CFG Group 4 Project Report

Hye Ji Lee

Lindelani Moyo

Nikita Simarata

Naimah Riaz

Sum Experts

Contents

1. Introduction

This report documents the group project undertaken as part of the Code First Girls Software Specialisation degree.

1.1 Aims and objectives

With the cost of living crisis, people are trying to make more financially conscious choices. Our system offers financial assistance for users depending on their needs addressing the problem of the current financial crisis. It also allows users to book an appointment with a financial advisor and also provides financial advice.
Users can make an appointment at any time and can view the available dates. The system provides a quicker and more efficient way of booking rather than the more traditional way of doing this via the telephone, where more often than not users may have to wait a long time to be connected to a person. There will be no need to have a call centre therefore reduces cost, also user issues are dealt with quickly and problems solved. The system will be easy to use and efficient.

To achieve our aim we will create an app that:

1. Enables the user to book an appointment with an advisor
2. Users can view/search advisors and their specialism
3. View cost of living news
4. Users can estimate their mortgage using the mortgage calculator.

1.2 Outline of the report

1. Introduction – Introduces the project and outlines the aims and objects of the application we are creating.

2. Background – A brief overview of the cost of living crisis and how our application can help.

3. Specifications and Design – Details the application's requirements, considerations, architecture and program flow.

4. Implementation and Execution – Explains how we have collaborated as a team and the steps we have taken to produce the application.

5. Testing and Evaluation – Details how we have ensured the application is of the highest possible quality.

6. Conclusion – Summarises the project and our experiences.

2. Background

The ‘cost of living crisis’ refers to the fall in ‘real’ disposable incomes (that is, adjusted for inflation and after taxes and benefits) that the UK has experienced since late 2021. It is being caused predominantly by high inflation outstripping wage and benefit increases and has been further exacerbated by recent tax increases.

The cost of living is continuing to rise dramatically in the UK. Rising inflation and proposed tax increases are hitting the average consumer’s budget. This was particularly hard felt in April 2022 when the Bank of England energy inflation increased by 31% to 58% after Ofgem’s energy tariff caps were raised. Additionally, the government has proposed an increase to the national insurance rate of 1.25%, equating to an over 10% rise in the amount of tax paid by employees. Both will increase first time buyer’s monthly costs, and the lack of sufficient wage growth to cover these costs will compound the problem.

The significant rise in the cost of living is important to first time homebuyers and it will affect mortgage lenders assessment of their ability to afford a mortgage. We wanted to create an app that could contribute to helping and supporting people during the crisis, our app is a financial consulting and educative app. This app is purely backend, and when it is ran, the user is given textual instructions to support them in navigating the system.

1. This app has been designed for Python 3.
2. This app utilises an SQL database.
3. This app uses two API's:
    a. https://api-ninjas.com/api/mortgagecalculator
    b. https://money-live.p.rapidapi.com/news/newspapers
4. Installation of Python 3 'requests' libraries.

3. Specifications and Design

3.1 Requirements technical and non-technical

We have designed our app requirements with the user in mind. Considering that users use our app to seek advice on how to navigate in this current crisis, we have added features in the app to do so, by enabling them to view the latest financial news from their preferred news source, give an estimation of their mortgage rate, and finally, booking an appointment with a financial advisor specialised in specific areas. As such, we have identified the requirements to be as follows:

- Users must be able to interact with the app and input the necessary data.
- App must have a main menu to help users to navigate the features in the app.
- App should store the user's personal data only to book the booking feature.
- For the mortgage feature:
    - App must be able to connect to the mortgage calculator API.
    - App must return data according to the user's input, which are the amount to be loaned, the interest rate, and how many years the loan is for.

    - Data returned must be formatted so that it is easy and clear to read.

- For the news feature:
    - App must be able to connect to the financial news API.
    - Users should be given the choice to choose from different news sources.
    - Users should be given the title of the article and a link where they can click and be directed to the website.
    - Data returned must be formatted so that it is easy and clear to read.
- For the booking feature:
    - App must be able to store the user's personal data (first name, last name, email, address, and phone number) to the customer database.
    - App should allow users to choose the advisor based on the area they are interested in (Investment/Mortgage/Financial Plan) to provide more targeted advice.
    - App must show the availability for a particular day and for the chosen specialisation, before the user books an appointment.
    - App must be able to store the booking details in the booking database.
- After having a successful return on a particular feature, users should have the option to be redirected back to the main menu so that they can explore other features.

3.2 Design and architecture

The app consists of a MySQL database (create_database.sql) containing customer information, advisors information, and appointment information. The backend logic is done using Python and we have created an API to communicate with the database. Furthermore, our app also interacts with external APIs: mortgage calculator and cost of living news API. The overall architecture is shown in Figure 1.

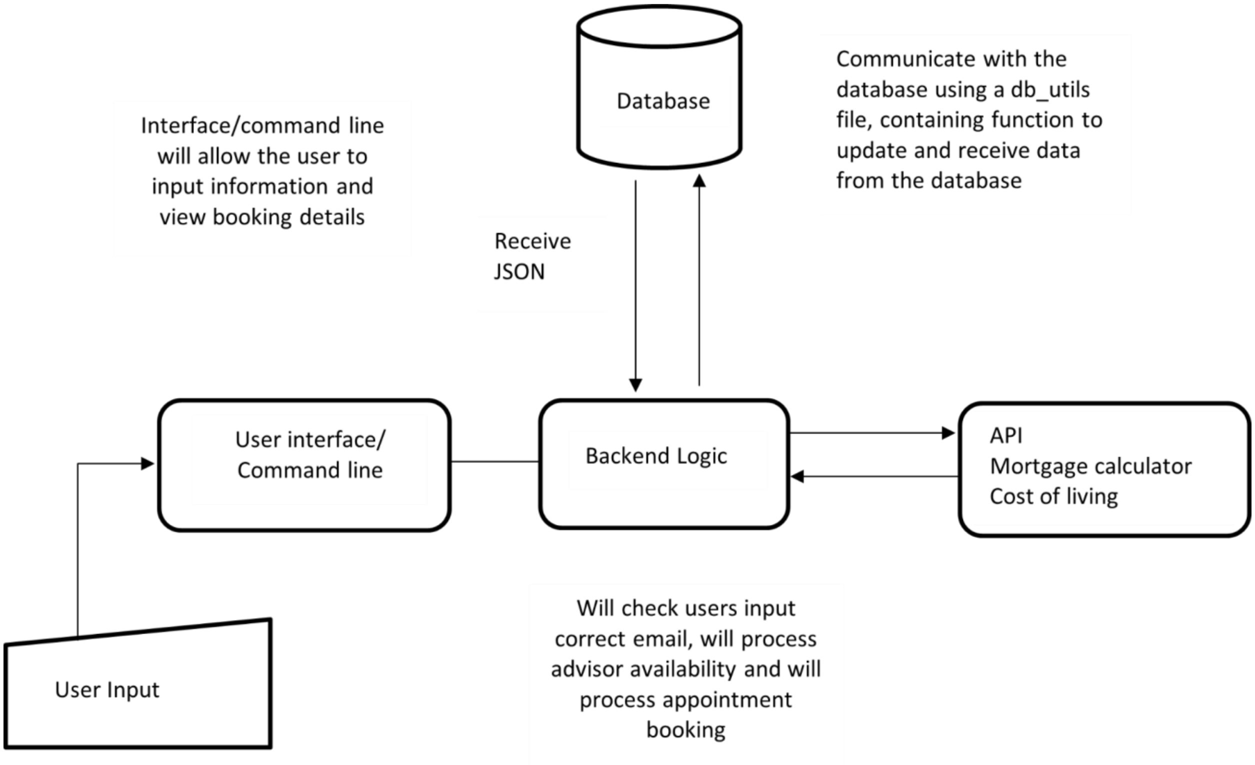
Database
Communicate with the database using a db_utils file, containing function to update and receive data from the database
Interface/command line will allow the user to input information and view booking details
Receive JSON
User interface/ Command line
Backend Logic
API
Mortgage calculator
Cost of living
User Input
Will check users input correct email, will process advisor availability and will process appointment booking

Figure 1. Diagram of the system architecture.

A flowchart describing how the app works is displayed in Figure 2.

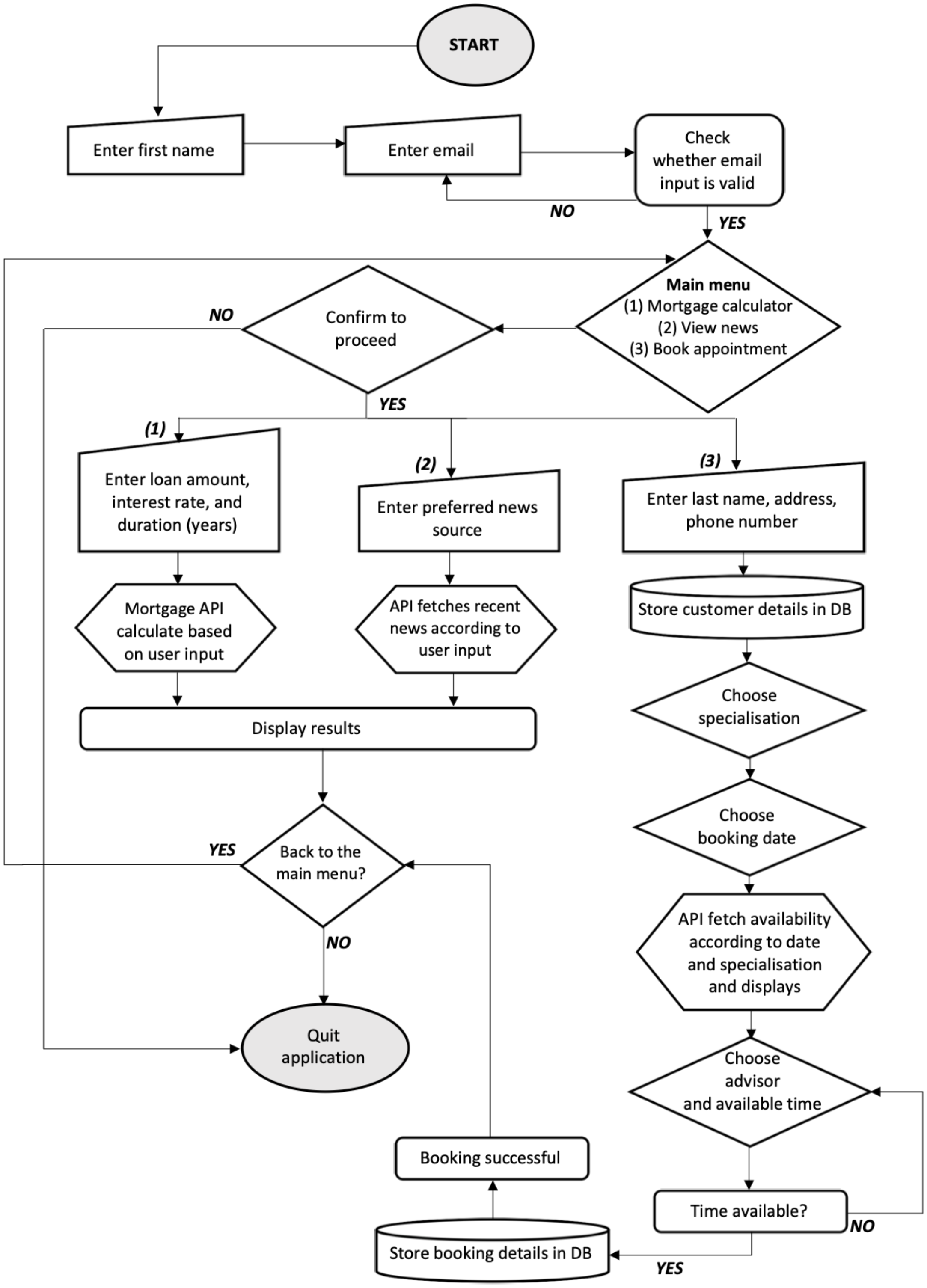
START
Enter first name
Enter email
Check whether email input is valid
NO
YES
Main menu
(1) Mortgage calculator
(2) View news
(3) Book appointment
Confirm to proceed
NO
YES
(1)
(2)
(3)
Enter loan amount, interest rate, and duration (years)
Enter preferred news source
Enter last name, address, phone number
Mortgage API calculate based on user input
API fetches recent news according to user input
Store customer details in DB
Display results
Choose specialisation
Choose booking date
Back to the main menu?
YES
NO
API fetch availability according to date and specialisation and displays
Quit application
Choose advisor and available time
Booking successful
Time available?
NO
Store booking details in DB
YES

Figure 2. Diagram of the process flow.

4. Implementation and Execution

4.1 Development approach

We shared all of our work via GitHub throughout the project. When a feature was completed, a pull request was submitted with every other team member assigned as a reviewer. A branch was only merged once every group member had had the opportunity to try out the code on their own computer and to leave any comments or suggestions. We also used Slack to communicate and had regular Zoom calls to talk through our work. This was very efficient when fixing bugs and implementing immediate feedback.

4.2 Tools and Libraries

Modules

Flask

When connecting with the database, flask serialized our data as JSON and jsonify was used to return our response object.

Libraries

Requests and Json

When working with moneylive and mortgage APIs, we used requests and json to ask for data and received it in json form.

Re

Regular expression was used to ensure that the input for customer’s email address is of a valid format.

Mysql.connector

This was used to connect to our MySQL database, in order to read and fetch from it and insert data into it.

4.3 Implementation Process

Decision to change something

Originally, we wanted our product to be just a booking system. We have later on decided to add live news viewing and mortgage calculator as side features. When we first implemented the code, our system would exit once the customer used one of the features. We decided that it would be better for customers to get an option to be redirected to the main menu because they might want to use more than one feature and starting the whole process from scratch is tiresome.

Achievements

We have made a functioning software that has all features we wanted to have. We have used git and github successfully to manage this project and worked collaboratively.

We have used git successfully to manage our code. Once pull request has been created, we made sure that it was merged after being checked by another team member.

We have developed many soft skills throughout this project, from time management to working as a team.

4.4 Agile Development

Through our project meetings, we implemented agile development by analysing our code and adapting it, such as turning our bugs into features to improve the functionality and quality of our code and adapt it to meet new deliverables that we felt would benefit our system, such as the system confirming the users booking.

In our meetings, everyone showed what they have been working on during the past week, what they will do next, and are there any problems to discuss with the rest of the group.

4.5 Implementation Challenges

It took a while to figure out how to connect mysql to python and achieve the result we wanted. It involved many hours of reviewing and research, as well as looking into similar examples. Many of the examples online involved SQLAlchemy instead of MySQL Connector, making this aspect particularly challenging.

Another challenge we faced was in adding concepts of OOP into our code. In the end, we were able to apply our knowledge of decorators using classes. We noticed that it would be useful to have each of our smaller codes wrapped in a “processing” and “process complete” block, so we used class decorators to achieve this.

As each of us has a job during the day, we found it tricky to manage our workload and find time to meet. Nevertheless, we met at least once a week (often, more than this) update each other on the work and discuss bugs.

5. Testing and Evaluation

5.1 Testing strategy/ Functional and user testing

We tested our code with simple print statements before each return statement and as we were writing it. We ensured two members of the group were assigned to each task so that we could pick up on anything that may have been missed. Some of this was done by pair programming where one person shared their screen and both members of the group contributed to the code. The final code was also completed using a pair programming technique where we exchanged ideas on how to best structure and consolidate it together. Each function after review, was transferred to the testing phase. Functionality was tested with various sample cases and one in each type was presented. Based on function, end cases were written and tested while for some of the functions, test cases were listed along with functionality requirements and were followed during the coding phase. Every function was finally tested for any undefined cases before finalising.

5.2 System limitations and future development

- Currently, our app is a console based text programme. We can incorporate frontend by using Tkinter.
- Have users sign in and out of their accounts, instead of navigating as guests
- Add feature for user to delete their booking
- Receive email notifications to confirm their booking with an advisor

6. Conclusion

With the ‘cost of living crisis’ continuing we hope that our application could be used to help alleviate some of the stress caused by the crisis and ensure our users are well informed about what they can do with their money. It was important to us to develop an application that would not only run efficiently but also contribute to improving people’s lives in some way.

Working within our team has been a valuable and enjoyable experience. We worked well together to achieve what we set out to do. Each member was able to use their skills to contribute to the project and ensured that it was successful. We are happy with the outcome and feel this experience has prepared us well for future software development projects.