PrepMate

Project plan

Group 15

Egle Sakalauskaite, 5566207
Gijs Six, 5574242
Hiba Azzim, 5304881
Ischa Hollemans, 5415039
Yasmina Aarkoub, 5416930

Contents

1. PrepMate: the concept 2
2. Requirements 4
3. Technical approach **9**
4. Development methodology 1**0**

1. PrepMate: the concept

Going into this project our group was tasked with making a tool that either aids the user with a problem or solves a problem for the user. This left us with a very broad field of topics and specific problems to choose from, but finally, after week 1, our group came up with the idea to make a tool that brings a solution to people who spend hours looking for recipes matching their food allergies, dietary restrictions and personal preferences, deciding what to eat and making a shopping list for the weekly visit to the supermarket. We decided to make a tool that helps them with cooking, planning meals and shopping: PrepMate.

Purpose of application

PrepMate is an application meant to aid users in their meal preparation. It is an app that will allow users to search for recipes in a database based on a set of requirements (think caloric value per serving, allergies, dietary restrictions, must-have ingredients, cooking duration, and the like). The app will then present the user with a list of recipes based on these criteria. However, the list will have a specific limit to prevent the user from choosing from a seemingly endless list of options and encourage fast decision-making.

With the user having chosen a recipe and completed the first step in meal preparation, the user can then insert the meal in a calendar to plan recipes for the coming days. In addition, the ingredient list will automatically be added to a shopping list generated by the app, which will update every time the user selects a recipe.

The app is meant for anyone who wants to prepare their meals quickly and consciously in as little time as possible, for example, students or anyone with an overfull schedule. Instead of having to spend Sunday morning looking through the entirety of the internet for recipes conforming to your dietary restrictions or preferences (often on websites with filters insufficient for your needs) to know what to cook the following week, and then manually going through every recipe to add up the ingredients, PrepMate can do all this for you and more. Thus, the app aims to encourage users to prepare their meals and simplify and facilitate the process, thereby saving them time and aiding them in their diet. The app will also have features to accommodate users with differing preferences and needs, such as adding your own recipes and editing the shopping list in case you already have some ingredients at home or would like to buy some snacks.

Ethical implications

One significant ethical dilemma is the source of the recipes from the database and the possible use of an API to provide the users with recipes from the web. While a barebones recipe consisting of simple ingredients and basic instructions is not copyrightable, recipe websites often contain much additional text that cannot simply be copy-pasted into our app. Additionally, even though a reference to the source is optional for barebones recipes, including such a reference to give credit would still be a good practice. These points will have to be kept in mind during the development process.

Another ethical implication that could come with this project is user data storage. The initial idea for our project is to make the application work locally so any personal information would only be stored on the user's device. This would bring little ethical implications. However, in later versions, the feature to create an account could be added so that users can access their shopping lists, choose recipes, and plan meals on multiple devices. In order to make this work, the user's data should be stored in an online database. This could be an ethical problem since the database contains personal information on many users and will be accessed by others or might be vulnerable to hacking. The user

should, therefore, be informed about their data being stored in an online database, which should be managed carefully.

Comparison to existing software

Already existing applications such as Paprika Recipe Manager, Yummly, and Plan to Eat have a lot of similar functionalities. They all provide meal planning for at least a week into the future, let you look up recipes, and add ingredients from the recipes to a shopping list. However, despite featuring filtering options and preference settings when searching for recipes, their filtering options are limited, and none of them permit filtering based on specific ingredients or calories per serving. Additionally, only some of these applications allow users to input their recipes and calculate the calorie content per serving based on the ingredients.

Another key difference between these existing applications and our project is that ours will be open-source and free to use. This distinction likely stems from differing goals among the creators: the developers of existing software aim to generate revenue, whereas our project's objective is centered around learning. Also, this means that we can not use any code from these existing applications but have to write all code by ourselves.

2. Requirements

The following requirements have been set up to showcase features that will be implemented in our software. They are categorized according to the MoSCow method.

Must-haves
These requirements are enough to create a functioning minimum prototype that already serves the project's primary purpose - to generate recipe recommendations and shopping lists according to user input.

Title: Database of recipes	**Priority:** Must have	**Estimate:** 2
As a user, **I want** to have access to a variety of recipes from the start of using the app, **so that** I can start planning my meals right away.		
Acceptance criteria: - A minimum of 30 recipes are stored in the database. **Tasks:** 1. Hardcode 30 recipes.		

Title: User information	**Priority:** Must have	**Estimate:** 3
As a user, **I want** to be able to specify my personal information, **So that** I can get recipe suggestions that consider this information.		
Acceptance criteria: - The user can input food allergies - The user can input dietary restrictions - The application has a window that takes user input through radio buttons, dropdown menus etc. and then saves this data in the database. **Tasks:** 1. Create the user interface window with widgets for entering data and a save button 2. Define a function that gets executed when the save button is clicked, that connects to the database 3. Define the table creation, data adding and editing for the database		

Title: Recipe suggestions	**Priority:** Must have	**Estimate:** 4
As a user, **I want** to receive recipe suggestions, **So that** I can create individualized meal plans.		
Acceptance criteria:		

- The application must filter out recipes that conflict with allergies and dietary restrictions and/or harshly deviate from the calorie intake goal the user has indicated and randomly displays a fixed amount of suggestions.

Tasks:

1. Create a display window that lists out recipe suggestions
2. Write the function that takes in all the recipes stored in the database and the user information and filters out recipes accordingly.

Title: Shopping list	**Priority:** Must have	**Estimate:** 3

As a user,
I want the app to generate a shopping list I can view based on the recipes I selected,
So that I have one shopping ready to go when I go grocery shopping.

Acceptance criteria:

- When the user selects a recipe, all the required ingredients and their quantities must be saved in the shopping list that is stored in the database.
- If an ingredient is already in the shopping list, the app increments the quantity instead of adding the same ingredient twice.
- The user should be able to view the shopping list

Tasks:

1. Create a shopping list page in the GUI
2. Create a connection between the database and shopping list page so the shopping list gets displayed on the page

Tasks:

1. UI: creating the window, defining the layout, adding the widgets
    a. Main window
    b. User information window
    c. Recipe Suggestions window
    d. Shopping list window
2. Database: defining the methods of creating tables and adding/editing data
    a. Recipe table
    b. User information table
    c. Shopping list table
3. Backend: defining the attributes
    a. recipe class
    b. ingrediënt class
    c. user class
4. Other
    a. Hardcoding recipes

Should-haves
These are the requirements that greatly improve and expand the functionalities of the minimum prototype version.

Title: User preference	**Priority:** Should have	**Estimate:** 3
As a user, **I want** to able to enter my preferences, **So that** they are taken into account when the recipe suggestions are generated.		
Acceptance criteria: - The user can input their liked and disliked ingredients - The user can input the preferred cooking duration - The user can input calorie intake goals - The user can input an ingredient that must be included in all recipe suggestions - The app should filter on based on the user's preferences **Tasks:** 1. Create a user preference page in the GUI 2. Add preferences which can be stored in the database		

Title: User's recipes	**Priority:** Should have	**Estimate:** 4
As a user, **I want** to be able to save recipes to the existing recipe database, **so that** I can have it suggested later for my meal plans.		
Acceptance criteria: - the user should be able to enter a recipe's name and list out all the ingredients and their quantities. This information should be saved in the recipe database. **Tasks:** 1. Create the user interface window with widgets for entering recipe information and a save button 2. Define a function that gets executed when the save button is clicked, that connects to the database		

Title: Manual editing of the shopping list	**Priority:** Should have	**Estimate:** 2
As a user, **I want** to manually edit the shopping list, **So that** I could adjust it according to the ingredients I already have at home, or add extra things, such as snacks.		
Acceptance criteria: the user should be able to add and delete items from the shopping list and save these changes in the database. **Tasks:** 1. Create a tool where the user can edit the stored shopping list in the database		

Title: Meal planning	**Priority:** Should have	**Estimate:** 3
As a user, **I want** to plan a selected meal in one of the upcoming days, **So that** I have a meal plan ready for the coming week.		
Acceptance criteria: - When the user has chosen the recipe, the user should be able to select the day for which the user wants to plan the meal - The user should be able to view a calendar of the current day and the next seven days, which contains the already planned meals. **Tasks:** 1. Create a calendar like structure page in the GUI		

Title: Cook book	**Priority:** Should have	**Estimate:** 4
As a user, **I want** to be able to get an elaborate instruction about how to prepare the given recipe, **So that** I don't have to consult another website to prepare my meals.		
Acceptance criteria: - When the recipes are chosen, then they have to be accessible in the recipe screen, where all the recipes for that week are stored. **Tasks:** 1. Create a recipe page in the GUI where all the recipes are displayed for the upcoming week.		

Could-haves

These are the requirements that mostly could add new features to the application. However, the project already serves its main purpose without these functions added, therefore they will stay as optional in case there is enough time to implement them.

Title: Calorie count for user-added recipes	**Priority:** Could have	**Estimate:** 3
As a user, **I want** to my added recipes to display their calculated caloric value, **So that** I know how many calories my recipe contains.		
Acceptance criteria: - The app should have a database of calorie information for each ingredient - The app should calculate the caloric value of user-added recipes based on the database of calorie information for each ingredient - The app should display the calculated caloric value in the user-added recipe **Tasks:** 1. Create database where all calorie information is stored for each ingredient 2. Create an algorithm that calculates the caloric value of user-added recipes		

Title: Varied diet	**Priority:** Could have	**Estimate:** 2
As a user, **I don't want** to get almost the same ingredients that I would have had yesterday/tomorrow, **So that** I can enjoy a varied diet.		
Acceptance criteria: - When the user already had a certain type of carb (pasta, potatoes, rice), then it won't be recommended - When two dishes have 80% overlap in ingredients, then it won't be recommended for the adjacent days. **Tasks:** 1. Create an algorithm that checks if the chosen recipe doesn't conflict with the recipe from the day before or the next day		

Title: Scaling ingredient quantities	**Priority:** Could have	**Estimate:** 1
As a user, **I want** to convert recipe quantities for a different number of servings, **So that** I have the correct quantities for the number of people I'm preparing a meal for.		
Acceptance criteria: - The user can input the number of servings - The app should scale the ingredients quantities based on the number of servings **Tasks:** 1. Create an algorithm that automatically calculates the amounts for the ingredients for different numbers of servings.		

Title: Extracting the shopping list	**Priority:** Could have	**Estimate:** 2
As a user, **I want** to get the shopping list as a file, **So that** I could send it to my email and have it on my phone when I go shopping.		
Acceptance criteria: - The shopping list gets formatted and saved in a convenient to read way **Tasks:** 1. Create a tool that is able to export the shopping list as a txt file.		

Won't-haves

These are the requirements that have been rejected by the development team because they deviate too much from the main purpose and functionalities of the application. Moreover, adding these requirements might cause conflicts with other functionalities or make the user interface too crowded.

- The application will not keep track of your calorie goals.
- The application will not keep track of a pantry list with things you already have at home.

Feasibility

The main focus of our application is to simplify the process of choosing and preparing a meal by giving recommendations according to the given preferences. The application's core can be described with the must-have requirements, which are relatively easy to develop. The app should have a graphical user interface, store given user input, and have a database complete with recipes.

The only must-have that would need more time to be realized is generating the recipe suggestions since it relies on the user's personal information and the content of the database. The more recipes in the database, the higher the chance that a recipe suggestion can be given that fits the user's input. Some group members have experience retrieving data from databases to implement its data in an application. They have a clear idea of how to realize this requirement.

By creating multiple screens in the GUI, most features of our software can be realized. Separate screens can be made for the user's personal information, saved recipes, recipe suggestions, and the shopping list. The backend would mainly focus on generating recipe suggestions by traversing the database and filtering based on the user's input. However, it would also include, for example, scaling ingredient quantities based on the number of servings. Possible obstacles could be encountered when developing an algorithm for the recipe suggestions because the suggestions are generated based on multiple user preferences and standards.

To conclude, we have a good idea regarding how we want to implement our requirements. With the programming experience of all developers and the structural approach, this project should be feasible.

3. Technical approach

Code libraries

The following libraries have been chosen for this project:

- SQLite: This library provides a reliable and easy-to-use solution for local data storage. Its seamless integration with Python makes it well-suited for your application's backend needs.
- PySide6: The stakeholders suggested this library because of its user-friendly nature and the substantial flexibility it offers. This is crucial for constructing an intuitive and visually engaging interface, ensuring a positive interaction experience for users.
- Mypy: This library is a type checker for Python, which is crucial to the clarity and predictability of the code. This library was also mandatory.

Architecture

We have chosen a layered architecture for this project due to the numerous benefits it brings to the table. This architectural choice simplifies the project's complexity by breaking it into smaller, more manageable tasks. The layered approach eases problem-solving and enhances the project's scalability. This is important as the project time is limited. The app can be further developed, or new features can be implemented if there is more time than expected or after the deadline has passed.

One potential drawback of this architecture could be performance, as it requires constant interaction with various layers. However, given the relatively small size of the project, the impact is expected to be minimal.

The architecture of the project will be split into the following three layers:

- User interface layer: This is the entry point where users interact with the application through the graphical user interface (GUI). It is responsible for presenting information to users and receiving their input.
- Business logic layer: The business logic layer resides at the core of the architecture. Here, critical functionalities of the project are housed, encompassing processes, calculations, and decision-making. This layer forms the engine that powers the application's core operations.
- Database layer: The database layer is the data storage and retrieval repository. It facilitates access to stored data, enabling the application to collect and modify information. The business logic layer orchestrates interactions with the database layer.

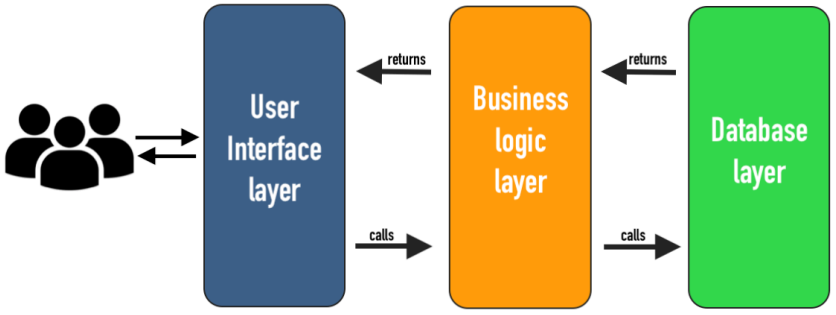

Figure 4.1. Visualized interaction between all architectural layers.

4. Development methodology

Development process
Our aim is to have a smooth development process with effective communication and collaboration during this project. In order to realize these key aspects of the process, we chose the software development approach Agile. By applying the Scrum framework we can easily keep track of the progress of each individual developer and the team as a whole.

While working as a group on developing an application, it is important that we are flexible and adaptable. It can be done by dividing tasks, integrating connected features together and providing assistance to each other when needed. This is one of the reasons Agile seems to be the best approach for this project. Flexibility could also entail fine-tuning the implementation of requirements when needed.

Since time is of essence, keep track of our goals and progress is of utmost importance. A clear overview can be realized through backlogs of the deliverable and the current progress. Weekly scrum meetings will ensure that each member is up-to-date and involved in the planning process. We collectively reflect on our app development, potential obstacles or issues and areas of improvement. By confirming all developers are on the same page, we can keep working towards our goal with any misunderstandings and take the key points from our reflection into account.

In Scrum there are different roles to guarantee an efficient process. These are the following: Scrum Master, Product Owner and Development Team. For our purposes, we will only be assigning the role of Scrum Master, who will chair the meeting and write an agenda beforehand. The only other role that will be assigned is the role of Secretary; someone responsible for writing the notes during meetings. In addition, every member of the group is a part of the development team.

The following table contains a schedule with the role of Scrum Master and Secretary for each week.

Week 3	**Scrum Master**	Egle
	Secretary	Gijs
Week 4	**Scrum Master**	Hiba
	Secretary	Ischa
Week 5	**Scrum Master**	Yasmina
	Secretary	Egle
Week 6	**Scrum Master**	Gijs
	Secretary	Hiba
Week 7	**Scrum Master**	Ischa
	Secretary	Yasmina

Week 8	**Scrum Master**	Egle
	Secretary	Gijs
Week 9	**Scrum Master**	Hiba
	Secretary	Ischa
Week 10	**Scrum Master**	Yasmina
	Secretary	Egle

User stories will be written based on the requirements to improve the development of the application. During the project, Gitlab will be used to manage the repository and aid development. All user stories and other To Do-items will be written as issues in Gitlab, and user stories will be labeled with the priority of the related requirement in the MoSCoW ranking. The status of these issues will be monitored using Gitlab's issue board, using the lists "To Do," "In Progress," and "Done". Each issue will be assigned to its developers using Gitlab's feature, and the time tracking tool will be used during development. Gitlab and its merge requests will also be used to manage the branches and for continuous integration.

Group interaction

General agreements:

- Attendance of the weekly meetings is mandatory.
- If there is a need, a team member can propose an additional meeting. If all team members agree on the meeting and it is time, that meeting becomes mandatory.
- We should do our best to be punctual.
- If a team member cannot attend the meeting, they should inform the team as early as possible.
- All deadlines should be respected. Those include both course-set and agreed-upon deadlines.
- We will make decisions by consensus.
- We do our best to follow the fundamental principles of Agile development.

During each weekly meeting:

- The rotating note taker and scrum master are appointed for the upcoming week.
- Scrum Master creates and leads the agenda for their assigned meeting.
- Secretary records all progress and key discussion points of their assigned meeting.
- Everyone can share their last week's work progress and any challenges faced.
- The workload for the next week will be distributed fairly. If a team member feels overwhelmed with their workload, they can ask for redistribution.

Communication:

- We communicate with each other openly and respectfully. That includes using proper language, not talking over one another, etc.
- All feedback given on each other's work should be constructive – instead of criticism, possible improvements should be suggested.
- We will acknowledge and appreciate each other's work.
- We should avoid off-topic discussions during the meetings unless a break is proposed and agreed upon.
- Direct contact with group members is via WhatsApp.
- Contact with the TA is via Mattermost.

Repository rules

To ensure that working together on the project in GitLab runs smoothly, we have drawn up a few guidelines for working with branches, merging, and keeping track of our progress:

Branches:

- Do not commit directly to the main branch
- Create a separate branch for each feature/part of the project or bug fix
- The branch's purpose should be evident from its name
- Only one person works on the same branch at the same time

Merging:

- Each merge request should be provided with a title and a clear description
- For merge requests implementing new features: only merge after getting two merge approvals from other group members

Progress tracking:

- Use the GitLab issue board to keep track of what tasks need to be done by whom and what user stories have already been implemented.