# Title Page :

**Project Title:** Ugot Bot [AI-Powered Query Solver]

**Name:** S. Mojesh

**Regd.No:** 23095A3306

**College:** Rajeev Gandhi Memorial College Of Engineering and Technology

**Department:** CSE-(Artificial Intelligence & Machine Learning)

**Date of submission:** 01-02-2025

**Guide :** V.P.Hara Gopal

# Acknowledgement :

The journey of developing **UGot Bot: AI-Powered Query Solver** has been both exciting and challenging, and I would like to take a moment to express my gratitude to V.P. Hara Gopal Sir [Asst.Professor], and my team those who made it possible.

First and foremost, I extend my heartfelt thanks to my mentors and instructors for their unwavering guidance and support. Their insights and encouragement have been invaluable in shaping this project and pushing me to explore innovative solutions.

I am also incredibly grateful to my peers and collaborators, whose feedback, brainstorming sessions, and shared enthusiasm played a crucial role in refining and improving the assistant. Their contributions made this journey both productive and enjoyable.

A special note of appreciation goes to the open-source developer community and researchers whose work served as a foundation for building this project. The availability of resources and knowledge-sharing platforms has been instrumental in bringing this idea to life.

Lastly, I would like to thank my family and friends for their constant motivation and belief in my abilities. Their support kept me inspired and focused throughout this endeavor.

This project has been a valuable learning experience, and I look forward to further exploring the potential of AI-driven voice assistants in real-world applications.

**S.Mojesh**  
RGM College Of Engineering and Technology  
01-02-2025.

# Abstract :

The **UGot Bot: AI-Powered Query Solver** is an intelligent voice-driven system designed to provide instant responses to user queries. This project leverages Natural Language Processing (NLP) and speech recognition technologies to enable seamless interaction between users and the assistant. UGot is capable of understanding voice commands, processing queries efficiently, and delivering accurate responses in real-time.

The system is built with a focus on user experience, ensuring smooth and natural communication. It integrates advanced AI models to interpret queries, retrieve relevant information, and generate meaningful responses. Additionally, UGot supports various applications, including general knowledge retrieval, task automation, and smart assistant functionalities.

This project highlights the growing potential of AI-powered voice assistants in enhancing productivity and accessibility. Future improvements may include multilingual support, personalized recommendations, and integration with smart devices. UGot represents a step forward in the evolution of AI-driven virtual assistants, offering a user-friendly and efficient solution for information retrieval and task management.

# Table of Contents

1) Introduction

* Background and Motivation
* Problem Statement
* Objectives

1. Methodology

* Tools & Technologies used
* Steps

1. Implementation

* Details of implementation
* Source Code
* Challenges faced

1. Results and Discussion

* Outcomes
* Images

1. Conclusion
2. Reference

# Introduction :

**Background and Motivation**

With the rapid advancement of Artificial Intelligence (AI) and Natural Language Processing (NLP), voice assistants have become an essential part of modern technology. From smartphones to smart home devices, voice-driven interfaces are transforming the way people interact with digital systems. The motivation behind developing **UGot Bot: AI-Powered Query Solver** stems from the need for an efficient, user-friendly assistant capable of understanding and responding to queries in real-time. Unlike traditional search engines that require users to type queries manually, a voice assistant provides a hands-free and intuitive way to access information.

**Problem Statement**

Despite the growing popularity of voice assistants, many existing systems struggle with accuracy, contextual understanding, and response efficiency. Some assistants lack flexibility in handling diverse queries, while others require constant internet connectivity or fail to provide meaningful responses. This project aims to address these limitations by developing a robust, AI-powered voice assistant that can process queries efficiently, understand user intent, and deliver relevant responses with high accuracy.

**Objectives**

The primary objective of this project is to design and implement a smart voice assistant that enhances user interaction through voice-based commands. The key goals include:

* Developing an NLP-based system that accurately interprets voice inputs.
* Ensuring fast and relevant query processing for real-time responses.
* Improving contextual understanding to provide meaningful and coherent answers.
* Creating a user-friendly interface for seamless interaction.
* Exploring future enhancements, such as multilingual support and smart device integration.

# Methodology :

**Tools & Technologies Used**

To develop the **UGot Bot: AI-Powered Query Solver**, a combination of advanced AI technologies, programming languages, and frameworks was used. The key tools and technologies include:

* **Programming Language**: Python, due to its strong support for AI and NLP libraries.
* **Speech Recognition**: Google Speech Recognition API for converting voice input to text.
* **Natural Language Processing (NLP)**: SpaCy and NLTK for processing and understanding user queries.
* **API Integration**: OpenAI API, Wikipedia API, and other knowledge sources for retrieving relevant information.
* **Development Environment**: Upython for coding and testing.

**Steps**

The development process followed a structured approach to ensure efficiency and accuracy:

1. **Speech Recognition & NLP Implementation**
   * Integrated a speech recognition system to convert voice input into text.
   * Implemented NLP techniques for intent detection and query classification.
2. **Query Processing & Information Retrieval**
   * Developed an algorithm to process user queries and extract key information.
   * Connected the system with APIs to fetch real-time data and responses.
3. **Response Generation & Text-to-Speech Conversion**
   * Designed a response generation module that formulates meaningful answers.
   * Converted the text responses into speech using a text-to-speech engine.
4. **Testing & Optimization**
   * Conducted multiple test cases to ensure accuracy and efficiency.
   * Optimized the assistant for better speed, accuracy, and natural language understanding.

# **Implementation :**

#### ****Details of Implementation****

The **UGot Bot: AI-Powered Query Solver** was implemented using a structured development approach, ensuring seamless integration of voice recognition, NLP, and AI-driven query handling. The system follows a pipeline that starts with capturing voice input, processing it using NLP techniques, retrieving relevant information, and generating a voice-based response.

The assistant was designed to handle diverse queries, including general knowledge, weather updates, and task automation. Speech recognition is enabled using the **Google Speech Recognition API**, while NLP tasks such as intent detection and response formulation leverage **NLTK** and **SpaCy**. The response is then converted into speech using the **pyttsx3** text-to-speech engine, making interactions more natural and user-friendly. API integrations, such as Wikipedia and OpenAI’s GPT-based models, allow the assistant to fetch accurate real-time information.

To improve performance, the system underwent multiple iterations of testing and optimization, ensuring minimal response time and accurate query interpretation

# Source Code :

**Chatbot\_assist.py**

|  |  |
| --- | --- |
| import time  from datetime import datetime  import requests  from ugot import ugot  # Initialize Ugot bot  bot = ugot.UGOT()  bot.initialize('192.168.135.49') # Replace with your robot's IP address  # Configuration  ACTIVATION\_PHRASE = "hi robot"  ROBOT\_GREETING = "Hi! What's up?"  VOICE\_TYPE = 0 # 0: female, 1: male  WEATHER\_API\_KEY = "ce89e1cabdee39be7f06489a3fe7d232" # OpenWeatherMap API key  NEWS\_API\_KEY = "9769065b683b483c97a987fb6c10f331" # Replace with your NewsAPI key  WEATHER\_LOCATION = "India" # Change as needed  NEWS\_COUNTRY = "in" # 'in' for India; use 'us' for the USA, etc.  # Utility Functions  def get\_current\_time():  """Returns the current time in a human-readable format."""  return datetime.now().strftime("%I:%M %p")  def get\_current\_date():  """Returns the current date in a human-readable format."""  return datetime.now().strftime("%A, %d %B %Y")  def get\_weather():  """Fetches weather information using OpenWeatherMap API."""  url = f"http://api.openweathermap.org/data/2.5/weather?q={WEATHER\_LOCATION}&appid={WEATHER\_API\_KEY}&units=metric"  try:  response = requests.get(url)  data = response.json()  if data.get("cod") == 200:  temp = data["main"]["temp"]  weather\_desc = data["weather"][0]["description"]  return f"The current temperature in {WEATHER\_LOCATION} is {temp}°C with {weather\_desc}."  else:  return "Sorry, I couldn't fetch the weather information."  except Exception as e:  return "There was an error retrieving the weather data."  def get\_news():  """Fetches the latest news headlines using NewsAPI."""  url = f"https://newsapi.org/v2/top-headlines?country={NEWS\_COUNTRY}&apiKey={NEWS\_API\_KEY}"  try:  response = requests.get(url)  data = response.json()  if data.get("status") == "ok":  articles = data.get("articles", []) | if articles:  headline = articles[0]["title"]  return f"Here is the latest news: {headline}."  else:  return "I couldn't find any news articles at the moment."  else:  return "Sorry, I couldn't fetch the news."  except Exception as e:  return "There was an error retrieving the news."  # Infinite Loop to Process User Inputs  while True:  try:  audio = bot.start\_audio\_asr() # Start mic and get user input  if audio and ACTIVATION\_PHRASE.lower() in audio.lower(): # Check activation  bot.play\_audio\_tts(str(ROBOT\_GREETING), wait=True, voice\_type=VOICE\_TYPE)  time.sleep(0.5) # Allow greeting to complete  while True:  query = bot.start\_audio\_asr().strip() # Get user query  if not query:  continue # If query is empty, skip iteration    # Handle queries  if "time" in query.lower():  response = f"The current time is {get\_current\_time()}."  elif "date" in query.lower():  response = f"Today's date is {get\_current\_date()}."  elif "weather" in query.lower():  response = get\_weather()  elif "news" in query.lower():  response = get\_news()  elif query.lower() in ["exit", "stop", "shutdown"]:  response = "Goodbye! Shutting down."  bot.play\_audio\_tts(str(response), wait=True, voice\_type=VOICE\_TYPE)  print("Exiting program...")  exit(0)  else:  # Fallback to AI NLP for other queries  response = bot.start\_audio\_nlp(query, wait=True)  # Ensure response is always a string before passing to TTS  bot.play\_audio\_tts(str(response), wait=True, voice\_type=VOICE\_TYPE)  time.sleep(0.5) # Allow processing time  except KeyboardInterrupt:  print("Program interrupted. Exiting...")  bot.play\_audio\_tts("Goodbye! Shutting down.", wait=True, voice\_type=VOICE\_TYPE)  break  except Exception as e:  print(f"Error: {e}")  with open("error\_log.txt", "a") as log\_file:  log\_file.write(f"{datetime.now()} - Error: {e}\n")  bot.play\_audio\_tts("An error occurred. Check logs for details.", wait=True, voice\_type=VOICE\_TYPE) |

**Challenges Faced**

During the development of **UGot**, several challenges were encountered:

1. **Speech Recognition Accuracy**
   * Background noise often interfered with voice recognition, causing misinterpretations. This was mitigated by using **ambient noise adjustment** and **multiple recognition attempts**.
2. **Handling Complex Queries**
   * Some queries required contextual understanding beyond simple keyword detection. Future improvements may involve integrating **advanced AI models** like GPT for better query interpretation.
3. **Response Speed Optimization**
   * Fetching real-time data from APIs introduced delays. Caching frequently requested queries helped improve response times.
4. **Integration Issues**
   * Combining multiple AI components (speech recognition, NLP, and API responses) required careful synchronization to ensure smooth interaction.
5. **Limited Knowledge Base**
   * The assistant initially struggled with answering diverse topics. Expanding the knowledge base and integrating multiple data sources helped enhance its capabilities.

# Result and Discussion :

**Outcomes**

The **UGot Bot: AI-Powered Query Solver** successfully achieved its intended functionality, providing real-time responses to user queries using speech recognition and natural language processing. The key outcomes of the project include:

1. **Accurate Speech Recognition**
   * The assistant effectively converts voice input into text with high accuracy in a quiet environment.
   * Minor misinterpretations occur in noisy surroundings, but ambientnoise adjustment improves recognition.
2. **Efficient Query Processing**
   * UGot handles general knowledge queries well, retrieving concise and relevant responses from sources like Wikipedia.
   * The assistant correctly interprets basic commands (e.g., greetings, weather requests).
   * More complex queries require enhancements in context awareness for better accuracy.
3. **Seamless Text-to-Speech Conversion**
   * The system generates natural-sounding speech responses using pyttsx3.
   * Speech rate and tone were optimized for clear and engaging interactions.
4. **Performance Evaluation**
   * The assistant processesand responds within 2-3 seconds for most queries.
   * API-based responses (e.g., Wikipedia searches) take slightly longer, averaging 4-5 seconds.
   * Overall user experience is smooth and interactive, making UGot a reliable voice assistant for general-purpose tasks.

# Conclusion :

The **UGot Bot: AI-Powered Query Solver** project successfully demonstrates the integration of speech recognition, natural language processing (NLP), and AI technologies to create an interactive and efficient voice-based system. Throughout the development process, the assistant was designed to understand and respond to a wide range of user queries, providing real-time answers using voice synthesis.

The project met its primary objectives of improving user experience by enabling hands-free interaction and quick access to information. The assistant showed solid performance in handling basic queries, offering a natural dialogue flow, and minimizing delays in response time. However, challenges such as noise interference and the need for enhanced contextual awareness highlighted areas for further improvement.

Despite these challenges, **UGot** proved to be a functional, scalable voice assistant. Future iterations will focus on expanding its capabilities, including multilingual support, deeper integration with smart devices, and enhanced AI models for improved context understanding and more sophisticated query handling.

Overall, the project has provided valuable insights into the potential of voice assistants and AI, setting the stage for future advancements in intelligent, interactive systems that can enhance productivity and accessibility.

# Reference :

Official Website : <https://ubtecheducation.com/ugot>

Upython download link : <https://edu.ubtrobot.com/downloads>

Github documentation link:<https://github.com/MojeshSetty/Ugot_chatbotAssist>