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

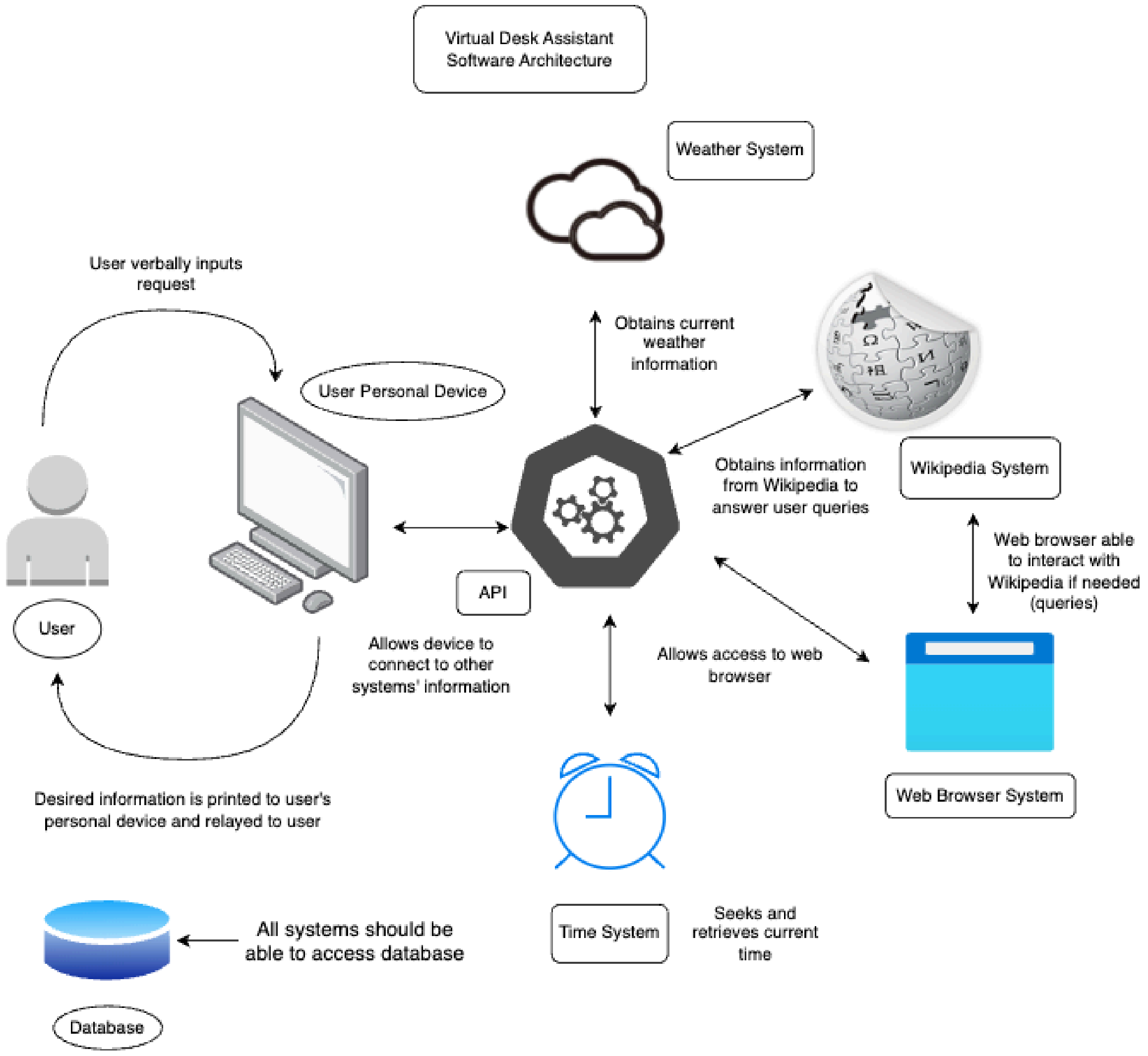
Darren Famisan

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.

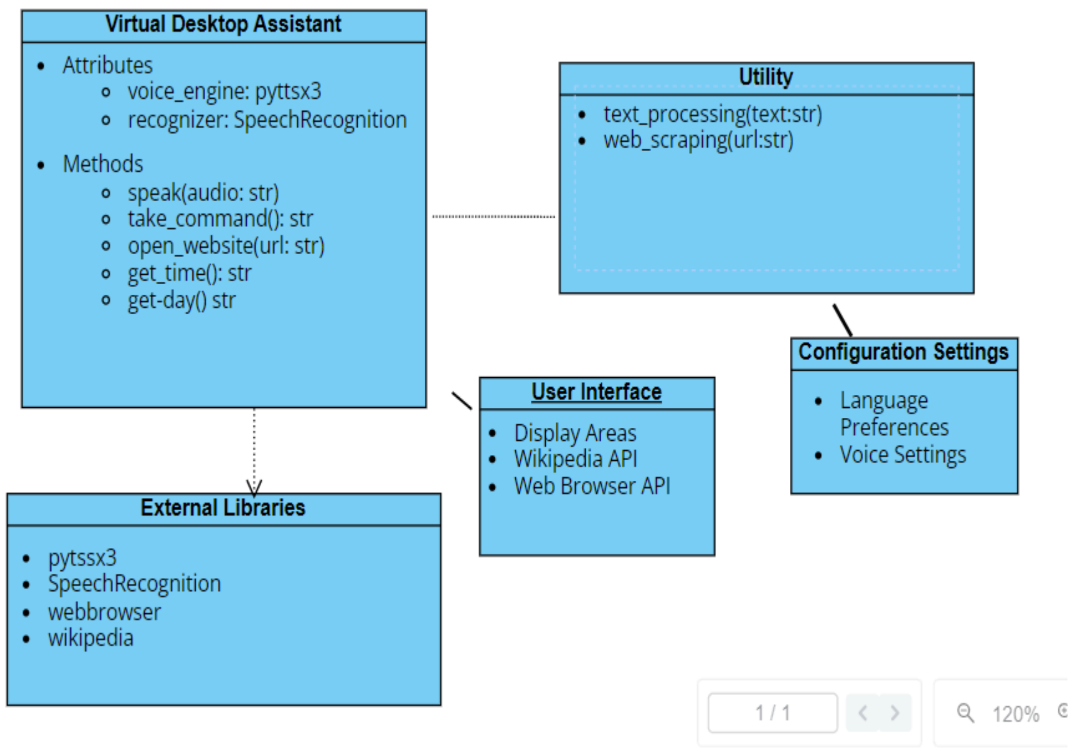
4. Software Architecture Overview

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)
API
User
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.

Group 23
Professor Donyanavard
CS 250
3/22/24

SDS: Test Plan

1. Verification Test Plan

Test Set 1: Test Voice Recognition Accuracy (Unit Test)

This test verifies the accuracy of the voice recognition feature in the virtual assistant. It ensures that the assistant correctly interprets and transcribes spoken commands from the user.

Test Vectors:

- Input a set of predefined commands (e.g., "What's the weather?", "Open Wikipedia", "Tell me the time") into the assistant.
- Observe if the assistant accurately recognizes and transcribes each command.
- Measure the accuracy rate by comparing the recognized text with the expected command.

Targeted Feature: Voice recognition functionality.

Coverage: This test set covers the accuracy and reliability of the voice recognition component. It ensures that the assistant correctly interprets user commands.

Potential Failures: Failures in this test could include misinterpretation of commands, incorrect transcription of speech, or failure to recognize certain words or phrases.

Test Set 2: Verify Text-to-Speech Conversion Quality

This test evaluates the quality of the text-to-speech conversion feature in the virtual assistant. It ensures that the assistant produces clear and natural-sounding speech output.

Test Vectors:

- Provide the assistant with various text inputs representing different types of information (e.g., weather forecast, Wikipedia article, time).
- Listen to the speech output generated by the assistant.
- Evaluate the clarity, pronunciation, and naturalness of the speech.

Targeted Feature: Text-to-speech conversion functionality.

Coverage: This test set ensures that the assistant delivers speech output that is understandable while ensuring that the information is accurate and concise.

Potential Failures: Failures in this test could include unclear or distorted speech output, mispronunciations, or unnatural-sounding speech.

Test Set 3: Test Web Browser Interaction (Integration Test)

This test verifies the virtual assistant and web browser module integration. It ensures that the assistant can open and navigate web pages based on user commands.

Test Vectors:

- Instruct the assistant to open specific websites (e.g., Google, Wikipedia, news sites).
- Verify that the web browser module successfully opens the requested pages in the default browser.
- Test navigation commands (e.g., scroll down, click on links) to ensure proper interaction with web content.

Targeted Feature: Integration between the virtual assistant and the web browser module.

Coverage: This test set verifies the seamless interaction between the assistant and web browser, enabling users to access online information efficiently.

Potential Failures: Failures in this test could include the inability to open requested websites, navigation errors, or compatibility issues with certain web pages.

Test Set 4: Test API Data Retrieval

This test validates the integration between the virtual assistant and external APIs (e.g., weather API, Wikipedia API). It ensures that the assistant can retrieve and process data from external sources.

Test Vectors:

- Request weather information for different locations using the assistant.
- Verify that the assistant correctly retrieves and displays weather data from the API.
- Query Wikipedia for various topics and confirm that relevant information is retrieved and presented to the user.

Targeted Feature: Integration between the virtual assistant and external APIs.

Coverage: This test set assesses the functionality and reliability of the API integration, enabling the assistant to provide accurate and up-to-date information to users.

Potential Failures: Failures in this test could include the inability to retrieve data from the API, incorrect data processing, or API authentication errors.

Test Set 5: Test End-to-End System Functionality (System Test)

This test evaluates the overall functionality of the virtual assistant system, including all integrated components and features. It ensures that the assistant operates smoothly and performs as expected in real-world usage scenarios.

Test Vectors:

- Engage in a series of interactions with the virtual assistant, including voice commands, data queries, and task executions.
- Test a variety of features and functionalities, such as weather updates, Wikipedia searches, web browsing, and time/date inquiries.
- Evaluate the responsiveness, accuracy, and user-friendliness of the system.

Targeted Feature: Overall system functionality and user experience.

Coverage: This test set provides comprehensive validation of the entire virtual assistant system, confirming that all components work together seamlessly to deliver a reliable and effective user experience.

Potential Failures: Failures in this test could include crashes or freezes, incorrect responses to user commands, or inconsistencies in system behavior across different usage scenarios.

Group 23
Professor Donyanavard
CS 250
4/12/24

Data Management Strategy

Database Choice:

For our Virtual Desktop Assistant project, we will be utilizing a SQLite database as our data management solution. Firstly, SQLite's lightweight nature ensures that it seamlessly integrates with Python through the built-in SQLite3 module, facilitating efficient data handling within our application. Secondly, the minimal setup overhead of SQLite aligns well with the project's goal of remaining as simple but efficient as possible by allowing us to focus on development without the burden of complex database configurations. Additionally, SQLite's file-based architecture eliminates the need for a separate server process, simplifying deployment and maintenance. This ensures that our virtual desktop assistant is designed for simple but efficient user interaction.

Logical Data Organization:

In organizing our data, we can adopt a single-database approach, consolidating all relevant data entities within a single SQLite database. This decision ensures simplicity and coherence in data management. Each type of data, including user queries, web scraping results, and API responses, is logically segregated into separate tables within the database. This organizational structure facilitates efficient data retrieval and manipulation for the smooth functioning of our virtual desktop assistant.

Design Decisions and Trade-offs:

By opting for a single database, we prioritize simplicity and ease of management, essential attributes for a virtual desktop assistant designed to provide seamless user experiences. While this approach may pose scalability challenges in the long term, the immediate benefits of simplicity and streamlined management outweigh potential scalability concerns for the current project scope.

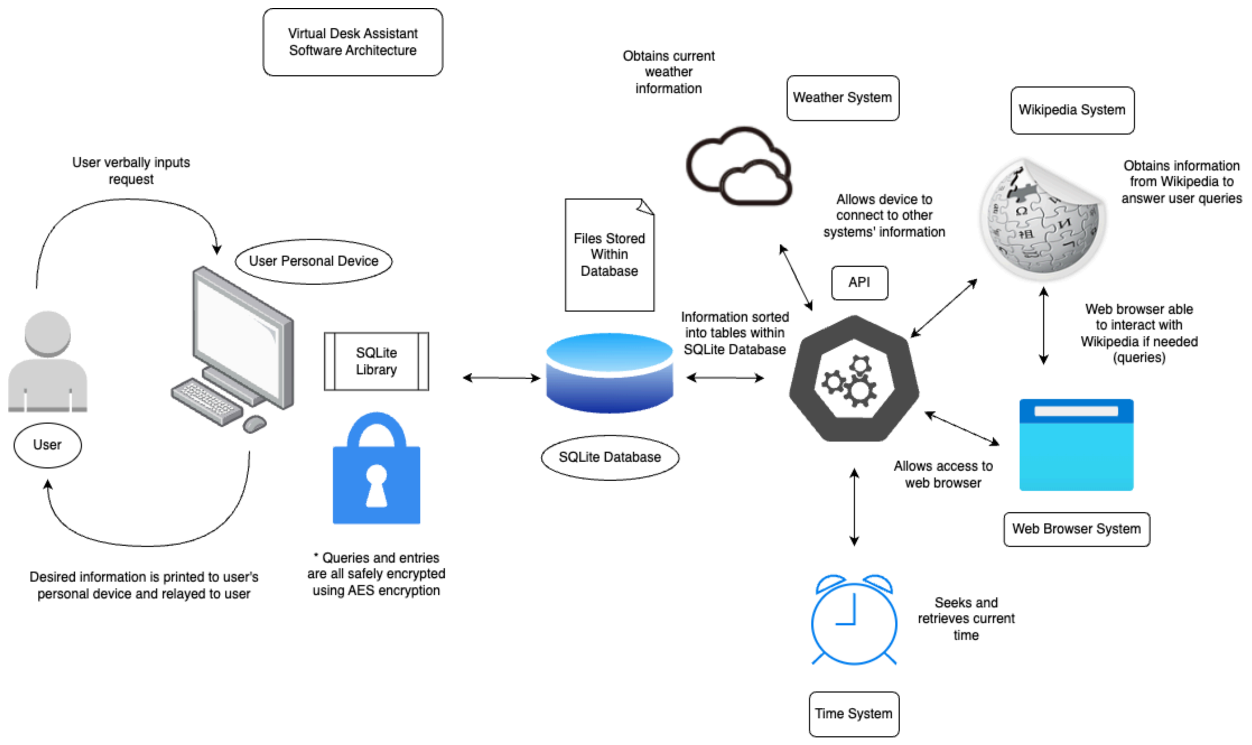
Encryption Overhead:

To ensure the security of sensitive data stored in the SQLite database, we’ll implement AES encryption. While encryption introduces computational overhead, potentially impacting performance during data encryption and decryption operations, the enhanced data security it provides is for safeguarding user privacy and confidentiality. Since the project is a desktop “assistant”, users will more than likely input personal data like journal entries or work schedules. The trade-off of increased computational overhead is justified by the significant benefits of data security in a virtual desktop assistant environment.

Access Control Complexity:

We’ll implement access control measures within the application logic to restrict access to sensitive data and operations. Despite the added complexity of the application code, access control enhances data security by ensuring that only authorized users or components can access sensitive data. This trade-off between complexity and security is essential for maintaining the integrity and confidentiality of user interactions within the virtual desktop assistant environment that we’ve referenced earlier.

Updated Software Architecture Diagram:



Virtual Desk Assistant Software Architecture
User verbally inputs request
User Personal Device
User
Desired information is printed to user's personal device and relayed to user
SQLite Library
* Queries and entries are all safely encrypted using AES encryption
Files Stored Within Database
SQLite Database
Information sorted into tables within SQLite Database
Obtains current weather information
Weather System
Allows device to connect to other systems' information
API
Wikipedia System
Obtains information from Wikipedia to answer user queries
Web browser able to interact with Wikipedia if needed (queries)
Allows access to web browser
Web Browser System
Seeks and retrieves current time
Time System

With the introduction of our SQLite Database and AES encryption, we have updated the software architecture to reflect those changes. In the previous diagram, the user's personal device was directly connected to the API, but here, we have the SQLite Database as a mediator between the two. The database stores information from the API, whether that be the time, the web browser searches, etc., in tables. The information is sorted into appropriate categories that match the data type, and the files are then stored within the database. Additionally, to reflect the safe management of data, AES encryption is included prior to the device's interaction with the database, thus the user's wall of privacy is maintained.