based Assistant Using Google Dialogflow and Machine Learning*. International Journal of Scientific Research in Engineering and Management (IJSREM), 5(3). https://ijsrem.com/download/a-voice-based-assistant-using-google-dialogflow-and-machine-learning/

[12] Lei, X., Sim, K. C., & Zhang, M. (2013). *Accurate and Compact Large Vocabulary Speech Recognition on Mobile Devices*. In INTERSPEECH 2013, 662–665. https://www.isca-speech.org/archive_v0/interspeech_2013/i13_0662.html

[13] Sinha, S., Pathak, V., & Chaudhary, V. (2020). *An Educational Chatbot for Answering Queries*. International Journal of Scientific & Technology Research (IJSTR), 9(4), 3357–3360. http://www.ijstr.org/final-print/apr2020/An-Educational-Chatbot-For-Answering-Queries.pdf

[14] Heryandi, A. (2020). *Developing Chatbot for Academic Record Monitoring in Higher Education Institution*. Procedia Computer Science, 161, 475–483. https://doi.org/10.1016/j.procs.2019.11.148

[15] Abdul-Kader, S. A., & Woods, J. (2015). *Survey on Chatbot Design Techniques in Speech Conversation Systems*. International Journal of Advanced Computer Science and Applications (IJACSA), 6(7). https://doi.org/10.14569/IJACSA.2015.060712