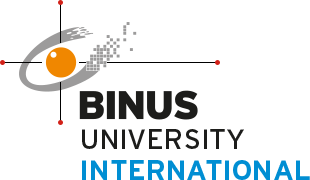
Algorithm and Programming Final Report

****

*Expense Tracker App Using Voice Assistant*

**Lecturer:**

Jude Joseph Lamug Martinez MCS

**Arranged by:**

Joshua Efraim Rawatan

**Algorithm and Programming**

**Computer Science Faculty**

**Binus International University 2022/2022**

**Table of Contents**

**1. Introduction —------------------------------------------------------------------------------------ 1**

**2. Use-case Diagram —------------------------------------------------------------------------------2**

**3. Activity Diagram —-------------------------------------------------------------------------------3**

**4. Class Diagram—--------------------------------------------------------------- —----------------4**

**5. Modules —-----------------------------------------------------------------------------------------5**

**6. Essential algorithms —-------------------------------------------------------------------------- 6**

**7. Screenshots of your application —------------------------------------------------------------11**

**8. Lessons learned/Reflection —------------------------------------------------------------------12**

**Chapter 1. Introduction**

**1.1 Genesis**

I have always loved superheroes, and one hero that stood out to me was Iron Man from the Marvel Cinematic Universe. Iron Man is a character that many people, especially guys like me, look up to. One of the most iconic features that Iron Man has is Jarvis. Jarvis is an artificial intelligence that helps him with his work and daily activities. By using Jarvis, Iron Man is able to minimise his workload purely by talking to Jarvis. The inspiration from Jarvis has led me to create this project.

**1.2 Purpose**

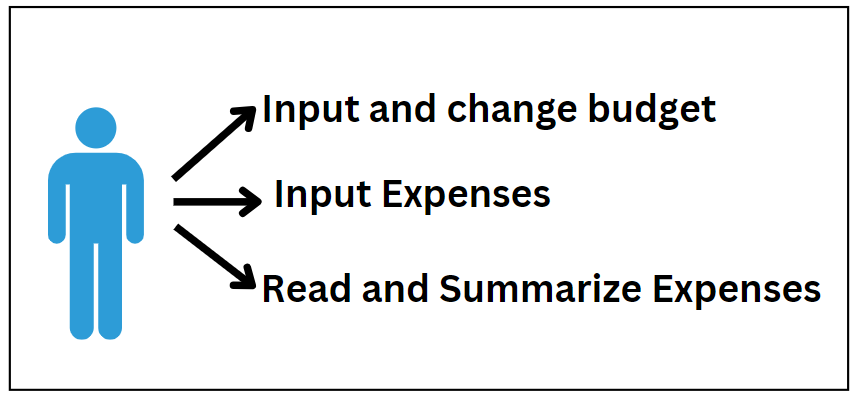
The purpose of this report is to explain the Algorithm and Programming Project of creating an expense tracker program that uses voice assistant. The goal of this project is to have a fully working expense tracker program with minimal effort or movement when using this. By implementing speech recognition and voice assistance in the project will help to achieve this goal.

**1.3 How it Works**

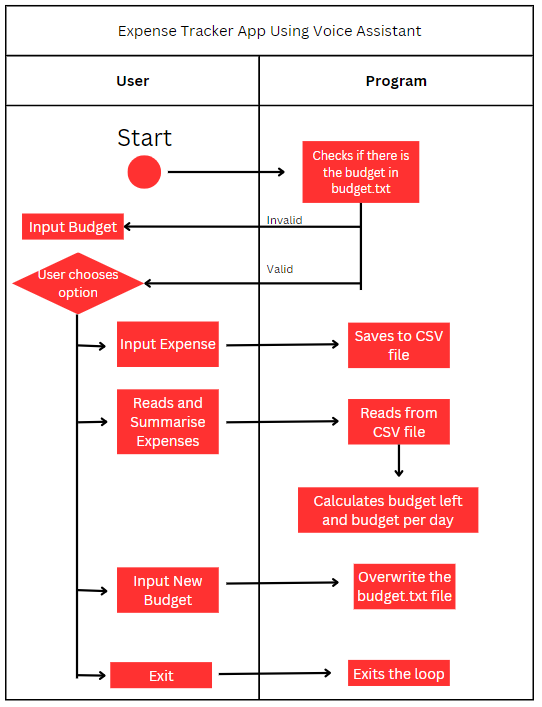
Python imports ‘speech\_recognition’ to understand and recognize what the user says and it also imports gTTS, winsound, and AudioSegment so that the program is able to respond to the voice command and say something back.

When the program is running, the user will initially be asked to set a budget which will then be saved into a txt file. The user is then able to choose 4 different options. The first option is to input expenses, when the user says “input my expense”, the program will start to get details of the expense which will then be recorded into a csv file. The second option is to read and summarise the expenses that were inputted, when the user says “read and summarise my expenses”, the program will start to read and say the expenses from the csv file and categorise them according to their category. It will then say the total money that has been spent, the remaining budget, the remaining days in the current month, and the budget per day. The third option would be when a user says “change my budget” where the budget from the txt file can be changed. The final option is exiting the program when a user says “exit”.

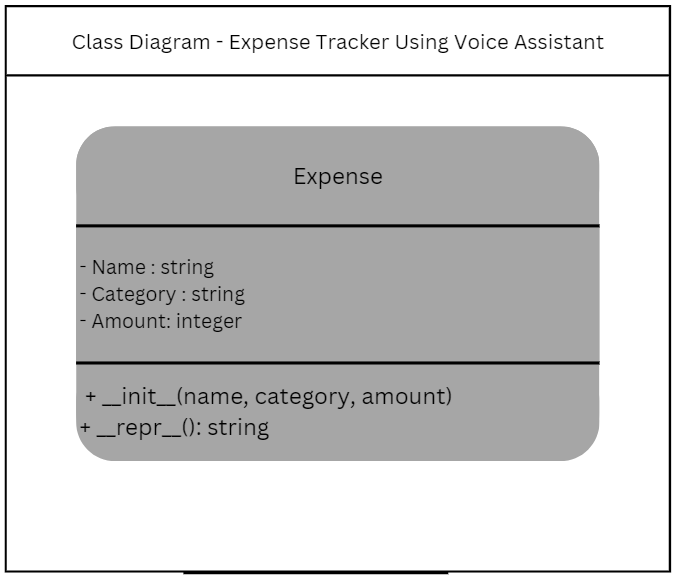
**Chapter 2. Use-Case Diagram**

****

**Chapter 3. Activity Diagram**



**Chapter 4. Class Diagram**



In this project there is only one class that is used which is the Expense class.

The Expense class contains the name, category, and amount for the expenses and it also helps with displaying the string into a readable format.

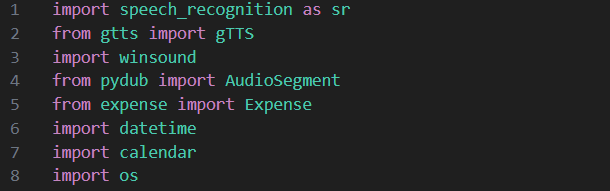
It has three attributes:

* Name (string) - The name of the expense
* Category (string) - The category of the expense
* Amount (Integer) - The amount of the expense

It has two methods:

* Init - This is the constructor method which initialises the attributes of the expenses.
* Repr - This method helps to create a readable format of the expenses.

**Chapter 5. Modules**



* ‘speech\_recognition’ is a module that is used to recognise the voice commands of what the person is saying.
* ‘gtts’ is a module that allows the program to generate voiced responses. In this case, Google's voice is used to sound the responses.
* ‘winsound’ is a module that is responsible for playing sound on Windows which in this case is used to generate the sound.
* ‘pydub’ is used to convert the MP3 sound to a WAV format in this case.
* ‘expense’ is the module that is created for the expenses with the attributes of name, category, and amount, and also has the repr method to display the expenses in a readable format.
* ‘datetime’ is a module that has date and time functions. In this case, it is used to get the current date and also helps with calculating the remaining days of the month.
* ‘calendar’ is a module that has functions related to the calendar. In this case, it is used to get the number of days in the current month.
* ‘os’ is a module that works with the files in the system. In this case it is used to check if the budget.txt file exists.

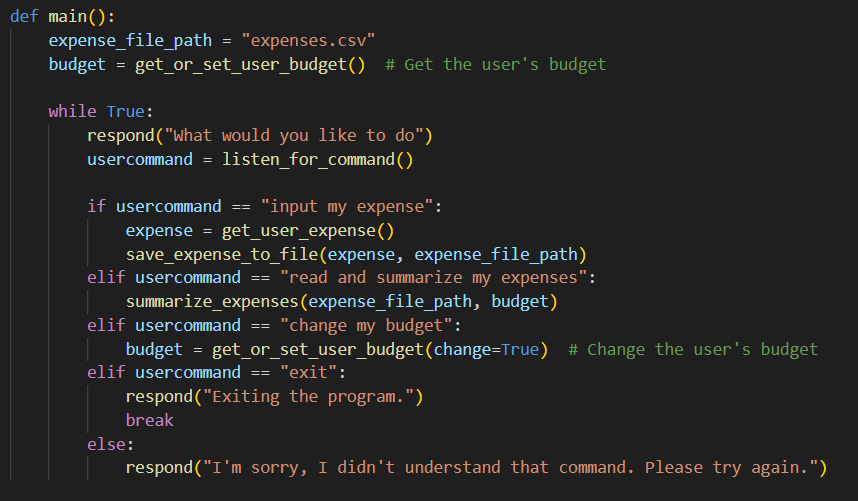
**Chapter 6. Essential Algorithms**

**6.1 Budget Parsing**



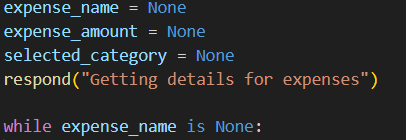
The screenshot of the code above shows how the program parses what the user says to set their budget. It first sets the “budget\_text” as listen\_for\_command() which is a function to listen to the phrase of the user. Then there is the try and except block where if the phrase of the user is not a digit, it will respond by saying “Invalid Input. Please provide a valid number for the budget.”. After that it assigns the variable “words” as budget\_text with lower case and is split into a list, “numeric\_value” and “the current\_number” is set as 0. Then a for loop occurs for each word that the user says. At the start, the for loop will make sure that there are no commas and full stops by replacing them with empty strings. An example is “3 million and 300 thousand” (there might be multiple other examples such as 3 million and 300,000 or 3 million and 300000 depends on how the program will register it), it processes the “3” and updates the “current\_number” as 3, then it processes “million” and adds the “numeric\_value” with multiplying 3 by 1,000,000 and resets the “current\_number” to 0 for later use , “and” is skipped as it won’t be calculated, after that “300” is processed and updates the “current\_number” to 300, and finally the “thousand” is processed which adds the “numeric\_value” by the “current\_number” which is 300 multiplied by 1,000. The final result is 3,300,000 which is then formatted and written in budget.txt.

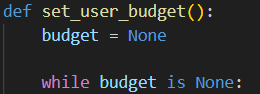
**6.2 User choice loop**

****

The screenshot above shows the while loop for the user to choose their options. The while loop starts by asking what the user would like to do, the listen\_for\_command() function is assigned to “usercommand”. There are 4 different choices which are “input my expense”, “read and summarize my expenses”, “change my budget” ,and “exit”, if the user says anything other than those choices, the program will respond by not understanding the command. Where for every choice which the user commands will do exactly what they wish for and the loop will break when the user says “exit”.

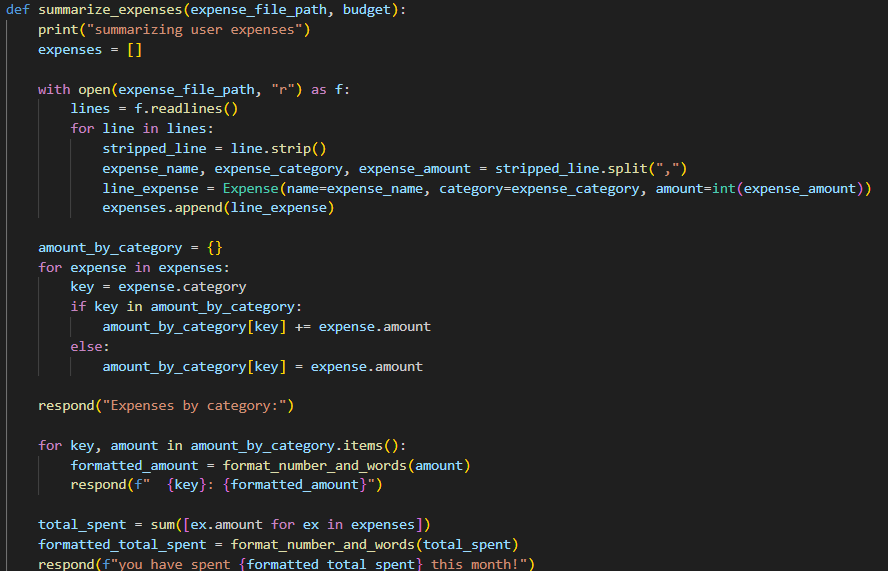
**6.3 Getting Expense Details and Setting Budget**





The screenshot above shows the logic on how to get the expense details and setting the budget. It first assigns the “expense\_name”, “expense\_amount”, “selected\_category”, and “budget” as None. Then it will use a while loop where if the variable is still None it will keep looping until the user inputs something in the variable.

**6.4 Expense by Category and Total Expense**

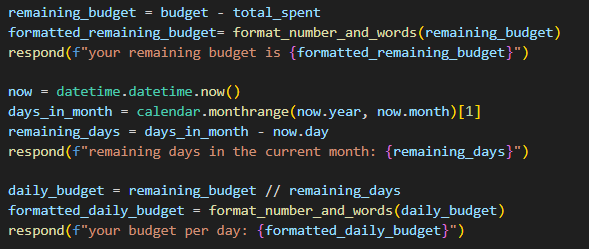
****

The screenshot above shows how the expenses are processed to produce the total expense amount by category and the total expenses. It first creates an empty list called “expenses”, then it opens the csv file and reads each line. For each line, all the white spaces are removed using the strip() function and are split into name, category, and amount based on the comma (“, “). It then creates a new “Expense” object which is then appended into the previously empty list, “expenses”.

Moving on, an empty dictionary is created called “amount\_by\_category”. A for loop is created which iterates through the expenses list. The expense.category which is the expense category from the expenses list is assigned as key and for each key (category) of each expense in expenses that the program finds, will have the amount added by expense.amount in the “amount\_by\_category”. While if the key is not found, a new category will be made.

The total spent is calculated by using the sum function which adds up all the amount of all expenses. It is then formatted and is spoken by the program.

**6.5 Calculating budget for the rest of the month**

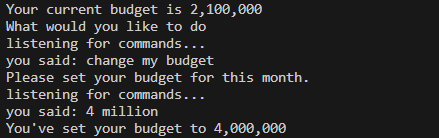


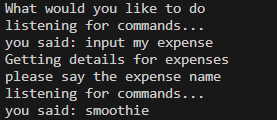
The screenshot above shows how the budget is calculated for the rest of the month. The variable “remaining\_budget” is created by subtracting “budget” by the “total\_spent”, which is then formatted.

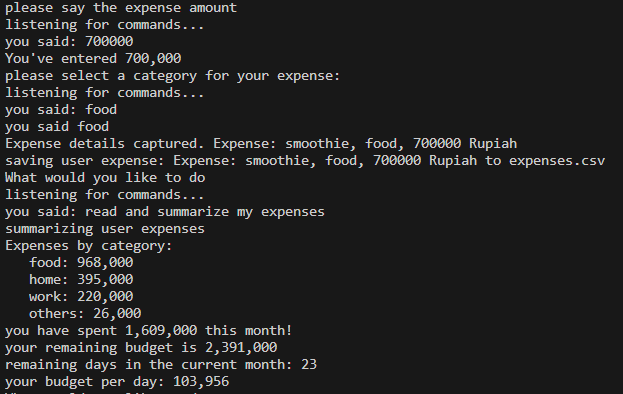
The variable “now” is created and calls the datetime module with the now() function which gets the current date. The second variable, “days\_in\_month”, is created and calls the calendar module with the monthrange function setting it to the current year and month, and calls the second tuple. The “remaining\_days” variable is created by subtracting the “days\_in\_month” variable with the “now” variable which gives the remaining days of the month.

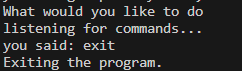
“daily\_budget” is created by dividing “remaining\_budget” with “remaining\_days” which gives the daily amount of budget for the remaining days of the month.

**Chapter 8. Screenshots of the application**









**Chapter 9. Lessons Learned and Reflection**

From this project I have learned a lot, not only about coding, but also about what it takes to create an app. Well obviously the big part of things that I have learned are from the coding aspects, especially because I am trying many different things that have not been taught in class such as :

1. writing code that can listen to my voice and respond to it
2. Using csv files
3. Using various modules (datetime, calendar, winsound, speech\_recognition, gtts, pydub, os)
4. Using more complex while and for loops, etc.

Understanding what it takes to create an app was quite irritating as it meant adding new features which means writing more code. I tried to make the app as simple but as perfect as possible.

There are still things that can be added into the program, such as:

1. The program can automatically convert the currency based on what the user wants the currency to be.
2. The expenses that are saved are tracked with the month during when they were added, as when a user moves on to a different month the expenses from the previous month are still calculated at the current month.

**Chapter 10.**

GithubLink: <https://github.com/JoshuaEfraim/AlgoProg_FinalProject>

References

“Learn Python • #12 Final Project • Build an Expense Tracking App!” *YouTube*, YouTube, 7 Apr. 2023, https://www.youtube.com/watch?v=HTD86h69PtE&t=1927s&ab\_channel=pixegami. Accessed 8 Jan. 2024.

Jake, director. *Make a Voice Assistant with Python*. *YouTube*, YouTube, 13 Aug. 2023, https://www.youtube.com/watch?v=iwVaAAEE4fo&t=108s&ab\_channel=JakeEh. Accessed 8 Jan. 2024.