IS600: Final Project

Uni Virtual Assistant

Mariam Yekini

December 2, 2020

Contribution of Members

At the beginning of the semester, our group assigned the roles below to each member:

- **Project Manager (John Nweke)** - oversees the project and make sure that the team delivers the expected outcomes on time
- **Quality Control Manager (Mariam Yekini)** - validates and verifies the delivered reports and programs to make sure that they are of high quality
- **Configuration Manager (Morgan Aldore)** - stores these versions and make sure that backups are taken appropriately

During the process of creating the Uni Virtual Assistant (UVA) program, each member was responsible for specific implementations in the program. Mariam was responsible for developing the commands.py and main.py files. Morgan was responsible for the weather.py and webbrowser.py files. Lastly, John was responsible for the emails.py file and conducting research on the feasibility of connecting the UVA program to a UMBC student account. However, John had no contribution to the project and Morgan had minimal. Each file was intended to be worked on equally by each member of the group, but due to the lack of input from John and Morgan, Mariam had to implement the majority of the project on her own.

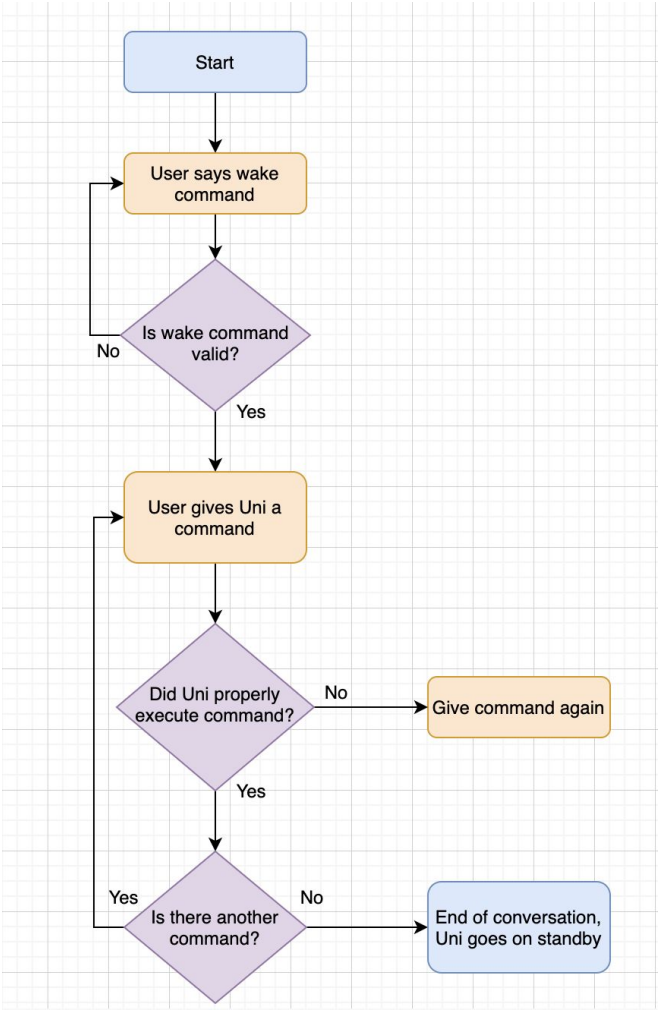
Goal, Motivation, and Approach

The goal of the UVA program was to create a program that is capable of providing college students and other professionals with information that can aid in their success. This assistant will be capable of greeting the user, taking notes, checking the weather, providing the date and time, searching wikipedia/google, opening Blackboard and YouTube.

The motivation behind creating this program is the firsthand experience of knowing how stressful balancing school, work, and extracurriculars can be. This program is designed to reduce the anxiety surrounding forgetfulness and works to take away small burdens so that students/professionals can focus on their productivity and success.

The approach taken to create the UVA program was aided with research. Through research I discovered the functionalities that virtual assistants usually execute and the flow needed to create it. As shown in the flowchart below, the first step in a successful virtual assistant program is saying a wake command to take the program off of standby. After the wake has been stated, if this is executed properly, the user can then give Uni another command and if it has not been executed properly the user will be prompted to give the wake command again. Following this same pattern, if Uni accurately hears the command, it will return the information associated with it and if not the user will have to repeat the command. Lastly, if there is another command to be executed, the user will again provide that to Uni and if there is no other command, Uni will go by on standby.

Flowchart



Code Concepts

In order to build the UVA program, there were a number of python concepts that were utilized. One important concept that I used was functions. Before I was able to create the main function that allows us to speak to our virtual assistant, there were other necessary functions that were created. For example, the recognizeAudio() function converts any input and converts it into speech.

Another concept that I utilized for this program is if/elif/else statements. This was utilized to conditionally execute a statement or another based on whether the first condition is true or false. For example, for the greetingType() function, if the time of the day is before 12 the if statement will be executed, otherwise the elif or else statement will be executed.

Exception handling was an important concept given the nature of the project. Depending on how loud the user is speaking, background noise and other factors, Uni may not accurately fulfill the intended command. In this case that this occurs, a try and except block is executed and the user can say their command again.

Furthermore, for loops and while loops were both used to iterate over a sequence. However, unlike the for the loop the while loop runs until the defined condition is no longer met. In the main program when Uni detects certain phrases in the speech such as “who is” or “where is” it will execute the intended command but if it is not detected it will not.

Lastly, the last important concept that was utilized in this program is operators. Arithmetic operators were used to concatenate strings, comparison operators were used to compare two values, logical operators were used to return true or false depending on the conditional statements, and membership operators were used to test if a sequence is presented in an object.

Dataset and Work Performed

In order to have a fully functional program, there were packages that needed to be installed and imported into the code. The speech recognition library was used to understand the user’s voice input and convert the speech to text. Pyttsx3 is a text to speech conversion library in python. Included in this was the SAPI5 which is a microsoft speech API that allowed for the customization of the voice for the program. The wikipedia library was used to summarize important information and is accessed through the web. Lastly, the pyjokes library was utilized to have Uni tell a range of jokes to the user on command. Along with the external datasets downloaded, various built-in python libraries were utilized as well. This includes datetime, calendar, and webbrowser.

The commands.py file contains the following functions needed to execute a range of commands. The recognizeAudio() function takes in an argument and converts it to speech. The takeCommand() function allows for microphone input and then returns a string output. The getCurrentDate() function allows us to ask Uni for the date and have it returned in speech. The greetingType() function allows Uni to greet us according to the time on the computer. After incorporating the necessary basic functions, the main function will call our recognizeAudio() function and any command asked of Uni will be executed with speech.

The main.py file then imports the functions created in the commands.py file in order to create a seamless, responsive program. With the use of a while loop and if/elif/else statements, Uni can take in voice commands and return responses depending on the conditional statements. The functionalities include: searching wikipedia, telling time, providing the date, taking notes, telling jokes, searching the internet for locations, opening YouTube and Blackboard.

The webbrowser.py requires a path to the web browser from a user’s local machine. Uni then requires a voice command for a request from the web. The user’s voice input is then recognized and is also outputted in string format for clarification before using the Web.get to fulfil the request.

The Weather.py is a text to text functionality where an Api key as well as the base url is required to link Uni with the weather website for information. It requires the user to input the city name where it then combines it with api key and the base url to give a text output with the weather information.

Results

Testing and Evaluation	
Stage	**Tasks**
1	- Ensured PC was working properly - Configuration of IDE - Packages
2	- Created basic functions needed in the main - Outputted basic functions with string before converting to speech
3	- Ensured basic functions were implemented correctly - Implemented speech functionality - Ensured main function works properly executes basic functions
4	- User Testing - Feedback
5	- Improvements - Maintenance

PHASE		DETAILS	Execution Status	Owner	Last Executed	
1	Project Definition and Planning	- Ensure PC is working	Passed	All	12/12020	PROJECT END
		- IDE Selection	Passed	All	9/21/2020	
		- Github Creation	Passed	Mariam	9/13/2020	
		Import Packages	Caution	Mariam + Morgan	12/1/2020	
		- Determine feasibility of connecting Uni to student account	Not executed	John		
2	Project Initial Development	- Output voice input in string format	Passed	Mariam	10/10/2020	
		Output current time in string format	Passed	Mariam	10/10/2020	
		- Output current date in string format	Passed	Mariam	10/10/2020	
		- Output greeting according to time in string format	Passed	Mariam	10/10/2020	
		- Output webbrowser input in string format	Passed	Morgan	11/30/2020	
		- Output weather in string format	Passed	Morgan	12/1/2020	
		- Output email content in string format	Not executed	John		
3	Project Launch & Execution	- Output current time with speech	Passed	Mariam	12/1/2020	
		- Output current date with speech	Passed	Mariam	12/1/2020	
		- Output greeting according to time with speech	Passed	Mariam	12/1/2020	
		- Ensure Uni responds to wake command	Failed	Mariam	12/1/2020	
		- Output webbrowser with speech	Failed	Morgan	12/1/2020	
		- Output weather with speech	Failed	Morgan	12/1/2020	
		- Execute all basic functions and commands through main function	Passed	Mariam	12/1/2020	
4	Project Performance & Control	- Final testing before user testing and feedback	Passed	Mariam	11/29/2020	
		- Final version control	Passed	Mariam	12/1/2020	
5	Project Close	- Improvements and testing	Passed	Mariam	12/1/2020	
		- Maintenance (continuous testing and debugging)	Passed	Mariam	12/1/2020	

Results Discussion

Overall, the UVA program was a success. The program can accurately capture what the user says and can execute a command. The program was tested on both OS and Windows systems and it runs properly on both. Furthermore, during user testing, the user shared that the functionalities that I included in the program was sufficient. However, I was able to make small changes such as having the time tell if it is am or pm and also indicating the date with month, day, and year.

One trade off in creating the program was deciding to leave out the Google Calendar API integration. With this integration, the user would have been able to ask Uni questions such as "am I busy tomorrow?" or "what do I have on December 8th?". However, given the time constraint and extensive debugging, the implementation was not successful. As a result, I included a functionality where Uni can take notes for the user in a text file and add the date and time that the note was taken. Another tradeoff was the wakeCommand() function. Although the function was capable of returning a string output during testing, I received errors when transferring that to speech. Instead, when the program runs, Uni greets the user before executing a command.

In the future as I continue to build on my python skills, I plan to incorporate more functionality into the UVA program. I plan to implement the Google Calendar API, wake command, include an interactive interface (GUI), include more functionalities such as playing music and explore the possibility of connecting Uni to Blackboard for students.

References

AskPython. (2019, September 5). *Python if else elif statement*. https://www.askpython.com/python/python-if-else-elif-statement

Gaddis, T. (2017). *Starting out with Python*. Pearson.

Python Packaging Authority. (n.d.). *Installing packages — Python packaging user guide*. Python Packaging User Guide — Python Packaging User Guide. https://packaging.python.org/tutorials/installing-packages/

Rockikz, A. (2020, October). How to convert speech to text in Python. PythonCode. https://www.thepythoncode.com/article/using-speech-recognition-to-convert-speech-to-text-python

Wigmore, A. (2019,December). How to Import Weather Data into Python. https://medium.com/analytics-vidhya/how-to-import-weather-data-into-python-scripts-7e9ff54f6aca