School of Computing

DIPLOMA IN INFORMATION TECHNOLOGY

Go Lunch Marathon

Assignment: Phase 2 Report

Class: DIT/FT/3A/89

Lecturer: Mr. Chong Chee Seng

Date: 5 August 2019

Team Members:

Tan Khang Hou 1618493

Ng Zheng Ning 1753383

J Shafeer Deen 1749212

Sherman Wong 1749085

Hau Sian Mung 1745755

Table of Contents

Project Overview **2**

Technology Stack **3**

Front-end 3

ReactJS written in TypeScript. 3

Backend 3

Express 3

Database 4

MySQL 4

Challenges Faced **4**

Implementation Schedule **4**

User Guide **6**

User 6

Registering 6

Then click register. You should be redirected to the login page with a success notification. However, if you use an email that's already registered with the application, then you will receive an error notification 6

Logging in 7

Tracking 7

Redeeming Rewards 11

Admin 15

Logging In 15

Accessing the admin portal 17

Adding Restaurants 18

Editing Restaurants 24

Deleting Restaurants 25

Searching Restaurants 26

General Configuration 27

Changing the menu 27

Creating new admins 29

Setting Up **30**

Install NodeJS and npm 30

Install MySQL Server 8.0 30

Install MySQL Workbench 8.0 31

Setting up the database 31

Deploying **31**

Deploying Server 31

API Documentation **32**
Auth.js 32
Register 32
Login 32
Tracking.js 33
Get Time 33
Scan QR 33
Shops.js 33
Get Shops 33
Get Promoted Shops 33
Get Shop 33
Add Shop 34
Edit Shop 34
Delete Shop 35
ShopRewards.js 35
Get Shop Rewards by Shop ID 35
UserRewards.js 35
Get User Rewards by User ID 35
Add User Reward 36
Claim Reward 36
ER Diagram **37**
Minutes of Meetings **38**
Test Cases **38**

Project Overview

Go Lunch Marathon is a tracking application developed for **kupittaan paviljonki**, a restaurant based in Finland. This was a project in joint collaboration with **theFIRMA**. Our project is a progressive web application that aims to promote healthy living while also improving business.

The application uses a point-to-point system to track the user's distance, and uses distance tracked to trade in for rewards such as free drinks or appetizers. Any distance that the user accumulated throughout the day must be saved by scanning a QR code that can be found in any affiliated restaurants. Only through this way will the distance be saved into the database

The distance accumulated can be used to redeem coupons from restaurants. For example, currently there are 10km, 22km, and 42km distance milestones, which will get you a free drink, a free appetizer and a free lunch respectively. Simply show the restaurant staff that you have the coupon and have claimed it to receive your reward.

Technology Stack

Front-end

React

ReactJS written in TypeScript.

React is a front-end progresssive web application framework written in JavaScript. React makes it easy to maintain and reuse code, allowing for smooth expansion & integration of additional features. It also has a vibrant ecosystem of packages which can be used to perform a large variety of functions, from client-side routing to http request handling.

Main packages used:

- React Router - client-side routing
- JWT - user authentication
- Formik - form handling
- axios - HTTP requests
- redux & react-redux - global state management (cleaner code)
- date-fns - date formatting and parsing
- framer-motion - animations
- lodash - complex array handling library

Backend

Express

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. Express is performant as it provides a thin layer of fundamental web application features such as routing, leaving the rest completely up to the developer.
Main packages used:

- mysql - connects express and mysql together
- bcrypt - password hashing
- body-parser - allows json and url-encoded form data to be sent to server
- dotenv - store sensitive information in a file that's untracked by git
- compression - compresses response bodies so that server can process requests quicker

Database

MySQL

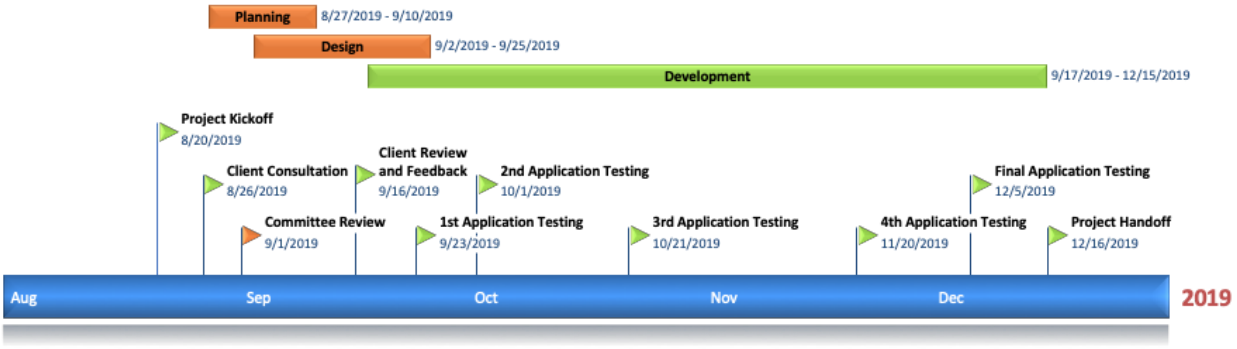
MySQL is one of the most popular relational database management systems out there. It is a time-tested and can run on PC, Mac and Linux. We used MySQL because it is cheap & simple to set-up, and we don't expect the application receive large amounts of information at a time.

Challenges Faced

Throughout the development of Go Lunch Marathon, our team encountered a few problems, though they were nothing we couldn't fix. However, a major problem in particular was the tracking system. We found it to be perhaps one of the most complex features we had to implement. Getting the location via navigator.geolocation was not very reliable as sometimes

it would just time out, and we would need to wait a minute or two before it finally returns us the location. We tried looking at other options, but we eventually stuck back with navigator.geolocation as we were running out of time.

Implementation Schedule



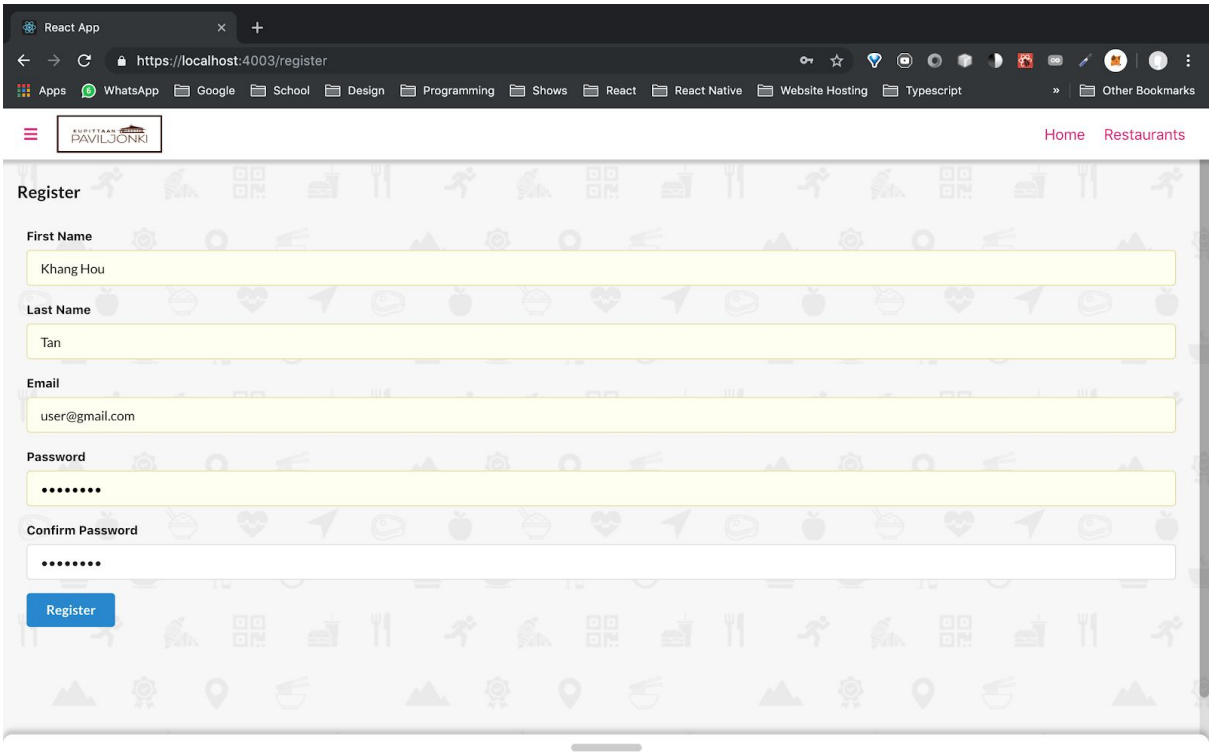
User Guide

User

Registering

At the login page, click on the bottom link: **Don't have an account? Click here.**
This will lead you to the register page. This is a standard register page with functions that you would expect from one.

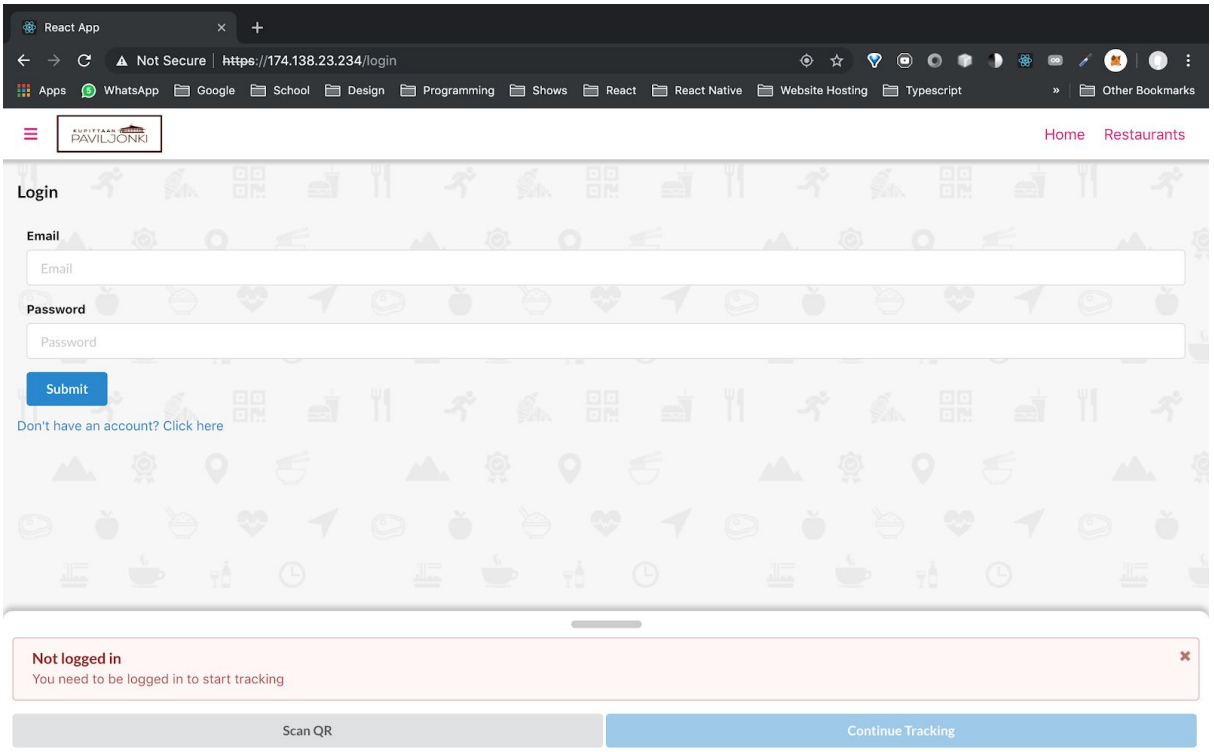
Fields:
First Name - this is required because a name has to be displayed in the sidebar menu
Last Name (optional)
Email
Password
Confirm Password



Then click register. You should be redirected to the login page with a success notification. However, if you use an email that's already registered with the application, then you will receive an error notification

Logging in

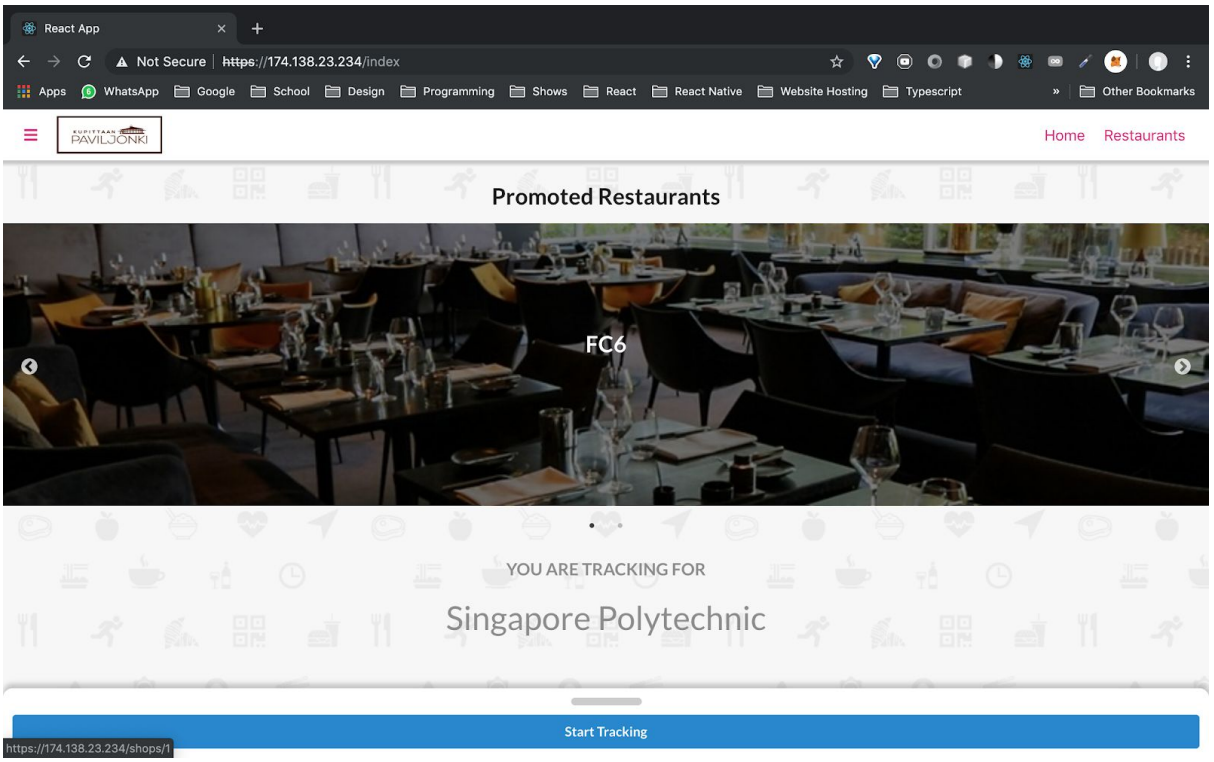
Username: user@gmail.com
Password: password



Tracking

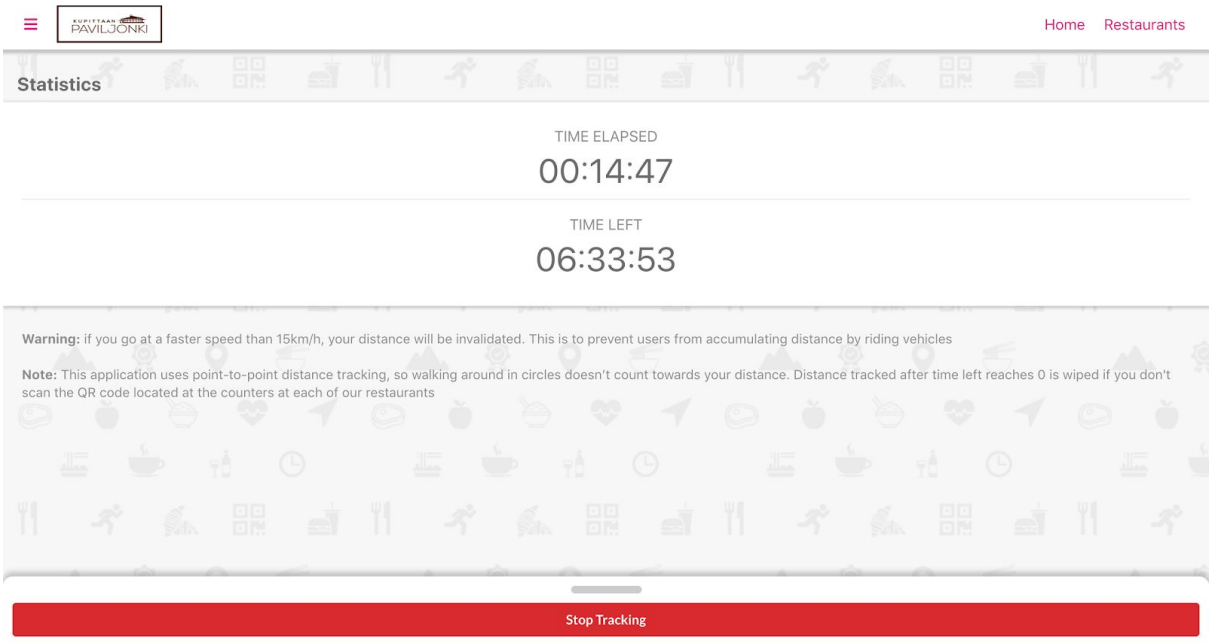
Click on the blue Start Tracking Button at the bottom. This might *occasionally* time out. Just wait for some time before you can track.

Additionally, you will need to enable location services before you can track. It will ask you for location permissions when you click on the button

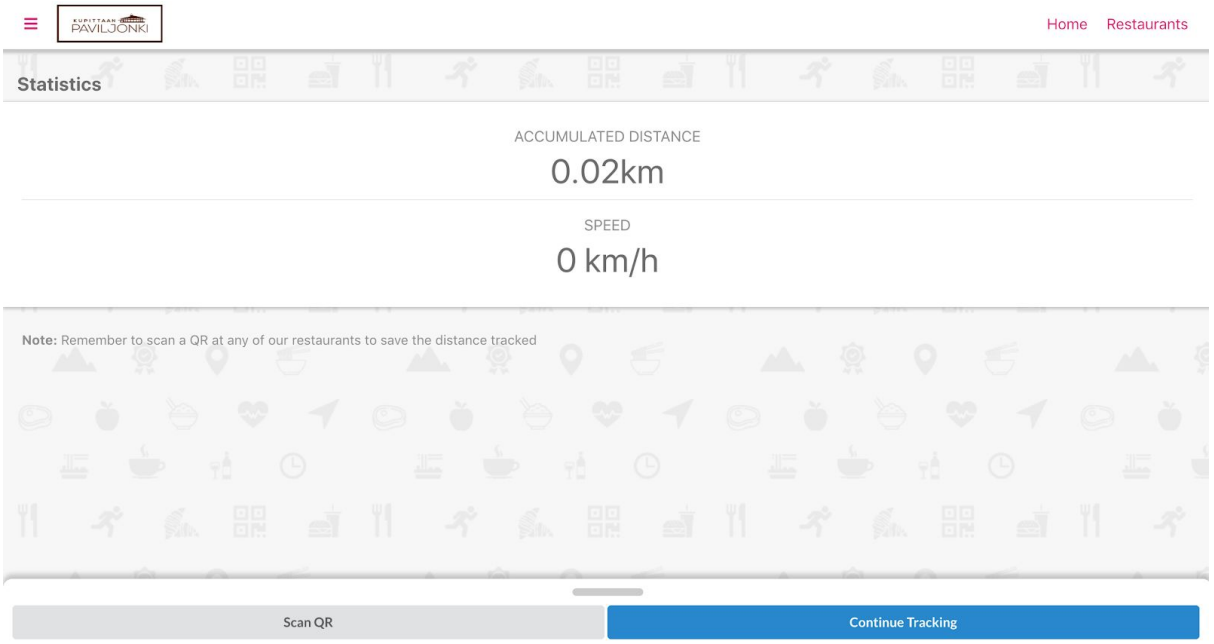


This page uses server time, and calculates the time elapsed since the user started tracking, and the time left before the tracking period runs out. Any distance tracked after the time left reaches 0 is wiped. The user needs to scan a QR code to save the distance tracked.

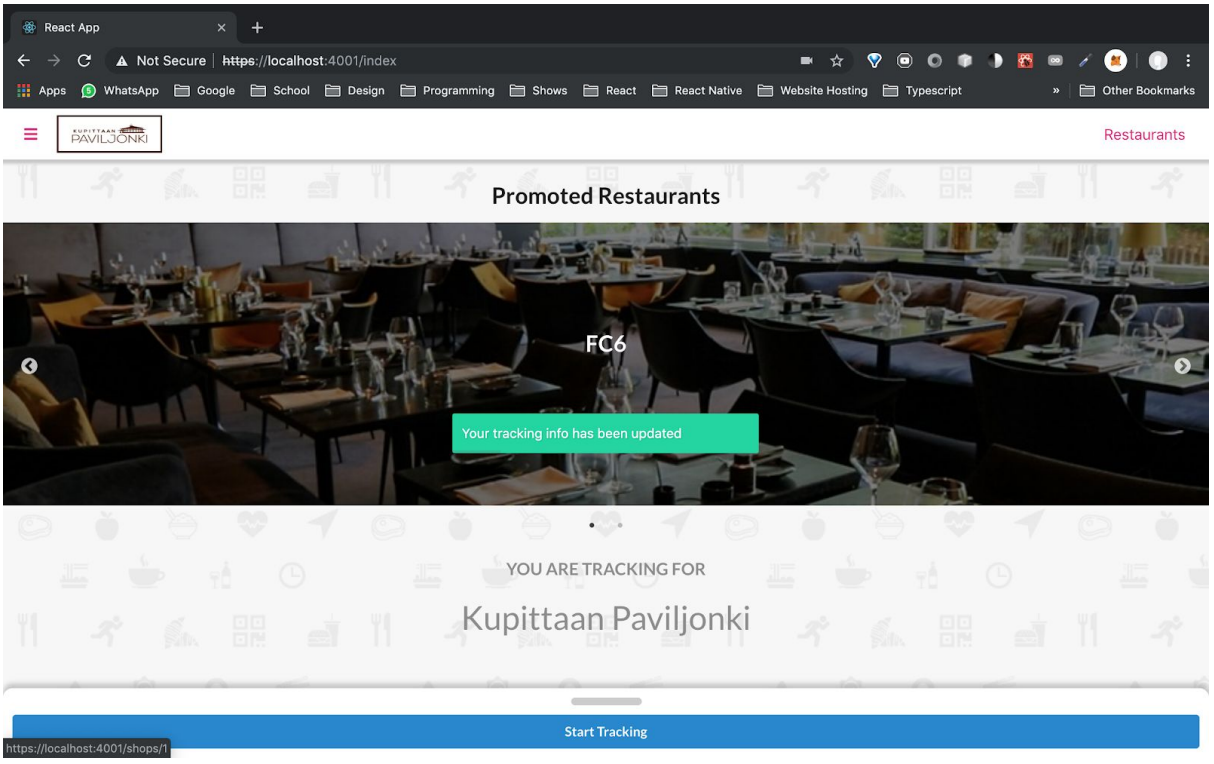
Note: if the time left reaches 0, automatically stop and invalidate the tracked distance



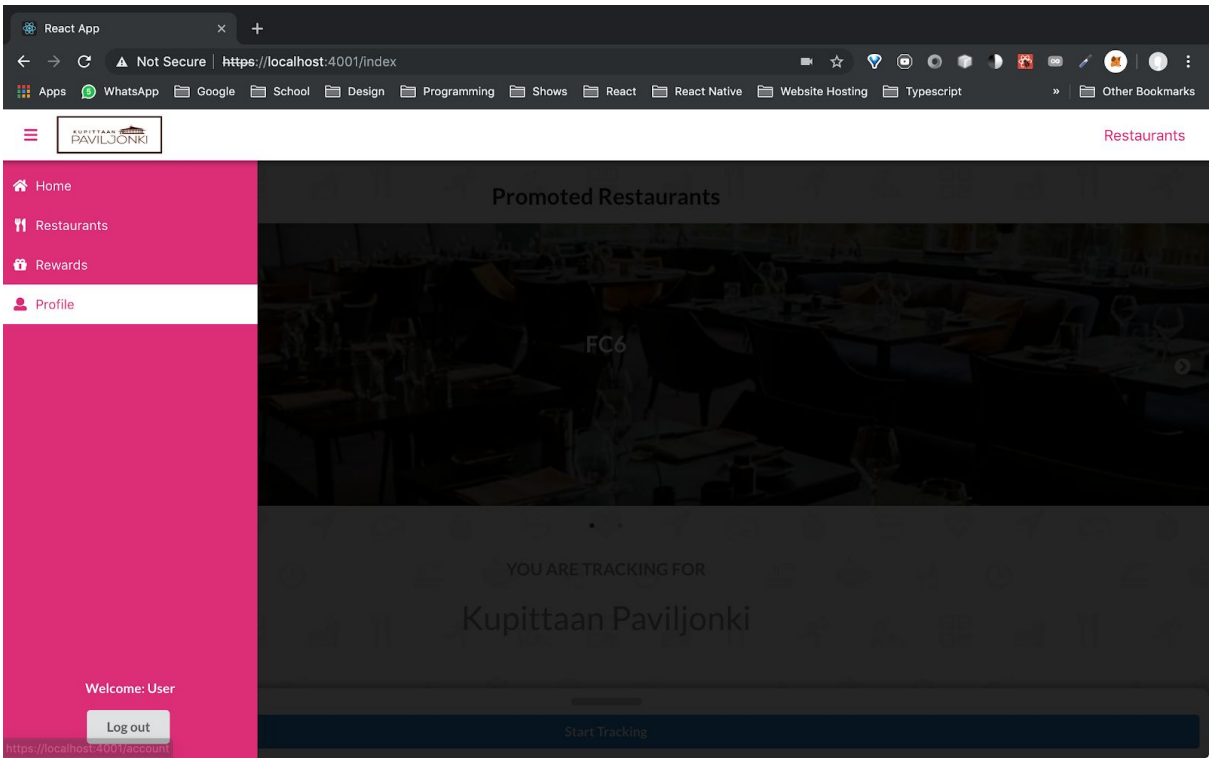
After the user stops tracking, it will display the distance travelled and speed. If the speed is above 15km/h, it will invalidate his tracked distance. This speed limit is configurable in config.json



Press the scan QR button at the bottom to collect the distance. Allow the website to access your camera. Scan a QR Code from the restaurant and the distance will be saved into the user's database record. This QR code will be generated from the admin portal

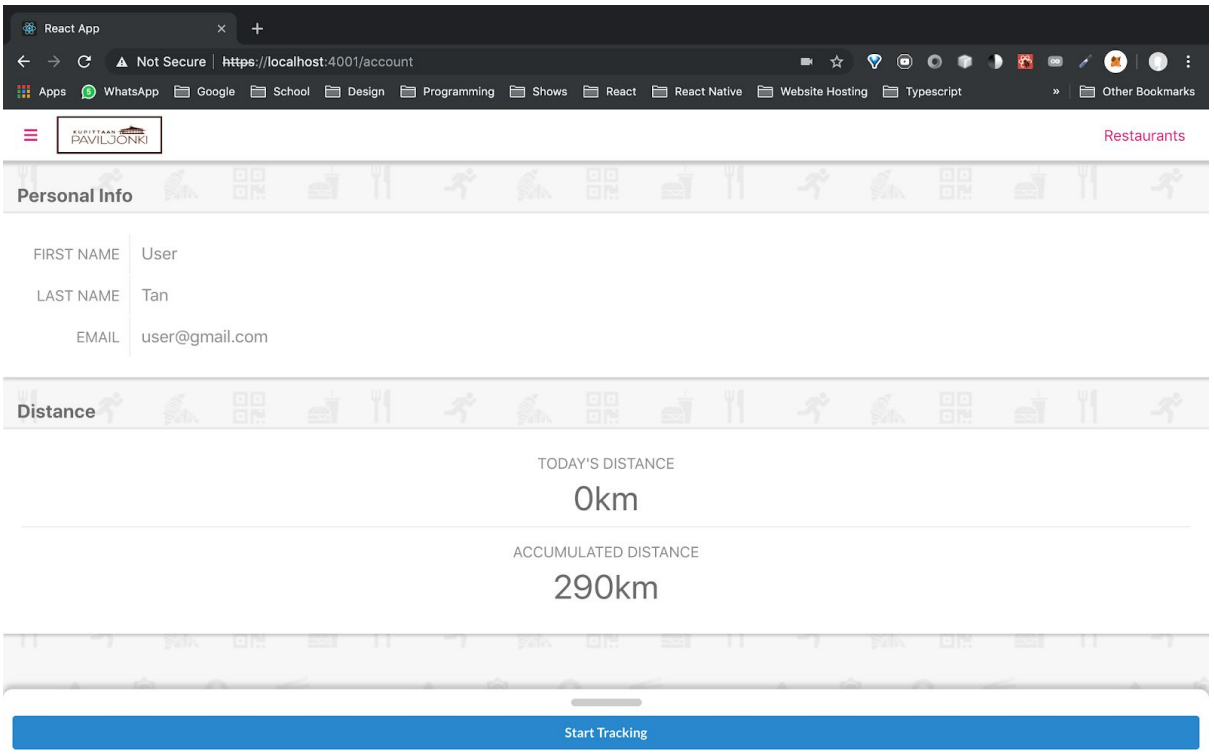


You can check the distance you walked in your profile page. Access the profile page by clicking on the top-left menu and clicking on Profile.



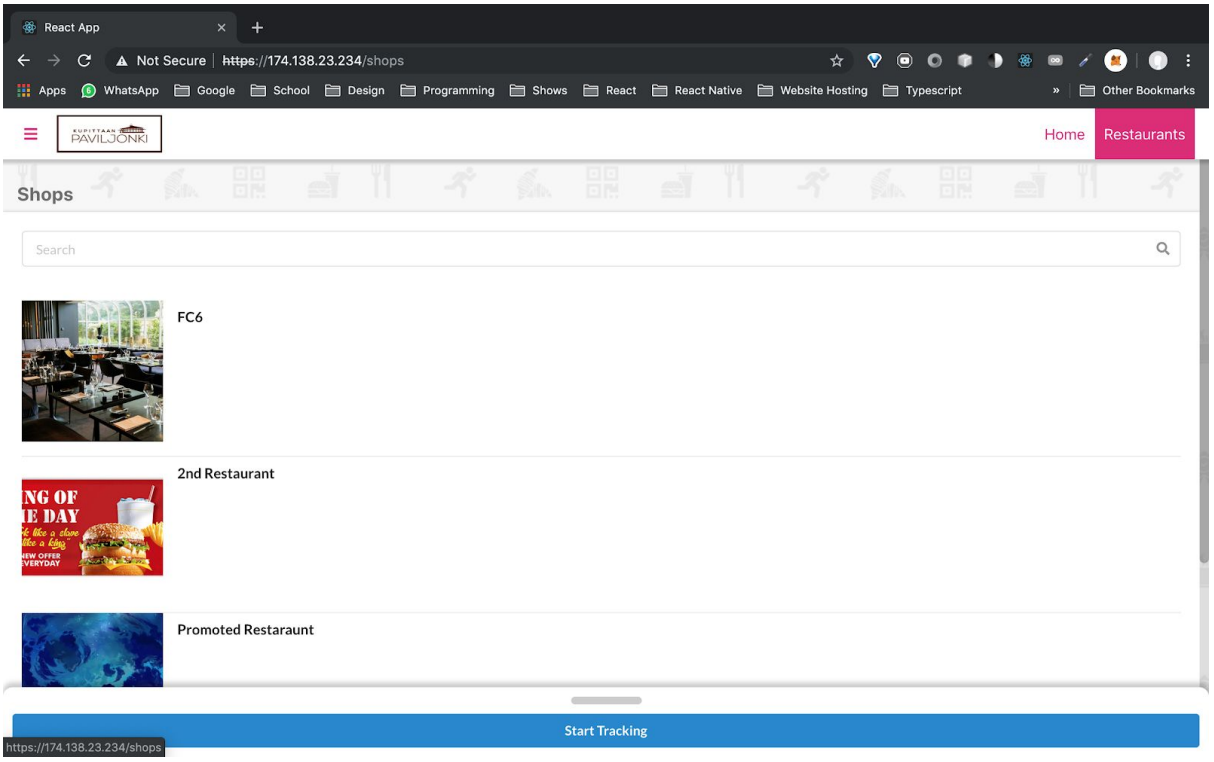
Today's distance refers to the distance that has been tracked, but not saved to the database.

Accumulated distance is the distance you have accumulated thus far, and can be thought of as your balance. Redeeming rewards will subtract from this amount.

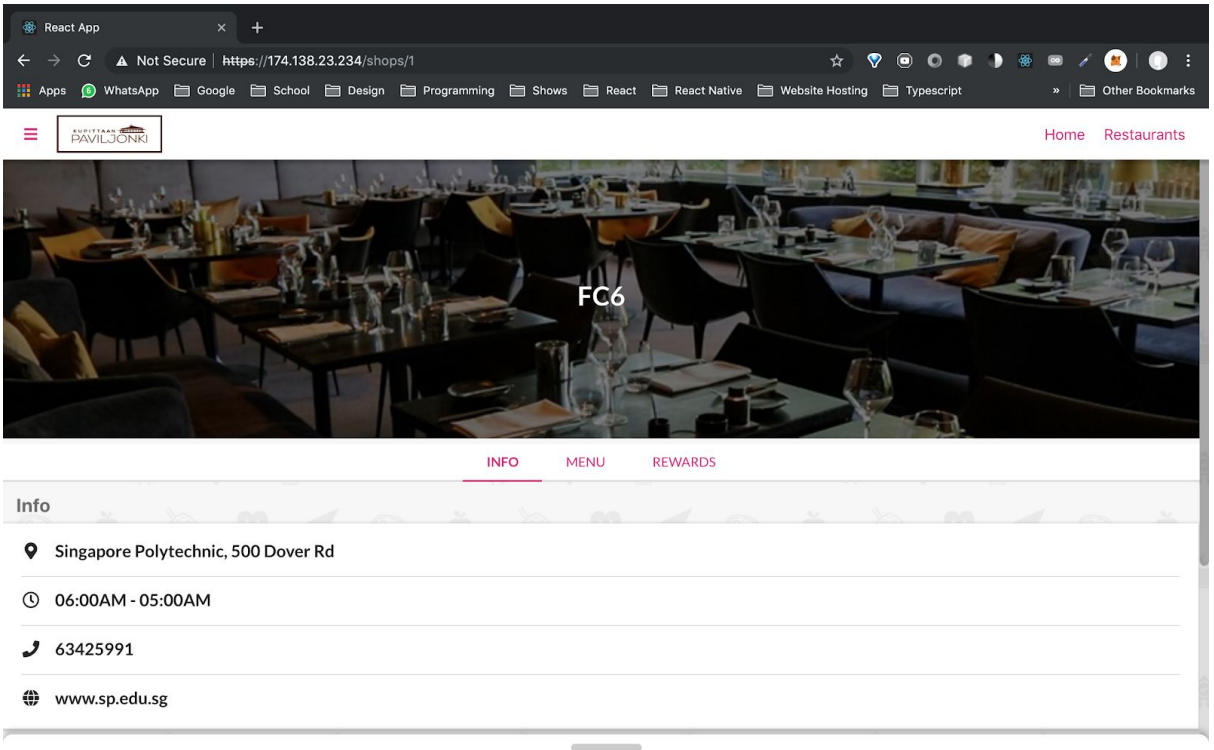


Redeeming Rewards

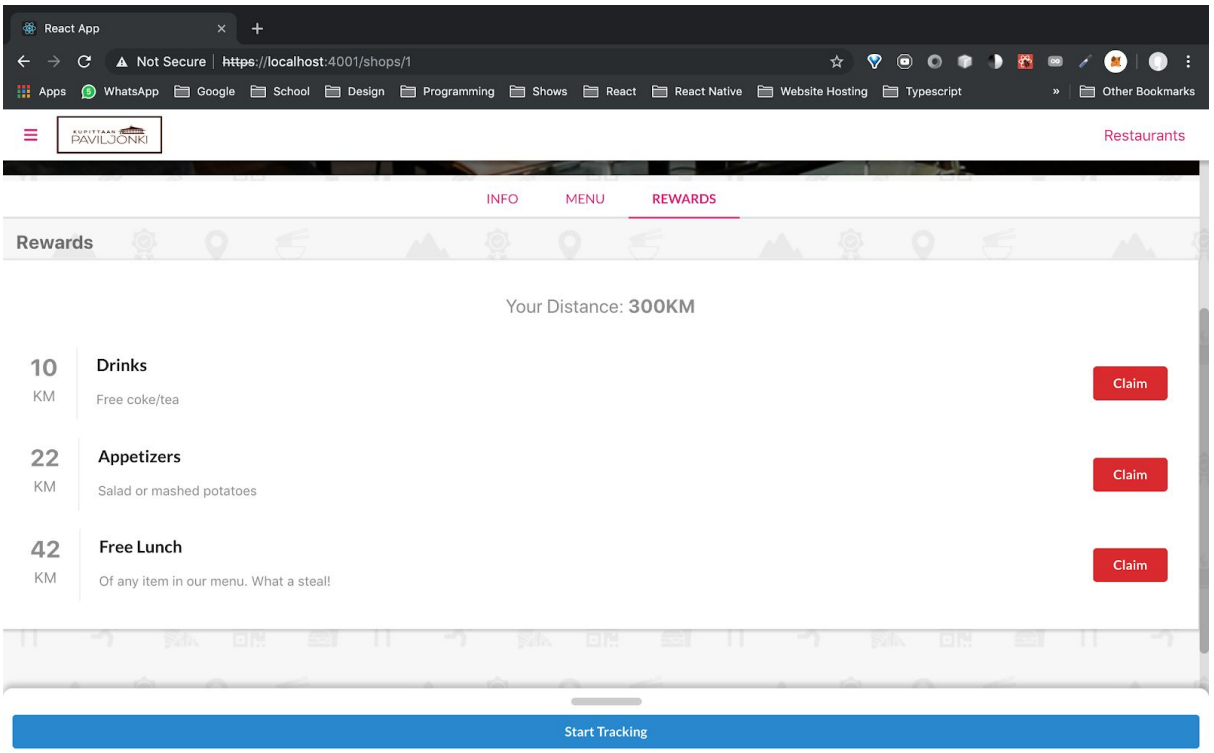
Click on the restaurants link at the top right. Then click on the restaurant you want to redeem a reward from



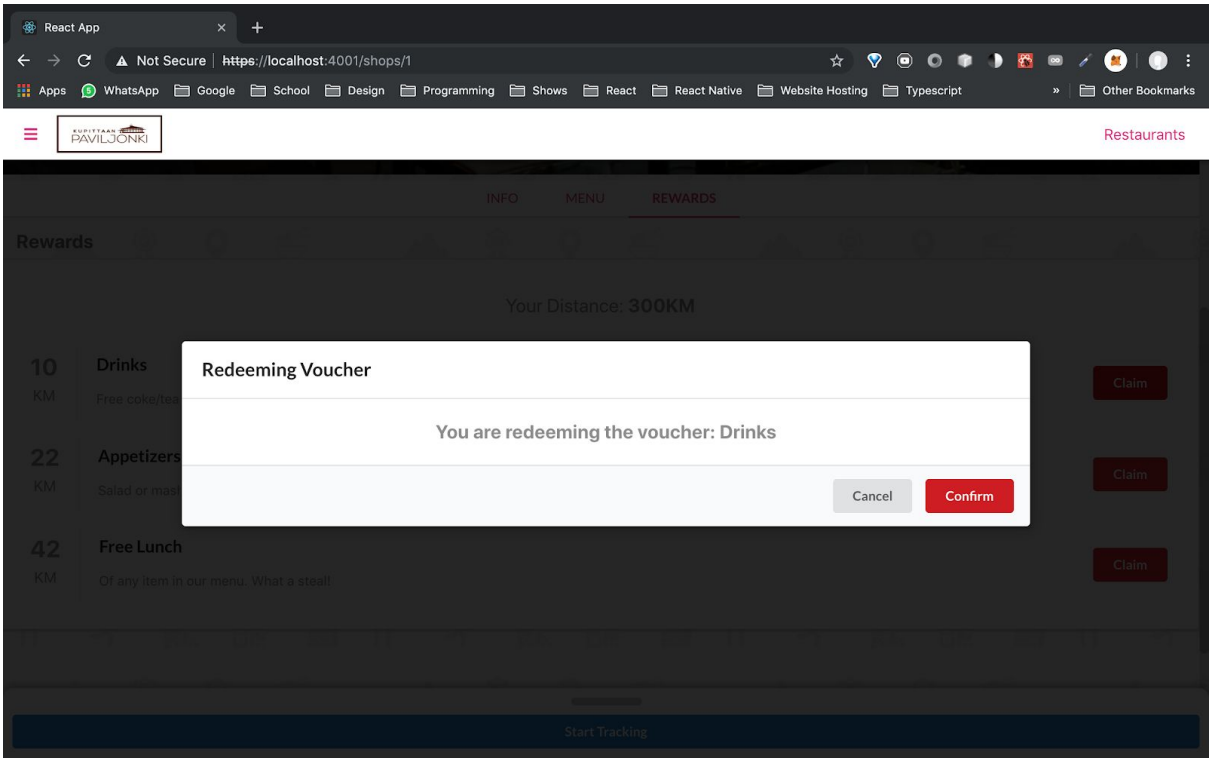
Click on the rewards tab

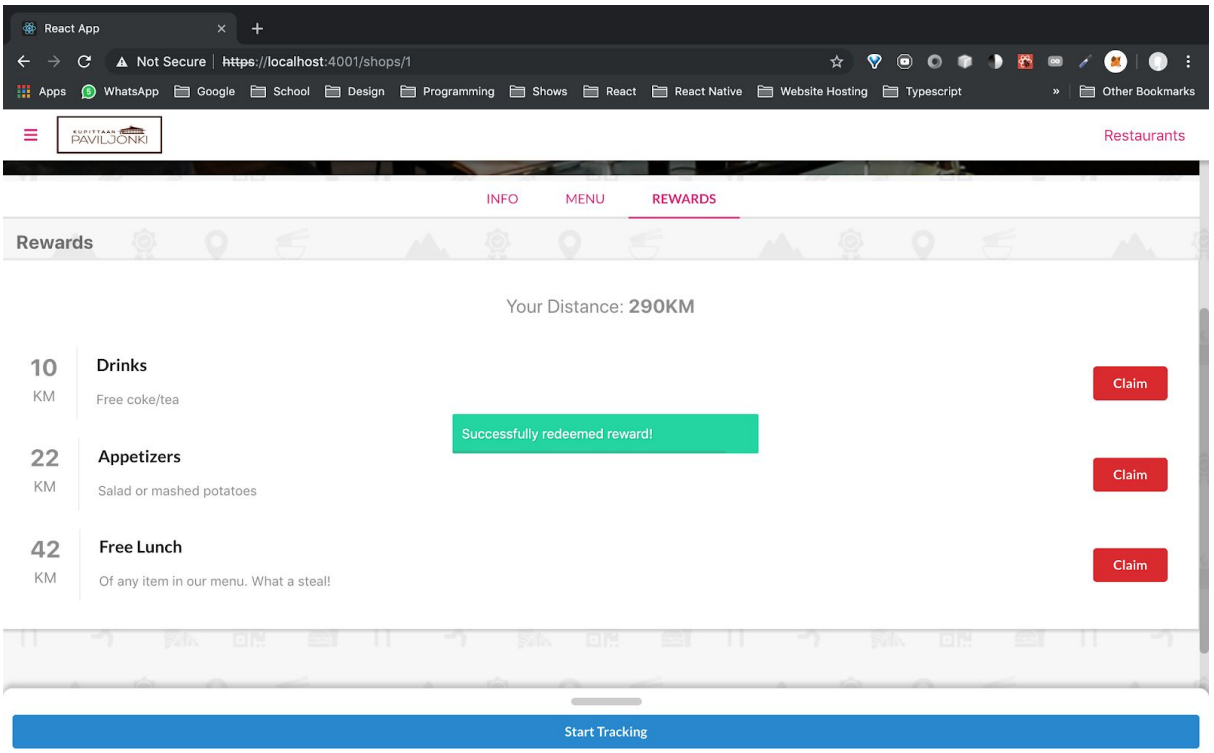


Click on the Claim button for the reward you want to redeem..

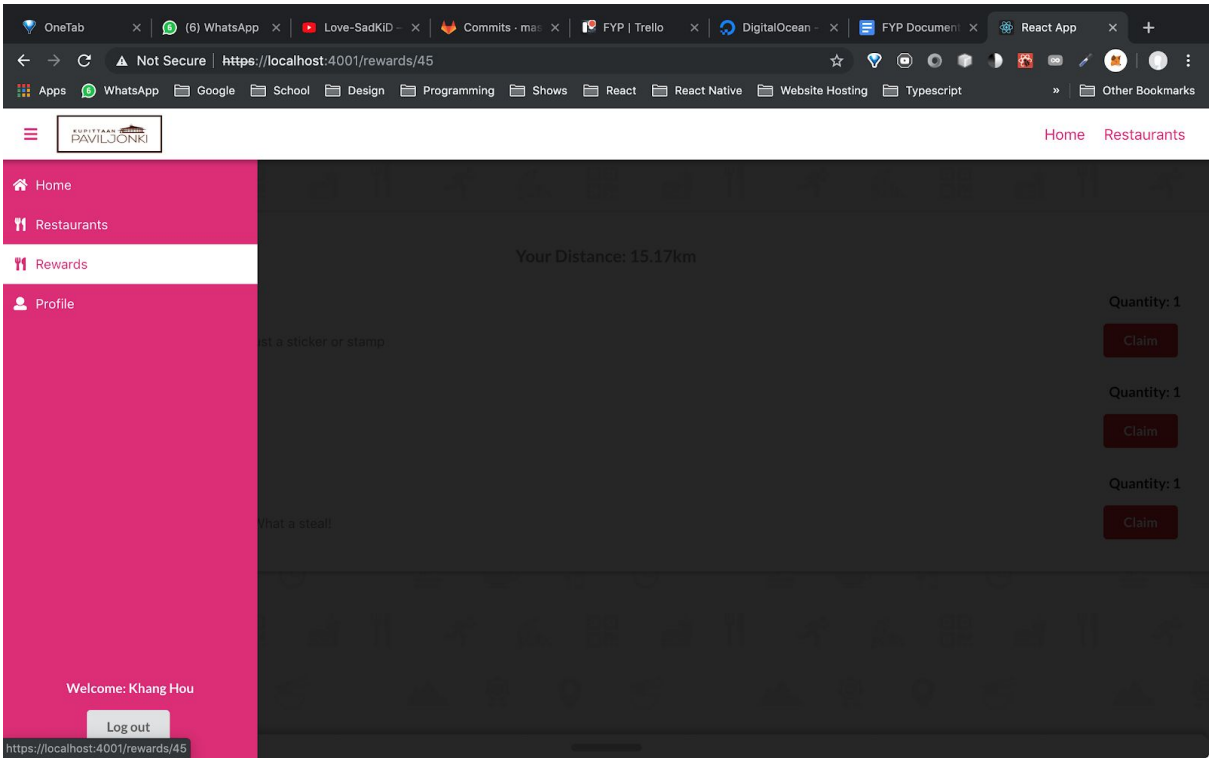


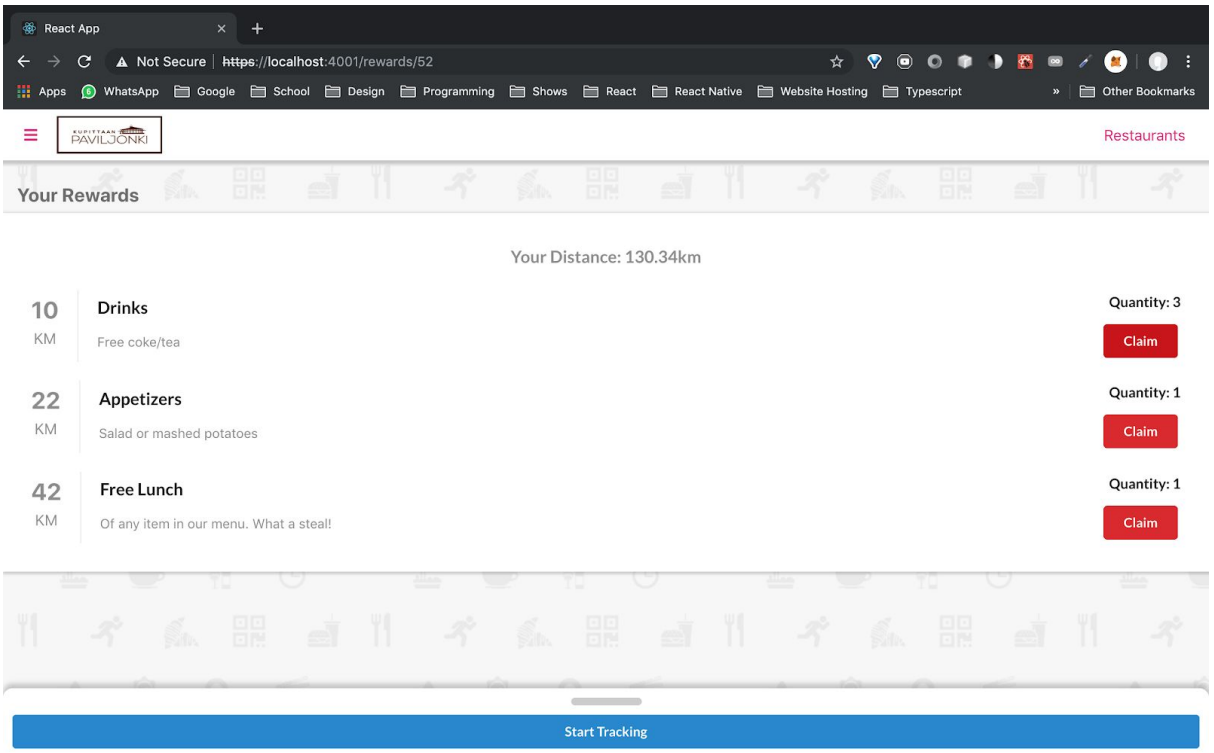
Click on confirm



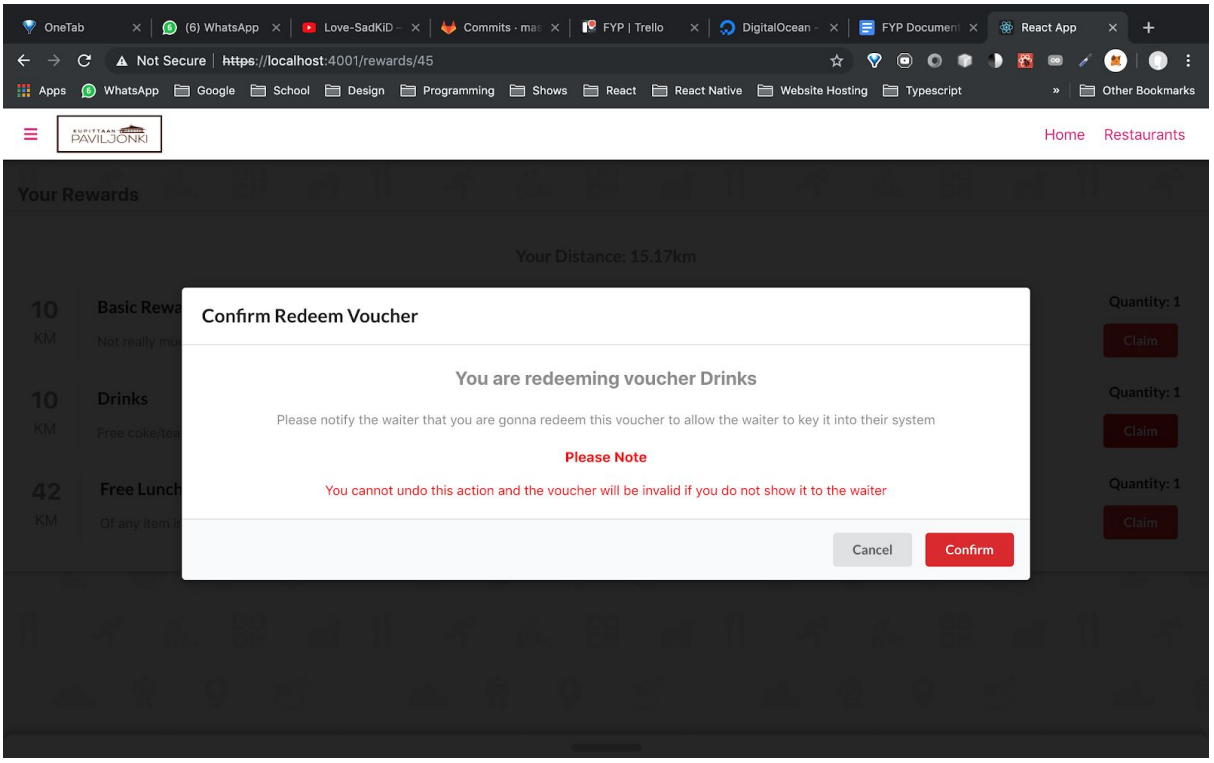


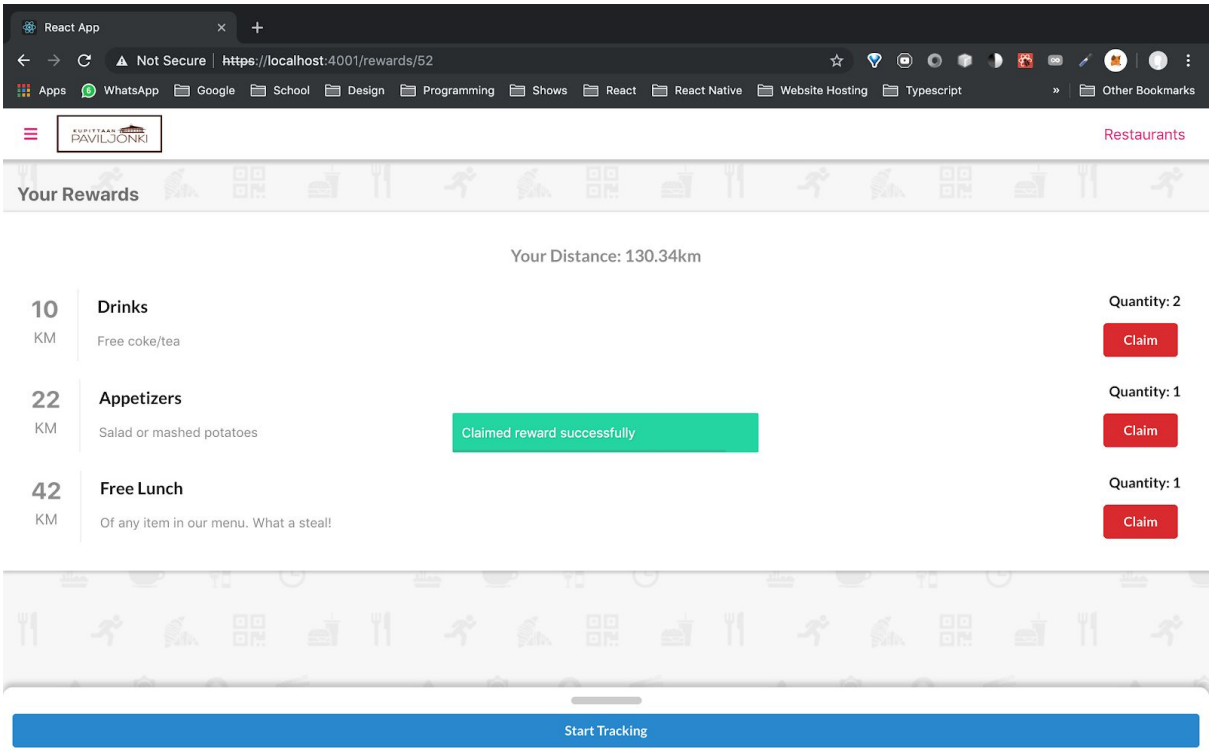
Afterwards, click on the top-left corner menu icon and click on the Rewards tab.





Then claim the reward you wish to claim. Be sure to show the claim dialog to the restaurant staff before claiming it. Afterwards, it will be removed from your inventory permanently.

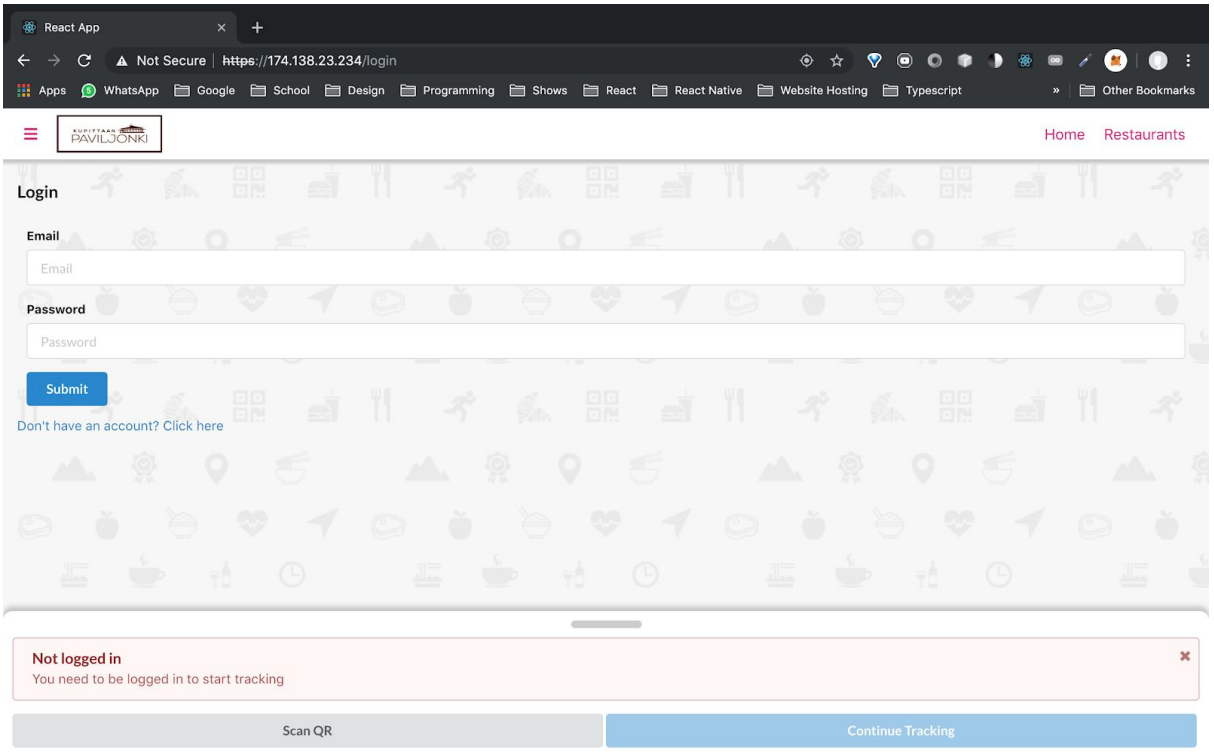


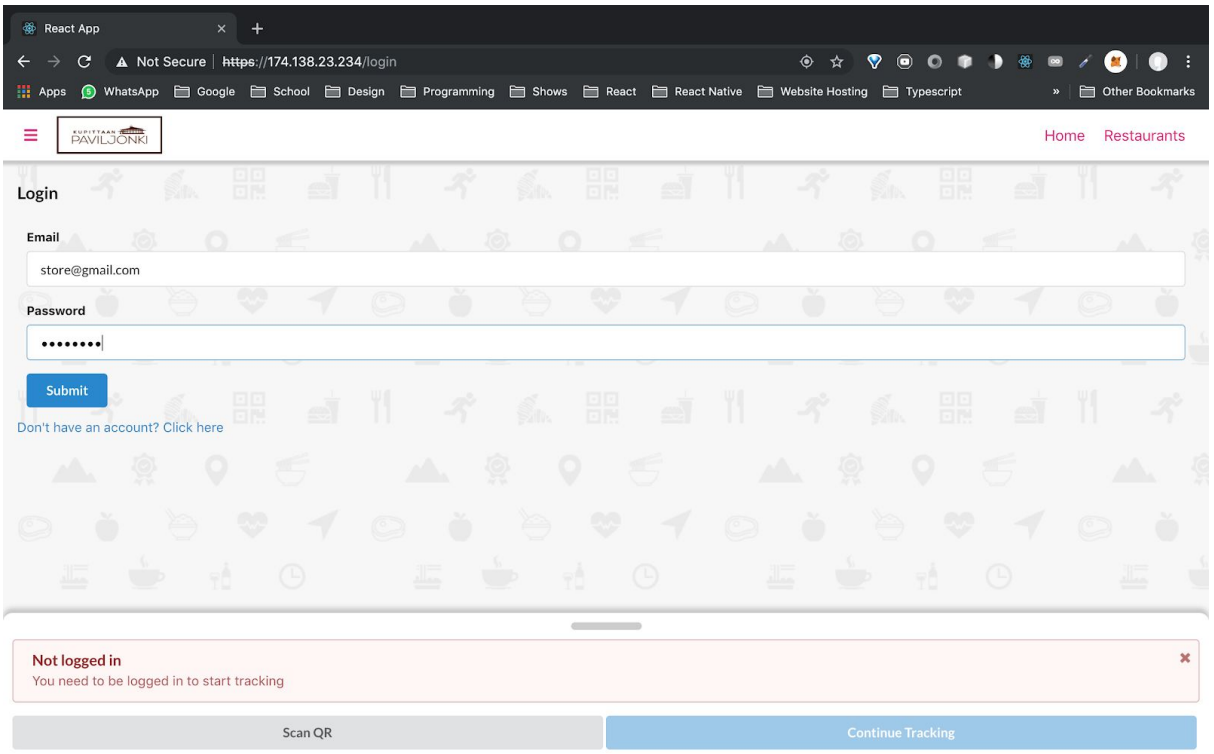


Admin

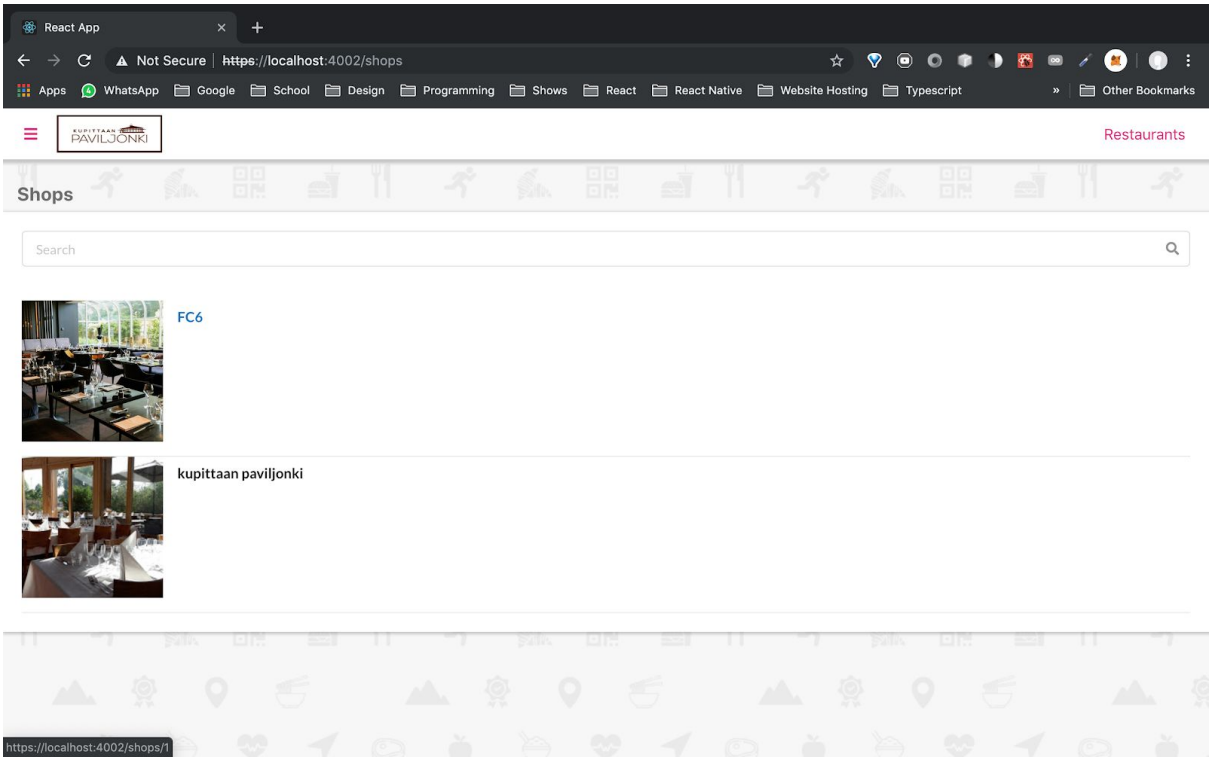
Logging In

Login as a store user
Username: store@gmail.com
Password: password



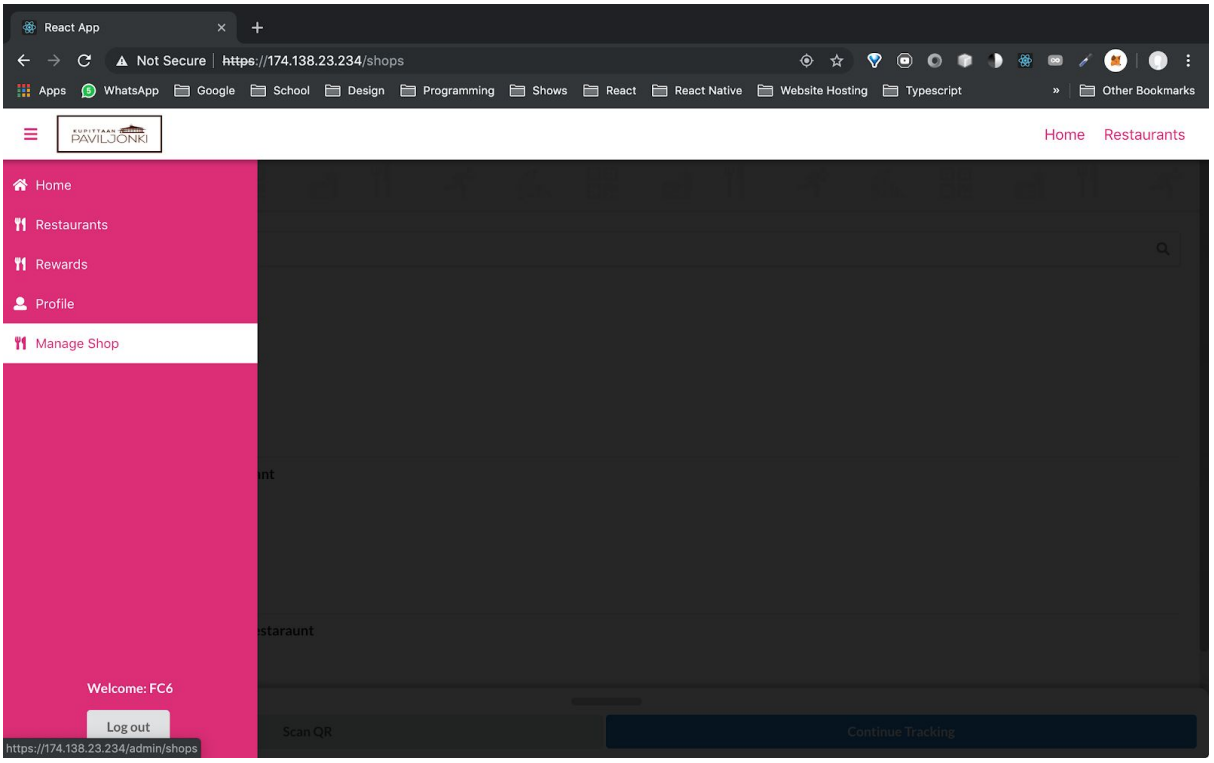


Click on submit, and you should be redirected to restaurants page



Accessing the admin portal

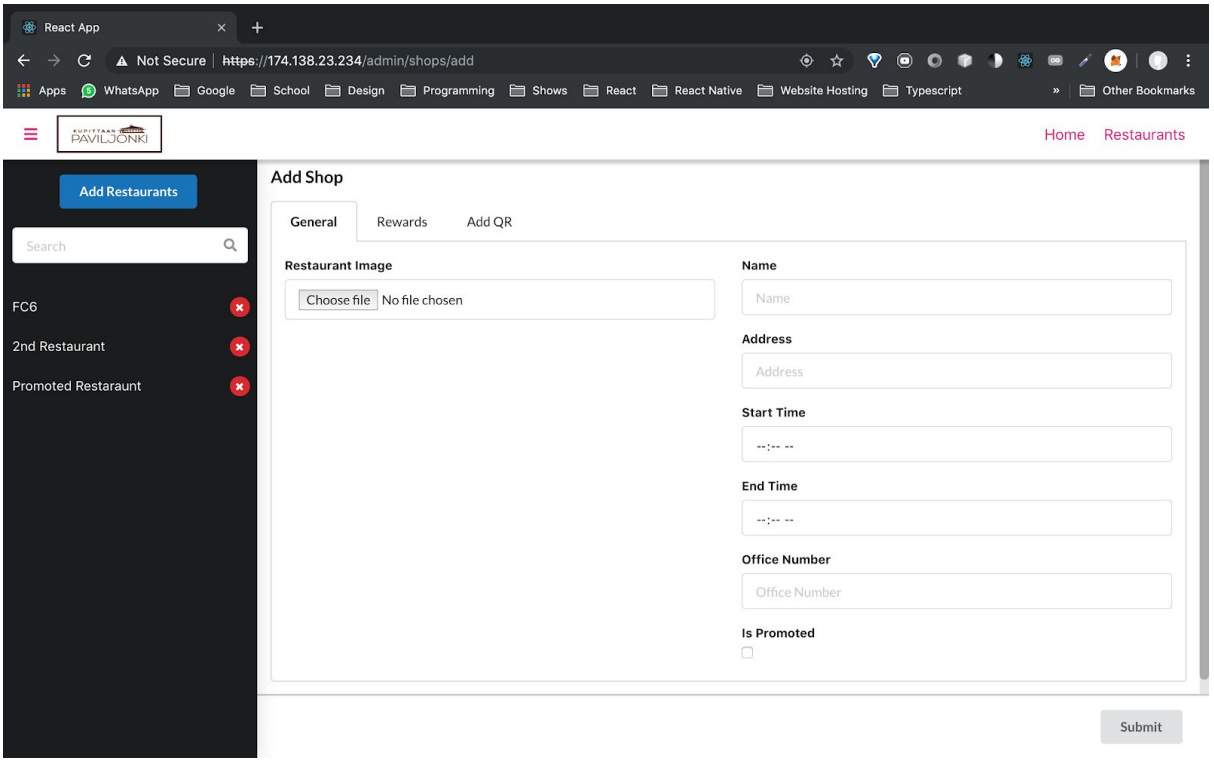
Click on the menu icon on the top-left corner and click on the manage shops link



Adding Restaurants

Click on the blue button named Add Restaurants. It will lead you to the add shop page. There are three tabs near the top of the form. Take note of these tabs for later steps.

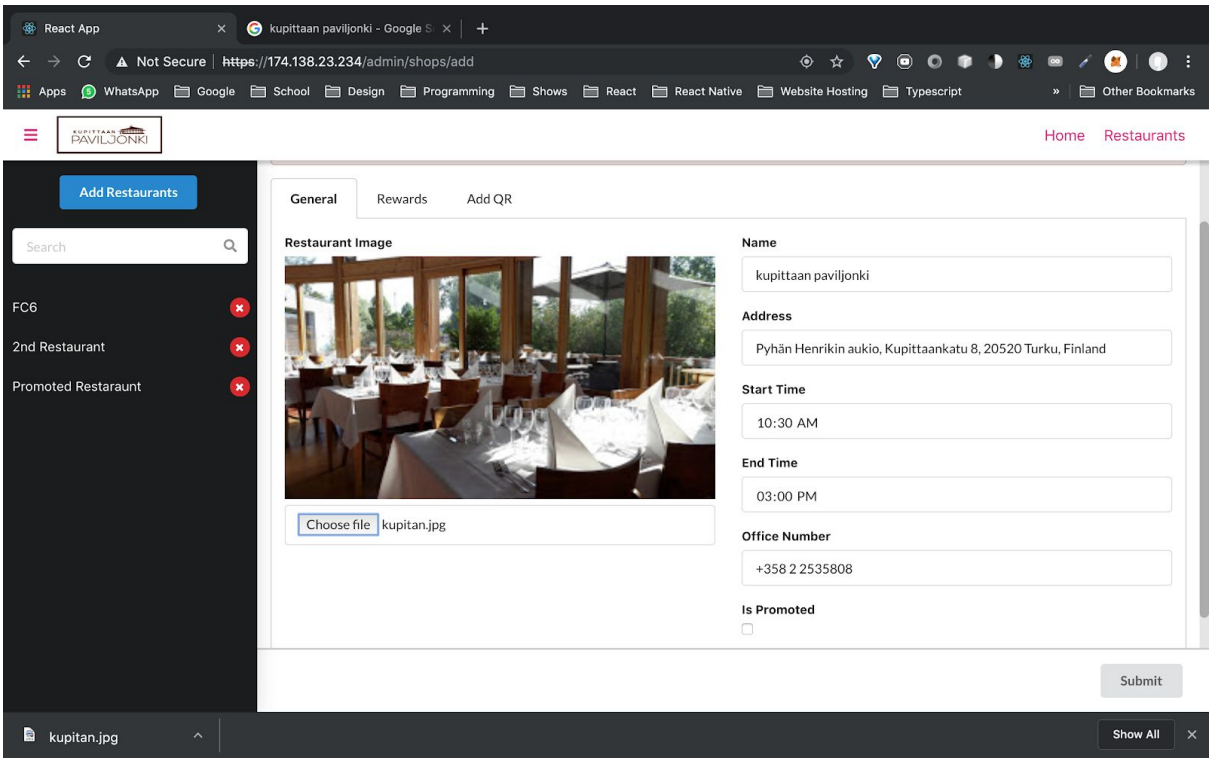
Note: the admin portal interface will only work desktops interfaces, not on mobile



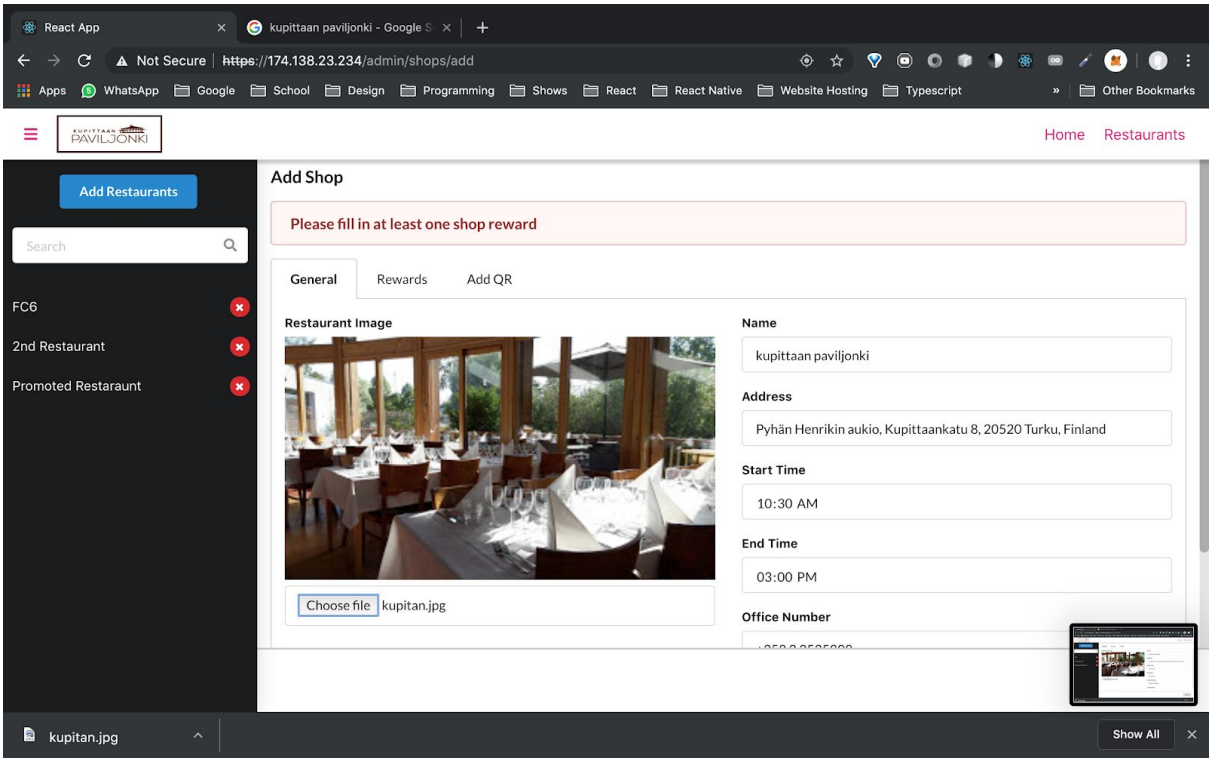
Fill in the following fields

Restaurant Image: the image of the restaurant
Name*: the name of the restaurant. The name will likely be used to indicate the location the store is based in
Address: the address of the restaurant
Start Time: opening time of the restaurant. Defaults to 12am
End Time: closing time of the restaurant. Defaults to 12am
Office Number: the office number of the restaurant
Is Promoted: whether the restaurant will be shown at the homepage or not.

** indicates required fields*



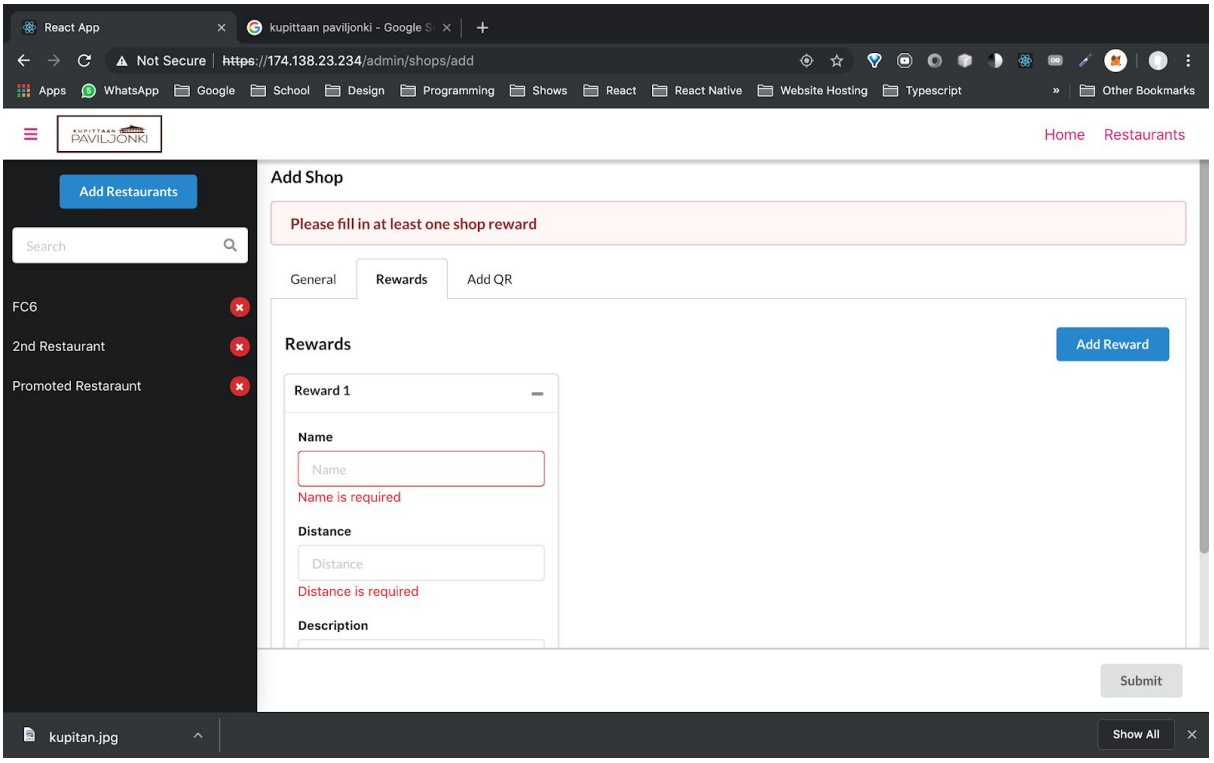
Once filling up the fields, you should find out that there is an error message at the top. It is complaining because at least one reward hasn't been filled. Click on the rewards tab at the top of the form

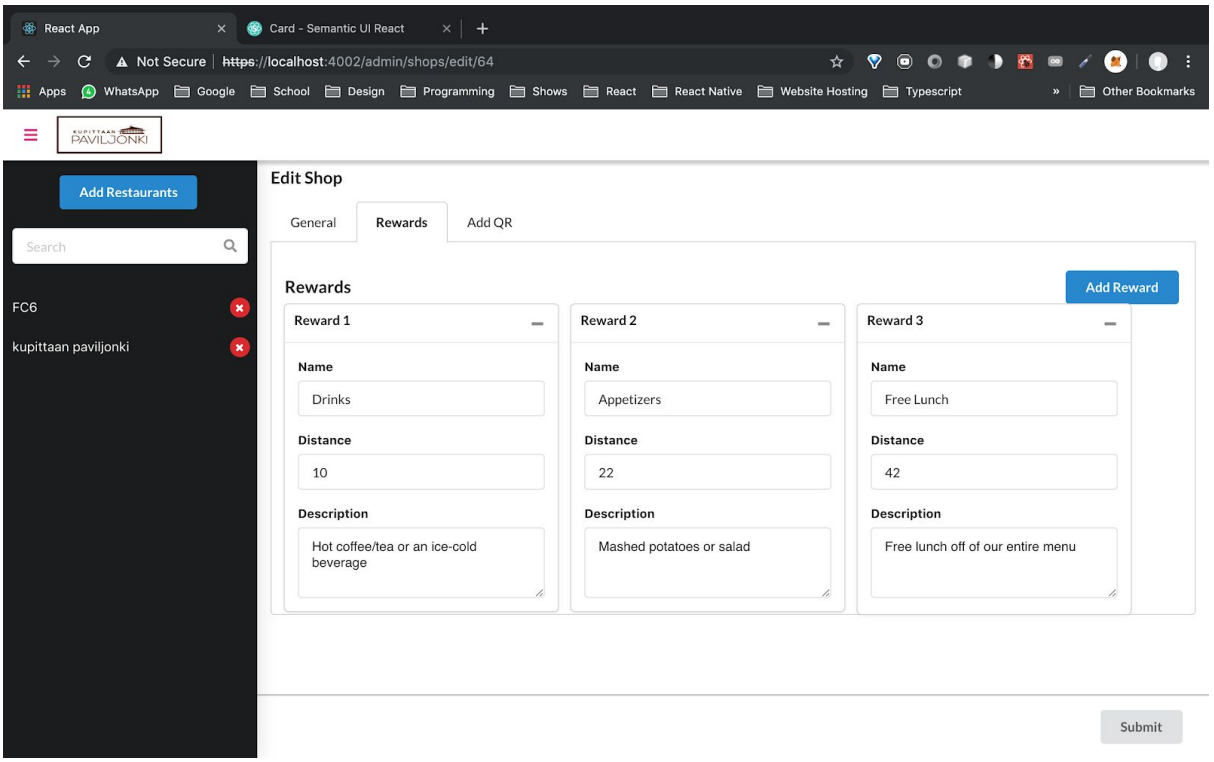


Fill in the following fields
Name*: the name of the reward
Distance*: the distance required to redeem the reward
Description: describes the reward
** indicates a required field*

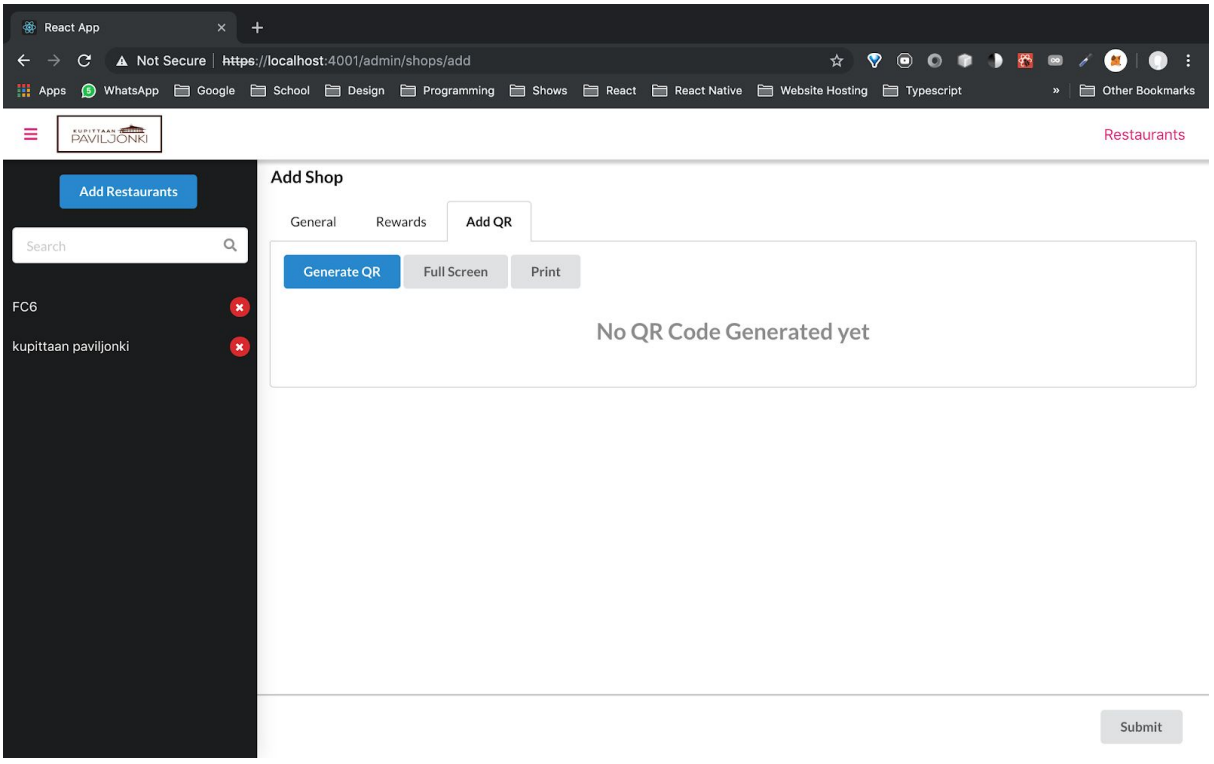
Click on the blue Add Reward button twice. You should now have 3 shop rewards to fill in.

*If you want to delete a reward, click on the **minus** icon at the top right corner of each reward*

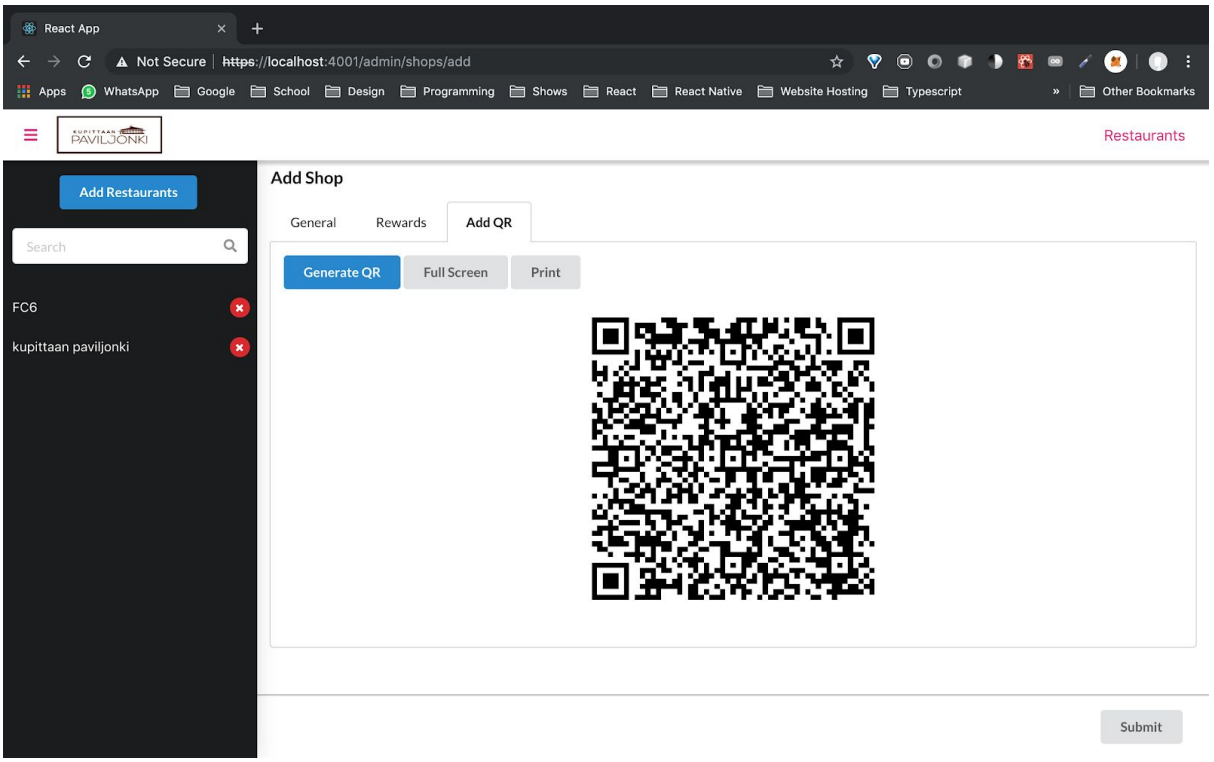




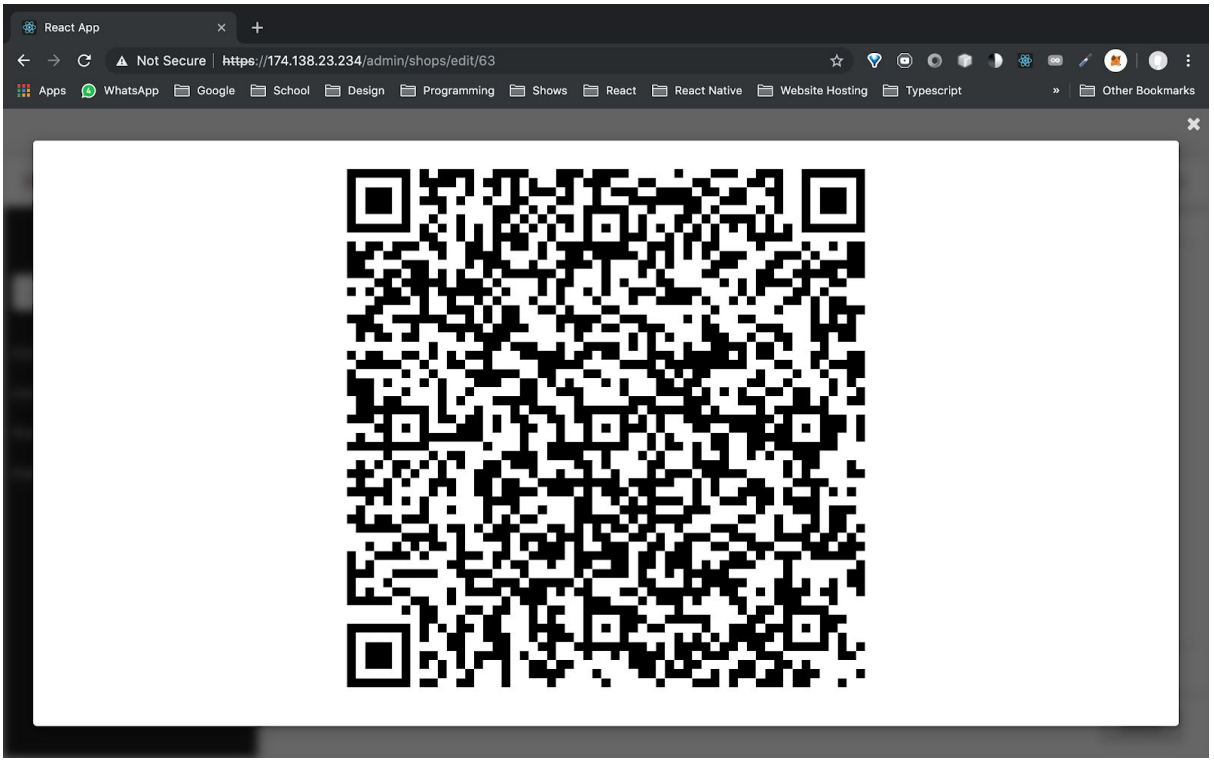
After filling up the reward fields, click on the Add QR tab around the same place where the rewards tab is. Clicking on it will lead you to this page. Click on the blue Generate QR button to generate a QR code.



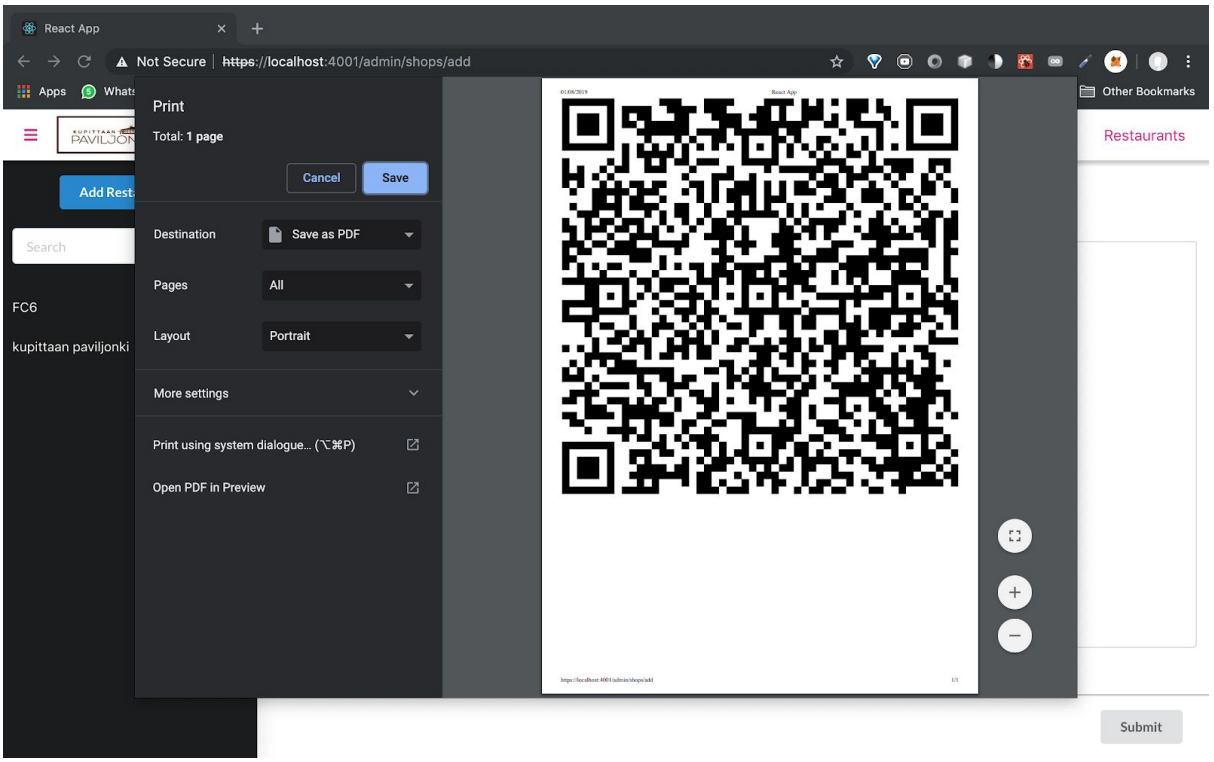
This QR Code is meant to be displayed at the counter of your restaurant, so that users can scan it after walking to your restaurants. If you want to refresh your QR code, click on the blue Generate QR button again.



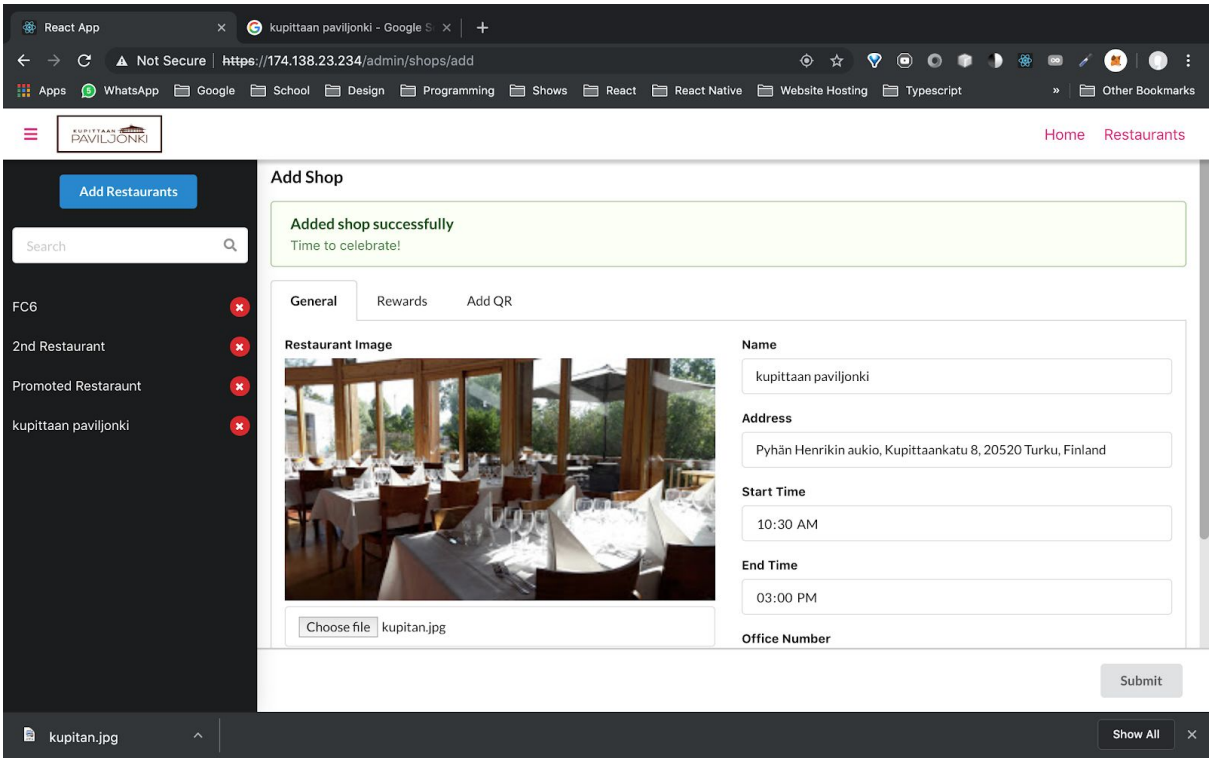
You can also click on full screen to make your QR code a little bit bigger. Click on the X icon at the top right to close the window



You may also want to print the QR code. In which case, click on the print button to open up the print dialog.

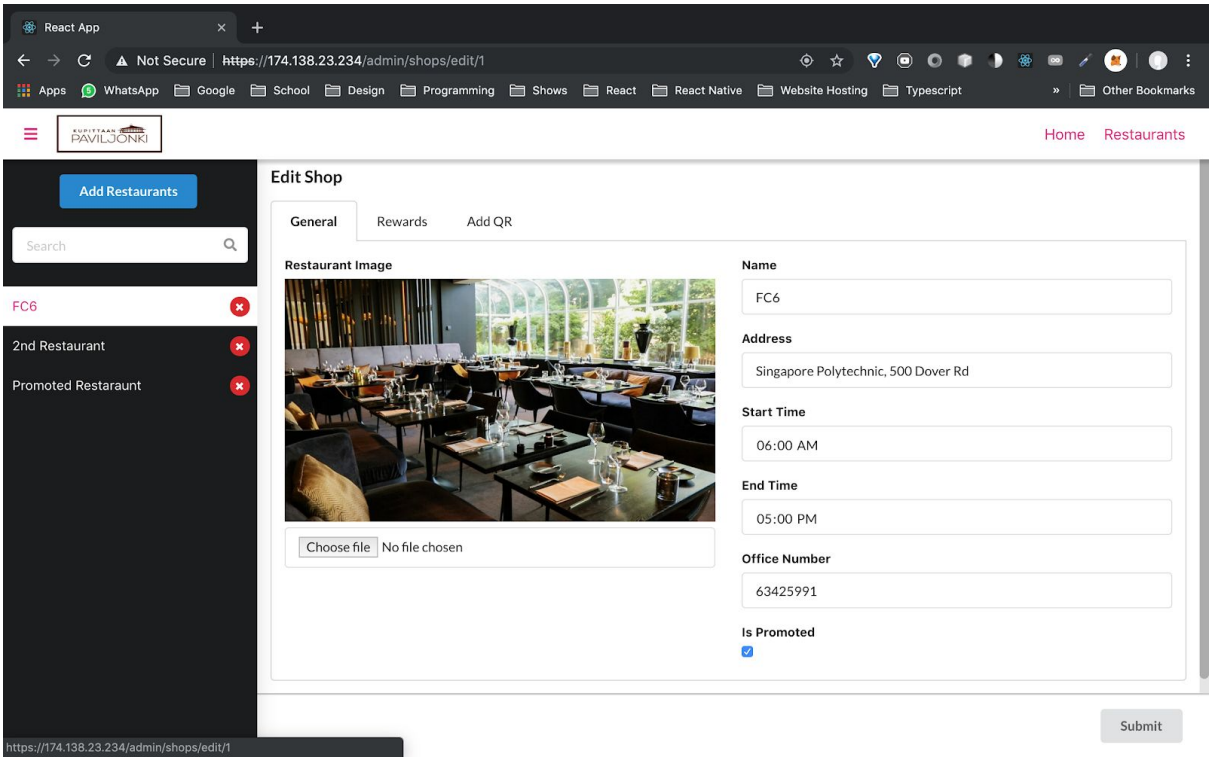


Once you're done, click on the Submit button at the bottom right corner and the shop should be saved properly. A success message should appear at the top, indicating your success

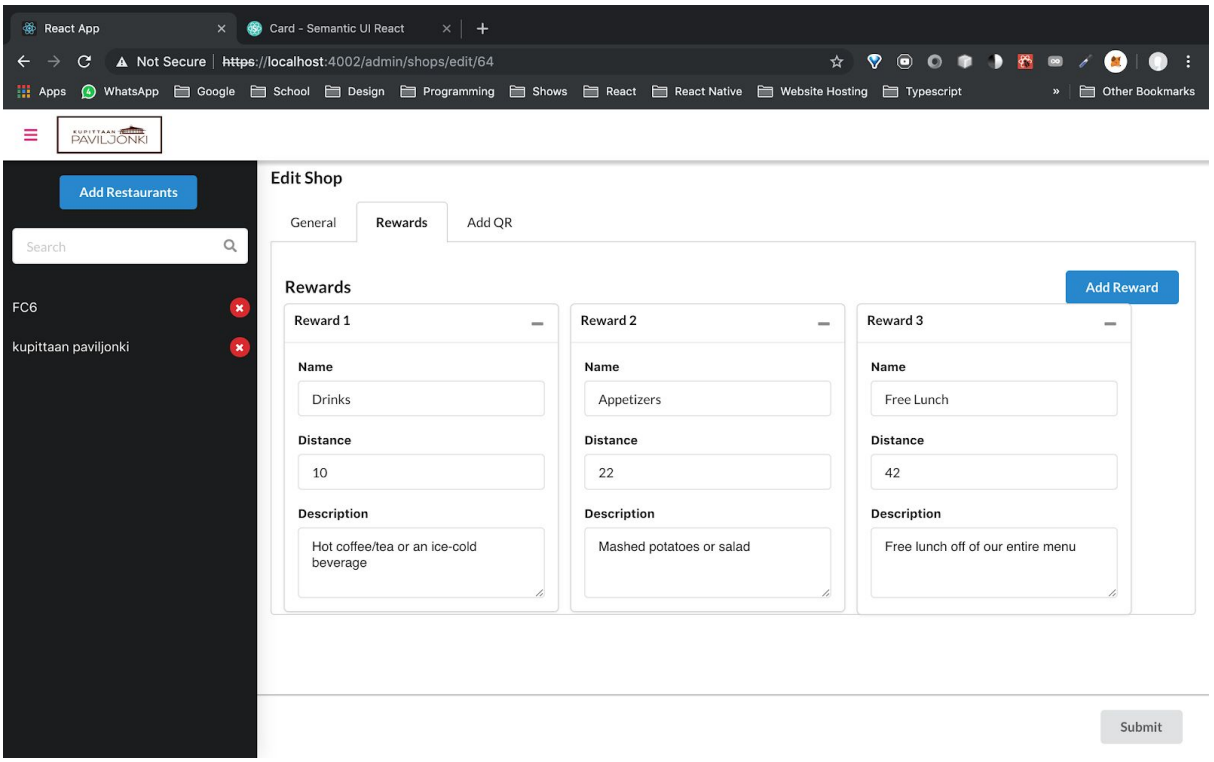


Editing Restaurants

Similar to adding restaurants, you can edit existing restaurants by clicking on their names at the left sidebar. Simply edit any field you wish to edit.



Perhaps you made a mistake while adding the shop rewards and want to update the offending reward, ør you want to add or remove rewards. Simply go to the restaurant that whose rewards you want to update, and access the rewards panel



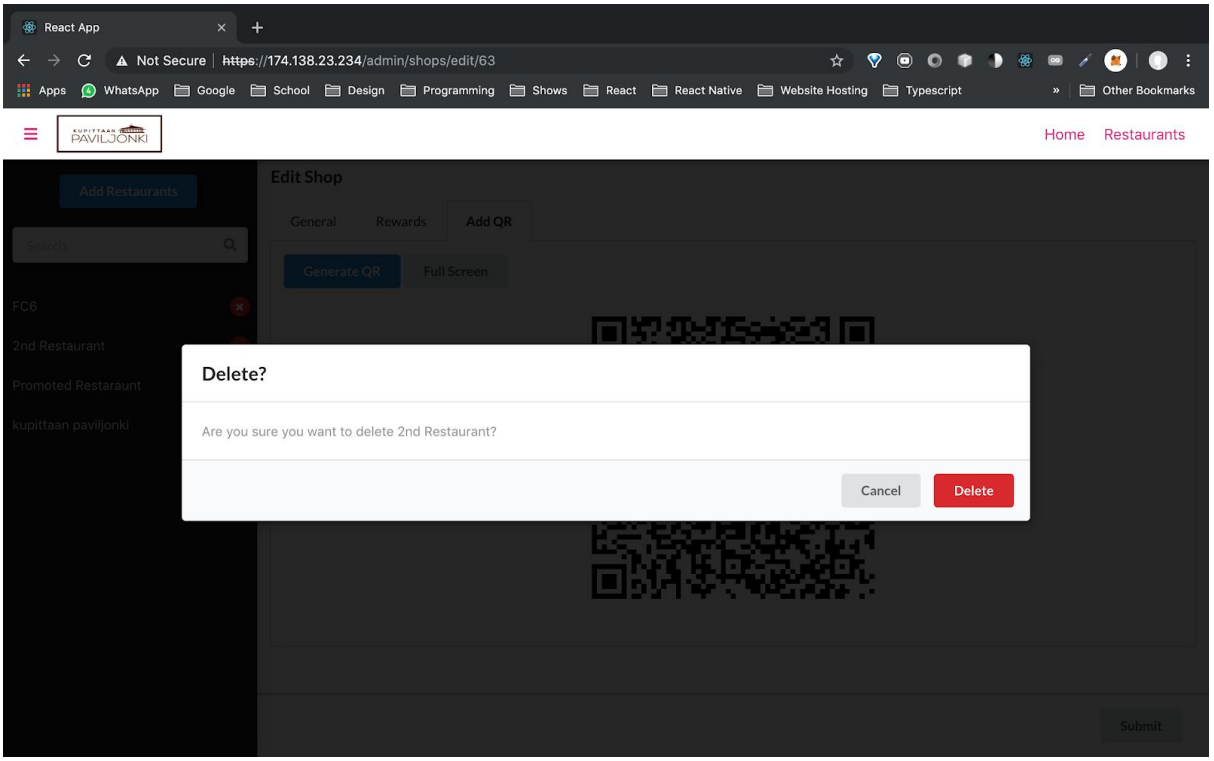
Simply edit any of the rewards and click on submit once you're done. The changes will be saved into the database. Additionally, if you delete any rewards by clicking on the top right minus icon on each reward, you must also click on submit to save the changes.

Deleting Restaurants

On the left sidebar, there is a red cross button to the right of each restaurant's name. Click on that and it will prompt you to delete the restaurant.

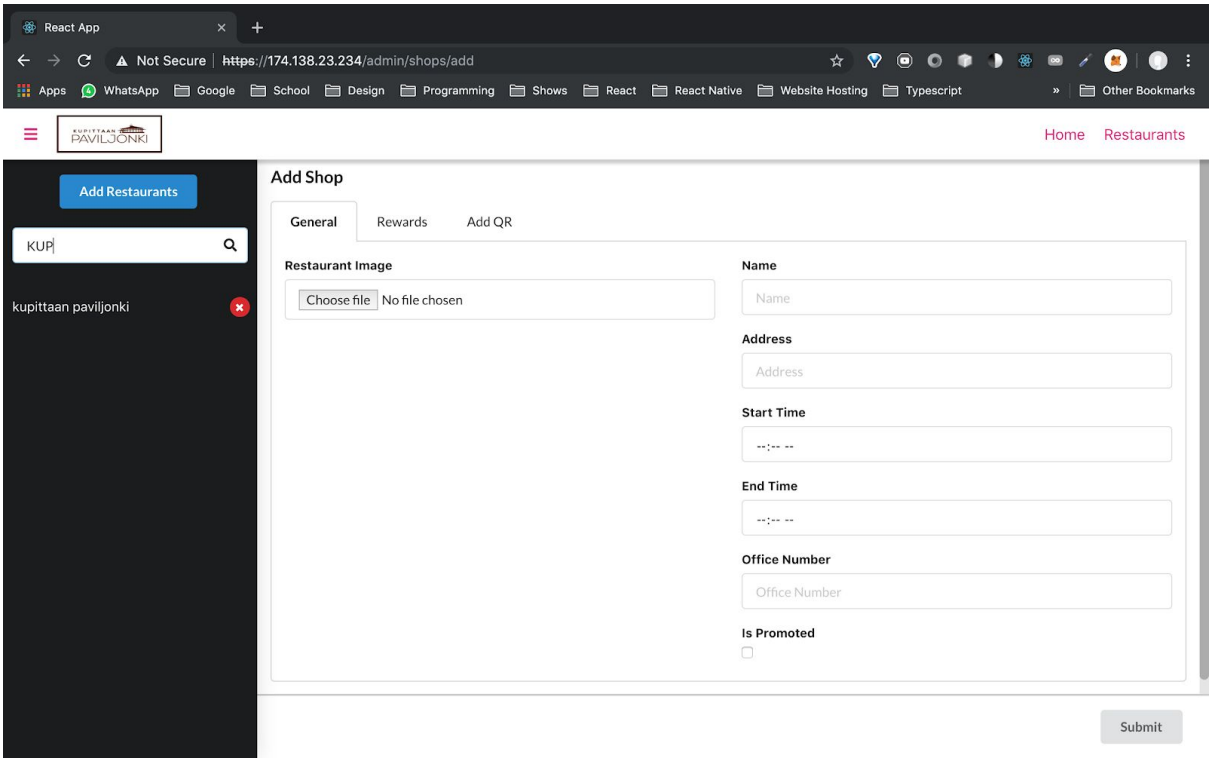
Note: Users who have already redeemed rewards from a deleted shop will still continue to keep records of the rewards. It can still be claimed from the user's rewards inventory despite the shop no longer existing, though it can no longer be redeemed from the store page..

Caution: **deleting a shop cannot be undone**



Searching Restaurants

Under the blue Add Restaurant button, is a search bar. Type the name of the restaurant you wish to find and it will show names that contain whatever you type. This is case-insensitive



General Configuration

Find the file in the following path: golunch/client/src/config.json
This file contains configuration options.

Start hour & End hour: You can set the start and end time of tracking period. Meaning you can change it from 10am - 3pm to 9am - 3pm for example. Keep in mind, startHour and endHour both use 24 hour format for time

Restaurant name: The name that's displayed at the homepage

Website: This website is optional and is only shown when a user views a shop's information

Speed Limit: This sets the maximum speed you allow the user to travel at. This concerns the user's tracking. Speeds that are higher than the set limit will invalidate the user's tracking info

Current configuration

```
{
   "website": "www.sp.edu.sg",
   "restaurantName": "Singapore Polytechnic",
   "startHour": 0,
   "endHour": 24,
   "speedLimit": 15
}
```

To change the values, simply change the right-hand-side values. Do not change the left-hand-side values like **"website"** or **"restaurantName".**

```
{
   "website": "https://www.kupittaanpaviljonki.fi/",
   "restaurantName": "Kupittaan Paviljonki",
   "startHour": 10,
   "endHour": 15,
   "speedLimit": 15
}
```

Changing the menu

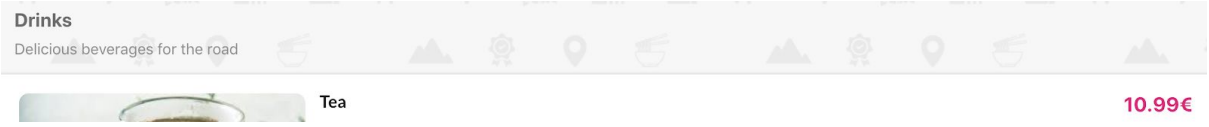
Locate the file ShopMenu.tsx. It should be found in the directory golunch/client/src//pages/shops/components.

There is a component called *MenuItemContainer* that holds information for each menu item, and a *ContentContainer* contains these *MenuItemContainers***.** The *ContentContainers* are supposed to group categories of food together.

ContentContainer takes in the following parameters:

1. header - name of the food category group
2. subheader (optional) - any extra description will appear under the header

Image Example:



Code Example:

```
<ContentContainer
   header="Drinks"
   subheader="Delicious beverages for the road">
   <MenuItemContainer
      name="Tea"
      description="It' just tea"
      price={10.99}
      imageUrl={`images/food/tea.jpeg`}/>
</ContentContainer>
```

MenuItemContainer takes the following parameters

1. name - the name of the menu item
2. description (optional)- the description of the menu item
3. price
4. imageUrl - the url of the image (the image url will be looking at the images folder directly under the src folder. Drop in your own images to customize the food images.

Image Example:

Code Example:

```
<MenuItemContainer
  name="Salad"
  description="A delicious, salad"
  price={5.00}
  imageUrl={`images/food/salad.jpeg`}/>
```

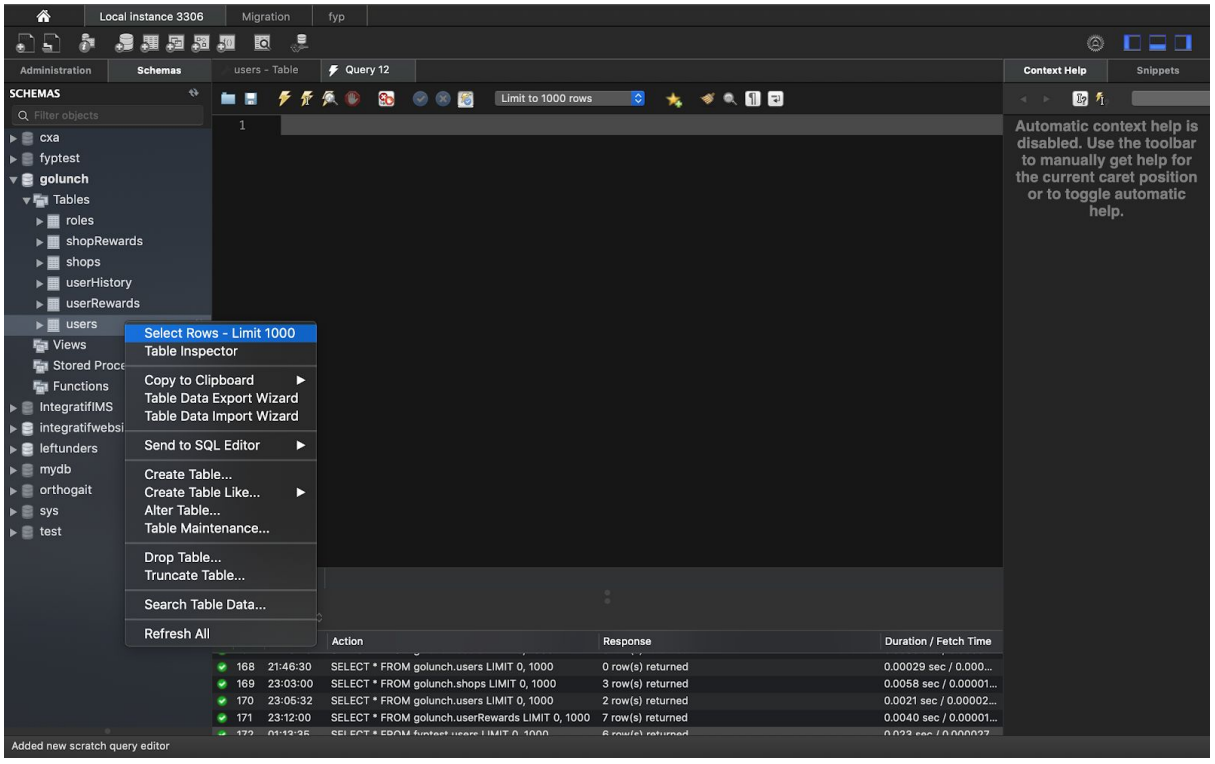
If you wish to change the currency, it's simple. Find the file MenuItemContainer.tsx within the directory: golunch/client/src//pages/shops/components. Scroll down and find the following code

```
<div className={css(styles.priceText)}>
    {price}€
</div>
```

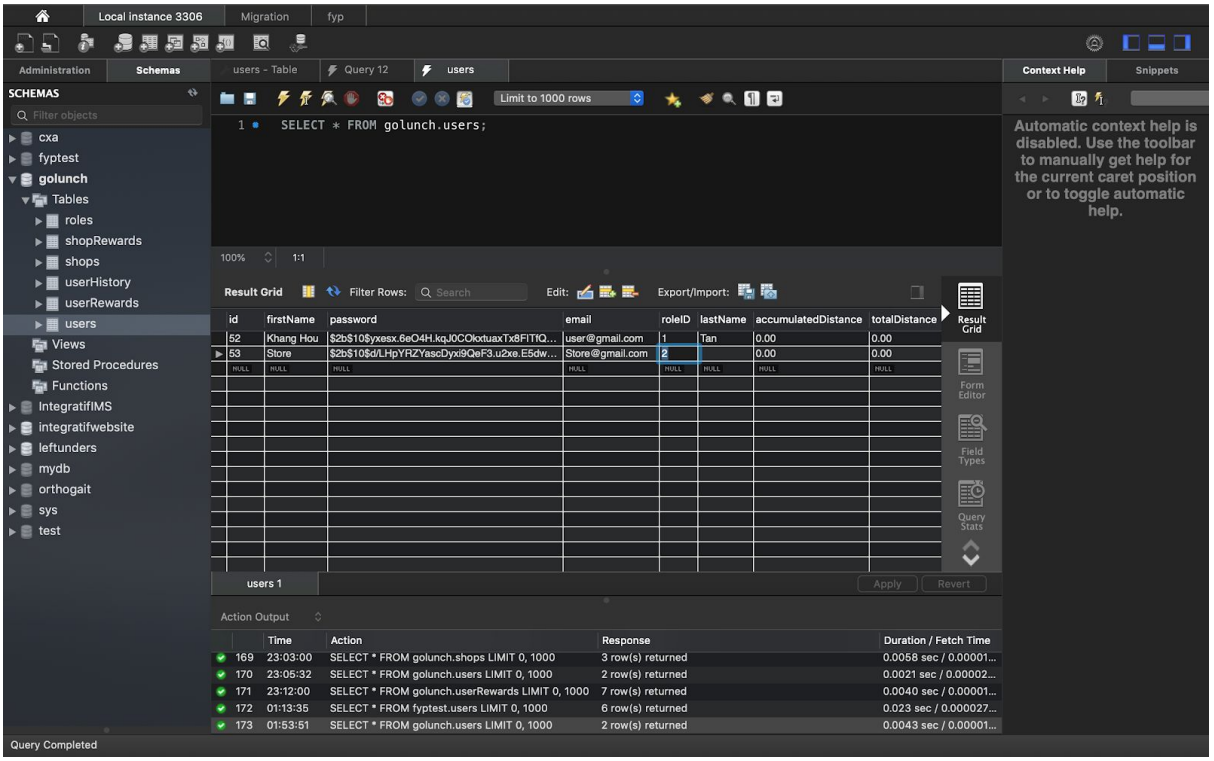
Replace the euro symbol with the currency you want.

Creating new admins

Register a new user at the register page. Then open up MySQL Workbench. Open up the golunch database, then tables and right click on the users table.



You should see a table of users at the bottom. Change the roleID of the newly registered user to 2. A roleID of 2 means that the user is treated as an administrator and is given the appropriate privileges



The registered user is now an admin, and can access the Manage Shops link from the navigation menu. You need to re-login back into your admin account if you logged in before the change.

Setting Up

Install NodeJS and npm

Run the following commands on your command prompt or terminal. Be sure to run it as administrator. This guide is performed on an Ubuntu or Linux OS (valid on Mac or Linux systems).

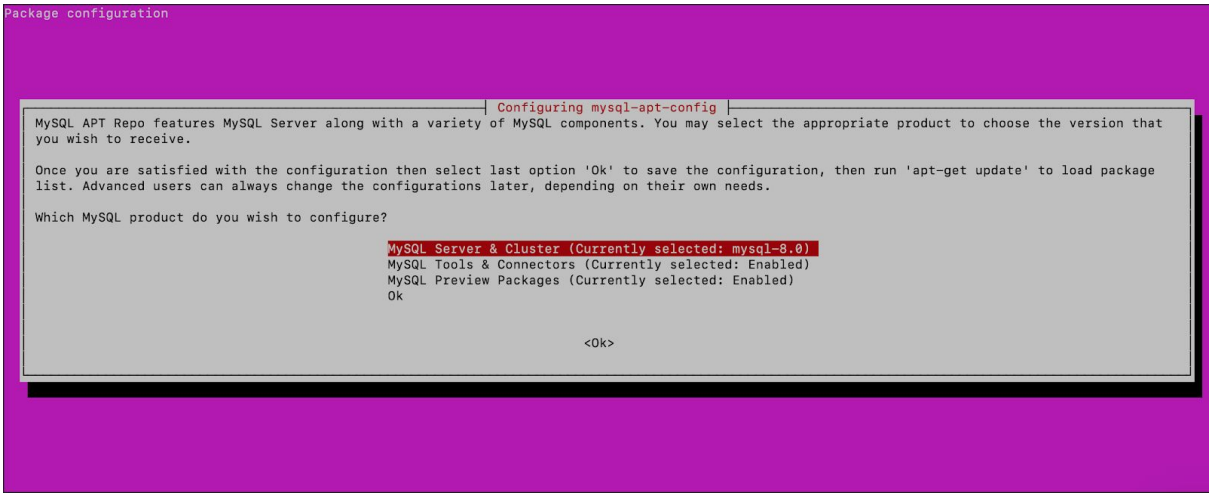
```
curl -sL https://deb.nodesource.com/setup_6.x -o nodesource_setup.sh
bash nodesource_setup.sh
apt-get install nodejs
apt-get install build-essential
apt-get install npm
npm install -g pm2
```

Install MySQL Server 8.0

```
wget -c https://dev.mysql.com/get/mysql-apt-config_0.8.11-1_all.deb
dpkg -i mysql-apt-config_0.8.11-1_all.deb
```

This should open up a window prompt as shown below. If the selected MySQL version is not 8.0, press enter and select 8.0. Enable MySQL Preview Packages as well.

Change selected option by pressing the up and down keys. Press enter to select the option



Once done, move down to the 4th option: OK, and press enter. After that, run these two lines

```
apt-get update
apt-get install mysql-server
```

After that, type in the following command to check if MySQL has been properly installed.

```
mysql -V
```

It should show the following line if your installation went withour problems
mysql Ver 8.0.17 for Linux on x86_64 (MySQL Community Server - GPL)

Install MySQL Workbench 8.0

Follow this link: https://dev.mysql.com/downloads/mysql/
Scroll down and click on the first installation link. It should be a standard installation

Setting up the database

Open up MySQL Workbench 8.0. A new query tab should automatically be opened. Type in the following MySQL commands

```
CREATE database golunch;
```

Copy paste the code inside the attached SQL script: *golunch.sql* and run it to populate the database

Deploying

Deploying Server

Open command prompt or terminal again. Be sure to run it as administrator.

You need to move into the directory that the project is in

```
cd path/to/project/client
```

You should've installed pm2 earlier. pm2 is used to run the server file. Your current directory should be within the client folder of the project directory. Example: golunch/client

```
pm2 server/server.js
```

It should show the following result

```
Use --update-env to update environment variables
[PM2] Applying action restartProcessId on app [0](ids: 0)
[PM2] [server](0) ✓
```

App name	id	version	mode	pid	status	restart	uptime	cpu	mem	user	watching
server	0	0.1.0	fork	22851	online	6	0s	0%	8.9 MB	root	disabled

API Documentation

Auth.js

Responsible for registering and authenticating users

Register

URL: /api/auth/register
Method: POST
Parameters:
- firstName: string
- lastName: string
- email: string
- password: string

Description: Inserts a new account with a roleID of 1 into the database. Automatically hashes the plaintext password value passed in the parameter

Login

URL: /api/auth/login
Method: POST
Parameters:
- email: string
- password: string

Description: Finds the user record that matches the email parameter passed. It then hashes the password passed using bcrypt, and compares it with the password hash of the user. If it is a successful comparison, it will return a JWT that contains the user's details to the front-end. This will be used to keep the user authenticated and logged in throughout sessions

Tracking.js

Responsible for doing a lot of the tracking-based backend work

Get Time

URL: /api/tracking/getTime
Method: GET
Parameters: None
Description: Returns the current server time. This is used to prevent the client from using their own time and cheating the system.

Scan QR

URL: /api/tracking/scanQR
Method: POST
Parameters:

- Array of trackingInfo. TrackingInfo object consists of
    - userID: number
    - startTime: Date
    - endTime: Date
    - date: Date
    - distance: number

Description: Loops through the array of trackingInfo. For each entry in the array, adds a single record to the userHistory table. It also adds up the totalDistance. Once it exits the loop, the user's accumulatedDistance is added by the totalDistance

Shops.js

Responsible for retrieving and managing shops.

Get Shops

URL: /api/shops/get
Method: GET
Parameters: None
Description: Gets all shops

Get Promoted Shops

URL: /api/shops/getPromoted
Method: GET
Parameters: None
Description: Gets all promoted shops

Get Shop

URL: /api/shops/get/:id
Method: GET
Parameters:
- Id: number

Description: Gets a shop by ID

Add Shop

URL: /api/shops/add
Method: POST
Parameters:
- Id: number
- Name: string
- Address: string
- officeNumber: string
- startTime: Date
- endTime: Date
- Coords: Object
    - lat: number

        - lng: number
        - E.g.: {lat: 1.311, lng: 103.4113}
    - imageUrl: string
    - isPromoted: boolean,
    - accessToken: string
    - shopRewards: Array of ShopRewards
        - ShopReward
            - Id: number,
            - Name: string
            - Description: string (optional)
            - shopID: number

Description: Processes the passed in variables. Stores the shop image (if any) into the server directory *golunch/client/src/images/shops.* Adds the shop into the database, then loops through the shopRewards array and adds it into the database one-by-one

Edit Shop

URL: /api/shops/edit
Method: POST
Parameters:

- Id: number
- Name: string
- Address: string
- officeNumber: string
- startTime: Date
- endTime: Date
- Coords: Object
    - lat: number
    - lng: number
    - E.g.: {lat: 1.311, lng: 103.4113}
- imageUrl: string
- isPromoted: boolean,
- accessToken: string
- shopRewards: Array of ShopRewards
    - ShopReward
        - Id: number,
        - Name: string
        - Description: string (optional)
        - shopID: number

Description: Similar to add shop, but more complex functionalities. Processes the passed in variables, then updates the shop. Then, it gets the old shop rewards using the shop ID, and loops through the old shop rewards. By comparing the old shop rewards and the newly passed shopRewards, it will be able to determine if the new shop reward still exists in the within the old shop rewards or not. It it doesn't exists, it deletes the old shop reward, but if it still exists, updates it with the new data. If there is a new shop reward that does not exist in the old shop rewards, the system adds it into the database.

Delete Shop

URL: /api/shops/delete/:id
Method: GET
Parameters:
- Id: number

Description: Deletes the shop using the passed in ID parameter. Gets the shop object using the passed in ID and checks if the shop has an imageUrl property. If there is, then deletes the image in the server first, before deleting the shop.

ShopRewards.js

Responsible for shop rewards

Get Shop Rewards by Shop ID

URL: /api/shopRewards/get/:shopID
Method: GET
Parameters:
- shopID: number

Description: Gets all shop rewards that has the corresponding shopID

UserRewards.js

Responsible for claiming and redeeming shop rewards

Get User Rewards by User ID

URL: /api/userRewards/getUserRewardsByUserID/:id
Method: GET
Parameters:
- Id: number

Description: Gets user rewards by user ID. The database query uses a left join, so it returns information from both userRewards and shopRewards table. Returns the following properties:
- userRewards table
    - Id
    - userID
    - rewardID
    - isClaimed
- shopRewards table
    - Name (of the shopReward)
    - Description
    - Distance

Also only gets userRewards whose isClaimed values are 0. Prevents users from claiming already-claimed rewards

Add User Reward

URL: /api/userRewards/addUserReward
Method: POST
Parameters:
- rewardID: number
- userID: number
- Distance: number

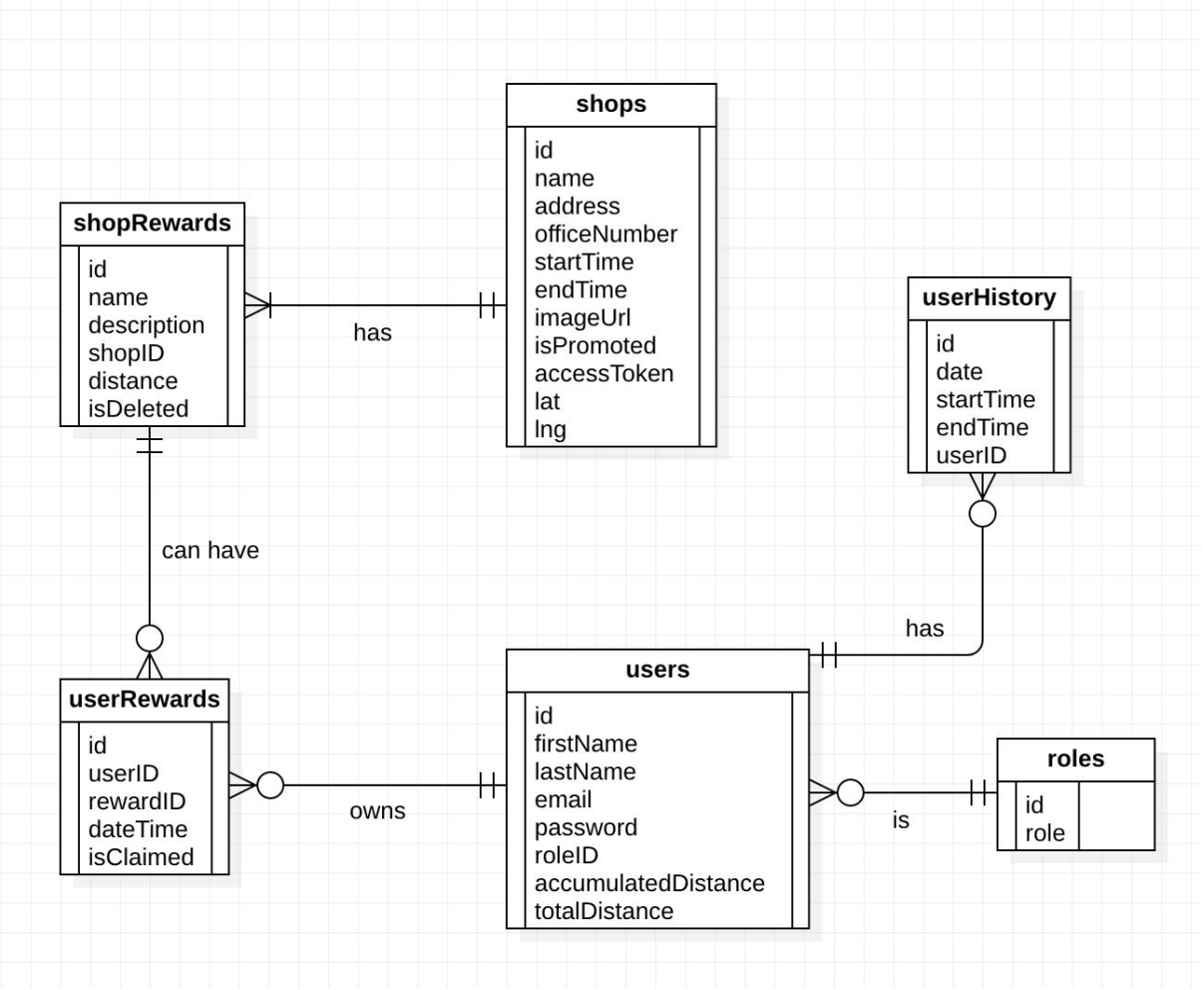
Description: Adds a user reward record according to the userID. The user's distance is subtracted by the distance parameter passed. It then returns the user's resulting accumulated distance.

Claim Reward

URL: /api/userRewards/claimReward/:id
Method: GET
Parameters:
- Id: number

Description: Sets the isClaimed property of the user reward from 0 to 1. This essentially prevents the rewards from showing up in the user's inventory, while still remaining as a record in the database.

ER Diagram


shops
id
name
address
officeNumber
startTime
endTime
imageUrl
isPromoted
accessToken
lat
lng
shopRewards
id
name
description
shopID
distance
isDeleted
has
userHistory
id
date
startTime
endTime
userID
can have
has
users
id
firstName
lastName
email
password
roleID
accumulatedDistance
totalDistance
userRewards
id
userID
rewardID
dateTime
isClaimed
owns
is
roles
id
role

Minutes of Meetings

Refer to attached file

Test Cases

Refer to attached file