Group 23
Professor Donyanavard
CS 250
3/8/24

Virtual Desktop Assistant Design Specification

1. Software Title

Virtual Desktop Assistant

2. Team Members

Aleksa Marie Ocampo

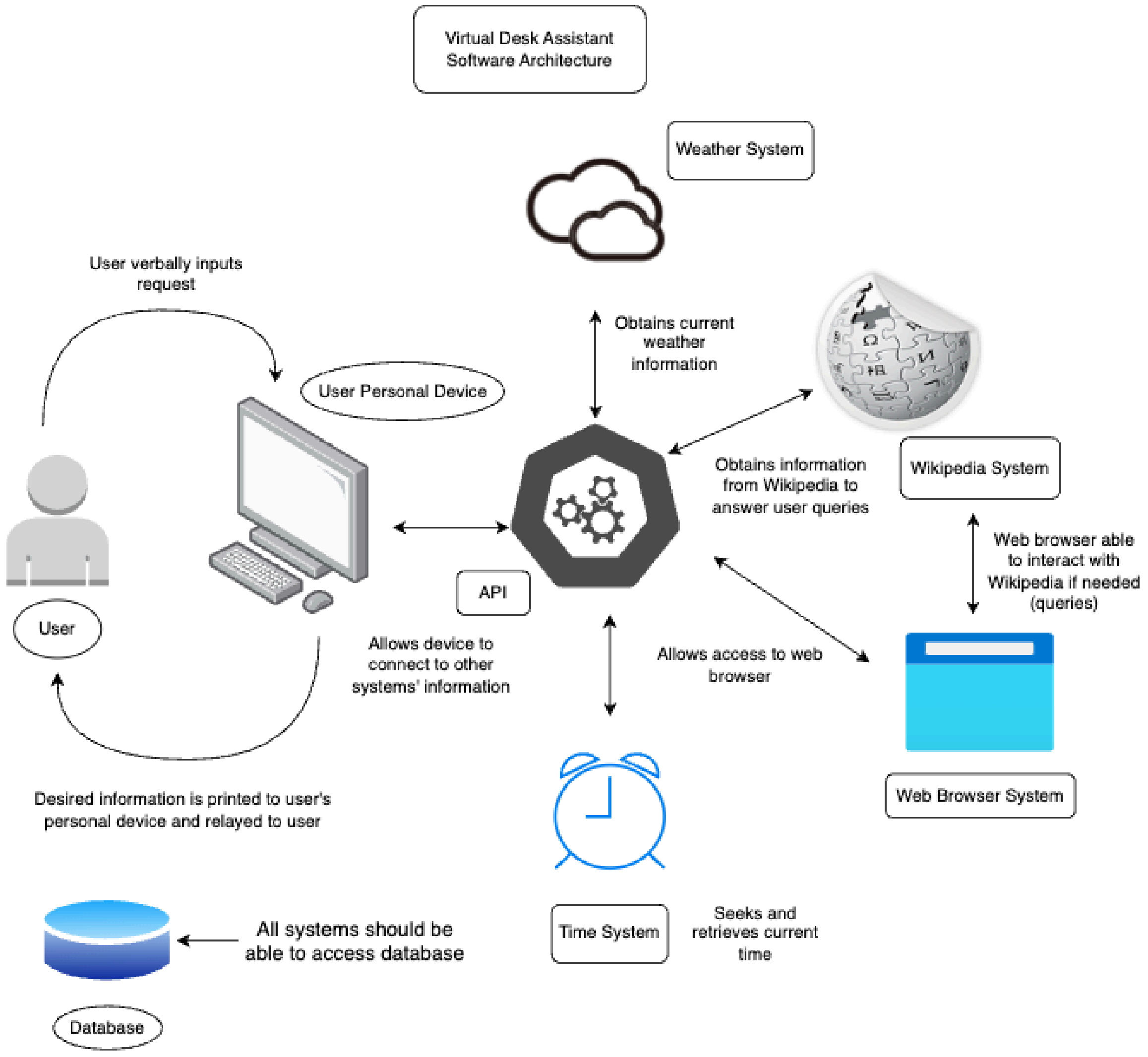
Darren Famisan

3. System Description

The Virtual Desktop Assistant is a system that enables users to control their computer through voice commands, akin to the fictional AI assistant Jarvis from the Iron Man series. Built using Python, it leverages major libraries to create a virtual assistant experience. While it operates as a weak AI, it provides users with the ability to interact with their machines seamlessly through voice commands.

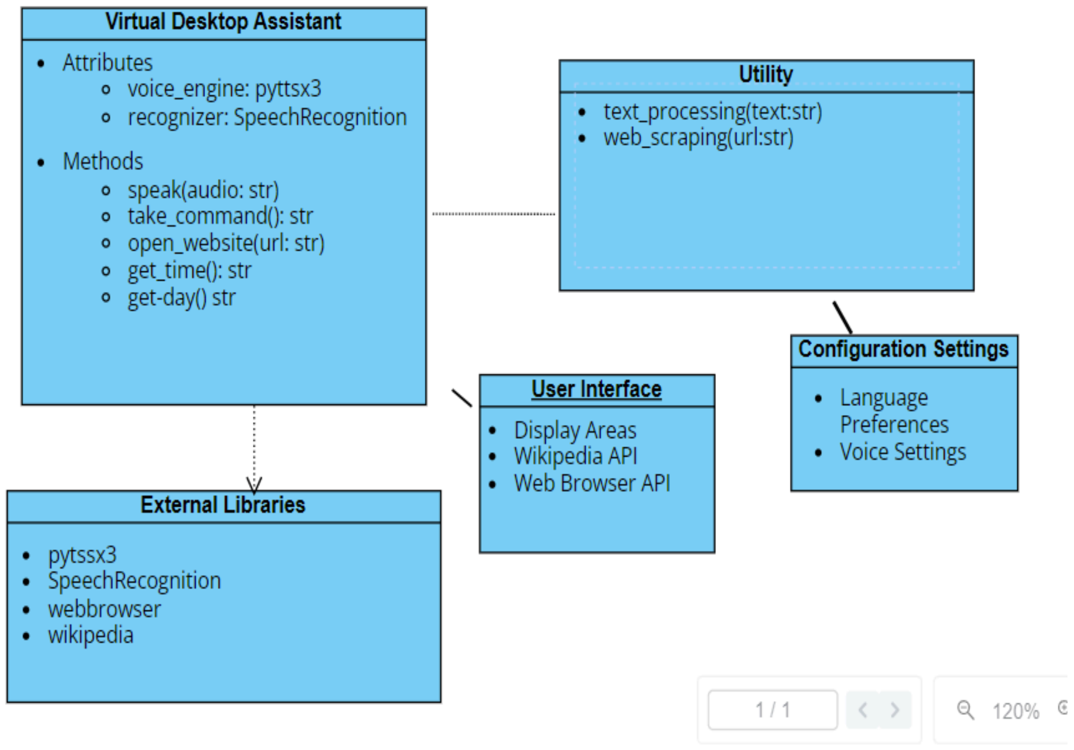
4. Software Architecture Overview

4.1 Architecture Diagram


Virtual Desk Assistant Software Architecture
Weather System
User verbally inputs request
Obtains current weather information
User Personal Device
Obtains information from Wikipedia to answer user queries
Wikipedia System
Web browser able to interact with Wikipedia if needed (queries)
User
API
Allows device to connect to other systems' information
Allows access to web browser
Desired information is printed to user's personal device and relayed to user
Web Browser System
All systems should be able to access database
Time System
Seeks and retrieves current time
Database

The first action the user must take is to verbally relay a request to the personal device, which is in this case, their computer. Depending on the request, the computer will use the API to access either the weather, Wikipedia, web browser, or time systems to obtain the proper information (all of these systems should be able to access the database). Once the software has gathered the correct information, it will relay its findings to the user by printing the information on the screen.

4.2 UML Class Diagram



4.3 Description of Classes

- **Virtual Desktop Assistant:** Represents the virtual assistant application that interacts with the user, processes voice commands, and performs various tasks
- **Utility:** Handles various utility methods for text processing and web scraping tasks. It encapsulates common functionalities to simplify the code and promote reusability.

4.4 Description of Attributes

- **voice_engine: pyttsx3:** This Python library provides a simple interface for text-to-speech conversion. It offers cross-platform support and offline functionality.

- r**ecognizer: SpeechRecognition:** This Python library facilitates the transformation of spoken language into text, enabling the interpretation and processing of speech input within applications.
- **speak(audio:str):** This method initializes a text-to-speech engine, sets voice properties, and converts the provided text into speech output, allowing the virtual assistant to communicate audibly with the user.
- **take_command(str):** This method captures audio input from the user via the microphone, recognizes the speech, and returns the recognized text as a string, enabling the assistant to understand and process verbal commands.
- **open_website(url:str):** This method accepts a URL as input, utilizing the webbrowser module to open the specified website in the default web browser, providing users with seamless access to web content directly from the virtual assistant interface.
- **get_time(): str:** This method retrieves the current system time, using the datetime module to fetch the hour and minute components, and then communicates this information audibly to the user, enabling the virtual assistant to inform users of the current time upon request.
- **get_day(): str:** This method retrieves the current day of the week and communicates it audibly to the user.
- **text_processing(text:str):** The text processing method manipulates text data, enabling tasks like breaking text into smaller components (tokenization) and identifying the root form of words (stemming), which aids in understanding and analyzing textual input.
- **web_scraping(url:str):** The Web scraping method will extract specific information from websites by analyzing their HTML structure, allowing the virtual assistant to retrieve relevant data such as text or links.

4.5 Description of Operations

- **User Input:** The assistant listens for user input via speech recognition, capturing spoken commands or queries.
- **Command Processing:** The input is processed to understand the user's intent, extracting relevant keywords or phrases to determine the desired action.
- **Action Execution:** Based on the processed input, the assistant executes various actions such as opening websites, retrieving information from sources like Wikipedia, fetching current time or day, or performing text processing tasks.
- **Data Processing:** If required, the assistant may process retrieved data further, such as extracting specific information from web pages or analyzing textual content.
- **Response Generation:** The assistant generates appropriate responses based on the executed actions or processed data, preparing to communicate with the user.
- **Text-to-Speech Conversion:** The responses are converted into audible speech using text-to-speech conversion techniques, ensuring the user receives a natural and understandable response.
- **Feedback Delivery:** The assistant communicates the generated response back to the user audibly and displayed through the user interface. The indication that the interaction loop between the user and the program is completed is done visually and audibly.

5. Development Plan and Timeline

The client requested the software to be built within six months, therefore the tasks need to be completed promptly. Ideally, a good chunk of time (one month or one-and-a-half) would also be allocated to testing towards the end of the timeline. In the case that there are unexpected outputs, there should also be enough time to revise the software.

5.1 Partitioning of Tasks:

Coding will be split between the two members. First, they will work together on making sure verbal requests from the user to the computer function properly. Both will ensure that the computer adequately connects to the database and API. Then, the remaining systems will be divided between the two. One will take the weather and time systems, which will accurately return information about the current weather and time to the user. The remaining members will code for the Web browser and Wikipedia systems. The Wikipedia system should be able to return information on the user's request, and the web browser system will allow access to search engines and other related information. The members will then test each other's functionality. In the case there are unexpected problems, there should be enough time to debug and revise the code.

5.2 Team Member Responsibilities:

Firstly, it is each member's responsibility to communicate concerns, revisions, and other observations to each other and other relevant parties. Should there be any problems encountered, concerns should be communicated immediately to ensure a timely manner of completion. Ideally, each member should be able to finish both of their systems within the four-month mark. For the methods that require collaboration, the remaining time should be properly allocated to completing those functions. Ideally, two months should be sufficient to complete those methods. The remaining two months should be used for testing and any necessary revising.