Wardrobe Planner

Team 5

Gemma Allwright

Kinga Bulsiewicz

Paulina Szymczak

Katarzyna Wojtaszczyk

Magdalena Zembik

INTRODUCTION

The group project for the Code First Girls Degree (Software Specialization) is documented in this report.

Aims and objectives of the project

- Solves the age old question: What to wear?
  - Saves time by presenting a shortlist of clothes that match your mood, occasion and weather, allowing for quicker decisions.
  - Finds the forgotten treasures in your closet. Sometimes they even still have tags on them!

- Allows better maintenance of your wardrobe.
  - Notifies about the need to do laundry – for those who are forgetful or too busy to remember
  - Add and remove clothes

- Lets you share your clothes with your friends. You don't need to go shopping and spend more money.
  - Lowers further Earth pollution – exchanging clothes enables people to wear different clothes without having to purchase them
  - Saves money for people who are on a low budget but still want to be fashionable, have a wide range of outfit choices without spending money

BACKGROUND

'According to the UN Environment Programme, the fast fashion industry is the second-biggest consumer of water and is responsible for about 10% of global carbon emissions - more than all international flights and maritime shipping combined.'[1] Its impact on the environment along with its poor record on human rights make it an industry we want to be part of the solution for.

A survey by TotalJobs.com found that women spend four months of their working life deciding what to wear and it's something that all of us relate to.[2]

The wardrobe browser project aims to help users organise their wardrobe, browse their clothes, and get outfit suggestions based on certain criteria.The goal of our team was to create an application with a simple and easy-to-use interface that simplifies decision-making, at the same time still allowing users to express their creativity and personal style.

The project also promotes sustainable fashion practices by enabling users to share their clothes with other users whom they can tag as 'friends' in the app. This way, people can borrow clothes from each other, reducing the need to buy new items and contributing to limiting waste.

One important feature of the wardrobe browser is its integration with a weather API that provides up-to-date weather information. This allows the application to suggest outfits that are suitable for the current weather conditions (and in the future development - when planning an outfit a short time in the future)

---

[1] https://earth.org/fast-fashions-detrimental-effect-on-the-environment/

[2] https://www.independent.co.uk/life-style/work-clothes-average-woman-four-months-lifetime-office wear-suits-skirts-jackets-dresses-fashion-a8090531.html

By using technologies like SQL databases for storing data, weather APIs for getting weather information, and Python for the programming part, the wardrobe browser is a simple but comprehensive tool.

Due to the scope of the project and time limitations the team decided to make certain simplifications. For example we made assumptions such as all users will be of the same gender (it is a Code First Girls project after all!), same size, and that exchange between users would be somewhat smooth. A more extensive version in the future would take gender, size and proximity also into consideration. As well as other criteria that could be identified by testing the app with real users.

To enhance the user experience, the wardrobe browser in the future could allow users to save their favourite outfit combinations, receive reminders that laundry is due, and create a catalogue of their preferred looks.

SPECIFICATIONS AND DESIGN

We used the MOSCOW method to decide what was essential and what might be added later as we knew that we had a limited timeframe to complete the project. This also allowed us to think about scalability of the app and what features could be developed in the future. It is worth noting that although we had listed the laundry feature as a Could, we did manage to implement it.

Requirements technical and non-technical

Must

- User profile
- Create an updatable wardrobe database for each user
- Give the user the option to share their wardrobe with others
- Search function to search your own wardrobe and the wardrobes shared with you (search criteria like weather, occasion, style, date)
- Retrieve current weather

Should

- Allow the user to become 'wardrobe friends' with someone and allow the items in your wardrobe to become shareable with them and vice versa
- Update item status
- Show you your selection based on the search criteria

Could

- Save User's Choices
- Laundry function and reminder
- Allow to include future weather conditions in the search
- Create a donation list
- Send a request to share an item of clothing from one of your friends' wardrobes
- Have a frontend

Might

- Chat with friends to request clothes
- Create clothes swapping events
- Create a catalogue of already created or favourite looks

Design and architecture

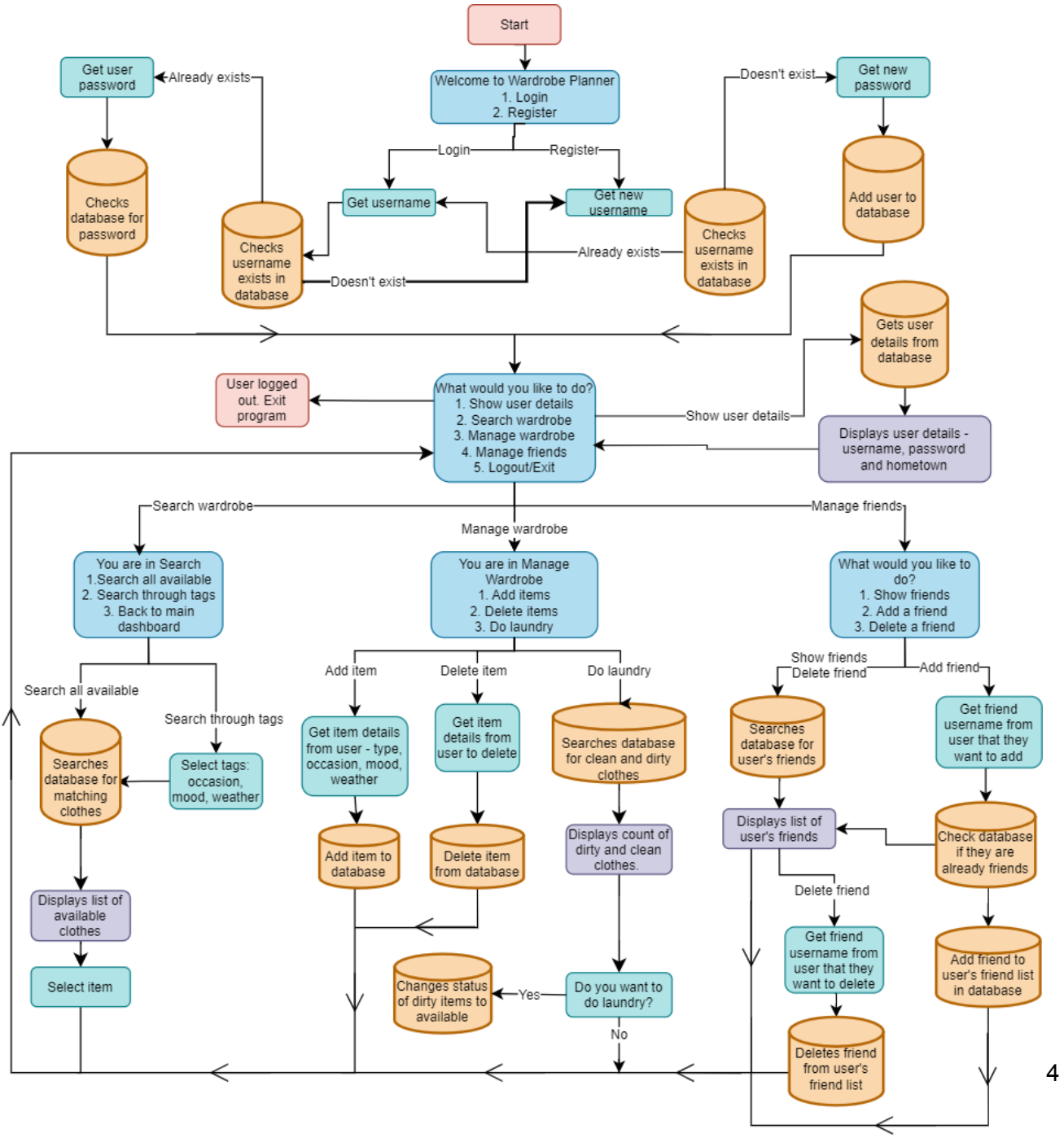
The SQL database "wardrobe_planner" consists of several tables that are designed to store and organise data related to users, their wardrobe items, friends, and availability status. The structure of the database follows the principles of normalisation to ensure efficient data storage and retrieval. The relationships between tables are established using primary key and foreign key constraints.

The tables in the database include:

- The "users" table, which stores user information, such as usernames and passwords.
- The "user_location" table, recording the location details of each user, including hometown and geographical coordinates.
- The "friends" table, establishing relationships between users and their friends (=those who the user wants to share clothes with)
- The "ownership" table, tracking the ownership of clothing items by users.
- The "clothes" table, storing information about individual clothing items, including item type, description, and associated tags.
- The "availability_status" table, tracking the availability status of clothing items.

In the **Python backend** we made a clear structure, grouping most of the files into 3 main categories: classes, functions (without class), and tests. Outside of the folders we only left the main file, along as wardrobeDB.sql, README and logo. As there was no time to implement a front end, the programme works as a console program. We added a 'wardrobe planner' logo to bring little bit of personality to the app.

The diagram below shows the flow of the program.



IMPLEMENTATION AND EXECUTION

Development approach and team member roles

All of us are well aware that in a professional environment there would be a lot of planning beforehand, and also implementing clear to do tasks for everyone, strict deadlines, probably very frequent morning stand ups, probably also an agile way of working with a project manager and ideally a scrum master. In reality, with only 4 weeks to implement from cradle to grave, with team members still learning the theory, doing homework, preparing for the final exam, and working full time, this was too hard to implement.

After deciding on our project theme we took time to look at the requirements of the program by using the MOSCOW method. This enabled us to decide on feature and task priority which aided our decisions as we started building the program. Initially the team was divided into two, one team looked at the database requirements and the other the python backend structure. Once these were established team members chose features of the program they wanted to work on through discussion on slack and Trello to ensure there was no duplication. Team members with similar tasks worked together on some things and everyone helped each other with code review.

To execute our project, we took a flexible approach where team members independently took ownership of different tasks based on their skills and interests. There was no rigid division of roles or a clear-cut split of tasks. Instead, team members contributed according to their expertise and time to work on different aspects of the project. Team members gravitated toward tasks that aligned with their skills, ensuring that each aspect of the project received attention from those who felt most comfortable with it. This also ensured all team members got experience working on all the aspects they wanted to. By using Trello, a bug tracking log and keeping in touch on Slack we ensured work was not duplicated and that each area of the project was covered.

Tools and libraries

We found these following tools and libraries essential to enable database connectivity, retrieve weather data, geocoding addresses, facilitate iterations, manage dates, and add a visual element at the beginning of our wardrobe search project.

- SQL database for storage
- Free Weather API for checking weather (https://open-meteo.com/)
- Python for backend
- Trello - keeping a list of who was doing what, to dos and what was in progress
- Github - used branches for different features and to review code before merging
- Slack - kept in touch via chat and frequent video chats
- mysql.connector - to allow our Python application to connect to and interact with a MySQL database.
- requests - to facilitate making HTTP requests to external APIs (in our case - weather API).
- geopy.geocoders.Nominatim - this geopy library provides geocoding services, the Nominatim geocoder allows to convert addresses into geographic coordinates (latitude, longitude).
- itertools - for efficient iteration and combination of elements.
- datetime - to handle and manipulate date-related data.
- art.text2art - was used to create our wardrobe planner ‘logo’ in the terminal.
- unittest - to create automated tests
- mock - to mock the database and user inputs while testing.

Implementation process

Ambitions were high, but only due to time constraints we decided we will not introduce some of the functionalities outlined in the beginning of this document.

We created an organisation on GitHub for our repository so that we could work together and use github's version control features to manage the project. We created branches for different features and then merged them together once the team was happy with the functionality.

Based on the functionalities we envisioned for the application we drafted a list of routines that divided these functionalities into possible functions and classes. Based on that we created separate files. Of course some of these plans were amended as we started to write code. We made a clear file structure, grouping most of the files into 3 main categories: classes, functions, tests. Functionalities that required several separate routines were grouped into sub folders in the functions folder. Outside of the folders we only left the main file, along as wardrobeDB.sql, README and logo.

Achievements

We successfully managed to create a working programme which integrated a lot of the features we wanted to, including the weather API, friends lists and laundry function. Our testing and debugging skills have improved significantly. We have worked well together as a team understanding each others' strengths and delegating where necessary.

Agile development

The team was aware of the ideal scenario to implement agile development practices, however due to a dynamic nature of the process, agile practice was not implemented. However, we did use some agile practices such as refactoring and code reviews. We are well aware there is room in the future for improvement in terms of overall efficiency.

Implementation challenges

As beginners, our team faced several challenges and teething problems that are common in the early stages of development for users with no experience. These challenges included learning new programming concepts, troubleshooting issues, breaking and fixing, and adapting to unfamiliar technologies and frameworks. The learning curve was steep, and it took a lot of time to overcome these hurdles.

We managed to address some of these challenges and steadily improve our skills. We supported each other where we could by sharing knowledge or seeking guidance from external resources.

TESTING AND EVALUATION

Testing strategy

Overall, our testing methodology involved a continuous cycle of manual testing, code review, bug fixing, and code rewriting if needed. We tested our code with print statements as we were writing it and tested out SQL queries in workbench prior to adding them to our db_utils file. We did a lot of manual user testing, working through each step of the program as a user would. After we were happy with most of the functionality of the functions and program we did unittests for each of the functions to make sure the program had integrity. Our team prioritised the quality of the code and worked hard to ensure that the application was functioning as intended.

System limitations

Due to time constraints and aiming to create a program which works there are some limitations to our program.

- Assumptions of same gender and size
- Assumptions of simplified exchange process
- No frontend interface
- Assumptions of user behaviour - we assume that users will keep their wardrobe consistently updated
- Reliance on an external API
- Limited number of features

CONCLUSION

To conclude, our team has successfully developed a wardrobe browser application that addresses the common struggle of deciding what to wear, and at the same time encourages sustainable fashion practices. The application saves time, reduces waste, and lowers its users' environmental impact.