**z**

Candidate Number: 0019

H446/04 Programming project:  
Virtual assistant

Joseph Bostock

Contents:

[Analysis: 3](#_Toc121777506)

[Who are the stakeholders? 3](#_Toc121777507)

[What problem do they have? 3](#_Toc121777508)

[How will you make sure that your proposed solution is appropriate to their needs? 3](#_Toc121777509)

[Computational Methods: 3](#_Toc121777510)

[Abstraction and Visualisation: 3](#_Toc121777511)

[Thinking Ahead: 4](#_Toc121777512)

[Thinking procedurally / Decomposition: 4](#_Toc121777513)

[Thinking Logically: 4](#_Toc121777514)

[Thinking concurrently: 4](#_Toc121777515)

[Research: 5](#_Toc121777516)

[Initial Research: 5](#_Toc121777517)

[The Amazon Alexa: 5](#_Toc121777518)

[Advantages: 5](#_Toc121777519)

[Disadvantages: 5](#_Toc121777520)

[Apple’s Siri: 5](#_Toc121777521)

[Advantages: 5](#_Toc121777522)

[Disadvantages: 6](#_Toc121777523)

[BMW: 6](#_Toc121777524)

[Advantages: 6](#_Toc121777525)

[Disadvantages: 6](#_Toc121777526)

[Interview Questions: 6](#_Toc121777527)

[Proposed Solution: 7](#_Toc121777528)

[Response from the Interviewee: 7](#_Toc121777529)

[Hardware and Software Requirements: 7](#_Toc121777530)

[Conclusion: 7](#_Toc121777531)

[Success Criteria: 8](#_Toc121777532)

[Design: 10](#_Toc121777533)

[The Decomposition Diagram: 10](#_Toc121777534)

[Interface: 14](#_Toc121777535)

[Activation: 15](#_Toc121777536)

[Speech Input: 16](#_Toc121777537)

[Speech Translation Module: 17](#_Toc121777538)

[Processing: 18](#_Toc121777539)

[Question Classification: 19](#_Toc121777540)

[Communications: 20](#_Toc121777541)

[Random Response Modules: 21](#_Toc121777542)

[22](#_Toc121777543)

[Mathematics: 22](#_Toc121777544)

[23](#_Toc121777545)

[Geographical: 23](#_Toc121777546)

[General Knowledge: 24](#_Toc121777547)

[Words: 25](#_Toc121777548)

[Output: 26](#_Toc121777549)

[Key Variables: 27](#_Toc121777550)

[Variable Name 27](#_Toc121777551)

[Data Type 27](#_Toc121777552)

[Usage 27](#_Toc121777553)

[Justification 27](#_Toc121777554)

[Detailed Summary of the Process: 28](#_Toc121777555)

[Test Plan: 29](#_Toc121777556)

[Prototyping: 32](#_Toc121777557)

[Prototype 1: 32](#_Toc121777558)

[UIs: 32](#_Toc121777559)

[Main: 34](#_Toc121777560)

[Intent Classification: 34](#_Toc121777561)

[Determine Most Similar: 34](#_Toc121777562)

[Conversational: 35](#_Toc121777563)

[Maths: 35](#_Toc121777564)

[Music: 36](#_Toc121777565)

[Speak and Listen: 37](#_Toc121777566)

[Words: 37](#_Toc121777567)

[Wikisearch: 37](#_Toc121777568)

[Voice: 38](#_Toc121777569)

[Test Table: 38](#_Toc121777570)

[Database: 41](#_Toc121777571)

[Plans for Prototype two: 41](#_Toc121777572)

[Fixing of current errors: 42](#_Toc121777573)

[New features: 42](#_Toc121777574)

[Prototype 2: 43](#_Toc121777575)

[Changes to original algorithms: 43](#_Toc121777576)

[Test Table: 46](#_Toc121777577)

[Plans for Prototype three: 49](#_Toc121777578)

[Fixing of current errors: 49](#_Toc121777579)

[New features: 49](#_Toc121777580)

[Prototype three: 50](#_Toc121777581)

[New Features: 50](#_Toc121777582)

[Test Table: 53](#_Toc121777583)

[Table of Figures: 58](#_Toc121777584)

[Bibliography: 60](#_Toc121777585)

# Analysis:

## Who are the stakeholders?

The main stakeholders for my project will be the general public as well as those who are virtual assistant enthusiasts. They will be consulted during the production of this project, so that I can make the project as good as it can be. This project will also be available to those that are disadvantaged in a way that prevents them from being able to type on a keyboard or for those that do not have a microphone, so that many people can still use the virtual assistant.

## What problem do they have?

The problems that the stakeholders have are that they do not know answers to certain questions, meanings of words or other general knowledge answers. The user could ask for the date, the time, the weather forecast etc… This will allow the user to have access to a simple and easy-to-use virtual assistant. The virtual assistant will allow them to have access to search any kind of question they may need to, or just provide themselves with a little bit of entertainment for a short while, it will be able to help with a variety of things for example, helping with homework, or checking spellings.

## How will you make sure that your proposed solution is appropriate to their needs?

To ensure that the needs of the stakeholders are met, I will allow the virtual assistant to have access to all of the WikipediaAPI to ensure that it can answer the questions that can be given to it, as well as having custom messages for entertainment purposes. By allowing the virtual assistant to accept inputs from both Speech to text, via a microphone and also by allowing written messages, this will maximise the amount of people that are able to use the product as it will be accessible for those who either don’t have access to a microphone or are disabled in a way that prevents them from being able to type using a keyboard.

# Computational Methods:

## Abstraction and Visualisation:

In order to prevent the code from becoming too resource heavy and to also maximise efficiency, a lot of code will be reusable throughout the project, via the use of procedures and functions. The data that will be obtained from the virtual assistant will be used in order to allow the virtual assistant to generate better responses, understand more questions and overall be more helpful.

## Thinking Ahead:

For my project, I will need to take an audio input or a written input, and then be able to produce both an audio and written output. I will have to use both speech-to-text and text-to-speech. I will also have to use a separate database in order to allow the program to learn from itself. I will use the database for machine learning and as a way to store questions that have been asked, and some basic set questions, to allow the program to categorise them into subsections to allow the code to provide a better response.

## Thinking procedurally / Decomposition:

I can break my project down into smaller parts, for example the initial input, for example text or speech recognition to allow all different types of users, with all different types of equipment and hardware to be able to use the virtual assistant. The virtual assistant being able to understand the question that is asked of it by separating all different questions into types of questions with subcategories, such as personal, date and time, weather, general knowledge, or mathematics. Furthermore, the virtual assistant will be able to generate an answer to any question that is given to it, no matter what it is, even in the event that it does not understand the question that is given to it, this will allow for a better interaction between the use and the virtual assistant as there will be no errors, just the virtual assistant telling the user that it does not understand. Moreover, the virtual assistant will be able to produce both text and speech output to allow all users to be able to access the information given via either a sound output or a visual output. Finally, the virtual assistant should be able to learn from itself and previous encounters from the user to create a better understanding of questions that are given to it, to allow the program to better classify the questions, in order to generate a better response.

## Thinking Logically:

Thinking logically entails sections of the program such as branching and repetition. These are two techniques that determine the state of the program whilst it is running, in order to determine what it will do next. The idea of branching will be used constantly throughout the program, for example when deciding whether to use text or speech as an input; deciding which functions or procedures to run based of a certain question, for example when the date or time is asked, it needs to branch off and run the correct procedures or functions to return the correct information. However, with the use of repetition, I can prevent a lot of errors for occurring, for example if nothing is said when talking to the virtual assistant, I can ask the user to repeat themselves, or if the question is not understood, thus creating a more user-friendly experience to allow for easier usage of the virtual assistant. Furthermore, the use of repetition will be useful to create as many different scenarios for the virtual assistant to learn from. By allowing it to be exposed to a lot of different scenarios and thus a lot of different questions, it will allow it to create a better database, and a greater understanding of what each question means, as the more data it has access to, the better it will be able to categorise each question when it is asked.

## Thinking concurrently:

The idea of thinking concurrently means that multiple things will be able to be completed and understood simultaneously, in my program I plan on being able to implement / learn how to implement several different things that are similar, at similar times. For example, how to take a speech input, and how to output speech, seem to be similar tasks, as they both involve audio. I plan on doing these at similar times as this will allow both input and outputs to the code. After this, libraries such as Wikipedia, or google or any kind of search API, will be able to be completed at a similar time. After this, I plan on having a simple GUI to allow it to be more user-friendly than just a screen of code, whilst building this, I will have to understand how I will link the code into the User-interface, and also link both of these into an executable application.

# Research:

## Initial Research:

## The Amazon Alexa:

To begin with, the Amazon Alexa is a digital voice that can recognise spoken commands and reply back with a response, it can answer questions and perform tasks such as adding items to a shopping lists, or playing music. For my project, I will use several features that Alexa offers, for example speech recognition, speech output, the ability to understand questions and come up with the best possible reply, I will use the features because these are some necessities for a virtual assistant.

### Advantages:

It has an advanced ai behind it to ensure that it can come up with a very good solution to the question that is asked of it. As well as having access to a large data base with all of the commands it can run, for example playing music, or adding an item to a shopping list.

### Disadvantages:

It doesn’t allow the user to type, thus meaning for those that are deaf using the Alexa is impossible, however with my program you will be able to both type and speak to use it, this means that anyone should be able to use it.   
It also requires an internet connection in order to function correctly.

Figure - Amazon Alexa

## Apple Siri | Logopedia | FandomApple’s Siri:

Siri is very similar to Alexa however, you can type and speak to communicate with Siri, this means that it allows a wider variety of people to have access to it and therefore opens it up to a larger market. From Siri, I will use the text feature which will allow a wider usage for my project, as it will allow those who do not have access to a microphone or are audibly impaired to use my project.

### Advantages:

It has access to a vast database behind it meaning it can determine what is the best possible response for each question that is given to it. Allows users to both type and speak, easily accessibly as its on all Apple products. The ability to use both speech and text is a feature that I plan on having inside of my project.

Figure - Siri

### Disadvantages:

Requires an internet connection in order to function. Furthermore, it is also only accessible if you have an apple product, a large disadvantage of Siri is that in loud environments it struggles to understand queries that are asked of it and sometimes you must repeat yourself several times.

## BMW:

The BMW personal assistant knows the car that it is in, the fuel levels and servicing requirements thus meaning that it can keep the user up-to-date on everything to do with their car. It learns and remembers your needs so that it can provide the best support possible. I am going to use the self-learning and data remembering side of the BMW IPA because it will allow my project to understand more questions the longer that it runs for and then also allow it to have a more personal connection with the user by remembering certain bits of information.

Figure - BMW IPA

### Advantages:

Easy to use, learns your personal needs in order to be a better assistant. Also, by having it in your car, this means that you can talk to someone on the phone, without having to actually use your phone, thus meaning there are less drivers using their phone when driving.

### Disadvantages:

Cannot type to it, meaning that those that are deaf will not be able to use it. Furthermore, it requires you to have a BMW, which are expensive.

## Interview Questions:

Q: What type of questions should the virtual assistant be able to answer?  
A: The interviewee has stated that the virtual assistant should be able to come up with a response to any question that is asked of it.

Q: What features should the virtual assistant have?  
A: The interviewee then stated that a virtual assistant should be able to answer both logical questions such as definitions and also arithmetic questions such as equations to, for example, help a child with their homework, as well as being able to answer more personal, simple questions such as how are you, or just simple responses to greetings.

Q: How can I ensure that the virtual assistant is user-friendly?  
A: The interviewee then stated that to ensure that the virtual assistant is user-friendly, it should be able to explain, in the case of an error, why this error has occurred, as well as if there is a way to better ask this question, it should also contain some form of interface.

Q: Should the virtual assistant be able to remember your information, in order to, for example call you by your name if you speak to it?  
A: The interviewee said that it should be able to remember this information if you would like it to, so this information can be stored, but not accessible by other users.

Q: How can I ensure that the virtual assistant can be accessed by anyone?  
A: To ensure that the virtual assistant can be easily accessed by anyone it will need to be free and be easy to use, for any age or experience with technology.

Q: What is a good way to allow a more personal connection with the user and the virtual assistant?  
A: To allow a more personal connection with the virtual assistant, the virtual assistant should be able to remember certain details, such as a name, and other things such as maybe ages of the user or other simple facts about them.

Q: How would you use the virtual assistant?  
A: The interviewee stated that the virtual assistant would more than likely be used for day-to-day questions, for example the weather or the time of day, moreover, it could be used for checking spellings of words, or helping with simple maths equations or just to ask for random definitions of words, events of things around the world.

## Proposed Solution:

In order to meet the requirements and the overall criteria of the stakeholders, I will attempt to ensure that the virtual assistant has an easy-to-use interface with simple error messages in the event that anything goes wrong, for example the virtual assistant not understanding the question, or it not being able to come up with an answer for whatever reason. Furthermore, it will be able to distinguish between logical and arithmetic questions to aid those that need it. Finally, it will be able to come up with an answer to any question that is asked of it even if it is a response telling the user that the virtual assistant cannot help with the question that has been asked of it. It will take an audio input or text input for those that do not have access to a microphone, it will process this information to provide the user with the best possible answer to the question that has been asked of it and then it will provide both a text and audio output. It will be able to learn from itself, by making a record of questions asked and what these questions mean, in an attempt to allow for a better experience for the user to ensure that any questions that couldn’t once be answered, can, in the future, be answered.

## Response from the Interviewee:

The interviewee stated that it is a great idea to ensure that there is an easy-to-use interface and that there shouldn’t be a lot of errors anywhere to ensure that it is easy to use for the user. However, they stated that it should be able to not only answer logical and arithmetic questions but also more personal questions to give it a sense of feeling and realism. Moreover, the interviewee asked if storing the user’s personal data is a good idea, as this could possibly lead to any issues.

## Hardware and Software Requirements:

For the virtual assistant, the user will need a functioning computer with access to the internet. I aspire to the fact that I will be able to compile the program and have it accessible either through a website, or through an application, thus meaning that the user will not have to install several libraries or other programs to allow it to run. In the event that the program is created into an application, it should be a rather small application and not take up a lot of disk space, so for the user to be able to run the program I would say that they also need to have at maximum 1 GB of free storage space. However, to create the program I will need to use my laptop and Visual Studio Code, as I will be working across several different python files and a database in order to create the virtual assistant.

## Conclusion:

Based on my research, and in conclusion, I aspire to have a program that can accept both text and speech inputs to allow a wider range of users; as well as having both a text and voice output to follow this. It will also be a program that can learn from itself to allow for a better understanding and providing a better service to the user. It will have access to be more personal with the user by having the ability to store information about the user if chosen and then this data can be used to create a better experience for the user. Additionally, the virtual assistant will have access to things such as a Wikipedia API to allow it to provide good answers to questions that are given to it. It should be able to hold a normal, basic conversation to allow for ‘a sense of feeling and realism’. It should also have the ability to understand simple arithmetic and maybe be able to do some more complicated arithmetic. It should also have a simple and easy to understand interface, to ensure that any user can very quickly determine how to ask the virtual assistant what they wish to ask it. However, a large drawback for the program is that you must have internet access in order to use it, otherwise the program will not be able to function correctly. I have not yet decided whether to make the program accessible via the internet as a webpage, or as an application that can be used on a computer’s desktop, as there are both advantages and disadvantages for both of these options. Finally, for the program to be deemed a success, I want it to be able to provide a good, acceptable response to any question that is asked of it. As well as allowing the user to traverse the program’s interface and understand the interface in under a minute. Finally, my last success criteria, is that the virtual assistant, will be able to understand, and speak several different languages, such as French, German, Spanish, Portuguese, Dutch, Japanese, Chinese, and hopefully more.

## Success Criteria:

|  |  |
| --- | --- |
| Condition | Met? |
| Can the user enter text as a form of inputting information or a question? |  |
| Can the user speak to virtual assistant as a form of inputting information or a question? |  |
| Can the virtual assistant recognise the speech that has been said to it? |  |
| Can the virtual assistant understand other languages? (French, German, Spanish, Portuguese, Dutch, Japanese, Chinese) |  |
| Can the virtual assistant take the statement or question given to it, and determine what type of question it is using machine learning and a database? |  |
| Can the virtual assistant add question examples to database if there is nothing in the database similar? |  |
| Can the virtual assistant tell the time? |  |
| Can the virtual assistant tell the date? |  |
| Can the virtual assistant do Addition? |  |
| Can the virtual assistant do Multiplication? |  |
| Can the virtual assistant do Subtraction? |  |
| Can the virtual assistant do Division? |  |
| Can the virtual assistant work out the square root of a number, or that number squared? |  |
| Can the virtual assistant do more advanced mathematical equations? |  |
| Can the virtual assistant understand the location of where it is accessed from? |  |
| Can the virtual assistant access the weather forecast of a certain location? |  |
| Can the virtual assistant create directions to a certain location based on where you are currently? |  |
| Can the virtual assistant translate into other languages? |  |
| Can the virtual assistant provide the correct spelling for words |  |
| Can the virtual assistant find definitions for words? |  |
| Can the virtual assistant find synonyms for words? |  |
| Can the virtual assistant find antonyms for words? |  |
| Can the virtual assistant provide responses to simple statements, such as ‘hello’ or ‘how are you?’? |  |
| Can the virtual assistant answer general knowledge questions? |  |
| Can the virtual assistant output text? |  |
| Can the virtual assistant output speech? |  |
| Can the virtual assistant provide simple, and easy to understand errors in the event that something goes wrong? |  |
| Is there a simple, and easy-to-understand interface? |  |
| Can the virtual assistant remember information about the user? |  |
| Can the virtual assistant remember previous encounters to allow it to learn from itself? |  |
| Can the virtual assistant be used without the use of the internet? |  |
| Is the virtual assistant an application that can be run on a computer? |  |
| Does the virtual assistant have a ‘keyword’ that it is listening for in order to be activated? |  |
| Do previous encounters with other users on other computers update the virtual assistant on all machines? |  |
| Can the virtual assistant tell simple jokes, in order to make it feel more personal? |  |
| Is there a User Interface? |  |
| Does the user interface have a simple colour scheme? No more than a few major colours |  |
| Does the user interface have buttons, that are easily accessible, to allow the user to input a question? |  |

# A picture containing graphical user interface Description automatically generatedDesign:

# The Decomposition Diagram:

Figure - Decomposition Diagram

My decomposition diagram is separated into 4 main parts, to begin with, there is the interface which is composed of 4 major components. The ability to have a user input, and an on and off switch, an interactive image and then also a rotating tips section. The ability to have a user input will have both a text box and will always be listening for the keyword. The interactive image will make it obvious when the virtual assistant is responding to a question and when it is not responding to a question. Furthermore, the rotating tips section will contain a list of things that the virtual assistant can do so that the user knows its full capabilities. Next is the input section, here there is an audio and text input. The audio input is always listening for the keyword; however, the text input is waiting for something to be put inside of the textbox. Once there has been an input, it will check what language the input is in, then It will translate it to English in order to allow it to be processed. Next is the processing, inside of the processing it will take the input from the inputs section and check the database to see if the question is in there, if it finds one then it will take the category that the question used and run the required modules. If it cannot find a question that is the same inside of the database, it will run the question classification module to assign a category to the question and then run the required modules. These modules may consist of the word module by which the user can ask for spellings, ask for definitions, synonyms or antonyms as well as translations into other languages. There is also the general knowledge module where the user can ask general questions and be given an answer that will be found on google or Wikipedia using an API. Each of these responses will be summarised to allow the user to be able to quickly understand what the answer is. After this, there is the maths module, this has the ability to do addition, subtraction, multiplication and division. As well as the ability to do exponentials and logarithmic calculations, with squares, cubes or numbers to the power of, with also square and cube rooting of numbers. Next is the interactions, by which the virtual assistant will be able to maintain a conversation with the user, by having the ability to give simple greeting and leaving statements as well as the ability to tell jokes. Finally, there is the geographical and time module. It will locate the user based on the longitude and latitude coordinates of the user and then provide a location on Earth for them, this will allow the virtual assistant to work out the time zone of the user. Using this information, the virtual assistant will be able to provide the time, date, and weather forecast for the user over a set period of time. The last section is the outputs, this is where the response from processing will be gathered and dealt with. The program will check to see what language the original input was in and then translate it back into that language thus allowing the user to understand it. Then, the virtual assistant will output this in both speech and text format, using print statements as well as text to speech libraries.

I have designed my decomposition diagram in this way to allow it to be easily followed by the programmer, thus allowing me to easily understand what bits of the code to do when. The way it is broken down, allows me, as the programmer, to follow and interpret which bits of the code will be separated and which bits to do together. Due to it all being decomposed and broken down into smaller, more manageable parts, it will allow the actual coding process to be much faster as I will know the logistics of which bit of programming to do when.

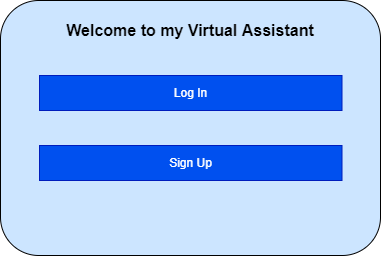


Figure - Home Page

Graphical user interface, application

Description automatically generated  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
The virtual assistant has a login page to allow users, to store their data, this will allow users that share a computer to have separate data stored inside of the virtual assistant. It will allow the virtual assistant to remember a name, and with the use of a username, in the event that the user doesn’t want to use a real name, the virtual assistant will be able to call the user by their username. It is made of 4 main components, the username input, the password input, the forgotten password and create an account. In the first two, you can enter the username and password, the program will check if the user has inputted data that corresponds to a user’s login. Next, the forgot password will allow the user to input an email and have a password reset available. As can be seen below.

Figure - Login UI Design

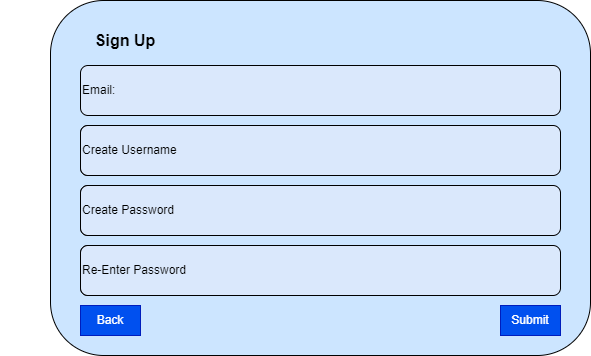
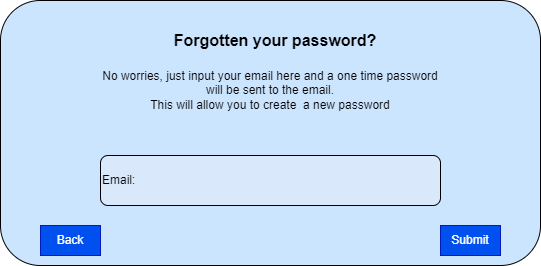


Figure - Forgotten Password UI Design

Figure - Create an account UI Design

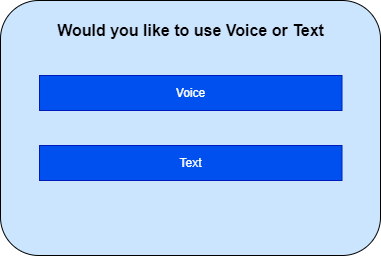
  
Next is the Create account. It will allow the user to input a username, and set a password, as well as confirm the password. It will check to make sure that password and confirm password are the same. It will run the passwords through a hash program and store it in a database. It will then be accessible by the program when someone wants to login. Every time that a user wishes to login, they will enter their username and password, it will run the password through a hash function, and compare this with the password stored in the database if they match then the user will be allowed to login.

Figure - Input Selection

## **Interface:**

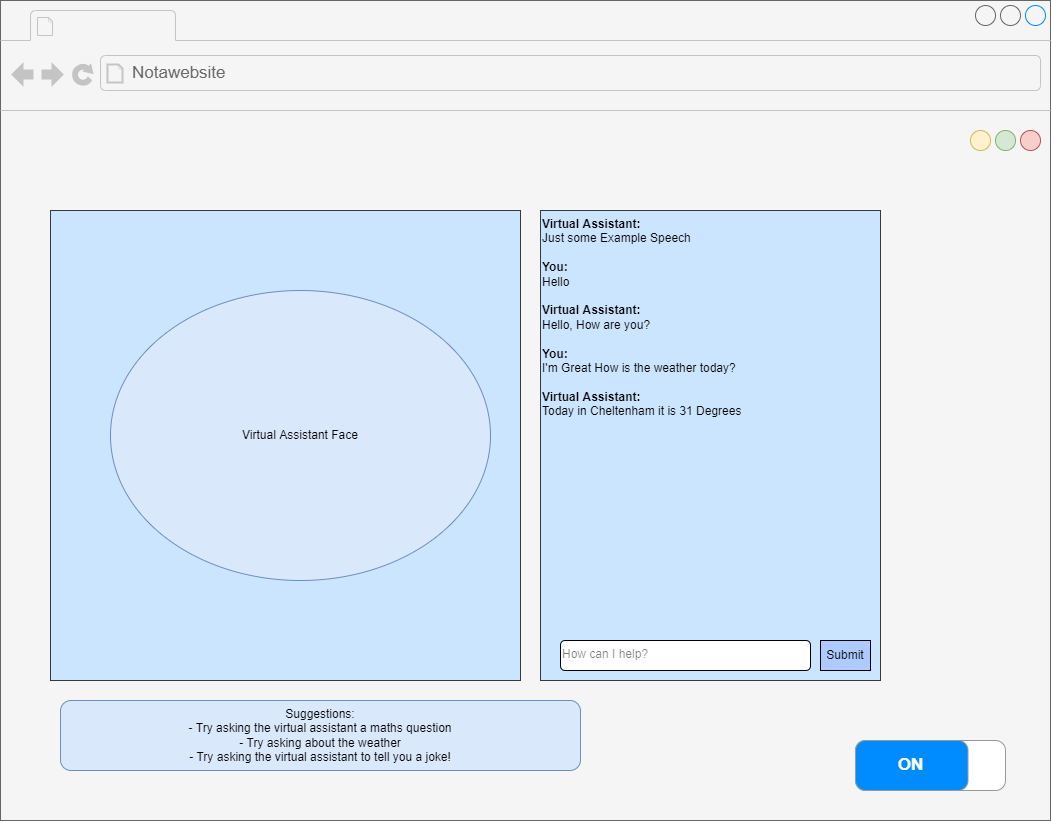
In figure 5, there is my initial idea for what the main page of the virtual assistant will look like. It is a very simple idea as it is my first attempt at ever designing something like this. It is clearly broken down into a few major parts, each fitting the criteria set out for it. There is a clear text box for written inputs from the user, and it will take audio input from the user if they have access to a microphone and say the activation keyword. There is an on or off switch to allow the program to be turned off or on depending on what the user wishes to do at the time. There is a suggestion box where the information will change over time to allow the user to see what the virtual assistant can do and to allow the user to know all of its capabilities. There will also be an interactive image where the circular shape is. The overall aesthetic is very blue themed as studies have shown that it is a calmer colour that doesn’t draw too much attention, this will allow the focus to not be on the colour scheme but on the virtual assistant itself. In my opinion the design of the UI can and will be changed quite a lot over the course of the project in order to create the best possible UI for the program.

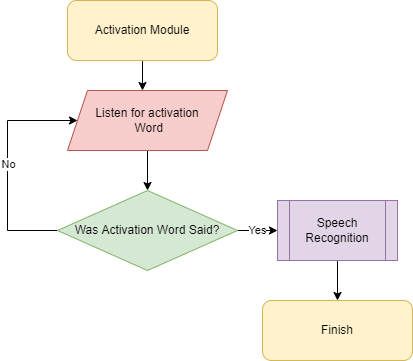
Figure - UI Design 1

### Activation:

This starts when the program starts, and it listens for the activation word that will be set in the program’s code. It checks to see if the word has been said, if not it continues to listen, however if it has been said then it will run the speech recognition module.

The activation module is one of, if not the most important modules inside of the code, as this will allow the program to not just take whatever is being said, but to listen out for a keyword and then begin to respond.

To create this module, I will use a python library known as pvporcupine, this is a wake word library that will allow me to create this module. I have decided to use an activation word to ensure that in the case that the user is having a conversation with someone else, that the virtual assistant is not trying to answer the questions or anything else, unless it’s being addressed.



#### Speech Recognition Module:

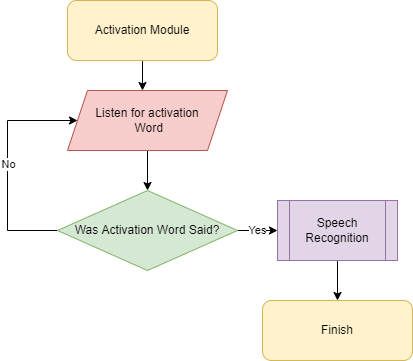
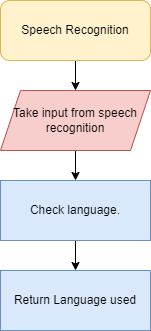
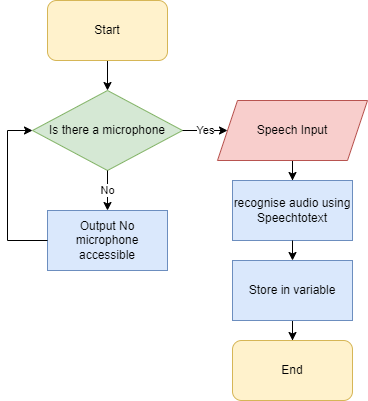
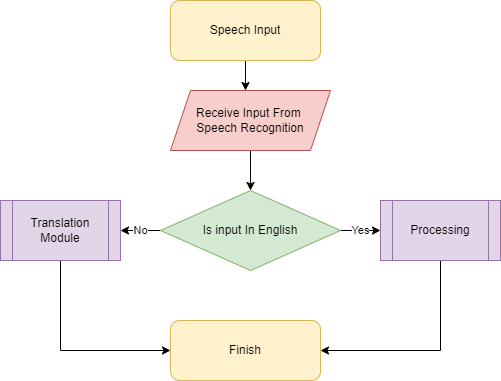
Checks if a microphone is in use, if not it will output that there is no usable microphone. If there is a usable speech input. It records this using speech to text and stores the text in a variable.

Figure - Speech Recognition Module Flowchart

Figure - Activation Flowchart

### Speech Input:

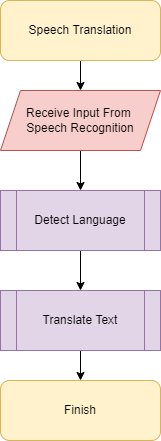
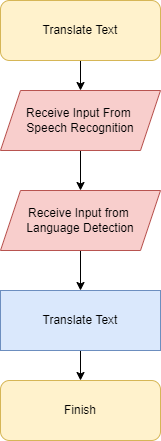
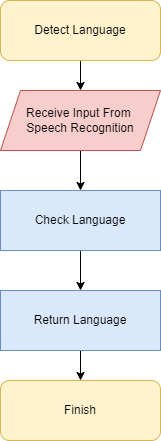
Next is the speech input module, this takes the input from the speech recognition module, and checks to see if the input is in English by running a check on it. If it is not in English, then it runs the translation module, however if it is in English then it begins the processing.

In order to create the speech input section of the program I will use a python library called speechrecognition. And to check if the language is English, I will use googletrans. I have decided to use speech input as well as text input to maximise the usability of the program, so that anyone can use it even if they don’t have a microphone, or they are blind, or for whatever other reason, cannot type. The ability to use a voice input will allow people to very quickly input larger questions or queries.

### 

#### 

Figure - Speech input Flowchart



## 

Figure - Language Detection Module Flowchart

Figure - Speech Translation Flowchart

Figure Translation Module Flowchart

### Speech Translation Module:

This takes the input from the speech input module that is above, it uses a python library called googletrans to detect what language the question is in, then translates from that language into English, then it returns this text to processing.

In order to create this module, I will use the googletrans python library to detect the language and then also googletrans to change the text into English so that it can be processed.

I have decided to add a speech translation module to allow users from all different nationalities and ethnicities to use the program, having the ability to speak different languages will allow the virtual assistant to help and assist people from all around the world.

#### Detect Language Module:

Takes an input from speech recognition, checks the language using google translates auto-detect feature, then returns the language used.

### Processing:

This flowchart takes the input from the translation, or speech recognition modules, and checks the database (CSV) for similar questions, if a question that is the same is found, it will run the required modules for that question, however if the question is not found then it takes the question and run the classification module and then run the required functions for that question. The choice for having the virtual assistant process information, questions and data in this way, in my opinion is a very good choice, it will allow the program to be fast, and not have to use millions of ‘if and else’ statements in order to properly respond to questions. Having a database with a lot of set questions that can then be built upon will allow the virtual assistant to really become personalised with the user.

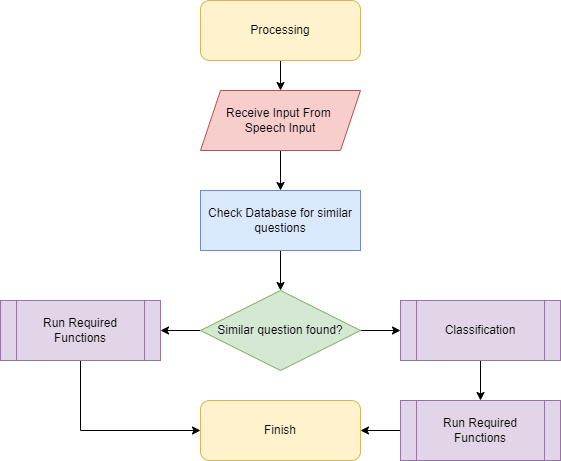


Figure - Processing Flowchart

### Question Classification:

The question classification module will be used when the question that has been asked by the user, is not already inside of the database. It checks the database once more, and then runs the code using Scikit learn to best categorize the question based on other questions it has received. Once it comes up with a suitable suggestion, it adds it to the database alongside its category. I have decided to classify questions in this way to allow the virtual assistant to very quickly, run the necessary modules in order to properly respond to the questions and queries provided by the user. As I mentioned above having the ability to classify questions, removes the need for thousands of if statements to generate a good response to a question.

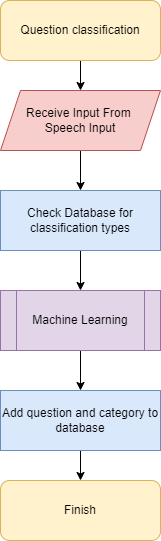


Figure Question Classification Flowchart

### Communications:

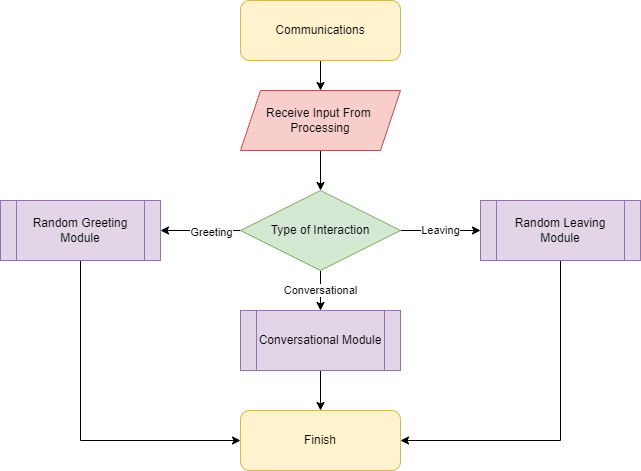
The communications module receives the question from processing and checks to see what type of question it is, whether the user is greeting, leaving or just having a conversation. Inside of each of the 3 there are many different possible responses. In greeting there will be responses such as hello, or how are you, or what’s up, just to name a few examples. In the leaving module there will be responses such as goodbye or see you later. And finally, in the conversational module there will be things such as simple responses to how are you questions, some jokes and other little conversational topics and responses that can be said to the user in order to maintain a conversation. Having the ability to hold a conversation is a necessary and vital part of a virtual assistant, as without the ability to maintain a conversation, it becomes lifeless. However, with the ability to maintain a conversation, the virtual assistant springs to life and allows the user to be entertained and assisted in the ways that they may need.

Figure - Communications Flowchart

### \\svr-stustore1\16intake\16BostockJ\Downloads\flowcharts.drawio (19).png\\svr-stustore1\16intake\16BostockJ\Downloads\flowcharts.drawio (18).pngRandom Response Modules:

Figure - Random Greeting Module

Figure - Random Leaving Module

Figure - Random Joke Module

Figure - Random Compliment Module

Figure - Random Insult Module

Each of these modules will allow the virtual assistant to have several responses for each statement, question or sentence said to it. It will allow it to not just repeat the same things over and over again and will also allow for the developer to quickly and easily add new possible responses.

## 

### Mathematics:

To begin with the maths module takes the question, and runs it through a set of if / elif statements to determine what type of question it is. To begin with it checks for simple signs or keywords to check if it is addition, multiplication, subtraction or division, if it is then it will run the required code, next it checks if it is to do with squaring, cubing to the power of or the inverse, a square or cube root, yet again if it is it runs the required modules, if not then it carries on. Next, it checks if it is asking for trigonometric values such as Sin, Cos or Tan, once more if it is then it runs the required modules if not then it runs the other maths modules, that will be added in the future.

#### Pseudocode:

Figure - Mathematics Flowchar

Determine\_most\_similar\_phrase(text)  
splittext(text)  
func(text)  
speak\_listen.say(output)

The mathematics module will allow the virtual assistant to run calculations to help the user with many different, simple forms of maths, the virtual assistant will be able to work out the calculation and complete it much faster than a human and will return the answer quickly, this will allow the user to have the response that it needs as quickly as possible.

## 

Figure - Geographical Flowchart

### Geographical:

The geographical module takes the input from processing once more, it locates the user based on latitude and longitude coordinates and runs these inside a simple module to locate the user. If the user wanted to know the date or time it then tells them this using the information that it got based on the user’s location and therefore the user’s time zone. If the user wanted the weather forecasts, the virtual assistant will run the required code and modules to tell the user the forecast in the user’s area using the locational data that it acquired earlier.

#### Pseudocode:

Location = Locateuser()  
if input == date or time:  
 datetime(input,location)  
elif input == weather:  
 weatherforecast(input,location)

I have decided to add a geographical module to the virtual assistant as having this will allow many features, the ability to locate a user, tells you there time zone, thus allowing the virtual assistant to provide a date, time and weather forecast as well as being able to provide any other kind of locational information, whether that may be local news or something else.

### General Knowledge:

Figure Words Module Flowchart

The Virtual assistant takes the input from processing, using the question it has received, it runs the keywords module to determine which of the words in the question are needed to search. It will then run the keywords through a search using the search module, after this the response will be summarised using the summarise module, and then the response will be returned.

#### Pseudocode:

Key = Keywords(input)  
answer = search (Key)  
summary = summarise(answer)  
return(summary)

I have decided to have a general knowledge section to allow the program to be able to come up with a response to any other question that has been asked. For example, if the question isn’t mathematical, or geographical or something to do with words, then the virtual assistant will be able to provide a good response to it using an API from either Wikipedia or a different provider of information. I also decided to have it be summarised to allow it to be easier to understand for the user.

### Words:

The word module takes the input from processing and allows the user to get definitions, spellings, synonyms or antonyms as well as translations into other languages.

The virtual assistant will run specific checks to determine what the user wants whether that is a definition or a translation, or anything else that the virtual assistant’s word module can offer.

#### Pseudocode:

If input == Definition:  
 DefinitionModule(input)  
elIf input == Spelling:  
 SpellingModule(input)  
elIf input == Translation:   
 TranslationModule(input,lang)  
elif input == Synonym or Antonym:  
SynonymantonymModule(input)

I have decided to have a words module to allow the virtual assistant to provide definitions, spellings, translations or synonyms as well as antonyms.

### Output:

Figure - Output Flowchart

It takes an input from the output of the modules that have been ran previously, and checks to see what language the code was in originally, and translates the response back into the original language. After this then it runs the Text-to-speech module and outputs the response back to the user.

#### Pseudocode:

if lang != English  
text = translate(input,lang)  
return(text)

I have obviously decided to have an output module to allow the information and response that has been generated by the virtual assistant to be given back to the user after it has all been processed, and then outputted as both text and speech, this will allow users with all different abilities or possible hinderances such as being deaf or blind, hence preventing the user to not read or hear. Having both outputs will allow as many people as possible to be able to use the virtual assistant.

## Key Variables:

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Data Type | Usage | Justification |
| Keywords | String | I will use this to have the activation, wake word that the program will be listening for. | Having a wake word will allow the user to be able to use the virtual assistant’s mode |
| input | String | Used to store information from the input text box. | Will allow the input to be processed. |
| said | String | Used to store information from the speech recognition module. | This will allow the input to be processed. |
| InputLang | String | Used to store the language of the input. | To translate from and to when processing or outputting the data. |
| Question | String | To store the question that has been posed after it has gone through the classification process | To allow the question to be processed. |
| Category | String | The category that is assigned to the question once it has gone through the classification process. | To allow the program to know which modules to run once it has decided how to classify the question. |
| Temp | String | Any temporary variable that will be switched out later | If a bit of information is just temporarily stored, it will use the Temp variable |
| Output | String | To allow the data to be outputted, via text and text-to-speech | To allow the user to see the response to the question that they asked the virtual assistant. |
| Longitude | Float / Real | To store the user’s longitude when finding the location | To allow the virtual assistant to find the location of the user. |
| Joke | String | To store a joke generated by the code | To allow the virtual assistant to tell jokes |
| Audio | string | To allow the speech recognition to listen to speech and return it as text | To allow the user to be able to speak to the virtual assistant. |
| Latitude | Float / Real | To store the user’s longitude when finding the location | To allow the virtual assistant to find the location of the user. |
| Text | String | Used to pass most data across files | Easy to remember variable with good, useful name. |
| Email | String | Used to have the user’s email stored inside of a variable that can be passed to the file | Needed in order to allow the user to use the OTP function when they forget the password. |
| Username | String | Used to have the user’s username stored inside of a variable that can be passed to the file, and used to login | Needed in order to allow the user to login. |
| Password | String | Used to have the user’s password stored inside of a variable that can be passed to the file, and used to login | Needed in order to allow the user to login. |
| Result | Float | Used to store the result of any equation done by the maths function. | Needed in order to store the value that is calculated |
| encmessage | Bytes | Used to store user’s data to the files in order to not have data stored in plaintext. | Needed to prevent data being stored in plaintext. |
| decmessage | String | Used to check if inputted information is the same as the information stored in the file. | Needed to check if the user can login using their inputted details. |

## Detailed Summary of the Process:

The project will be broken down into key segments: The interface, The inputs, all of the processing and the outputs. To begin with, the interface. It will be a simple graphical user interface that will open when the application is run. It will have a simple colour-scheme, that I am yet to decide on, this is to make the virtual assistant the main focus as opposed to distracting people with too many colours. It will also have very bold and easy to find buttons, this will allow the user to work with the virtual assistant, ask it questions or do what the user needs. The User interface will also have an image of the virtual assistant to allow it to be more interactive, in theory the image will open its mouth when it talks, and possibly have its eyes follow the mouse cursor, just for an added level of interaction. Each of the buttons that are on the screen should have obvious explanations as to what they are to allow simple and easy use of the virtual assistant to all of those who wish to use it, even if they don’t have much experience with computers. The inputs will be taken through both speech and text, thus allowing a greater variety of people with different types of equipment to use the virtual assistant. There will be a large text box on the right-hand side of the interface that has a clear indication of where to type the text if the user wishes to input text, however if the user would rather converse with the virtual assistant via speech, this will also be possible as all that will have to be done is to say the activation keyword and then ask the question. After this the program is broken down into several modules. To begin with there is the simple response module, this takes the samples in the json files and randomly choses one to respond with from the correct file depending on what is inputted. There is a maths module that works by taking the input, determining which function it should run, taking the numbers from the string, adding them to an array and then running the correct calculation, then outputting it. There is a word module that scrapes responses from dictionary.com and provides spellings by taking the string splitting it and adding each letter to an array and then outputting them. It also scrapes the web for synonyms and antonyms if the user asks for those. The virtual assistant can also play music. It works by first removing the work play from any sentence, for example play x song by y artist. It takes the name of the song, searches for it on YouTube, finds the first 12 options that appear on a full YouTube page. Takes the URL from the first video and using vlcmediaplayer, streams it. I believe that having the ability to play music is a key part of any Virtual Assistant. There are many conversational modules within the program that make use of json files for set responses, for example all of the greeting, compliment, leaving, personal questions and insults, these all have predefined responses to possible questions. It determines which category the user is saying and then randomly selects one of them to output. Along with the common replies / response code there is also a joke module. It tells computer science related jokes. The wikisearch module is also very important, this module uses the WikipediaAPI to take a keyterm, if it canot find a correct wikisearch then it suggests a possible option and asks you to try again. By having a wikipedia module it allows the user to ask any questions and it will provide a response.

## Test Plan:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Test Type** | **Possible Inputs** | **Desired Outcome** |
| Account Creation | Normal | Valid Login credentials, valid email, valid username and two valid passwords that are the same as each other. | Allow the user to create an account. Store all of the login details inside of the login credentials file, with each variable separated by a comma. |
| Account Creation | Erroneous | Blank Inputs. Invalid emails, for example without @.  Passwords that don’t match. | Provide the user with a popup, telling them that they haven’t inputted the correct information. For example, “invalid email”, or “passwords don’t match” |
| Account Creation | Boundary | A very long email, username or password input | Allow the user to create an account using the very long credentials, and then store these credentials inside of the login credentials file. |
| Login | Normal | Correct username and password that have been used when creating an account | Allow the user to login by checking the login credentials file and comparing it to the values that have been inputted. |
| Login | Erroneous | Incorrect Username and password | Provide the user with a popup, telling them that they haven’t inputted correct login details. |
| Login | Boundary | A very long password or username | It will allow the user to login with a very long password or username as long as the credentials are stored in the file. |
| One Time password email | Normal | A normal email that’s been used to create an account previously. | It will send an OTP to the user’s email. |
| One Time password email | Erroneous | An email that hasn’t been used to create an account previously | It will say that there isn’t an account with this username. |
| One Time password email | Boundary | A very long email | It will send the user an email as long as they have created an account using that email |
| OTP password change | Normal | A correct OTP and a new password | It will change the password associated with the email. |
| OTP password change | Erroneous | An incorrect OTP | It will tell the user there is an incorrect OTP |
| OTP password change | Boundary | A very long OTP | Tell the user it is an incorrect OTP. |
| Listen for wakeword | Normal | User says ‘Oi Badger’ | The program starts listening to what else is said. |
| Listen for wakeword | Erroneous | User says nothing | Nothing happens. |
| Take text input | Normal | User inputs a question | The program responds accordingly |
| Take text input | Erroneous | User inputs nothing | Nothing happens |
| Answer Maths questions | Normal | A normal maths calculation e.g.  6 x 8  124 x 25  23 + 2358 | Provide a correct response |
| Answer Maths questions | Boundary | Two very large numbers in a maths calculation, or several numbers. E.g.  124 + 235 + 12 + 24 + 234 | Provide a correct response |
| Provide a spelling | Normal | User asks for a spelling of word | Spelling is said letter by letter |
| Provide a definition | Normal | User asks for definition of word | Definition is provided |
| Provide a translation | Normal | User asks for a word or sentence to be translated into another language | Word or sentence is translated |
| Tell a joke | Normal | User asks for joke | Joke is told |
| Play a song | Normal | User asks for song | Song is played |

# Prototyping:

### Prototype 1:

Prototype one consisted of a lot of key features. To begin with it had UIs for account creation, log in, as well as one-time passwords that had the capability to email the user if they had an account created. Furthermore, there was also a user interface for the main program where you could input the questions that you had.

The program can Listen for input, or you can text in order to input into the program. It has a variety of modules ranging from maths questions to general knowledge questions. It can also control my smart light that I have at home. Furthermore, it can also play any song that is online and also has the ability to tell jokes and maintain a conversation. Once the virtual assistant has come up with a response, it will both print out a response and say it aloud by utilizing pyttsx3.

### UIs:

Figure – Sign up or login screenshot

Graphical user interface, application

Description automatically generatedThis window allows you to select whether you have an account or would like to create an account. It checks if either of the buttons is pressed and then runs the correct window based on what is pressed.

Text

Description automatically generatedBackground pattern

Description automatically generated Below can be seen the login system, which checks if the username and passwords are inputted, and then checks them in the database. If they are found then it opens the next menu, else it tells you that the login you have entered is incorrect.

Figure - Login Code

Figure - Login Screenshot

Background pattern

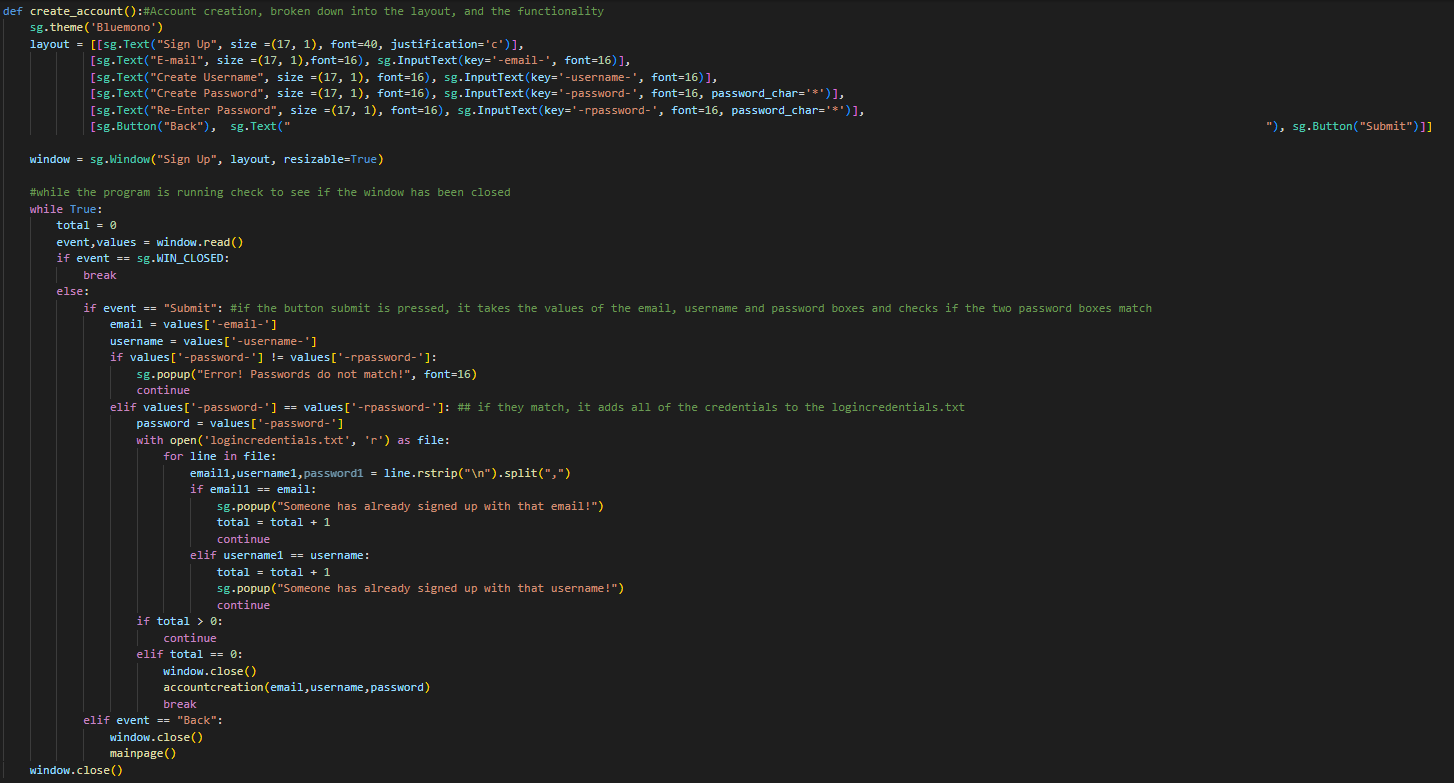
Description automatically generatedThis code defines the menu, opens the menu and waits for an input. It checks if the window is closed, if it is closed then it breaks from the code. Next, if the submit button is pressed, then it takes the values from the inputted boxes, it first checks if the email is alreay in use, and then I checks if the username is in use. After that, it checks if the two passwords are the same, if they are not the same then it returns a message telling them that they need to make sure that the two passwords match. If the two passwords match, then it will close the window and run into the backend for the account creation where it opens the file and writes it into the file.

Figure - Sign up Screenshot

Figure - Create an account front end code

Main:  
Text

Description automatically generated This code is defining a class called Assistant, which takes a name as input and assigns it to the instance variable self.name. The Assistant class has one method called reply, which takes three arguments: text, uuid, and choice. This method uses the intentclassifier object to predict the intent of the text input, and then uses a dictionary called replies to map the predicted intent to a function. If the function associated with the predicted intent is callable, the function is called with text and intent as arguments. If the function is not callable or if there is no function associated with the predicted intent, the Assistant says "Sorry, I didn't understand". Any exceptions that occur during this process are caught and handled.

Figure - Main python code

### Text Description automatically generatedIntent Classification:

The code above is a machine learning model that uses a Naive Bayes classifier algorithm to classify the intent of a given piece of text. The model is trained on a dataset of sample text and their corresponding intents, which is provided in a CSV file. The model uses the CountVectorizer and TfidfTransformer classes from the scikit-learn library to transform the training data into numerical feature vectors, which are then fed into the LinearSVC classifier to train the model. Once trained, the model can be used to predict the intent of new text by using the same transformation steps on the new text and then passing the resulting feature vector to the classifier's predict method. The model also has a filecheck method that checks if the given text already exists in the training dataset and if not, appends it to the dataset for future training.

Figure - Intent Classification Python Code

### Determine Most Similar:

Text

Description automatically generatedThe code above is a function that takes in two inputs, a text string and a dictionary of phrases (intent\_dict). It compares the input text to the phrases in the dictionary and returns the phrase that is the most similar to the input text. The function uses the difflib library's SequenceMatcher to calculate the similarity between the input text and the phrases in the dictionary. It then sorts the dictionary of phrases by the similarity scores in descending order and returns the most similar phrase. If the input dictionary only contains one phrase, the function simply returns that phrase. If the input dictionary is empty, the function returns None.

Figure - Determine Most Similar Code

### Conversational:

A screenshot of a computer

Description automatically generated with medium confidenceThere are many conversational modules within the program that make use of json files for set responses, for example all of the greeting, compliment, leaving, personal questions and insults, these all have predefined responses to possible questions. It determines which category the user is saying and then randomly selects one of them to output. Along with the common replies / response code there is also a joke module. It tells computer science related jokes.

Figure - Common response code

Text

Description automatically generated

Figure - Jokes Module Code

Maths:  
Next is the maths module, this is quite a large module so I will break it down into several sections.

Text

Description automatically generatedText

Description automatically generatedTo begin with, it takes the input statement and uses the determine most similar module from earlier to determine which statement it is most alike, then it runs the module defined next to it.

Figure - Maths module second

Figure - Maths Module Start

It uses the split text function to separate the string and find any digits or floats inside of the string, and adds them to a list, after this it returns the value to the correct function, does the math calculation and outputs it. I have a maths module as I believed it would be an easy module to start with as it doesn’t require too much knowledge of external APIs and other functions. It was set out in my original design, and I have kept mostly to the idea of how I set out to do it.

### Music:

Text

Description automatically generated The code is a class called music that has a main function and a play function. The main function is called when the user wants to play a song on YouTube. It takes the user's request, which should contain the word "play" and the name of the song and passes it to the play function. The play function uses the pafy and vlc libraries to search YouTube for the song specified by the user and play it. It first uses the urllib library to encode the user's request and search YouTube with it. It then uses the re (regular expression) library to find the first matching video on YouTube and extract its URL. Next, it uses the pafy library to create a video object for the YouTube video and gets the best audio version of the video. It then creates a vlc media player and plays the audio. The function uses the time library to sleep for the duration of the video and then stop the media player.

Figure - Music Module Code

### Speak and Listen:

Text

Description automatically generatedThis next module is one of if not the most important module in the program. It has the ability to take what the user is saying and convert it to text, hence Speech recognition or speech to text, and it also has the capabilities for text to speech, to allow the user to hear what is being outputted as well as being able to see it. This is very important as it links back to the original idea of allowing anyone to use the program. Having this here allows a user that cannot see to utilize the code and not have to type or attempt to read.

Figure - Speech Input and Output Code

### Words:

Text

Description automatically generatedNext module is the words module, this provides spellings, synonyms and antonyms for the user depending on what they ask for. To begin with, like most others it runs through the determine most similar module in order to determine which function is most likely needed to be used. It has spelling functions that works by separating the word into a list of characters and says them one by one. Next, there is a definition, using the dictionary module, it scrapes dictionary.com to find an appropriate definition for the word. Next there is both synonyms and antoyms that work the same way as the definitions.

Figure - Words Code

### Wikisearch:

Next is the wikisearch Module, this module uses the WikipediaAPI to take a keyterm, if it canot find a correct wikisearch then it suggests a possible option and asks you to try again. By having a wikipedia module it allows the user to ask any questions and it will provide a response.

### Text Description automatically generatedVoice:

Text

Description automatically generatedThis code is always running when the program is on. It works by listening to everything being said and only activates once the activation word ‘oi badger’ is said and then it prints (“I Heard my name”) and then uses the listening function from the speak and listen module to determine what the user is saying and change it into text. If the wake word is not heard, then the program just continues listening.

Figure - WikiSearch Code

Figure - Voice Code

### Test Table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Number** | **Test** | **Test Type** | **Desired Outcome** | **Pass or fail** | **Proof** |
| 1 | Account Creation | Normal | Stores Details to files | Pass |  |
| 2 | Account Creation | Erroneous | Provide Error message | Pass |  |
| 3 | Sign In | Normal | Allow a login | Pass |  |
| 4 | Sign In | Erroneous | Provide Error message | Pass |  |
| 5 | One time password | Normal | Send me an email | Pass |  |
| 6 | One time password | Erroneous | Send an error | Pass |  |
| 7 | Password change | Normal | Change the password | Pass |  |
| 8 | Password change | Erroneous | Return an error | Pass |  |
| 9 | Listen for Wake word | Normal | Return I heard my name | Pass |  |
| 10 | Listen for wake word | Erroneous | Nothing | Pass |  |
| 11 | Take text input | Normal | Respond | Pass |  |
| 12 | Answer Maths questions | Normal | Provide the correct answer | Pass |  |
| 13 | Provide a spelling | Normal | Provide correct spelling | Pass |  |
| 14 | Provide a definition | Normal | Provide correct definition | Fail | It failed this test due to the fact that I believe the web scraper used inside of the module isn’t up to date or correctly working. In order to fix this inside of prototype 2 I am going to either need to find a new module, or I am going to need to create my own web scraper in order to collect the data that the user wants. |
| 15 | Provide a translation | Normal | Provide correct Definition | Fail | This failed the test as I believe I haven’t used the module correctly, there are many modules similar to this, and yet again in prototype 2 I am either going to need to find a new one, or I am going to need to make my own web scraper to use google translate or something similar. |
| 16 | Tell a joke | Normal | Give Funny Joke | Pass |  |
| 17 | Play a song | Normal | Play Song | Pass |  |
| 18 | Respond to Simple Conversation statements | Normal | Respond normally | Pass |  |
| 19 | Encryption of personal data | N/A | Encrypt data such as passwords when storing them to the database | Fail | This is a plan for prototype 2. I will have all of the information of the user encrypted when stored to the database, and then decrypted when the user inputs their login details in order for the program to check if the user has entered the correct login details. |

### Database:

During the process of the program, I created a database that stores all of the questions that have been asked of the virtual assistant, and it stores them alongside the intent that it has classified them with. This means that new data can be added with ease and without having to manually do it. Having a database allows the virtual assistant to use the old data that has been inputted, in order to classify the new data and generate the correct intent for the statement.

A screenshot of a computer

Description automatically generated with low confidence

Figure - Database Example

## Plans for Prototype two:

During prototype two there are several new things that I want to implement.

### Fixing of current errors:

Obviously to begin with I would like to remove the current errors in the tests 14 and 15, like I said in the test table above I will have to revisit the documentation on each of the modules and review if I have used them correctly, or if it is a problem within the module itself.

### New features:

As I have mentioned in test 19 inside of the test table above, I would like to add a function by which I can add it encrypts the user’s email, username and password before storing them inside of a document. This will increase security and then not have the usernames and passwords not stored in plain text.   
Continuing on with the idea of security and passwords, I would like to add minimum requirements for a password, for example having a minimum amount of characters, at least one capital letter and then also a special character.

However, on the flip side I would also need to add a decryption feature by which it would decrypt the data when the user goes to login so that the program can allow the user to login.   
Another new feature would be having personal profiles, depending on who is logged in, it will store the questions that they ask in order to create a profile more personal to them. By doing this it will allow the virtual assistant to create a better idea of who they are dealing with and create a better idea of the person.

This will be a challenging task, but by doing it, it will allow the virtual assistant to really engage with the user, by creating preferences, and getting to know what they are like and what they normally need the virtual assistant’s help with.

## Prototype 2:

### Changes to original algorithms:

Figure - Encryption Pseudocode

Text

Description automatically generatedGraphical user interface, text

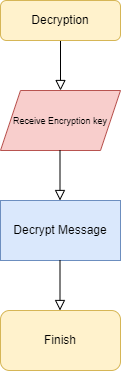
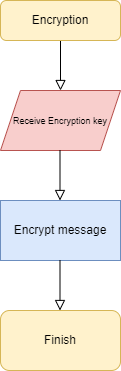
Description automatically generated  
  
  
  
  
  
  
  
  
  
  
  
  
  
After programming prototype two, I have now added a few changes to the original program.

Figure - Decryption Pseudocode

Figure - Decryption Flowchart

Figure - Encryption Flowchart

To begin with, I decided not to change the look of my UI, as I decided that I am actually very happy with how my current UI looks. The program now stores all emails, usernames, passwords and UUIDs inside of a file after they are encrypted and then it decrypts them outside of the file using the unique key. They are only decrypted if they are needed to be checked, e.g., to allow a user to login or if a user is trying to use the OTP feature.

Text

Description automatically generatedThe encryption code above takes the message and encrypts it using Fernet and cryptography.

Figure - Encryption Code

Text

Description automatically generatedText

Description automatically generatedThe decryption code is very similar and just works the other way. However, I ran into several issues when trying to write the code originally. So, when the string is encrypted, it becomes a ‘bytes’ data type, but when being stored to the file, it must be stored in a string. So, I convert it to a string when storing it to a file, but once it is taken from the file in order to try to decrypt it, it first has to be converted back into a ‘bytes’ data type, and then decrypted back into a string. This is done with the code below.

Figure - Integration of Decryption into code

Figure - Decryption Code

The next addition that was made to prototype 2, was the addition of personal profiles.  
To begin with for a personal profile to work, each user needs a Universally Unique Identifier, or for short UUID. So, upon the creation of an account, it also generates a random string as a UUID so each account has a unique identifier.

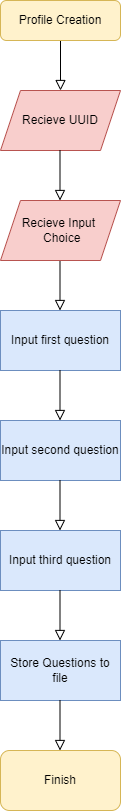
It is a simple statement making use of the random.choice function and also making use of the string class as a dataset.

Figure - UUID Generation

Text

Description automatically generatedNext part of Prototype two is the profile creation stage. In this part of the program if the user requests to create a profile the program first checks if they have a personal profile by checking if there is a file with their uuid. Next it asks them if they would like to create a profile, asks a few questions and then checks with the user if they are all correct. Finally, if everything is correct, it stores it to a file with the user’s uuid as the file name.

Figure - Questions

Text

Description automatically generatedText

Description automatically generatedThis code allows the user to have their name, age and location stored to the file. In the future I plan to add uses for this data, for example language translations based on where they are from, and further personalisation with the user’s name and other possible bits of information.

Figure - Profile Creation flowchart

Figure - Saving to file for profile creation

Figure - Main Profile Creation

### Test Table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Number** | **Test** | **Test Type** | **Desired Outcome** | **Pass or fail** | **Proof** |
| 1 | Account Creation | Normal | Stores Details to files | Pass | Graphical user interface, text  Description automatically generatedGraphical user interface, application  Description automatically generated |
| 2 | Account Creation | Erroneous | Provide Error message | Pass | Table  Description automatically generated with medium confidence  Graphical user interface, text, application, chat or text message  Description automatically generated |
| 3 | Sign In | Normal | Allow a login | Pass | Graphical user interface, application  Description automatically generatedA picture containing table  Description automatically generated |
| 4 | Sign In | Erroneous | Provide Error message | Pass | Graphical user interface, text, application  Description automatically generatedA picture containing table  Description automatically generated |
| 5 | One time password | Normal | Send me an email | Pass | Graphical user interface, text, application, email  Description automatically generatedGraphical user interface, text, application  Description automatically generated |
| 6 | One time password | Erroneous | Send an error | Pass | Graphical user interface, text, application  Description automatically generatedText  Description automatically generated |
| 7 | Password change | Normal | Change the password | Pass | A picture containing graphical user interface  Description automatically generated |
| 8 | Password change | Erroneous | Return an error | Pass | Graphical user interface, text, application  Description automatically generatedGraphical user interface  Description automatically generated with medium confidence |
| 9 | Listen for Wake word | Normal | Return I heard my name | Pass | Text  Description automatically generated |
| 10 | Listen for wake word | Erroneous | Nothing | Pass |  |
| 11 | Take text input | Normal | Respond | Pass | Graphical user interface, text, application, chat or text message  Description automatically generatedGraphical user interface, application  Description automatically generated |
| 12 | Answer Maths questions | Normal | Provide the correct answer | Pass | Graphical user interface, text, application  Description automatically generated |
| 13 | Provide a spelling | Normal | Provide correct spelling | Pass | A picture containing graphical user interface  Description automatically generated |
| 14 | Provide a definition | Normal | Provide correct definition | Fail | Text  Description automatically generated with medium confidenceIt failed this test due to the fact that I believe the web scraper used inside of the module isn’t up to date or correctly working. In order to fix this inside of prototype 3 I am going to either need to find a new module, or I am going to need to create my own web scraper in order to collect the data that the user wants. |
| 15 | Provide a translation | Normal | Provide correct Definition | Fail | Text  Description automatically generatedThis failed the test as I believe I haven’t used the module correctly, there are many modules similar to this, and yet again in prototype 3 I am either going to need to find a new one, or I am going to need to make my own web scraper to use google translate or something similar. |
| 16 | Tell a joke | Normal | Give Funny Joke | Pass | Graphical user interface, text  Description automatically generated |
| 17 | Play a song | Normal | Play Song | Pass | Graphical user interface, text, application, Word  Description automatically generated |
| 18 | Respond to Simple Conversation statements | Normal | Respond normally | Pass | Graphical user interface, text, application  Description automatically generated |
| 19 | Encryption of personal data | N/A | Encrypt data such as passwords when storing them to the db. | Pass |  |
| 20 | Profile Creation | Normal | Create a personal profile for the user, storing name, age and country | pass |  |
| 21 | Profile Creation | Erroneous | Tell the user that the inputted values are not correct. | fail |  |

## Plans for Prototype three:

### Fixing of current errors:

In prototype 3, I plan on fixing both the tests 14 and 15 to ensure that all of the modules inside of the virtual assistant are working. I would also like to add validation to test 21 as currently there is no validation.

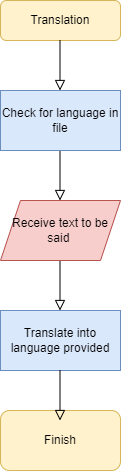
### New features:

A new feature that I would like to add is the ability to utilise the data collected from the profile creation module in order to change the program for the ease of access of the user. For example, in translations, it will translate to the language that a user could set as a default. It could use the user’s name in order to personalise the program more. I am also planning on adding a timer function as that will be an easy-to-use feature.

## Prototype three:

### New Features:

### Translation:

In Prototype three there has been some rather large changes and additions. This code is a function that takes two arguments: text and uuid. The function first tries to open a file named UserProfiles/{uuid}.csv, where {uuid} is the value of the uuid argument passed to the function. If the file is successfully opened, the function reads the first three comma-separated values from the file, which are assumed to be the user's name, age, and language. The language is then converted to a lowercase string and stored in the target\_language variable. Next, the function checks the value of target\_language and converts it to a two-letter language code if necessary. For example, if target\_language is "french", it is converted to "fr". Once the target language is determined, the function uses a Translator object to translate the text argument passed to the function into the target language. The translated text is then returned. Note that if the UserProfiles/{uuid}.csv file cannot be opened, the target\_language variable is set to 'en' (English) by default.Text

Description automatically generated

Figure 59 - Translation Flow Chart

Figure - Speak translation code

### Text Description automatically generatedText Description automatically generatedWeb Search:

Figure - Profile example

Figure - Suggested results and related questions code

Figure - Main and normal search code

This code is a class called Searches, which defines several functions that perform different types of searches on Google. The main function, main(), takes four arguments: text, intent, uuid, and choice. The function first defines a dictionary, samples, that maps various sample phrases to the corresponding function that should be called to handle that phrase. The main function then calls the determine\_most\_similar\_phrase() function from the assistant\_functions.determine\_most\_similar module to find the sample phrase that is most similar to the text argument passed to main(). The function associated with the most similar sample phrase is then called, passing the text and uuid arguments to it.

### Timer:

Text

Description automatically generatedText

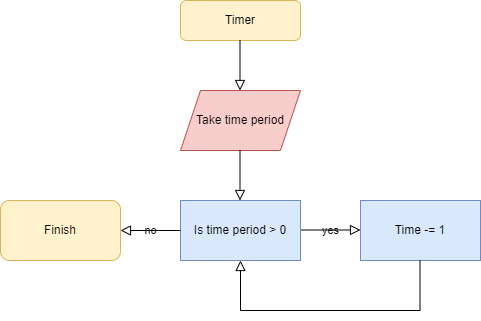
Description automatically generatedThis next bit of code defines a class called Timer that contains several function for creating and starting a timer. The main method takes four arguments: text, intent, uuid, and choice. First, the main method defines a samples dictionary that maps several different phrases to the timerset or timercreate function. Then, it calls the determine\_most\_similar\_phrase function to determine which of the phrases in the samples dictionary is most similar to the text argument passed to the main method. Next, the main method retrieves the func value associated with the most similar phrase and stores it in a variable called func. Then, it calls the func method, passing in the text and uuid arguments. This allows the main method to dynamically call the correct method (either timerset or timercreate) based on the most similar phrase. The timerset method takes in two arguments: text and uuid. It first converts the text argument to lowercase and splits it into a list of words. Then, it iterates over the list of words and looks for the first word that is a digit (which is assumed to be the time period specified in the input) and the first word that is a time unit (such as "hours", "minutes", etc.). These values are stored in the timeperiod and periodoftime variables, respectively. Next, the timerset method calls the timechecker method, passing in the periodoftime and timeperiod variables as well as the uuid argument. The timechecker method converts the time period specified in the input to seconds and then calls the timerscreen method, passing in the time in seconds and the uuid argument. The timercreate method takes in two arguments: text and uuid. It creates a graphical user interface (GUI) using the PySimpleGUI library that allows the user to input the time for the timer in hours, minutes, and seconds. When the user clicks the "Submit" button, the method retrieves the user-specified time and converts it to seconds. Then, it closes the GUI and calls the timerscreen method, passing in the time in seconds and the uuid argument. The timerscreen method takes in two arguments: time and uuid. It creates another GUI that displays the time remaining on the timer and counts down the time in seconds. When the time reaches zero, the GUI closes, and the method returns.

Figure - Timer definition code

Figure - Timer creation and UI

### Shopping List:

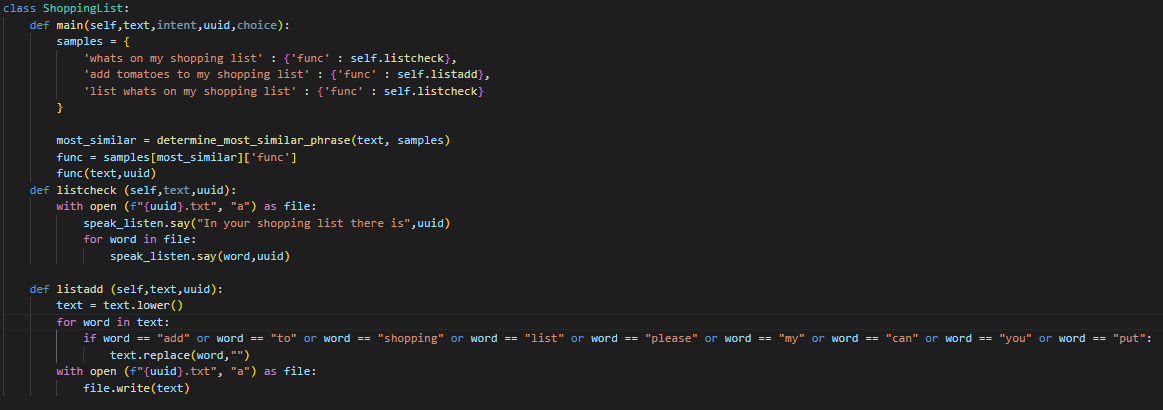
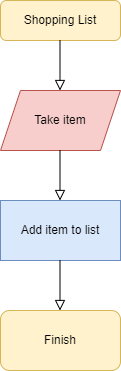
This code defines a class called ShoppingList that contains two function: main and listcheck. The main method takes in four arguments: text, intent, uuid, and choice. First, the main method defines a samples dictionary that maps several different phrases to the listcheck or listadd function. Then, it calls the determine\_most\_similar\_phrase function to determine which of the phrases in the samples dictionary is most similar to the text argument passed to the main method. Next, the main method retrieves the func value associated with the most similar phrase and stores it in a variable called func. Then, it calls the func method, passing in the text and uuid arguments. This allows the main method to dynamically call the correct method (either listcheck or listadd) based on the most similar phrase. The listcheck method takes in two arguments: text and uuid. It opens the assistant\_functions/shoppinglists/{uuid}.txt file in read mode and reads each line of the file. For each line, it prints the line and calls the say method of the speak\_listen object, passing in the line and the uuid argument. The listadd method also takes in two arguments: text and uuid. It first converts the text argument to lowercase and splits it into a list of words. Then, it iterates over the list of words and adds each word to a new list called new if it is not one of the specified words (such as "add", "to", etc.). After the loop, the method opens the assistant\_functions/shoppinglists/{uuid}.txt file in append mode and writes the new list to the file, with each word separated by a space. Finally, the code creates an instance of the ShoppingList class and assigns it to the shoppinglist variable.

Figure - Shopping List code

### Test Table:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Number** | **Test** | **Test Type** | **Desired Outcome** | **Validation** | **Pass or fail** | **Proof** |
| 1 | Account Creation | Normal | Stores Details to files | Make sure passwords match and email and username haven’t been used before | Pass | Graphical user interface, text  Description automatically generatedGraphical user interface, application  Description automatically generated |
| 2 | Account Creation | Erroneous | Provide Error message | Make sure passwords match and email and username haven’t been used before | Pass | Table  Description automatically generated with medium confidence  Graphical user interface, text, application, chat or text message  Description automatically generated |
| 3 | Sign In | Normal | Allow a login | Makes sure username and password are located in the login credentials file | Pass | Graphical user interface, application  Description automatically generatedA picture containing table  Description automatically generated |
| 4 | Sign In | Erroneous | Provide Error message | Makes sure username and password are located in the login credentials file | Pass | Graphical user interface, text, application  Description automatically generatedA picture containing table  Description automatically generated |
| 5 | One time password | Normal | Send me an email | Makes sure email is in the database | Pass | Graphical user interface, text, application, email  Description automatically generatedGraphical user interface, text, application  Description automatically generated |
| 6 | One time password | Erroneous | Send an error | Makes sure email is in the database | Pass | Graphical user interface, text, application  Description automatically generatedText  Description automatically generated |
| 7 | Password change | Normal | Change the password | Two New passwords match and otp is what was sent | Pass | A picture containing graphical user interface  Description automatically generated |
| 8 | Password change | Erroneous | Return an error | Two New passwords match and otp is what was sent | Pass | Graphical user interface, text, application  Description automatically generatedGraphical user interface  Description automatically generated with medium confidence |
| 9 | Listen for Wake word | Normal | Return I heard my name | Don’t activate unless it’s said | Pass | Text  Description automatically generated |
| 10 | Listen for wake word | Erroneous | Nothing | Don’t activate unless its said | Pass |  |
| 11 | Take text input | Normal | Respond |  | Pass | Graphical user interface, text, application, chat or text message  Description automatically generatedGraphical user interface, application  Description automatically generated |
| 12 | Answer Maths questions | Normal | Provide the correct answer |  | Pass | Graphical user interface, text, application  Description automatically generated |
| 13 | Provide a spelling | Normal | Provide correct spelling |  | Pass | A picture containing graphical user interface  Description automatically generated |
| 14 | Provide a definition | Normal | Provide correct definition |  | Pass |  |
| 15 | Provide a translation | Normal | Provide correct Definition |  | Pass |  |
| 16 | Tell a joke | Normal | Give Funny Joke |  | Pass | Graphical user interface, text  Description automatically generated |
| 17 | Play a song | Normal | Play Song |  | Pass | Graphical user interface, text, application, Word  Description automatically generated |
| 18 | Respond to Simple Conversation statements | Normal | Respond normally |  | Pass | Graphical user interface, text, application  Description automatically generated |
| 19 | Encryption of personal data | N/A | Encrypt data such as passwords when storing them to the db. |  | Pass |  |
| 20 | Profile Creation | Normal | Create a personal profile for the user, storing name, age and country |  | pass |  |
| 21 | Profile Creation | Erroneous | Tell the user that the inputted values are not correct. |  | fail |  |
| 22 | Shopping List | Normal | Adds something to the shopping list |  | Pass |  |
| 23 | Shopping List | Normal | Lists everything in the |  | pass |  |
| 24 | Searches | Normal | Ask for football score |  | pass |  |
| 25 | Translate to different language? | Normal | Be in French Mode and print everything in French |  | Pass |  |
| 26 | Timer | Normal | Ask for a 30 second timer |  | Pass |  |
| 27 | Timer | Normal | Create a timer |  | pass |  |
| 28 | Date and Time | Normal | Say the day |  | Pass |  |
| 29 | Date and Time | Normal | Say the date |  | Pass |  |
| 30 | Date and Time | Normal | Say the time |  | pass |  |
| 31 | Quiz | Normal | Answer correctly |  | pass |  |
| 32 | Quiz | Normal | Answer incorrectly |  | pass |  |

# Table of Figures:

[Figure 1 - Amazon Alexa 5](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783583)

[Figure 2 - Siri 5](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783584)

[Figure 3 - BMW IPA 5](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783585)

[Figure 4 - Decomposition Diagram 9](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783586)

[Figure 5 - Home Page 10](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783587)

[Figure 6 - Login UI Design 11](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783588)

[Figure 7 - Forgotten Password UI Design 11](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783589)

[Figure 8 - Create an account UI Design 12](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783590)

[Figure 9 - Input Selection 12](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783591)

[Figure 10 - UI Design 1 13](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783592)

[Figure 11 - Speech Recognition Module Flowchart 14](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783593)

[Figure 12 - Activation Flowchart 14](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783594)

[Figure 13 - Speech input Flowchart 15](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783595)

[Figure 14 - Language Detection Module Flowchart 16](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783596)

[Figure 15 - Speech Translation Flowchart 16](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783597)

[Figure 16 Translation Module Flowchart 16](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783598)

[Figure 17 - Processing Flowchart 17](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783599)

[Figure 18 Question Classification Flowchart 18](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783600)

[Figure 19 - Communications Flowchart 19](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783601)

[Figure 20 - Random Greeting Module 20](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783602)

[Figure 21 - Random Leaving Module 20](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783603)

[Figure 22 - Random Joke Module 20](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783604)

[Figure 23 - Random Compliment Module 20](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783605)

[Figure 24 - Random Insult Module 20](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783606)

[Figure 25 - Mathematics Flowchar 21](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783607)

[Figure 26 - Geographical Flowchart 22](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783608)

[Figure 27 Words Module Flowchart 23](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783609)

[Figure 28 - Output Flowchart 25](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783610)

[Figure 29 – Sign up or login screenshot 31](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783611)

[Figure 30 - Login Code 31](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783612)

[Figure 31 - Login Screenshot 31](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783613)

[Figure 32 - Sign up Screenshot 32](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783614)

[Figure 33 - Create an account front end code 32](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783615)

[Figure 34 - Main python code 33](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783616)

[Figure 35 - Intent Classification Python Code 33](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783617)

[Figure 36 - Determine Most Similar Code 33](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783618)

[Figure 37 - Common response code 34](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783619)

[Figure 38 - Jokes Module Code 34](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783620)

[Figure 39 - Maths module second 34](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783621)

[Figure 40 - Maths Module Start 34](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783622)

[Figure 41 - Music Module Code 35](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783623)

[Figure 42 - Speech Input and Output Code 35](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783624)

[Figure 43 - Words Code 36](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783625)

[Figure 44 - WikiSearch Code 36](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783626)

[Figure 45 - Voice Code 36](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783627)

[Figure 46 - Database Example 40](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783628)

[Figure 47 - Encryption Pseudocode 41](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783629)

[Figure 48 - Decryption Pseudocode 41](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783630)

[Figure 49 - Decryption Flowchart 41](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783631)

[Figure 50 - Encryption Flowchart 41](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783632)

[Figure 51 - Encryption Code 41](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783633)

[Figure 52 - Integration of Decryption into code 42](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783634)

[Figure 53 - Decryption Code 42](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783635)

[Figure 54 - UUID Generation 42](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783636)

[Figure 55 - Questions 42](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783637)

[Figure 56 - Profile Creation flowchart 43](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783638)

[Figure 57 - Saving to file for profile creation 43](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783639)

[Figure 58 - Main Profile Creation 43](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783640)

[Figure 59 - Speak translation code 48](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783641)

[Figure 60 - Profile example 48](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783642)

[Figure 61 - Suggested results and related questions code 48](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783643)

[Figure 62 - Main and normal search code 48](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783644)

[Figure 63 - Timer creation and UI 49](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783645)

[Figure 64 - Timer definition code 49](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783646)

[Figure 65 - Shopping List code 49](file:///C:/Users/Owner/Downloads/Joe%20Bostock%20Component%203%20(1).docx#_Toc120783647)

# Bibliography:

* How to use python WikipediaAPI, Stackabuse. Available at: <https://stackabuse.com/getting-started-with-pythons-wikipedia-api/> (Accessed: 4 May 2022).
* Geeksforgeeks, how to use the google speech recognition API for python, geeksforgeeks. Available at: <https://www.geeksforgeeks.org/speech-recognition-in-python-using-google-speech-api/> (Accessed: 4 May 2022).
* Mural, Decomposition Diagram Creator, Mural. Available at: <https://app.mural.co/t/cleeve1757/m/cleeve1757/1654677218899/435bb4e55bb1309f80a6614f8c3503400040eb35?fromVisitorModal=true&sender=u397f6ae116bda0e34ae99881> (Accessed: 5 May 2022, 8 June 2022).
* Amazon ‘Amazon Alexa Image’. google. Available at: <https://www.google.com/search?q=alexa+png&source=lnms&tbm=isch&sa=X&ved=2ahUKEwi1vrOBrNz3AhWHbsAKHVVMApsQ_AUoAXoECAEQAw&biw=1920&bih=969&dpr=1#imgrc=V4c8a-zT9Mxy8M> (Accessed: 13 May 2022)..
* Apple “Siri Image.” Google. Available at: <https://www.google.com/search?q=siri+png&tbm=isch&ved=2ahUKEwiR1YOCrNz3AhVHZxoKHUlRA2YQ2-cCegQIABAA&oq=siri+png&gs_lcp=CgNpbWcQAzIECAAQQzIFCAAQgAQyBQgAEIAEMgUIABCABDIGCAAQBRAeMgYIABAFEB4yBggAEAUQHjIGCAAQBRAeMgYIABAFEB4yBggAEAgQHjoGCAAQBxAeOgcIABCxAxBDOggIABCxAxCDAToLCAAQgAQQsQMQgwE6BAgAEAM6CAgAEIAEELEDULAEWOwNYLwPaABwAHgAgAFdiAHSA5IBATmYAQCgAQGqAQtnd3Mtd2l6LWltZ8ABAQ&sclient=img&ei=IT1-YtFAx85pyaKNsAY&bih=969&biw=1920#imgrc=xygsrF_toNoVsM> (Accessed: 13 May 2022).
* BMW “BMW IPA.” Google. Available at: <https://www.google.com/search?q=bmw+intelligent+personal+assistant&rlz=1C1GCEB_enGB1005GB1005&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjx8NeJrtz3AhWJhFwKHaMECDgQ_AUoAnoECAEQBA&cshid=1652441083023227&biw=958&bih=959&dpr=1#imgrc=vKKuINn77ScnXM> (Accessed: 13 May 2022).
* BMW BMW IPA, BMW. Available at: <https://discover.bmw.co.uk/help/technology/what-is-ipa> (Accessed: 5 May 2022).
* Ratnesh Amazon Alexa – Pros, Cons and Competitors, Seeromega. Available at: <https://seeromega.com/amazon-alexa-pros-cons-competitors/> (Accessed: 5 May 2022).
* Soffar, H. Apple Siri features, use, advantages, disadvantages & using of Siri For Learning, Online Sciences. Available at: <https://www.online-sciences.com/technology/apple-siri-features-use-advantages-disadvantages-using-of-siri-for-learning/> (Accessed: 5 May 2022).
* Microsoft. (2016). Visual Studio Code. <https://code.visualstudio.com/>
* Bhat, N. M. (2020). *PythonTextToSpeechx3 Module*. <https://pypi.org/project/pyttsx3/>
* Carriere, D. (2018). *GeoCoder Python Module*. <https://pypi.org/project/geocoder>
* Goldsmith, J. (2014). *Wikipedia Python Module*. <https://pypi.org/project/wikipedia>
* Han, S. (2020). *GoogleTrans Python Module*. <https://pypi.org/project/googletrans/>
* Kenarsa. (2022). *PVPorcupine Python Module*. <https://pypi.org/project/pvporcupine/>
* Marks, T. (2021). *Playsound Python Module*. <https://pypi.org/project/playsound/>
* Mueller, A. (2022). *Scikit-Learn Python Module*. <https://pypi.org/project/scikit-learn/>
* Pham, H. (2018). *PyAudio*. <https://pypi.org/project/PyAudio/>
* The Pandas Development Team. (2022). *Pandas Python Module*. <https://pypi.org/project/pandas/>
* Zhang, A. (2018). *Speech Recognition Python Module*. <https://pypi.org/project/SpeechRecognition/>
* Zope Foundation and Contributors. (2022). *DateTime Python Module*. <https://pypi.org/project/DateTime/>
* Colorbux *Most eye-catching colour study*, *Colorbux*. Available at: <https://www.colorbux.com/articles/what-color-catches-human-eye-most> (Accessed: 24 June 2022).
* DrawIO *DrawIO UI design*, *DrawIO*. Available at: <https://app.diagrams.net/> (Accessed: 24 June 2022).
* Smartdraw *Flow Chart diagram creator*, *SmartDraw*. Available at: <https://cloud.smartdraw.com/editor.aspx?credID=-39109915&depoId=35340836&flags=128#depoId=35340836&credID=-39109915> (Accessed: 22 June 2022).
* GeeksForGeeks (2020) *Encryption and Decryption*, *GeeksForGeeks*. Available at: <https://www.geeksforgeeks.org/how-to-encrypt-and-decrypt-strings-in-python/> (Accessed: 11 July 2022).
* PicoVoice (2020) *PicoVoice*, *Picovoice*. Available at: <https://picovoice.ai/> (Accessed: 11 July 2022).
* PyData (2020) *Pandas*, *PyData*. Available at: <https://pandas.pydata.org/> (Accessed: 11 July 2022).
* Pysimplegui (2022) *PySimpleGUI*, *pysimplegui*. Available at: <https://www.pysimplegui.org/en/latest/> (Accessed: 11 July 2022).
* Python (2020a) *MultiProcessing*, *Python3*. Available at: <https://docs.python.org/3/library/multiprocessing.html> (Accessed: 11 July 2022).
* Python (2020b) *Tkinter*, *Python3*. Available at: <https://docs.python.org/3/library/tkinter.html> (Accessed: 11 July 2022).
* RealPython (2019) *Async Programming*, *realpython*. Available at: <https://realpython.com/async-io-python/#:~:text=Asynchronous%20routines%20are%20able%20to,look%20and%20feel%20of%20concurrency>. (Accessed: 11 July 2022).
* SciKit (2020) *Machine Learning*, *Scikit*. Available at: <https://scikit-learn.org/stable/auto_examples/index.html#classification> (Accessed: 11 July 2022).