2021

Macro Manager Environment

Contents

Executive Summary

The following document will detail the environment which the Macro Manager app and platform will be implemented in. A high level overview of the tech stack will be discussed (not including APIs and libraries) with justifications for each technology, platform etc. These technology platforms are as follows:

- Kotlin and Android Studio
- Golang API
- PostgreSQL
- Heroku

A detailed wireflow diagram will also show the process of a user moving through the app with mockup designs of the apps pages. Logical and physical entity relationship diagrams will be provided for insight into the design and implementation of the database. Several risks will be highlighted and accounted for. These include:

- Unfamiliarity
- Bugs and Errors
- Mistakes in the Design Process

Finally, the expectations for iteration 1 will also be discussed.

Technology Platform

Overview

The Macro Manager environment will be developed in and utilise many different platforms and hosting services. The first prototype will be developed in Android Studio using Kotlin. The API will be developed in Golang and data will be stored in a PostgreSQL database. Hosting will be done through Heroku.

Kotlin and Android Studio

Android Studio is an integrated development environment for the development of Android apps. It provides the options to use Java or Kotlin as the development language. Kotlin will be used for the development of this application. The Android application will be the interaction point for users to make use of the systems which will be developed.

Golang API

The API (Application Programming Interface) will be developed in Golang. Golang is a statically typed language developed by Google employees and is used heavily within Google. The API will act as a bridge between the application frontend (Android app) and the Potsgres database. This is to prevent tampering, hacking and bad user inputs from affecting the database in a negative way. For instance SQL injections may be used to retrieve data from the database which a user should not have access to, such as other users data, which will no longer be possible when acting through an API.

PostgreSQL Database

The database will be designed using PostgreSQL an open-source, relational database management system. The database will hold all of the data each user stores and will have communication with the API in order to show users their information within the Android app.

Heroku

Hosting of the API and database layers of the Macro Manager system will be on Heroku. Heroku is a cloud platform which offers a generous free tier with support for many programming languages.

Justification

There are many reasons for choosing the platforms mentioned above. Firstly, the system requires an app. This app could be developed for Android, iOS or a hybrid model (platform agnostic and works on all operating systems). Initially a hybrid app seemed to be the best way forward as it would allow multiple platforms to be reached, however it might lead to a limitation of features. Due to this an Android app was chosen as that is the phone which the team has access to.

Golang has been chosen as the language for the API as it is fast and light-weight. It is a fantastic language for developing light-weight microservices and has an easy to follow syntax being very similar to C. The decision to design an API is mostly due to the fact that it will remove a lot of the work required in implementing the Macro Manager app in other forms such as for iOS or web broswer platforms as all of the logic will be handled on an external backend server by this API and can be accessed by any system capable of sending API calls. The language that the API could be programmed in is mostly at the discretion of the developer. A lot of developers might choose to make their API in JavaScript. This is neither right nor wrong and often comes down to preference and what exactly the team wants out of the API. Some of the benefits of writing this API in Go include better scalability and optimisation and the fact that base Go (the language without any added modules which are similar to packages in Java) provides you with a multithreaded server which makes use of all processor cores, making it the fastest and most efficient it could possibly be. (mohelgamal, 2019)

Initially the database was to be designed in MySQL. However, the native database of Heroku is PostgreSQL and so it made more sense to use Postgres as it would be more straightforward and it is far more scalable. Should this application grow in size Postgres would be able to handle the traffic and use much better than a MySQL database could. On top of this, Postgres offers more data type options than MySQL allowing more freedom of data within the tables as well as having top notch user forums and documentation made by the community. (Chen, 2021)

Finally, Heroku is the hosting site of choice as it is a familiar platform for the team and offers a free tier with 5 apps to be hosted. As such Heroku is more than capable of hosting each of the layers of the application. Heroku is also very user friendly and is ideal for small teams or individual projects. As the team size of this project is 1, Heroku is perfect for this project. (Philogène, 2020) Heroku also allows you to not have to worry about the infrastructure of your servers. (Almeida, 2019) With many other cloud hosting platforms such as AWS, Azure and, Digital Ocean, the user must specify their storage and computing needs which is not necessary with Heroku, allowing the user to focus on the development of their platform without spending time figuring out the processing power required to run it.

GUI

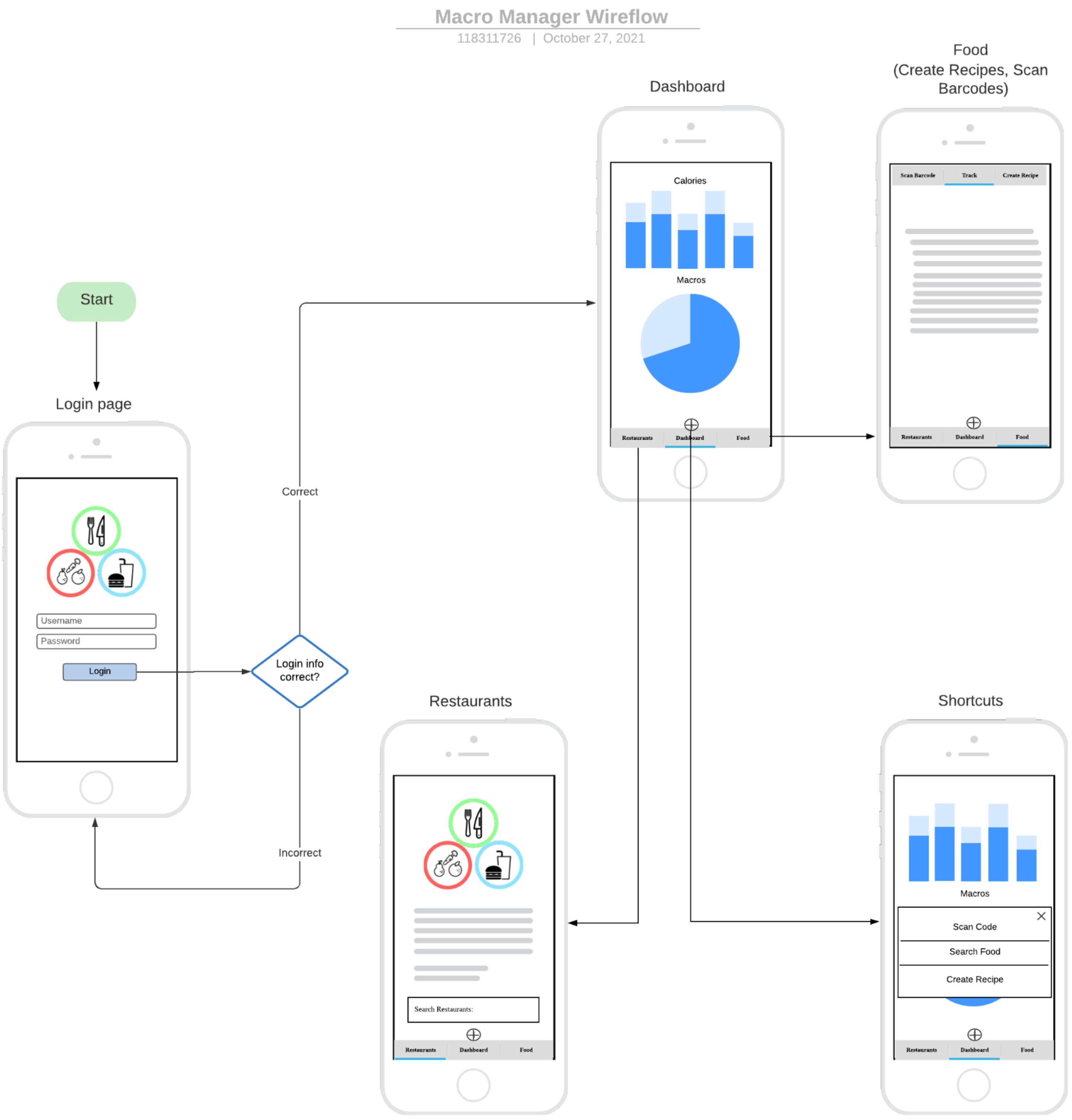
The Graphical User Interface for Macro Manager will benefit greatly from simplicity. One of the problems which the app is to solve is the complexity of tracking calories that exists with the other apps currently on the market. As such the proposed design of the apps pages are very simplistic and minimalist. Below is a wireflow which shows the basic designs of each page (subject to change) and the page which each action item, usually a button or tab, takes you to upon being pressed. When a user starts the app they will be taken straight to the login page. From here you may login and if successful you will be taken to the dashboard. The dashboard will house several types of graph to show calories eaten, macro percentage (percentage of calories coming from each of the three macro nutrients: protein, fats, carbohydrates) and more. There is also a tab at the bottom for navigating to each page.

On this tab there are several options:

- Dashboard, which the user is landed on upon login.

- Restaurants, where the user will look at a list of restaurant cards and filter and search them in order to decide where to eat.
- Food, where the user will be able to view what they have eaten throughout their day so far (and for previous days as well), scan foods barcodes and create recipes out of foods which they have scanned or created.

Wireflow


Macro Manager Wireflow
118311726 | October 27, 2021
Start
Login page
Username
Password
Login
Login info correct?
Correct
Incorrect
Dashboard
Calories
Macros
Restaurants
Dashboard
Food
Food (Create Recipes, Scan Barcodes)
Scan Barcode
Track
Create Recipe
Restaurants
Search Restaurants:
Shortcuts
Macros
Scan Code
Search Food
Create Recipe

Database

Overview

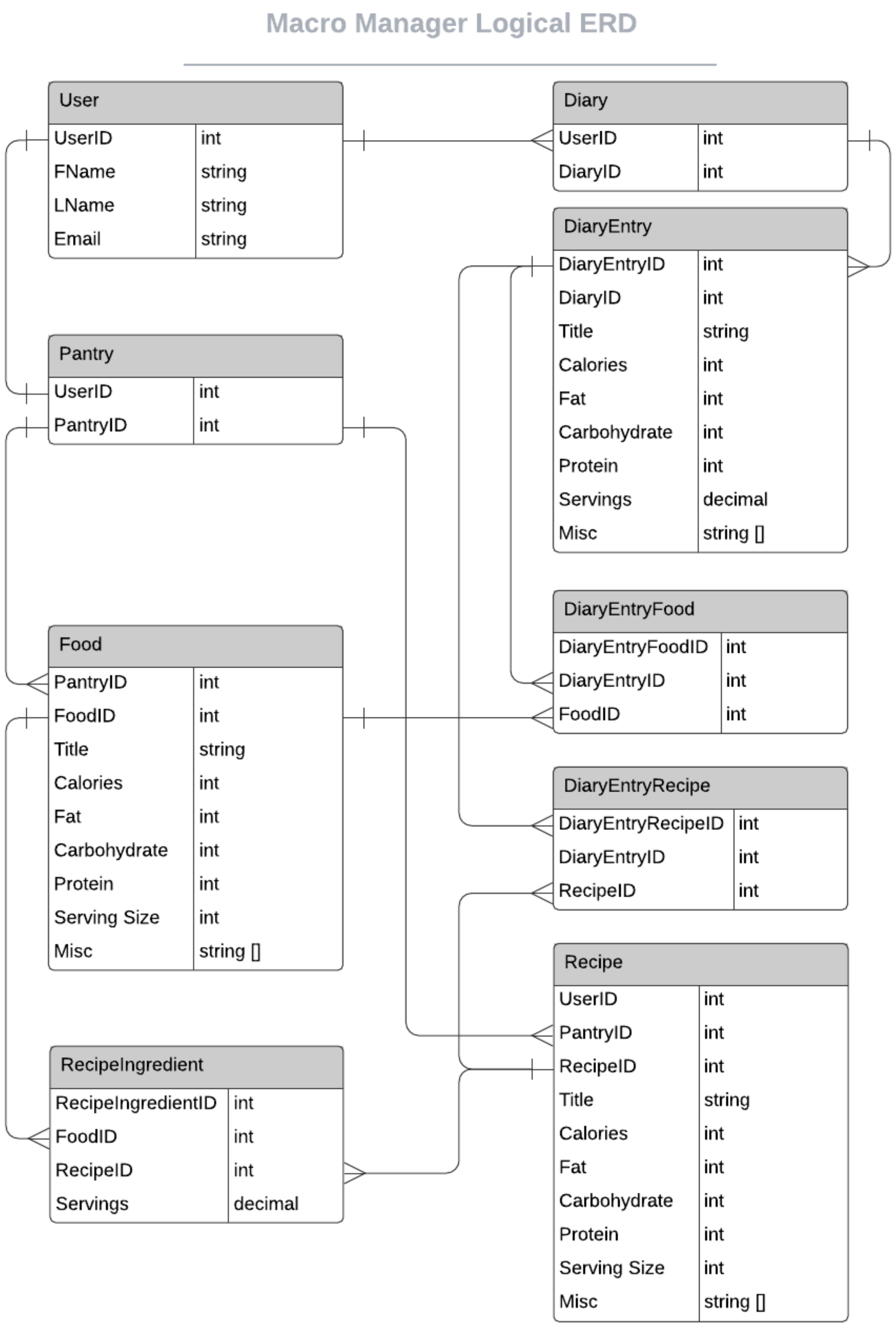
The database for Macro Manager contains several many to many relationships, as such there are several join tables (an entity connecting two entities which would otherwise enter into a many to many relationship) as relational databases struggle with these types of relationships. Below are two entity relationship diagrams detailing the logical and physical shcemas of the database. The logical schema details the entities, relationships, fields and field types. The physical schema details the above and includes primary and foreign keys, indicating which field in each table will be the unique identifier and which keys create relationships with other tables. (Visual Paradigm, n.d.).

As stated in the High Level Design document for this project; The tables to be present are as follows:

- User
- Diary
- DiaryEntry
- Pantry
- Food
- Recipe
- DiaryEntryRecipe
- DiaryEntryFood
- RecipeIngredient

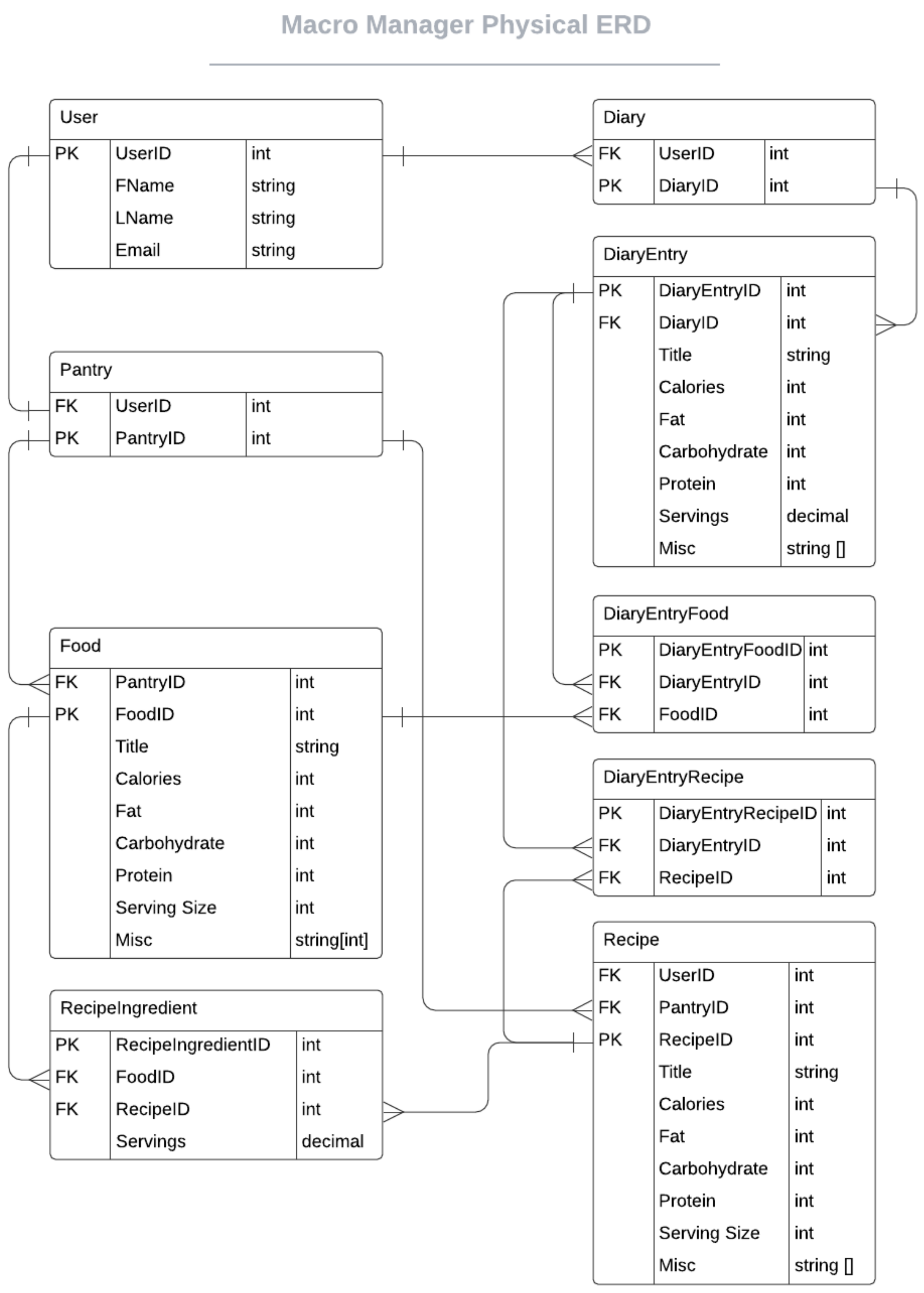
The User, Diary, Recipe and, Pantry tables are to contain the details of the user, each days calorie entries (contained within each individual entry in the DiaryEntry table), a users recipes and, a users scanned foods (contained within each individual entry in the Food table) respectively. The DiaryEntryRecipe, DiaryEntryFood and, RecipeIngredient tables are there to eliminate the many to many relationships which would occur otherwise. These relationships are due to multiple ingredients/food items (Food table entries) being present in a recipe and each ingredient being present in multiple recipes [PantryRecipe], multiple recipes appearing in any one diary entry, recipes appearing in multiple diary entries and, diary entries containing the same recipe more than once [DiaryEntryRecipe] and, finally, food items in the pantry appearing in multiple diary entries or multiple times in the same diary entry [DiaryEntryFood]. (Sallam, 2021) There have been several changes to the ERD betweenthe previous iteration of the ERD. Notably several fields have been added to tables which were previously only populated by primary and foreign keys. These fields are mostly titles, calories and macro nutrients in order to track the values of custom recipes and amounts easier. Also a miscellaneous field has been added to several of the tables to add more descriptive tags and details, for instance if a user wanted to mark a recipe as Keto, or gluten free or organic etc. this would be saved in the miscellaneous field.

Logical



Macro Manager Logical ERD
User
UserID int
FName string
LName string
Email string
Diary
UserID int
DiaryID int
DiaryEntry
DiaryEntryID int
DiaryID int
Title string
Calories int
Fat int
Carbohydrate int
Protein int
Servings decimal
Misc string []
Pantry
UserID int
PantryID int
DiaryEntryFood
DiaryEntryFoodID int
DiaryEntryID int
FoodID int
Food
PantryID int
FoodID int
Title string
Calories int
Fat int
Carbohydrate int
Protein int
Serving Size int
Misc string []
DiaryEntryRecipe
DiaryEntryRecipeID int
DiaryEntryID int
RecipeID int
Recipe
UserID int
PantryID int
RecipeID int
Title string
Calories int
Fat int
Carbohydrate int
Protein int
Serving Size int
Misc string []
RecipeIngredient
RecipeIngredientID int
FoodID int
RecipeID int
Servings decimal

Physical



User

PK	UserID	int
	FName	string
	LName	string
	Email	string

Diary

FK	UserID	int
PK	DiaryID	int

DiaryEntry

PK	DiaryEntryID	int
FK	DiaryID	int
	Title	string
	Calories	int
	Fat	int
	Carbohydrate	int
	Protein	int
	Servings	decimal
	Misc	string []

Pantry

FK	UserID	int
PK	PantryID	int

DiaryEntryFood

PK	DiaryEntryFoodID	int
FK	DiaryEntryID	int
FK	FoodID	int

Food

FK	PantryID	int
PK	FoodID	int
	Title	string
	Calories	int
	Fat	int
	Carbohydrate	int
	Protein	int
	Serving Size	int
	Misc	string[int]

DiaryEntryRecipe

PK	DiaryEntryRecipeID	int
FK	DiaryEntryID	int
FK	RecipeID	int

RecipeIngredient

PK	RecipeIngredientID	int
FK	FoodID	int
FK	RecipeID	int
	Servings	decimal

Recipe

FK	UserID	int
FK	PantryID	int
PK	RecipeID	int
	Title	string
	Calories	int
	Fat	int
	Carbohydrate	int
	Protein	int
	Serving Size	int
	Misc	string []

Risks

Several risks exist in this project. Several of these risks have been discussed in the High Level Design document for this project. Additionally, there exist risks surrounding the environment specifically. These include:

- **Unfamiliarity:** The team is mainly unfamiliar with many of the technologies to be used in this implementation. As such this may cause issues during the development of the project.
- **Bugs and Errors:** Bugs and errors will be found in every project. This is non-negotiable. However, it is very unlikely that these bugs and errors will be impactful enough to cause any large delays. (Sallam, 2021)
- **Mistakes:** It is possible that mistakes were made in the database schema and planning of other platforms which will result in delays and time needed to make fixes.

Iteration 1

Overview

At each iteration of this assessment, the team must present a working project. As such the first iteration will requirea lot of work to be done on all parts of the project. The tasks to be completed in this iteration are:

- An implementation of the database schema in PostgreSQL with hosting set up
- An implementation of the wireframe mockups as shown above
- An implementation of the feature to scan barcodes on foods
- Integrate the database and app via a simple API which will allow food to be save from the app to the database

Assumptions

- User logins will not be available for the first iteration
- The database may not be fully complete during this iteration
- The API will have minimal functionality during this iteration
- The Android app may not have all of the pages which will appear in the final version during this iteration

Conclusion

In conclusion, the environment which Macro Manager is to be developed and implemented in has been well thought out and documented. The key points of this document are the technologies which will be used to implement the solution, the risks involved and the deliverables for iteration 1.

To recap, the technologies to be used are Kotlin, Android Studio, Golang (API), PostgreSQL and Heroku (Hosting).

There are a few risks involved in the development of the project, namely a lack of familiarity with several of the above technologies, however, being familiar with other similar technologies should be of help to the development team.

Finally, deliverables for iteration 1 include implementations of the database schema, wireframe mockups and, the barcode scanning feature and, the integration of these project layers through a basic implementation of the API.

References

Chen, B. (2021, September 2). *Fivetran*. Retrieved from PostgreSQL vs. MySQL: What You Need to Know: https://fivetran.com/blog/postgresql-vs-mysql

Sallam, O. (2021, October 15). Macro Manager High Level Design.

Visual Paradigm. (n.d.). *Visual Paradigm*. Retrieved from Data Modeling: Conceptual vs Logical vs Physical Data Model: https://online.visual-paradigm.com/knowledge/visual-modeling/conceptual-vs-logical-vs-physical-data-model/