INTEGRATED PROJECT DEVELOPMENT

SOFTWARE REQUIREMENT SPECIFICATIONS - Stage II

FOOGLE

Version: 1.0

Developed By

Akhil Surendran (17BCE7087)

Dedipyaman Das (17BCE7080)

Suhruta Malireddy (18BCD7003)

Ajeeth Paul (18BCD7058)

Taha Junaid (18BCD7072)

VIT-AP

Date: 27th July, 2019

FP Batch ID: 8

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this application is to make the restaurant experience hassle-free and customer-based. It aims to simplify menu-reading and allows for a more personalised experience. It encourages restaurants to raise their standards, therefore benefitting both consumers and the business.

1.2 Intended Audience

This document is used by developers for understanding the requirements of the project; also the intended users of this document are project managers for planning and scheduling, testers to generate test cases, document writers for preparation of user manual and for other end-users/stakeholders to validate their requirements.

1.3 Project Scope

Menu Reviewing is an innovation which makes it easy for people to know more about the dish they are about to spend on. This application is aimed at helping people to be informed appropriately of the myriad of dishes available for consumption at the eatery. This application is also simple to use and hands customers ample information at their fingertips.

The goal of the project is to reduce the time and effort of the customer spent in reading long and strenuous menus and build a database based on user provided data to dispense appropriate results to the queries of the customer.

1.4 System Requirements

- A smartphone with Android Marshmallow and above.
- 1 GB Memory
- An internet connection.

1.5 Design and Implementation Constraints

Foogle is a 3-tier application implemented on the Android platform. Foogle is modular in design as opposed to a monolithic structure. The layers follow a well defined interface which lays out the contracts for each module to communicate with each other in a systematic way.

The Front-End

The user facing application is built on top of **Android SDK API level 27** designed with **XML** and **Gluon** for the User Interface design. The front end is managed by a set of Controllers written in **Java, C++** and **Kotlin**. C++ has been injected into the project for optimal performance by utilizing low level control structures.

Java and Kotlin are responsible for sending requests and accepting responses from the server and interpreting it for the front end to use.

The Communication Channel

The application communicates with the server using a **REST API** using **JSON** to transfer data between the server and the client. All the connections are encrypted by a **Secure Socket Layer** based on **RSA-2048 bit encryption** and **SHA-256 HMAC** signature.

The Back-End

The back-end is built on top of the **LAMP** stack with **PHP 7.3** as the main driver and **MySQL** to provide the relational database. The back-end is based on a Cloud Server hosted on **Amazon Web Services - EC2** as the **Compute Server**, an **S3** instance as **object storage** and **CloudFront** as the **Content Delivery Network** to speed up static asset delivery.

2.System Features

#	*Title of System Feature*	*Description*	*Priority*	*Functional Requirements*
	The Global Menu	*While a plate’s cost-to-price ratio is important, it’s not the sole factor in determining an item’s success. By tracking restaurant menu analytics, you’ll be able to determine which menu items drive return visitors, larger check sizes, and even a greater frequency of visits.*	High, 9	As a user starts the app, they will be presented with the menu of the restaurant they are currently in. The app will automatically scan the user’s location and ask for confirmation. If it fails to detect the location, as a fallback, it will prompt the user to scan the QR code on the Foogle supported menu. The user can then choose the dish from the menu that they want to know about. They can apply filters to the results based on their preferences, votes by other users etc. After choosing the dish, the user will be provided with information about the dish and comments/reviews by other users.
	User Preferences	*The user’s choices at each restaurant visit are stored in order to help the user make better dish selections each time, based off of his/her preferences.*	High, 9	A login screen allows a user to log in with password based authentication. If the user isn’t already registered, they are redirected to a view for registration in which they provide their credentials and basic profile information. A logged in user can then enter their preferences manually using an interactive

				view. Preferences are also updated automatically as they use the app while they are logged in.
	Recommended restaurants/dishes	*The data collected from the user preferences and usage history allows us to send periodic notifications and recommendations.*	High, 8	Notifications are sent to the user via Firebase Cloud Messaging service based on a CRON job.
	Add a restaurant/unlisted dish	*If a user visits a new restaurant not listed in Foogle, they can opt to add it. Similarly, if a new dish/special is added to a menu, users can add this to the listing in the app.*	Medium, 7	Users (who are logged in) can add to the Database of dishes and restaurants via a form, in which they must fill up the details requested. The new information is then added to the database.
	Review dishes/restaurants	*The app aims to give back the control of the menu from the restaurant to the customers. Users can anonymously add reviews/comments about their experience whilst gaining points on their contributions as incentives.*	High, 8	After consuming a dish, users can share their experience (w.r.t taste, spice levels, accuracy of menu description, etc). Once feedback is given, this can be viewed by other users in the future. This shared input will be stored in the database, as well as in the user's history.

3. External Interface Requirements

3.1 User Interfaces

3.2 Hardware Interfaces

3.3 Software Interfaces

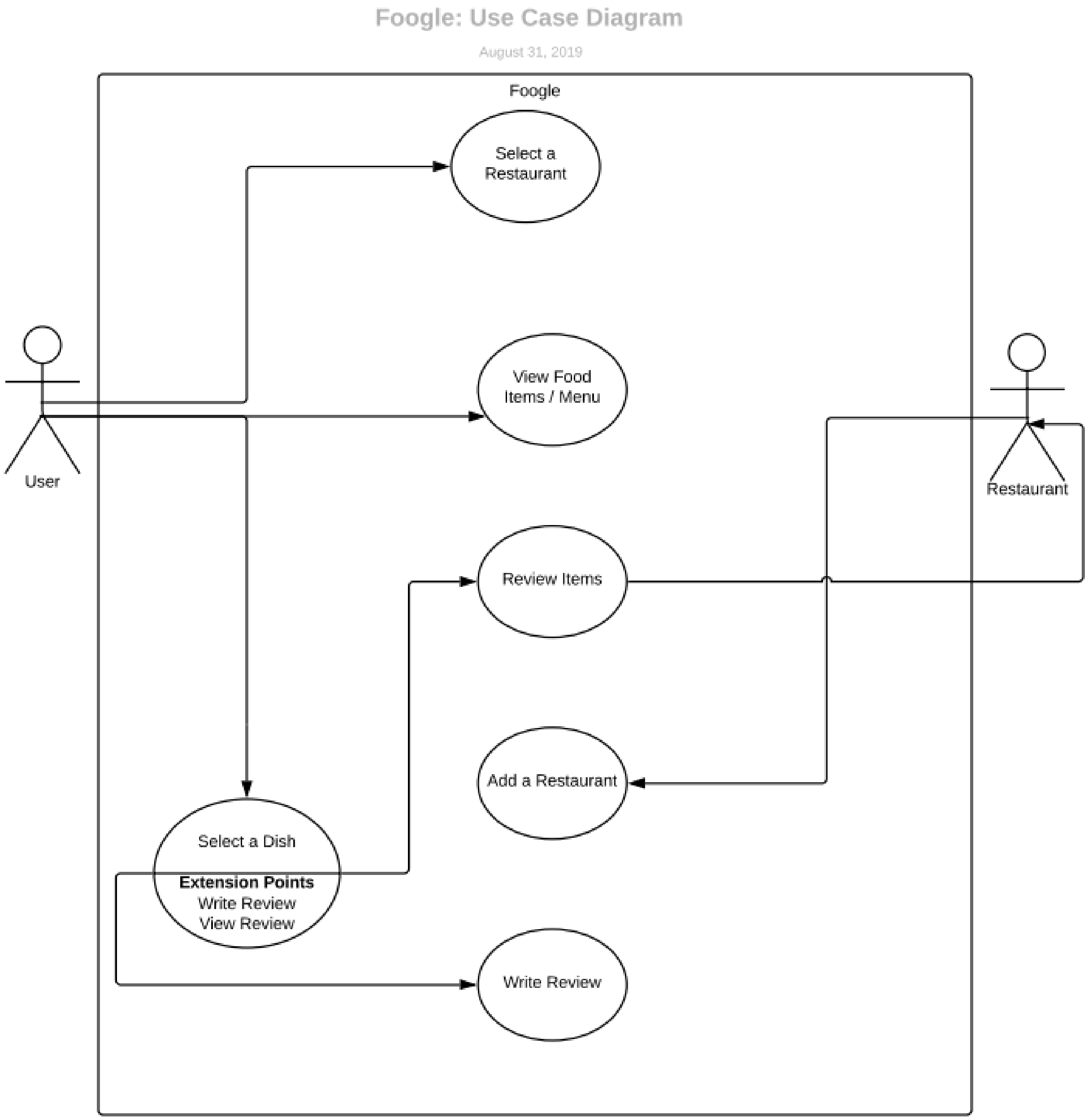
3.4 Communications Interfaces

4. Other Nonfunctional Requirements

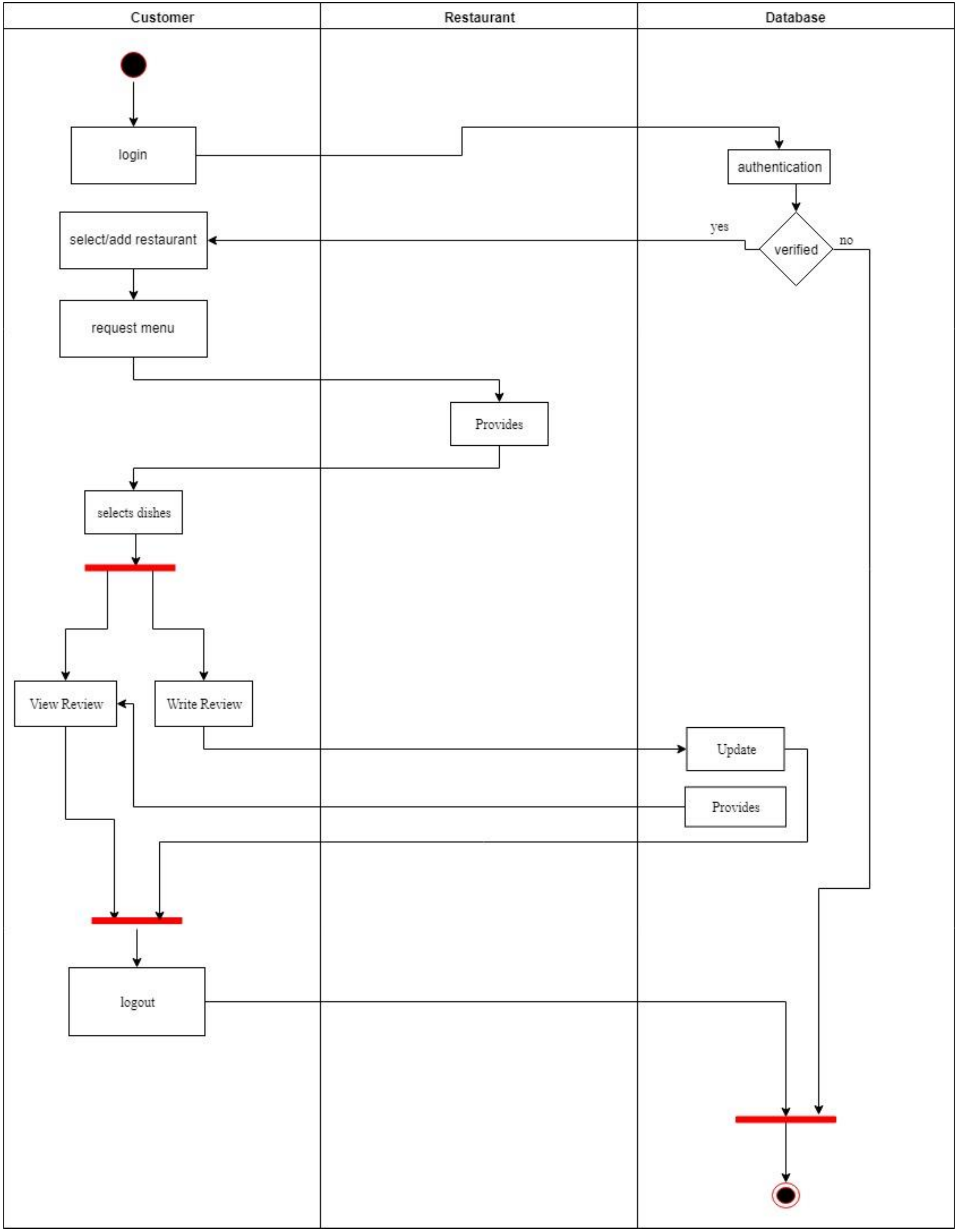
5. References

UML DIAGRAMS:

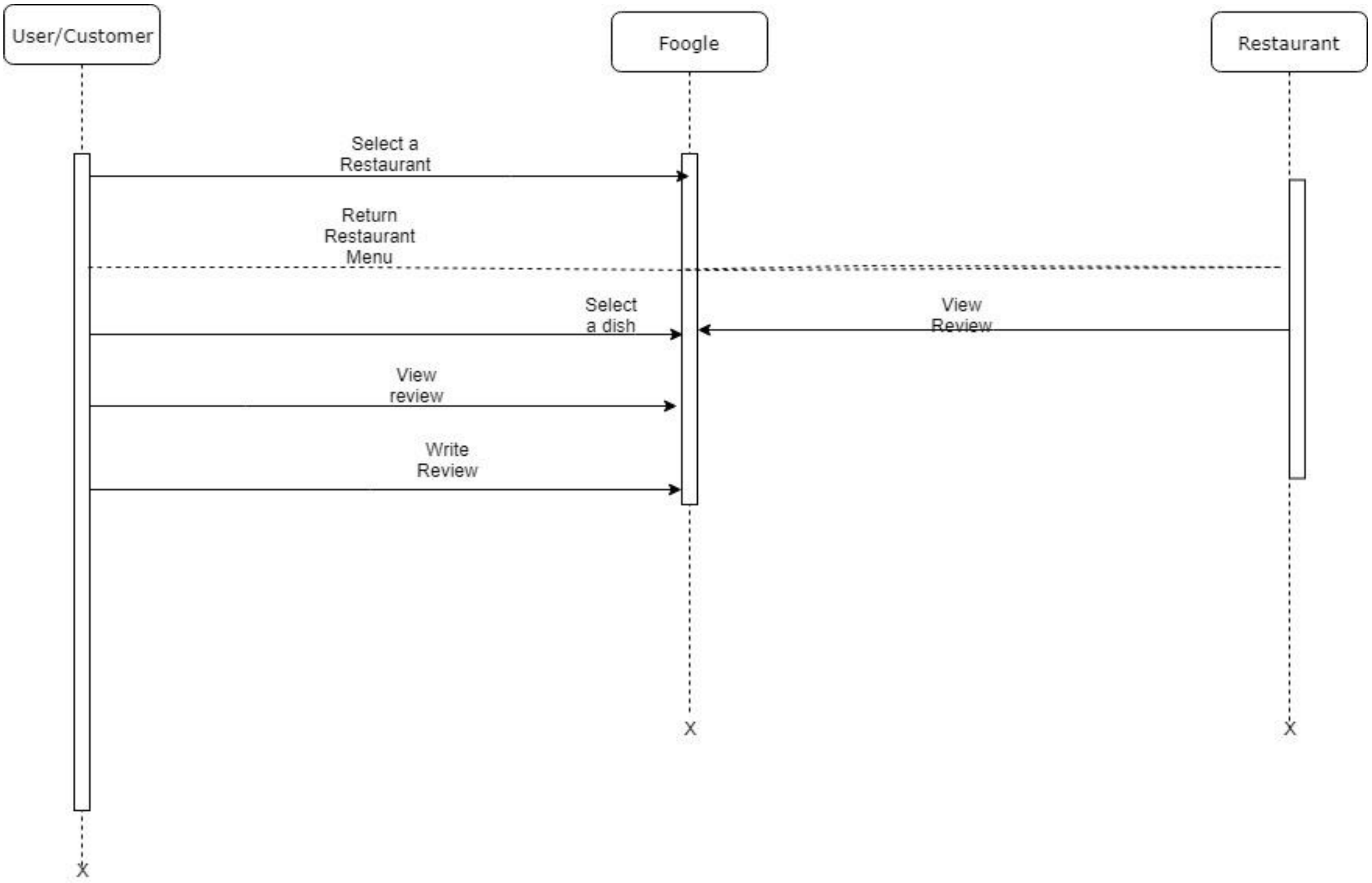
1. Use case



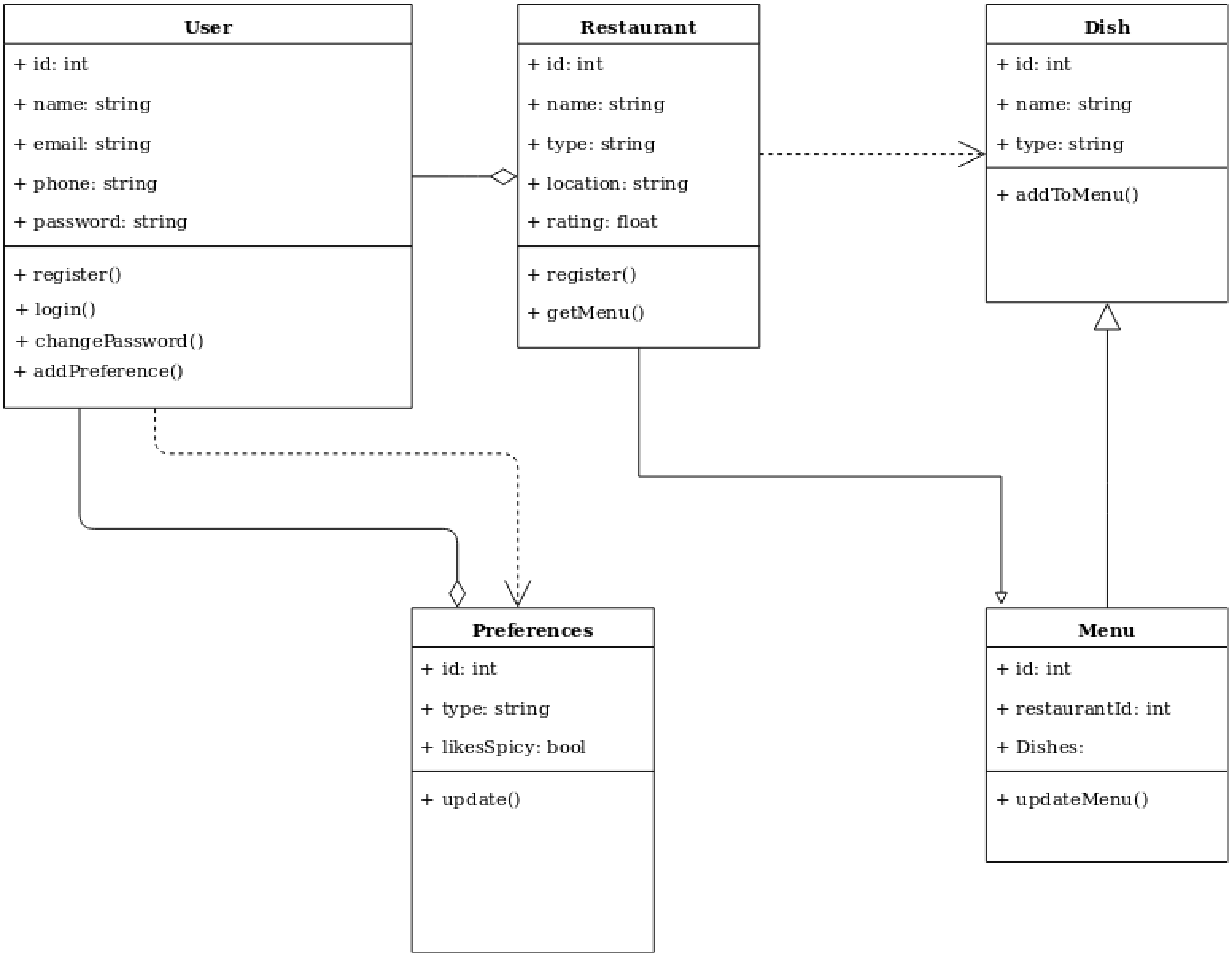
2. Activity diagram:



3. Sequence Diagram



4. Class Diagram


User
+ id: int
+ name: string
+ email: string
+ phone: string
+ password: string
+ register()
+ login()
+ changePassword()
+ addPreference()
Restaurant
+ id: int
+ name: string
+ type: string
+ location: string
+ rating: float
+ register()
+ getMenu()
Dish
+ id: int
+ name: string
+ type: string
+ addToMenu()
Preferences
+ id: int
+ type: string
+ likesSpicy: bool
+ update()
Menu
+ id: int
+ restaurantId: int
+ Dishes:
+ updateMenu()