# THAT GIRL's Personal Finance App

Group 1 (Aisling R, Jessika P, Layla C, Onyeoma A, Sophie W, Yi-Chun W)

## 📍 Introduction

---

Our project is a simple personal finance application, designed to empower individuals by giving our users the tools to effectively manage their savings and investments.

Our aim is to create an accessible and user-friendly platform which provides our users the tools and resources to confidently handle their financial affairs, irrespective of their background or experience level.

## 📍 Background

---

In light of the cost of living crisis, we recognise the growing anxiety and hesitance many people face when it comes to financial management. Our application is a selection of tools used to help track savings, spending and investments. In **Savings**, users are able to determine the amount of interest you can earn on your savings over a specific period of time. With **Budgeting**, you can upload your payslip and a text file of your expenses to visualise the break down your monthly spending. And with **Investments**, track and manage various investments such as stocks, bonds, and mutual funds using the Alpha Vantage API.

## 📍 Specifications and Design

---

**Functional Requirements:**

- Ability for user to check interest on savings over a user specified amount of time
- Ability for user to track and manage various investments
- Ability for user to upload payslip to visualise monthly expenditure

**Non-functional Requirements:**

- Application should be accessible and easy to use
- Application should be reliable and operate consistently
- Application should be built to be modular and maintainable

# 📍 Implementation and Execution

---

**Workload Distribution**
Our team distributed the workload based on the different tasks of the project, namely writing Python code for the Savings, Budgeting and Investments sections, creating an app interface, conducting code tests, writing README file documentation, and creating a PowerPoint for the final presentation. We decided on these roles during our weekly meetings and communicated adjustments as needed through Slack. We adopted an Agile methodology to ensure we were working on our project asynchronously.

**Code Management**
All of our code is stored and managed in a GitHub repository that we created. This repository serves as our central hub for code version control and collaboration. Every member has access to clone, push, pull, and merge changes into the repository. This enabled our team to work on different parts of the project simultaneously without overwriting each other's work. Regular commits helped us maintain a clear history of changes, facilitating easier debugging and ensuring accountability for all changes.

**System Testing**
For the Python code parts, we wrote test cases for each function and module, aiming for high test coverage to ensure that our code behaves as expected under various conditions. The results of these tests are saved within our GitHub repository. We tested the interface by performing actions and verifying that the outcomes align with our expectations. This involved tasks such as navigating between different screens, ensuring that data is accurately displayed, and confirming that the interface is responsive and user-friendly.

**Documentation**
We have created thorough README files that explain the purpose of our project, its architecture, and instructions on how to install and use it. This ensures that anyone who accesses our repository will understand what we're working on and how to use it.

**Final Presentation**
Lastly, we created a presentation to summarise our work. This includes an overview of our project, the processes we used, the challenges we faced, and our key takeaways.

## 📍 Testing and Evaluation

Our testing strategy for the **Investments** code encompasses both unit and integration testing, focused on validating the individual classes of the system and the interaction between them respectively. The **unittest** module is being used for the suite of tests, including the **Investment**, **Bond**, **Stock**, **MutualFund**, and **Portfolio** classes. The code is using mocks to mimic the responses of external services, like the Alpha Vantage API, to ensure controlled and reliable responses for the tests.

Functional and user testing are achieved through the individual test cases for each class. Each test verifies the instantiation and expected behaviour of the objects. For example, the **test_investment** ensures that the object attributes are assigned correctly upon instantiation. User testing is indirectly covered in the tests that validate different types of user inputs, such as valid and invalid quantities. The **test_negative_quantity** and **test_invalid_data_type** tests are examples of this approach.

## 📍 Conclusion

Our initial aim in the project was to create a personal finance application that's not only accessible and user friendly but also empowering for our users. We wanted to provide them with the tools and resources to confidently manage their finances, especially in light of the current cost of living crisis. Our focus was drawn to three critical areas in financial management: Savings, Investments and Budgeting.

Over the course of our project, we navigated several challenges which tested our abilities. Time was a scarce due to the short duration of the project, so learning how to manage our workload and maintain momentum was an ongoing task. We grappled with finding the best techniques to build our solution in a way that wouldn't compromise efficiency or scalability. Debugging and testing required a meticulous approach to ensure our application functioned as expected.

Overall, we were able to deliver a personal finance application that met the essential requirements we initially laid out. We created a tool that allows our users to track savings and manage their investments, and we developed a feature for users to visualise their monthly expenditure by uploading their payslips.

We learned invaluable lessons in collaborative software development, financial application design and project management. Through exploring the nuances in financial application design, we developed skills that will no doubt be useful in future projects. To conclude, despite the challenges we faced, our team demonstrated resilience and adaptability, delivering a successful personal financial application with limited prior knowledge of finance. As we will continue to develop our technical skills, we will look to enhance this application further by leveraging user feedback to make our tool more robust and user-centric.

# THAT GIRL's Personal Finance App

README

## 📍 Savings

---

This code is a simple calculator to determine the amount of interest you can earn on your savings over a specific period of time. It offers two types of accounts: Instant Access Savings and Fixed Bonds. The user is prompted to select the desired account type and provide the necessary information for the calculation.

### How to Use

1. Run the code in a Python environment.
2. The program will display a menu with the following options:
    a. **Savings Account:** Calculates the amount of interest you'll earn on your savings.
    b. **Stocks Portfolio:** Calculates the amount of interest you'll earn on your investment.
    c. **Exit:** Terminates the program.
3. Enter your choice by typing the corresponding number and pressing Enter.

### Savings Account

1. If you select **Savings Account**, the program will display information about the two available account types: **Instant Access Savings** and **Fixed Bonds**.
2. Enter 'a' for **Instant Access Savings** or 'b' for **Fixed Bonds** and press Enter.
3. Based on your selection:
    a. For **Instant Access Savings**:
        i. Enter the amount you want to deposit and press Enter.
        ii. Enter the number of years you want to invest for and press Enter.
        iii. The program will calculate and display the amount of interest you'll earn with each of the available banks (NatWest, Aviva, Lloyds).
    b. For **Fixed Bonds**:
        i. Enter the amount you want to deposit and press Enter.
        ii. Enter the number of years you want to invest for and press Enter.
        iii. The program will calculate and display the amount of interest you'll earn with each of the available banks (NatWest, Aviva, Lloyds) based on the duration of the bond.
4. After displaying the results, the program will show a "Thank you" message.

# 📍 Investment

This code is a simple investment portfolio management system. It allows users to track and manage various investments such as stocks, bonds, and mutual funds. The code interacts with the Alpha Vantage API to fetch current price information for investments.

## Investment Classes

The code defines the following investment classes:

- **Investment**: The base class for all types of investments. It contains common attributes like symbol, name, quantity, and purchase price. The current_price() method retrieves the current price of the investment using the Alpha Vantage API, and the current_value() method calculates the current value of the investment based on the quantity and current price.

- **Stock:** A subclass of Investment representing stocks. It includes an additional type attribute set to stock.

- **Bond:** A subclass of Investment representing bonds. In addition to the attributes inherited from Investment, it includes a type attribute set to bond and a coupon_rate attribute representing the bond's coupon rate.

- **MutualFund:** A subclass of Investment representing mutual funds. It includes a type attribute set to mutual_fund and an expense_ratio attribute representing the mutual fund's expense ratio.

## How to Use

1. Make sure you have Python installed.
2. Run the code in a Python environment.
3. The program will prompt you to enter investment details. The supported investment types are stocks, bonds, and mutual funds.
4. For each investment, enter the investment type, a keyword to search for the investment symbol, the quantity, and the purchase price.
5. If the code successfully retrieves the investment symbol based on the keyword from the Alpha Vantage API, it will display the found investment.
6. For bonds, you need to enter the coupon rate, and for mutual funds, you need to enter the expense ratio.
7. Repeat the process to add more investments. Enter "done" to finish adding investments.
8. After adding investments, the program will display the portfolio data, including detailed information for each investment and the total portfolio value.

# 📍Budgeting

This code reads, organises and visualises your financial information based on your payslips and expenses. The program imports and organises your expense data from a CSV file, extracts and formats your payslip data rom a PDF file and merges both data sets based on the month. It also plots a visual summary of your budget.

## Requirements

You will need to install the following dependencies before running the script:

- 'os'
- 'pandas'
- 'pdfplumber'
- 're'
- 'unicodedata'
- 'collections'
- 'matplotlib'

## Functions

- **Organise_Expenses**: This function reads in your expense CSV file
- **get_payslip:** This function extracts and formats your payslip information from a PDF file.
- **get_required_info:** This function applies regular expressions to match specific patterns in your payslip information to extract necessary information.
- **get_payslip_dataframe:** This function organises payslip data into a pandas data frame and prepares it for further processing.
- **Merge_Pay_Expense:** This function merges the payslip data frame and the expense data frame based on the common key of 'Month'.
- **plot_summary:** This function uses matplotlib to plot a summary of your budget information based on the combined data frame

## How to Use

1. Run the program - you will be prompted to enter your expense CSV file.
2. You will need to input your payslip file in PDF format.
3. After inputting both files, the program with process your data and output a summary bar chart, which will be stored as **Budget Summary Chart.png**

# 📍Personal Finance App

---

This Python/Flask application is designed to assist our users with navigating our application.

## Requirements

You will need to have Python 3 and Flask installed on your system.

## How to Use

Run the application by executing **app.py** in your Python environment. The application will run on **localhost:5000** by default.

## Routes

- **'/':** Home page
- **'/savings':** This page allows the user to calculate the final amount from savings based on the entered initial amount, number of years and interest rate.
- **'/savings_result':** Displays the result of the savings calculation.
- **'/about':** Displays information about the application.
- **'/budgeting':** User can view a sample payslip and expenses file.
- **'/get_summary':** This route is used to get a summary of the user's budget.
- **'/investment_search':** User can search for investments using a symbol and quantity. The route will return the total value of the investment.
- **'/investment_result':** Displays the result of the investment search.