

# Mixbook

**Mixology Application**

**Design Document**

**Team 3:**

**Alexander Coll (acoll@purdue.edu), Matthew Fouts  
(foutsm@purdue.edu),  
John Tyler Preston (presto@purdue.edu), Nicholas  
Zenobi (nzenobi@purdue.edu)**

## **Table of Contents**

<b>1. Purpose</b>	<b>2</b>
<b>2. Design Outline</b>	<b>2</b>
<b>a. Client</b>	<b>3</b>
<b>b. Server</b>	<b>3</b>
<b>c. Database</b>	<b>4</b>
<b>d. Miscellaneous</b>	<b>4</b>
<b>3. Design Issues</b>	<b>4</b>
<b>a. Functional Issues</b>	<b>4</b>
<b>b. Non-Functional Issues</b>	<b>5</b>
<b>4. Design Details</b>	<b>7</b>
<b>a. Class Design</b>	<b>8</b>
<b>b. Description of Classes</b>	<b>9</b>
<b>c. Event Sequence Diagrams</b>	<b>10</b>
<b>d. UI Mockups</b>	<b>14</b>

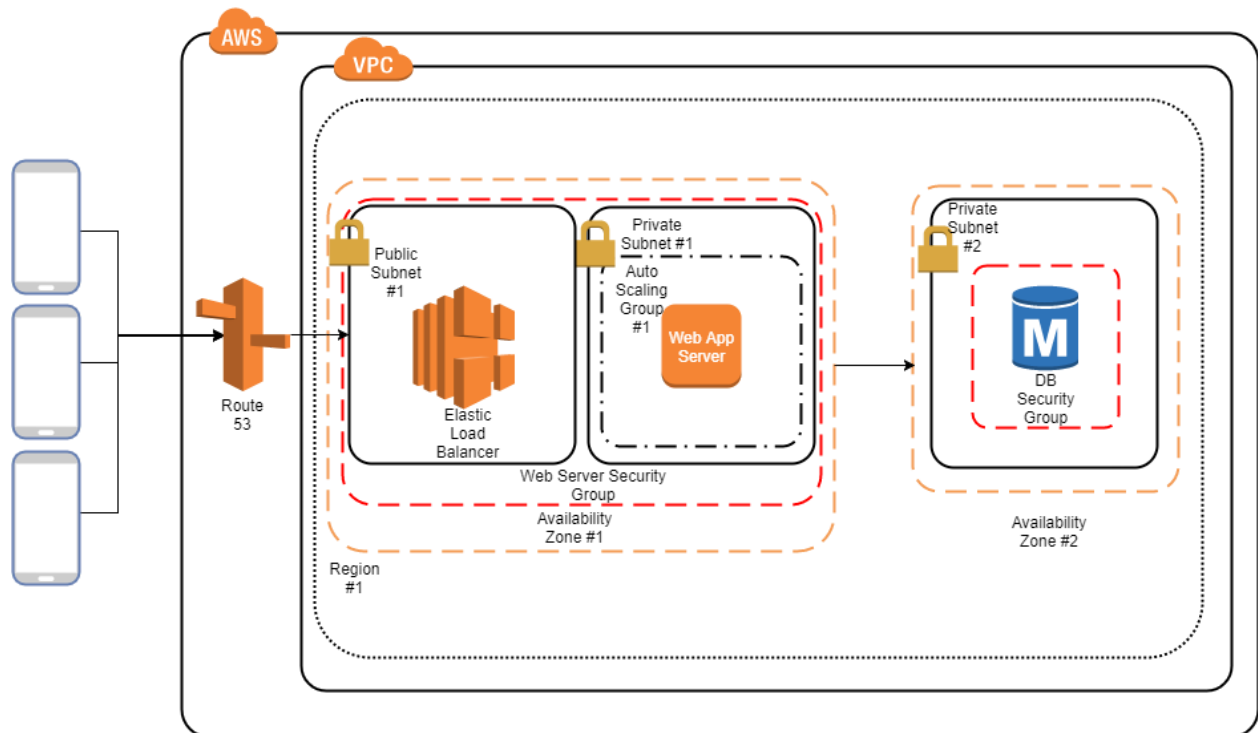
## Purpose

No drink beats a well mixed cocktail at your favorite bar; however, what if you don't want to leave your house or pay \$10 for drinks that you can make yourself for the fraction of the cost? You could mix your own drinks at home, but you're not a bartender: you don't know what drinks you can make with what you have in your kitchen; our mixology app aims to solve that problem: give it a list of ingredients and Mixbook will give you detailed recipes using only those ingredients. Currently there are a multitude of mixology apps that just solely list recipes. Our application serves to create a social media where users can share recipes and review drinks through a simple user interface.

For the longest time, drink recipes were in paper-bound books and usually only bartenders would have these books. With the advent of the internet, websites started cropping up, although shoddy, with some drink recipes here and there, though nowhere near complete or user friendly. As mobile applications came about, this technology shifted to become mobile app based, though none were anything more than static drink recipe books. Currently there is no existing mobile application that allows a user to find recipes based on the ingredients that they have as well as allow the sharing and reviewing of drink recipes in a user friendly manner. It is frustrating searching for hours to find drink recipes that are craftable with only the ingredients that you currently have, and this mobile application aims to change that.

## Design Outline

The application will use a client-server model. The client operates through an Android interface by connecting to the server. The server will handle requests by the client; examples include: editing a recipe, rating a review, and changing their password. The server will also handle the queries to the database. The database will store the necessary information about recipes, reviews, and about each user. Below is a diagram about the setup of the client-server model (which uses AWS for everything on the backend).



## Client

The client sends requests to the server, as well as receives responses from the server. The client then interprets the response and displays them using the Android interface.

## Server

The server receives requests from the client. The server then interprets the request from the client and completes the action needed for the request. The server also fetches the necessary information from the database to complete the request. The server then formats the response, and sends the response to the client to be interpreted.

## **Database**

The database will be implemented in SQL, which should allow for easy queries that reduce the work done by the server. The database should handle data relating to the recipes, reviews, and users. The fields of the database will be further explained in the Design Details section.

## **Miscellaneous**

The domain is hosted in AWS Route 53, which maps the domain using an A record to an Elastic Load Balancer that is in Public Subnet #1 that obeys the Web Server Security Group and in Availability Zone #1. The Elastic Load Balancer then conducts SSL termination at the Auto Scaling Group #1 that is in Private Subnet #1 (to only allow access from the Elastic Load Balancer and not directly from the outside world) that obeys the Web Server Security Group and also in Availability Zone #1. Inside the Auto Scaling Group #1 in the web server (using an auto scaling group for failover redundancy), which then communicates with the database that is in Private Subnet #2 (to only allow access from the web server and not directly from the outside world) that obeys the DB Security Group and in Availability Zone #2 (using two different availability zones for failover redundancy). Everything that comes after AWS Route 53 falls within a VPC which falls in the same Region. There is also end-to-end SSL encryption at play from the client to the database and back.

## **Design Issues**

Below are some of the issues we encountered when designing our system.

### **Functional Issues**

1. In what format should users provide feedback on the application?
  - a. Email link

**b. In-app feedback page**

We chose to provide an in-app feedback page that sends an email to an account that we will create for the application. In this way, the user is not directed out of the app in order to send feedback which tends to be disruptive and can inhibit the user experience.

**2. How should users be able to navigate between recipes, their feed, and other views?**

**a. Tabbed view**

**b. Hamburger menu**

While a tabbed view along the bottom of the screen would allow the user to be able to access everything very easily, it could become cluttered if too many tabs were added. For this reason, using a hamburger menu for navigation between views allows us the ability to add more views without adding clutter, and it also keeps unnecessary information out of sight.

**3. How should users be able to reset their password if forgotten?**

**a. Recovery questions**

**b. Email recovery**

While recovery questions are the simplest to implement, they're also the least secure. For this reason, using email recovery for resetting a password if forgotten ability to accomplish the same functionality without hurting account security, and also preventing needing to persist additional information.

## **Non-Functional Issues**

**1. Which language should we use for building an Android application?**

**a. Java**

**b. React Native**

**c. Xamarin**

We have decided to build our application using React Native because it allows us to potentially create both iOS and Android versions of the app

without having to build two completely distinct applications. While Xamarin also allows you to build iOS and Android versions of an app using C#, we chose React Native because the components that you write in Javascript interface directly with the equivalent components for iOS or Android. While developing the app using native Java and the Android SDK would also be an option, it makes it more difficult to reuse our work if we decide we ever want to build an iOS application. It also worked nicely in the previous iteration of the project, and the cost of overhauling our frontend to use a different technology does not make sense.

2. What login options should users have for the app?
  - a. Facebook
  - b. Google
  - c. Create a username and password unique to our system

While providing a variety of login options would allow us to serve the widest variety of users, we have decided to allow users to login with a username and password exclusively for our system. This would still allow us to serve everyone while simplifying the types of keys we would have to store. Providing our own login system will allow us to provide for all users including those who do not have Facebook or who like to refrain from logging in with Facebook.

3. Which database implementation should we use?
  - a. MySQL
  - b. MariaDB
  - c. Oracle SQL

When deciding which database implementation we should use, we ultimately chose MySQL. In comparison with Oracle SQL, MySQL stands out because it is open source, so is available to us to use at no cost, while still offering the features of Oracle SQL that we need. While MariaDB does still offer all of the features of MySQL plus some extra, we still chose

**MySQL in this instance because the additional features are not necessary for our purposes. It also worked nicely in the previous iteration of the project, and the cost of overhauling our system to use a different RDBMS does not make sense.**

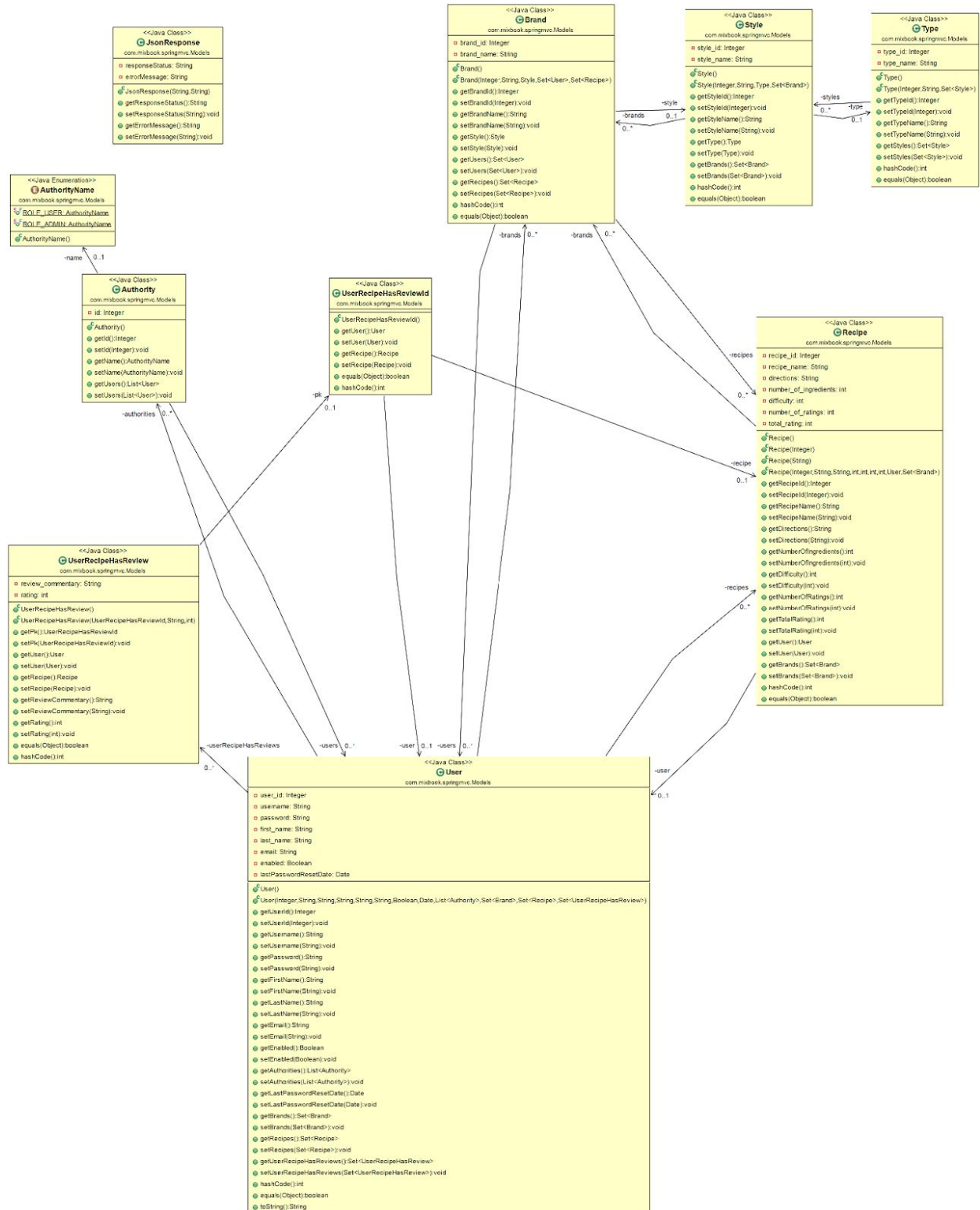
- 4. Which push notification service should we use?**
  - a. Google Firebase Cloud Messaging**
  - b. AWS SNS**

**When deciding which push notification service we should use, we ultimately chose Google Firebase Cloud Messaging. In comparison with AWS SNS, Google FCM stands out because it is available to us to use at no cost forever, while still offering the features of AWS SNS that we need.**

## **Design Details**



# Class Design



## **Description of Classes**

### **User**

- Contains all the pertinent info related to an individual user
- Significant impact on all other classes as virtually all other classes derive their ability from the User class

### **Authority/AuthorityName**

- Used to determine user roles (permissions) such as admin or standard user

### **JsonResponse**

- Used to provide custom JSON responses back from the server for a variety of requests

### **Brand/Style/Type**

- Used to provide filtering of different mixers from the broadest (Type, i.e. food, liquor, beer, wine, etc.) to the narrowest (Brand, i.e. Jack Daniels Old No. 7)
- Significant impact on recipes and users as it is used to determine what recipes a user can make as well as indicate what is needed in a recipe

### **Recipe**

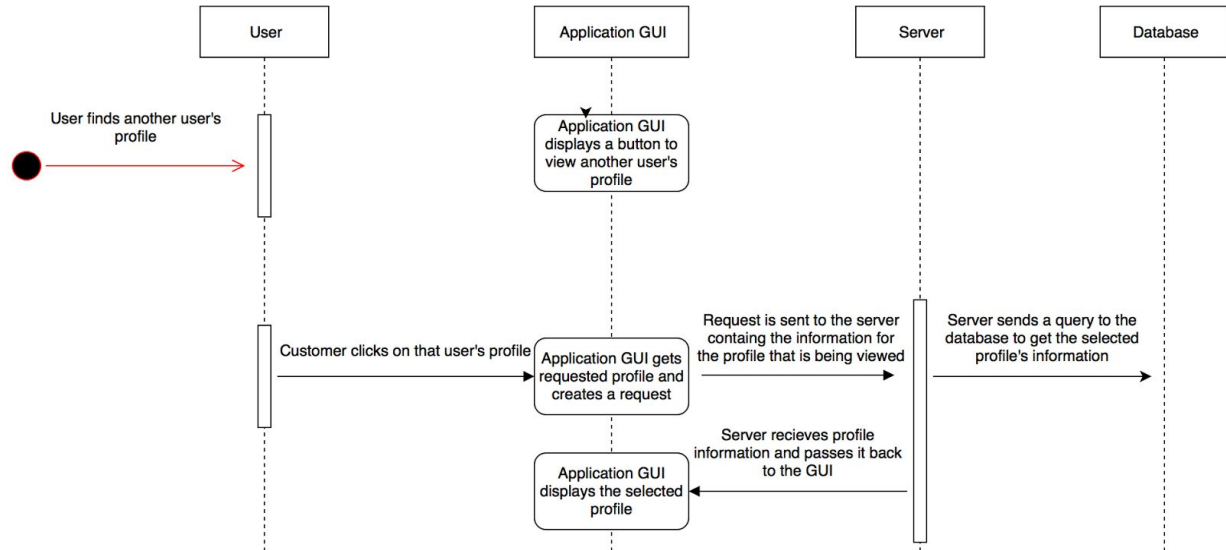
- Contains all of the pertinent information about a recipe (outside of the ingredients themselves)
- Used as the base for all recipe interactions

### **UserRecipeHasReview/UserRecipeHasReviewId**

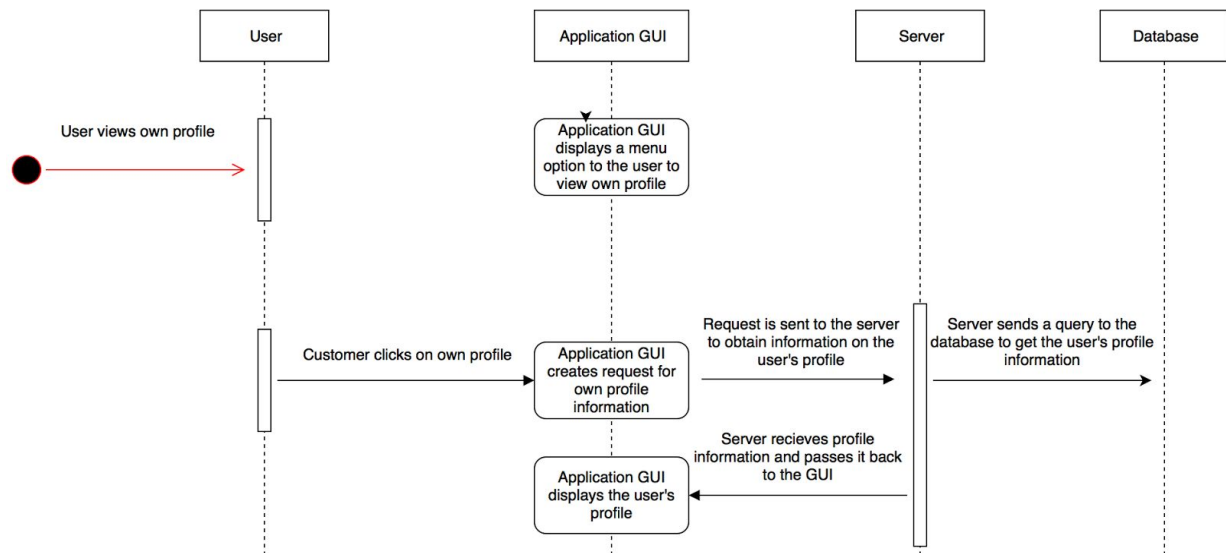
- Contains all of the pertinent information about recipe reviews
- Used to link recipe reviews to users and the recipes respectively
- Serves as a base for the social interaction of the application

# Event Sequence Diagrams

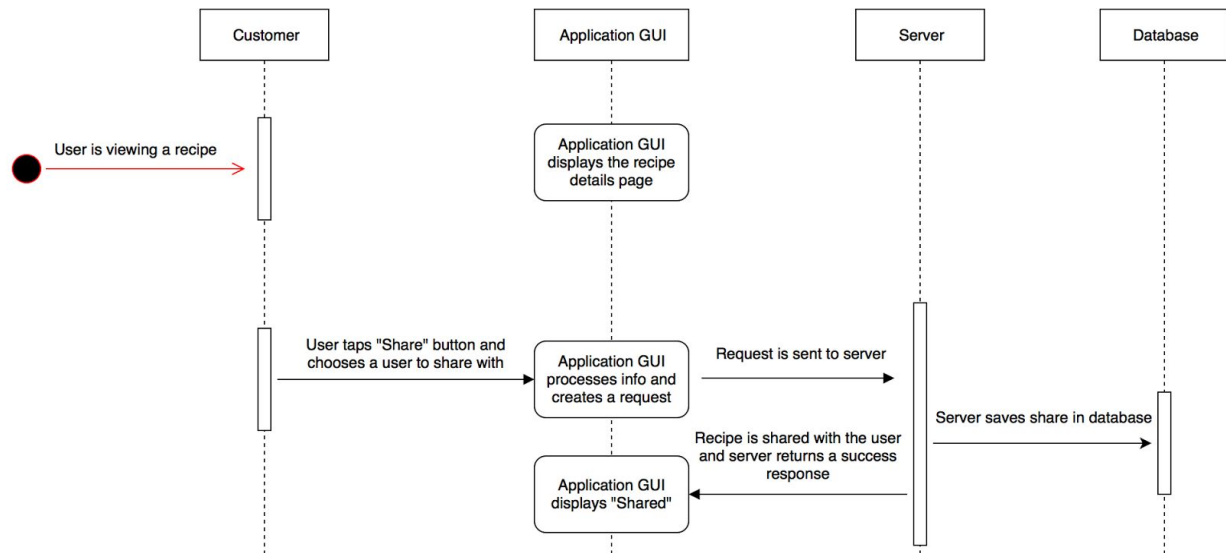
## Viewing other user's profiles



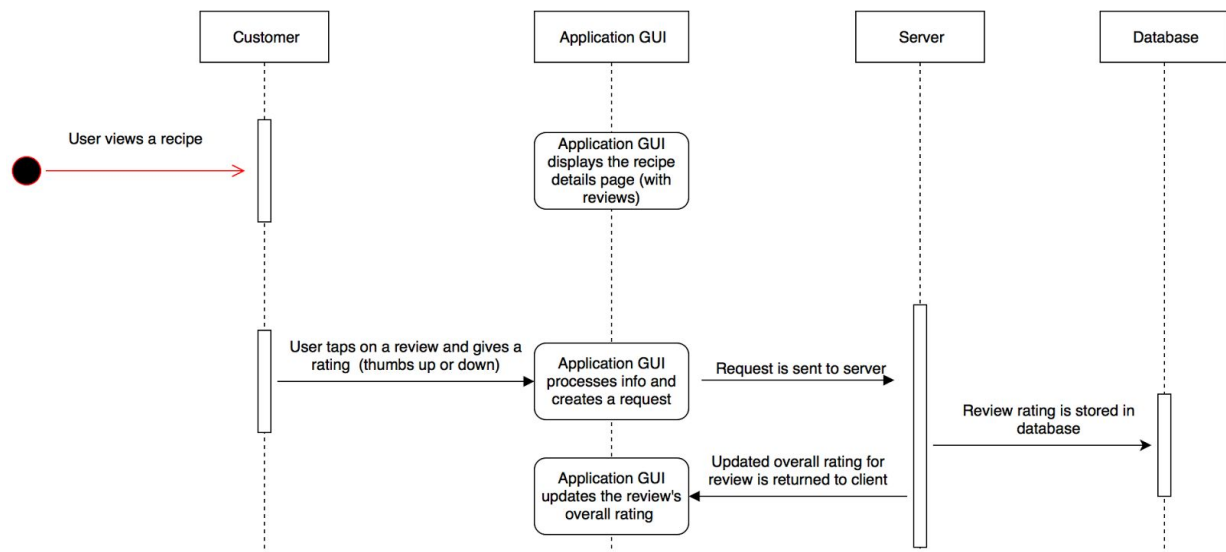
## Viewing user's own profiles



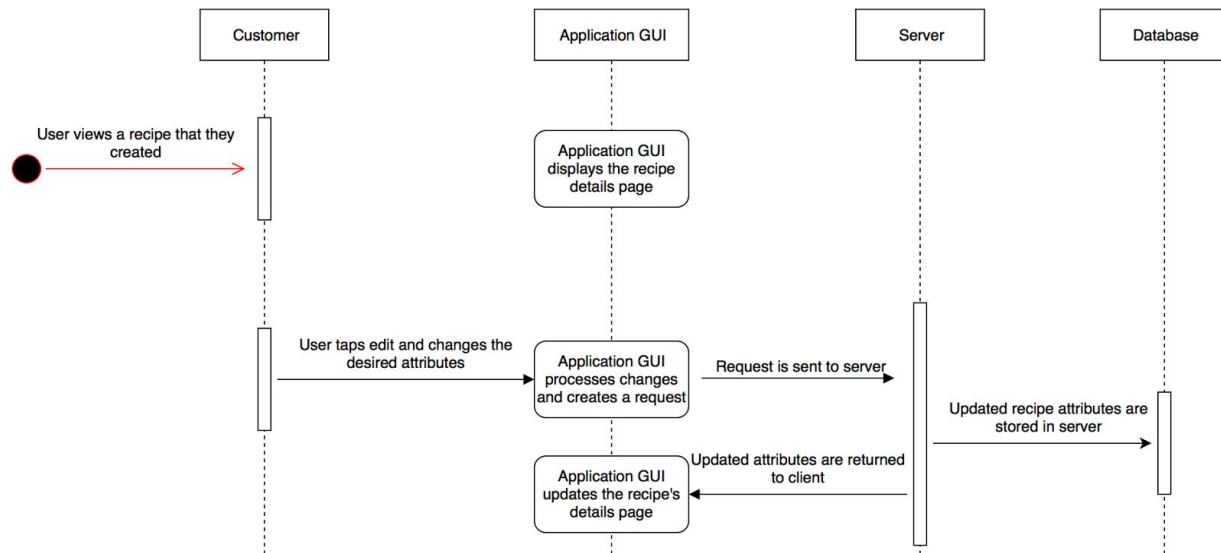
## User sharing a recipe



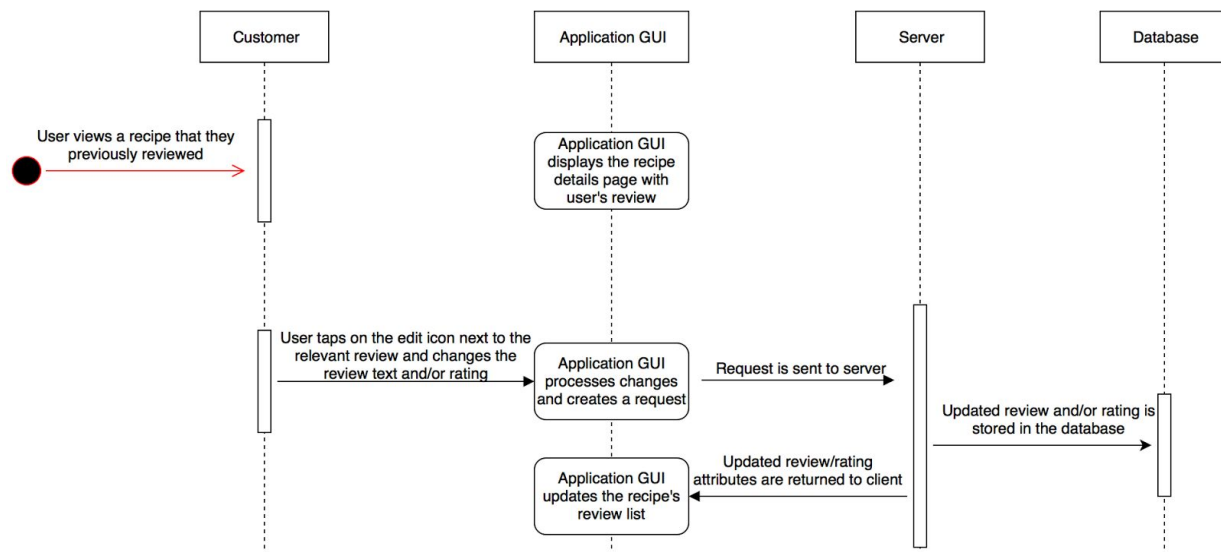
## Rating another user's review of a recipe



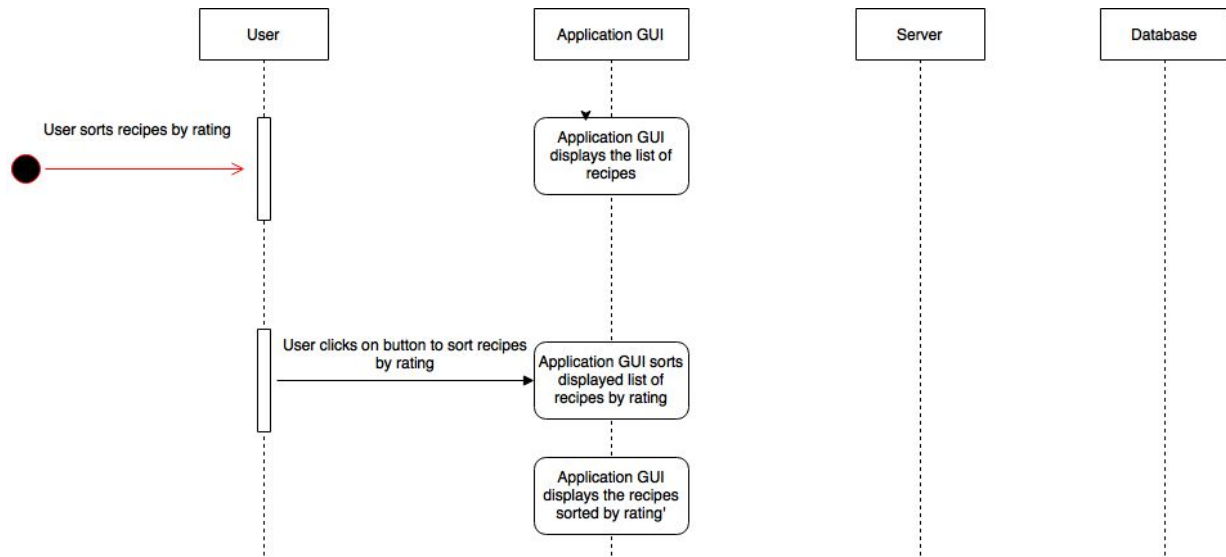
## Editing a user's own recipe



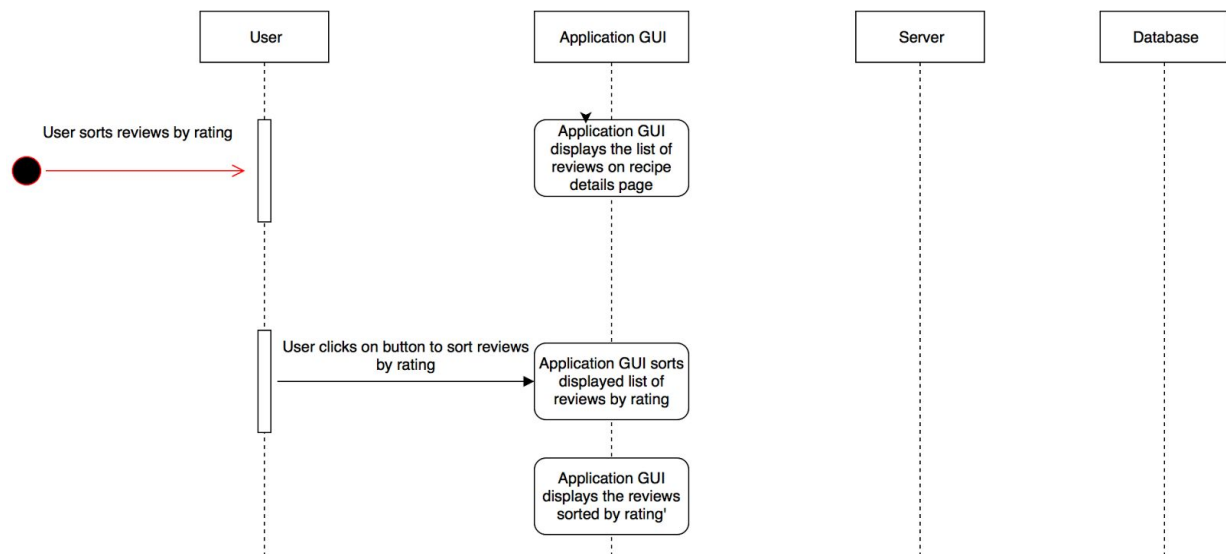
## Editing a user's review of a recipe



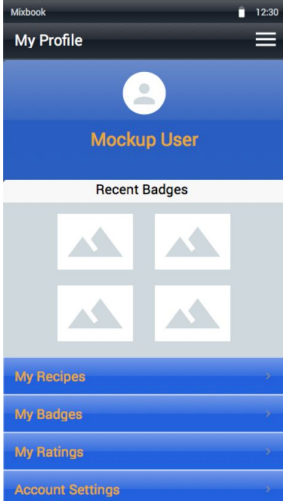


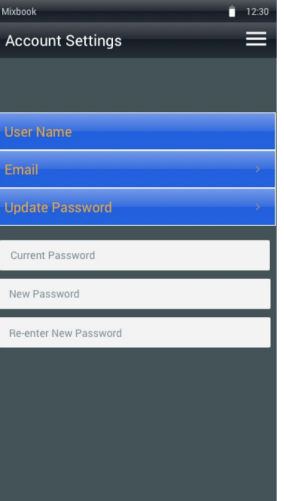

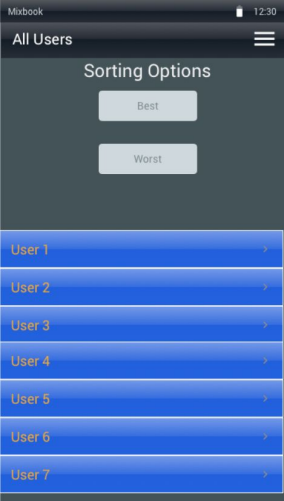
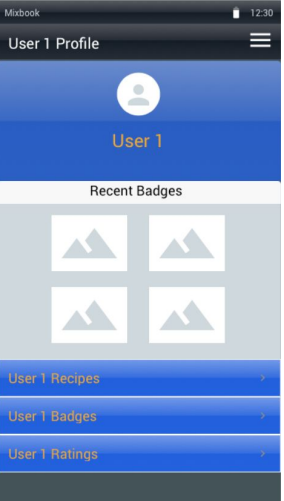
## Sort recipes by ratings



## Sort reviews by ratings



# UI Mockups

<b>My Profile Page</b> 	<b>My Recipes</b> 	<b>Edit One of My Recipes</b> 	<b>My Account Settings</b> 
<b>View All Recipes</b> 	<b>Sort Users by Rating</b> 	<b>See Other Users Profile</b> 	<b>Rate Reviews</b> 