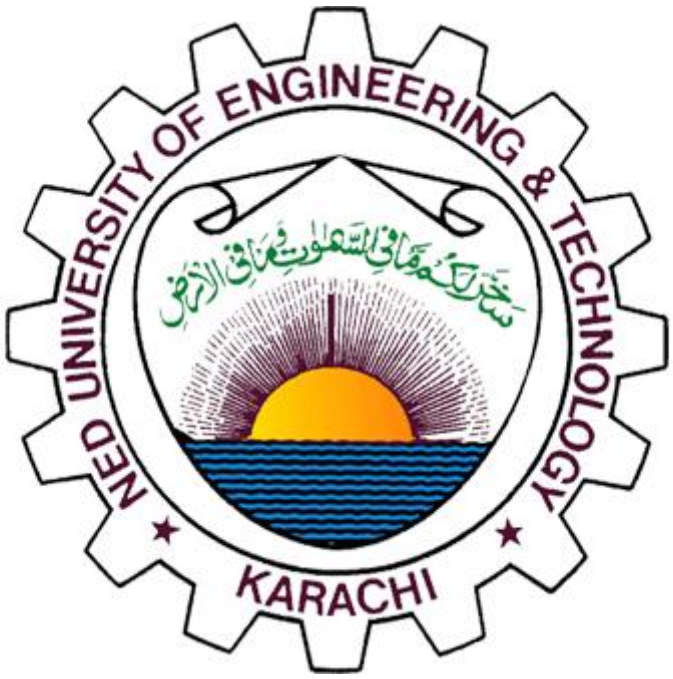


# AIES CCP Project Report

## Project Title: Dennis - An AI-Powered Desktop Assistant

Course Code: AIES-CCP

Course Title: Artificial Intelligence & Expert Systems

Group Members:

1. Syed Daniyal Ali | CT-22025
2. Muhammad Anus | CT-22039

## 1. Introduction

In recent years, the evolution of artificial intelligence has reshaped how users interact with digital systems. A significant portion of users, especially non-technical ones, find it challenging to operate system-level functions such as managing files, accessing data, or executing utilities through command-line interfaces. Traditional systems are often too rigid, requiring precise syntax and lacking natural user-friendly interaction.

To address these challenges, this project presents Dennis, an intelligent desktop assistant capable of bridging the gap between user intention and system functionality. Dennis listens to user input (typed or spoken), interprets the command using AI-based natural language understanding, and executes the appropriate actions. It leverages modern NLP models (Google Gemini), speech recognition, text-to-speech synthesis, and GUI programming to deliver an all-in-one assistant solution.

Dennis offers practical solutions like opening software, retrieving weather data, setting alarms, managing files/folders, and browsing the internet—all through natural conversation. It empowers users to interact with their desktops in an intuitive, productive, and efficient manner.

---

## 2. Problem Statement

A key bottleneck in user productivity on desktop environments stems from the complexity of operating system-level tasks through conventional interfaces. Simple tasks like launching an app, setting a reminder, or retrieving local file data can become overwhelming for users unfamiliar with command-line syntax or system navigation.

The lack of an intelligent, voice/text-enabled system that understands natural language and maps it to backend automation scripts necessitates an AI solution. Therefore, this project aims to solve the following problem:

"How can we enable seamless human-computer interaction through voice/text-based intelligent assistants that understand and execute system-level tasks using AI principles?"

---

## 3. Objectives

- To develop a desktop assistant that operates based on natural language commands.

- To integrate voice recognition and text-to-speech capabilities.
- To utilize Google Gemini API for NLP-based intent understanding and task classification.
- To modularize the assistant's functionality for easy scalability and maintainability.
- To support dynamic and real-time operations such as weather updates, file manipulation, and application launching.
- To bridge AI language models with backend automation for a hybrid intelligent system.

---

## 4. Technologies Used

| Technology | Role |
|---|---|
| tkinter | Build GUI for interactive chat between user and assistant |
| speech_recognition | Capture and convert voice input into textual commands |
| pyttsx3 | Convert assistant's responses into speech |
| google.generativeai | Use Gemini LLM for intelligent response and JSON instruction generation |
| requests | API integration for weather, IP location |
| webbrowser, subprocess, os | Execute system commands and open files or applications |
| win32com.client | Integrate with Windows Media Player for media tasks |
| queue, threading | Handle asynchronous communication and real-time events |

---

## 5. AI Capabilities Breakdown

### 5.1 Natural Language Understanding

The assistant uses the Google Gemini LLM to interpret natural user language into actionable tasks. This involves:

- Semantic parsing
- Intent extraction
- Task identification (e.g., "open", "create", "read", "delete")

The assistant distinguishes between model-handled (conversational) and developer-handled (executable) commands. For example:

- "Tell me a joke" → conversational
- "Open Notepad" → structured JSON for backend

5.2 Speech Recognition and Voice Response

Dennis supports hands-free usage:

- Voice Input: Uses speech_recognition to capture commands from a microphone.
- Voice Output: Uses pyttsx3 to speak replies aloud.

This enables users with accessibility challenges or multitasking needs to interact conveniently.

5.3 Application Launching

Commands like:

"Launch Chrome", "Open VS Code"

Are converted into structured JSON:

{"task": "open_app", "operation": "open", "name": "vscode"}

Dennis checks:

- Windows registry
- System directories
- Command shell fallback

This flexible approach ensures maximum compatibility with system-installed apps.

5.4 File and Folder Management

Operations include:

- Create new files or folders
- Open or delete existing ones

Commands:

- "Create file report.txt in Documents"
- "Open folder Downloads"

Dennis uses:

- os and subprocess for file ops
- os.path for directory verification

5.5 Weather Retrieval

"What's the weather in Karachi?"

Dennis:

- Extracts location
- Sends request to OpenWeatherMap
- Parses and formats temperature, humidity, wind, and conditions

This shows the integration of LLM NLP with real-world API data retrieval.

5.6 Location Detection

"Where am I right now?"

Dennis uses IP-based API ipinfo.io to:

- Fetch city, region, country, coordinates
- Return formatted geographic data

5.7 Document Access (PDF/DOCX)

"Read file thesis.pdf"

Dennis locates and opens:

- PDF via default reader
- Word docs via Microsoft Word or system default

Helpful for document review tasks using natural commands.

5.8 Web and Internet Commands

"Open github.com"

Dennis validates URL structure and opens the site in the default web browser using webbrowser.open().

---

6. Modularity and Software Architecture

Dennis is designed with clear separation of responsibilities:

- Frontend UI (DennisAssistantUI)
    - Built with tkinter
    - Handles user input, displays messages, GUI components
- Backend Logic (DennisAssistant)
    - Receives text/speech input
    - Sends to Gemini LLM for interpretation
    - Executes developer-handled tasks using local methods
- Gemini LLM Integration
    - Processes language via generate_content()
    - Returns structured JSON for task routing
- Task Execution Handlers
    - Methods like create_file(), get_weather() or play_music() handle exact command logic

---

7. Assumptions

- Users issue commands in clear English.
- Internet connection is available for Gemini API and weather/location APIs.
- System has necessary applications and libraries installed.
- Mic and speakers are present for full voice interaction.
- Gemini consistently returns well-formed JSON for developer-handled tasks.

---

## 8. Conclusion

Dennis demonstrates the power of artificial intelligence to transform user interfaces by combining:

- Speech interaction
- NLP-based command understanding
- System-level automation

It bridges the gap between AI language models and user-level automation on a desktop. The modular design, use of threading, and real-time interaction showcase strong principles of expert system design and AI application.

This project proves that AI-powered desktop assistants are not just futuristic concepts, but viable real-world tools today.

---