PROJECT REPORT

COURSE CODE	**:**	**WIX3004**
COURSE NAME	**:**	**MOBILE APPLICATION DEVELOPMENT**
LECTURER	**:**	**DR. ONG SIM YING**
SEMESTER / SESSION	**:**	**SEMESTER 1, SESSION 2020/2021**
TUTORIAL	**:**	**3**
GROUP NAME	**:**	**JETPACK**
TITLE	**:**	**MOBILIZED QR MENU APPLICATION**

NO.	**NAME**	**MATRIC NUMBER**
1.	DYLAN SALIM	17123480/1
2.	NG JIH YANN	17060539/1
3.	FANG WAI NAM	17183792/1
4.	DONG YIRUI	17102165/1
5.	TAN QING LIN	17074063/1

Table of Contents

a. Introduction

QR Menu mobile application is an application that migrates conventional menu listing to digital version of menu listing. There are two main types of users in this application which are Merchant and Customer. Merchant can upload their business details and items list (menu) to the application and Customer can search the menu published by Merchant through scanning the QR code generated from the application. Users are no longer needed to reach to a shop or restaurant for food or item menu.

User can become a Merchant just by selecting Merchant option during the account registration. Besides, a Customer can also become a Merchant by role switching setting in the Account page. A setup new store message will pop up in the QR Scan page if the Merchant never setup a store yet. A store must have information like store name, contact number, address, operating hour to complete the registration. Merchant can set up their store item list in the Merchant Item page. Besides, Merchant can create category for the items they want to show in their store. For items on promotion, Merchant can activate the promotion button of the item and enter the promotion price of the item. After that, Merchant can generate the QR code of their store and share the QR code through social media if they have set up the store. Customer can access the store menus by scanning the QR code using the QR code scanner of the application. QR code scanner can be accessed in the Home page or in the QR Scan page.

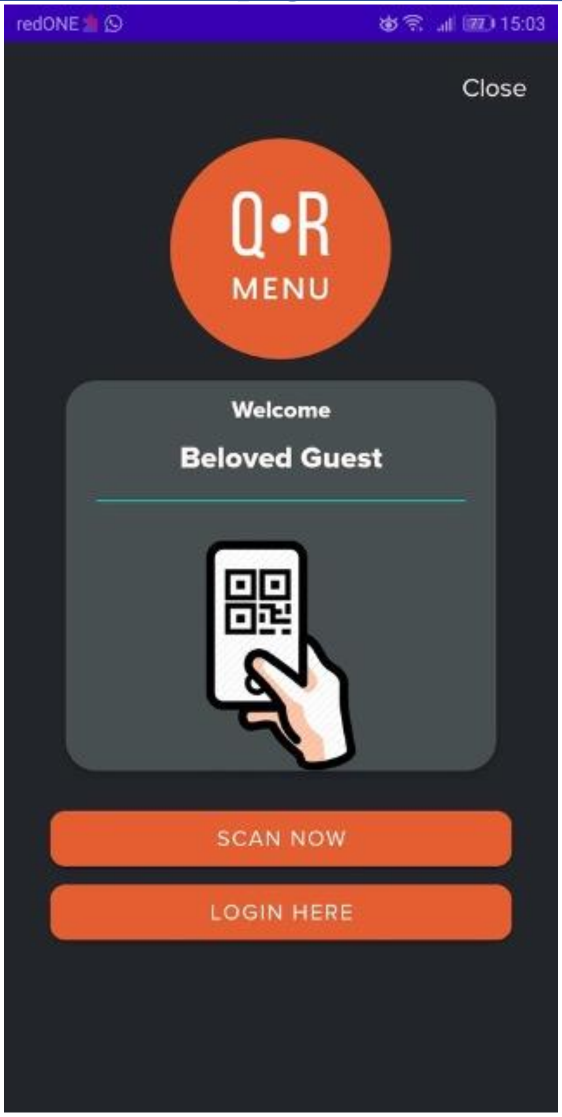
Customers have the right to file a report on a store with reasons. The report will be emailed to QR Menu Admin and the admin will check the problems out. Admin has the right to penalize the store that violates the rules and regulations of the application. In addition, Customers can rate a store and the store average rating will be shown clearly to the Customers. There are two types of Customer, registered user, and one-time user. A registered user can save some menu as Favorites and perform several functions like profile editing, manage notification setting and role switching. One-time user is also allowed in this mobile application. However, one-time user is unable to set up profile and saving menus as Favorites. One-time user is also not allowed to manage the notification setting. Nevertheless, one-time user and registered user can do feedback to the application on Google Play Store.

b. Functions and Accomplishment

Functionalities Proposed	**Accomplishment**
Menu Creation - Merchant can use the application to create a menu and generate a QR code that will redirect user to the shop catalogue page.	Menu Creation - Merchant can create a store with menu and generate a QR code that will redirect user to the store.
Menu Modification - Merchant can update, modify, or delete the created shop's catalogue.	Menu Modification - Merchant can update, modify, or delete the items in the menu list of the store.
Menu Discovery - Customer can use the application to access the shop's catalogue page through the generated QR code. The catalogue will contain a list of items or services with description, pricing, and sample photo.	Menu Discovery - User can access the store containing a list of items or services with description, pricing, and item's sample image through scanning the generated QR code using the QR code scanner in the application.
Account Registration - Merchant is required to register for a merchant account before performing any actions. Customer could also register for a regular account to receive new notification from the subscribed merchants. However, the customers are not required to register an account to view shop catalogue.	Account registration - User can register as Merchant or Customer by selecting the user type during registration. Registered Customer can receive new notification from the subscribed merchants. - User can be one-time user that can view the store but unable to use the subscribe functionalities.
Favorite Merchant - The customer can subscribe to a merchant and the customer will receive notification from the merchant if there is new update or promotion from the subscribed merchant.	Favorite Merchant - The Customer can add a store into Favorite list (subscribe) and the customer will receive notifications from the merchant if there is new update or promotion from the subscribed merchant.
Map - Merchant can provide their shop location in the catalogue. Once customers click on the address in the description page, the application will	Map - Merchant can provide their store location in the store's details. Application will route the customer to show the shop location in Google Map

route the customer to show the shop location in Google Map.	once customer click on the address in the description page.
Rating - Customer can view and give rating to a shop. The rating section will have a censoring swear words function.	Rating - Customer / One-time User can view and rate a store. The rating section have a censoring swear words function.
Report - Customer could report a merchant if the merchant is selling illegal items or providing false information on the item they are selling.	Report - Customer / One-time user can report a merchant with reasons and the report is emailed to the mobile application admin.
Share merchant via Social Media - Customer can share a particular shop catalogue through social media by the means of sharing generated QR code.	Share merchant via Social Media - User can share a store through social media by sharing the generated QR code for the store.

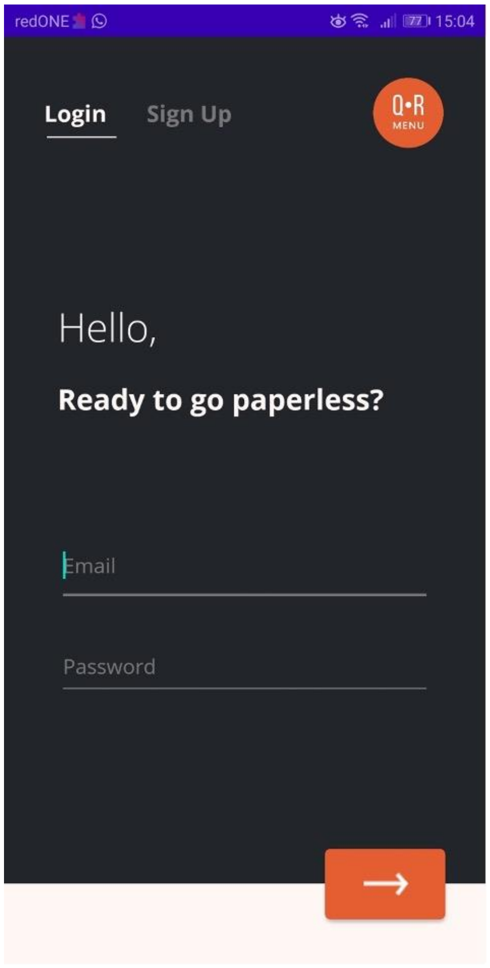
c. Screenshot of all activities/pages with descriptions



Figures 1 QR Scan Activity

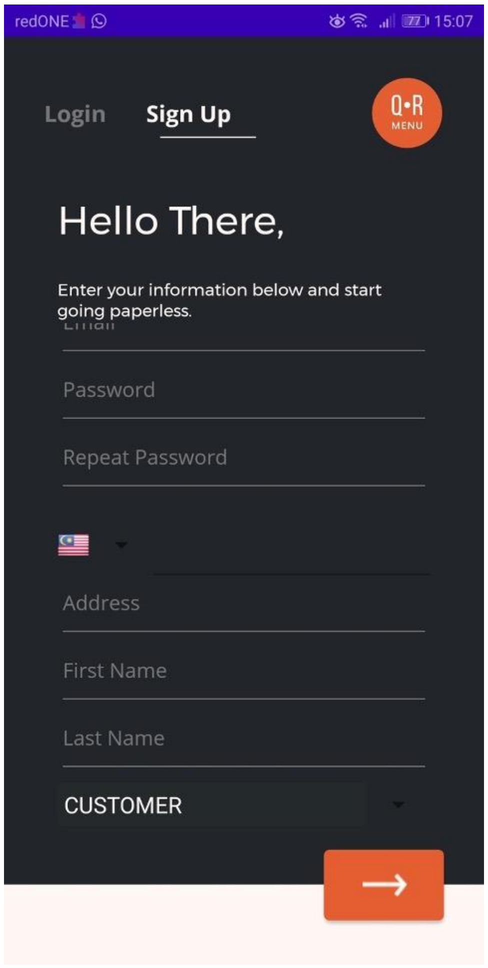
Figures 1 shows the screenshot of the QR Scan Activity. The Users can view menus without registration as some one-time users may not be willing to, however they will not be able to perform several operations such as saving menus as favorites.

In the QR Scan Activity, user will be able to use his mobile phone camera to scan QR code by simply locating the QR code in the QR scanner located in middle of the screen If user scans a valid QR code for menu, he/she will be redirected to the Merchant Item Activity where menu information is shown.



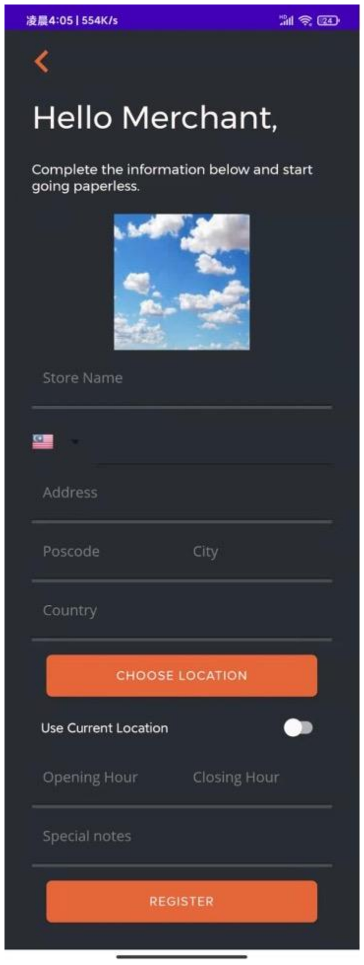
Figures 2 Login Fragment

Figures 2 shows Login Fragment of the Login Registration Activity. By entering a valid email address and respective password into the fields located in the center, users will be able to login to the mobile application. If user wants to create an account instead, he can press on the 'Sign Up' tab beside the highlighted 'Login' tab to be redirected to the Registration Fragment.



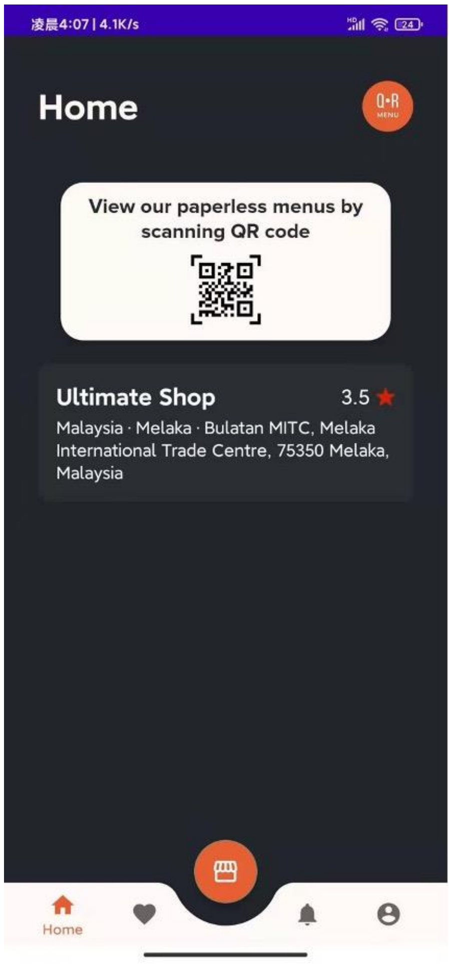
Figures 3 Registration Fragment

Figures 3 shows the Registration Fragment, where users can create an account by entering valid information. Users can select their roles to be either ‘customer’ or ‘merchant’.



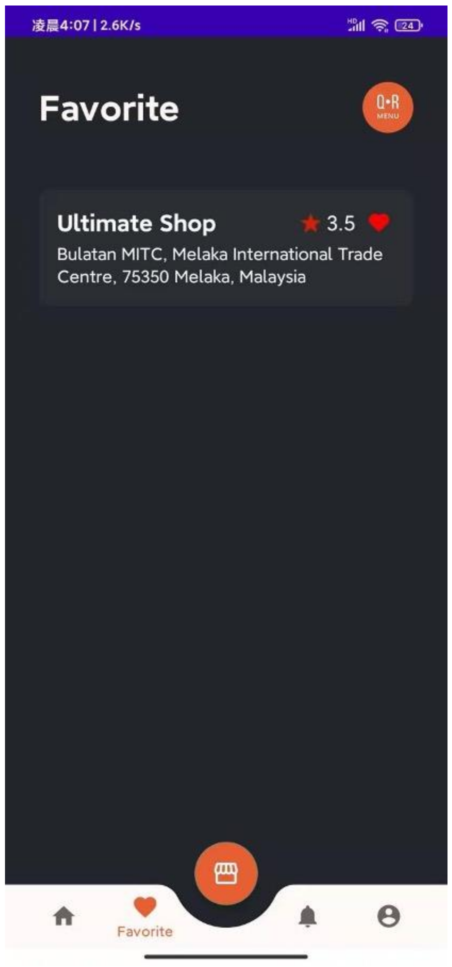
Figures 4 Store Registration Activity

Figures 4 shows the Store Registration Activity. After merchants logged in, if they have not registered their store, they need to insert information such as the shop name, location, merchant phone number and opening hour in the Store Registration Activity.


凌晨4:07 | 4.1K/s
Home
Q·R MENU
View our paperless menus by scanning QR code
Ultimate Shop
3.5
Malaysia · Melaka · Bulatan MITC, Melaka International Trade Centre, 75350 Melaka, Malaysia
Home

Figures 5 Home Fragment

Figures 5 shows the Home Fragment in the Main Activity. Home Fragment of Main Activity acts as the home page for our mobile application. In the middle of the user interface, users are able to scan QR code by clicking on the QR button, redirecting users to the QR Scan Activity. Below the QR scan field shows the list of the recently visited menus of the users. If users press on one of the menus, they will be redirected to the Merchant Item Activity which shows the items of the menus.

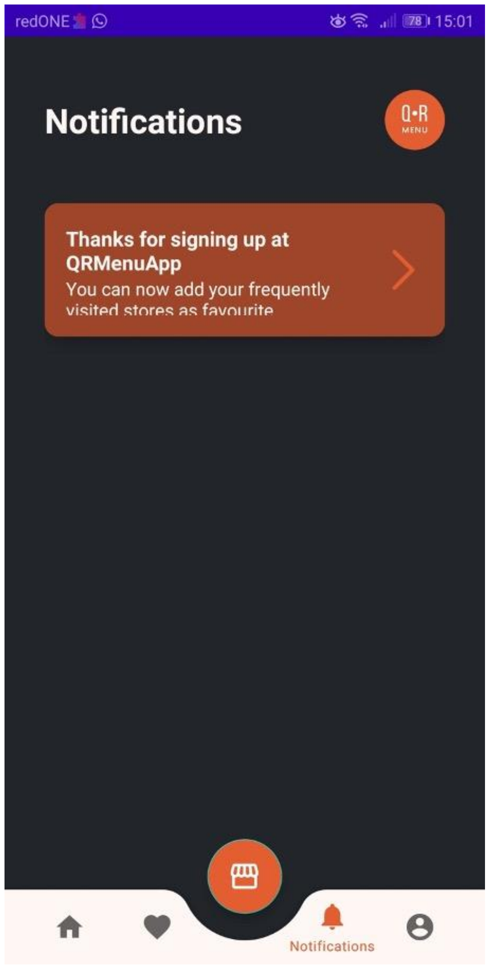


Ultimate Shop 3.5

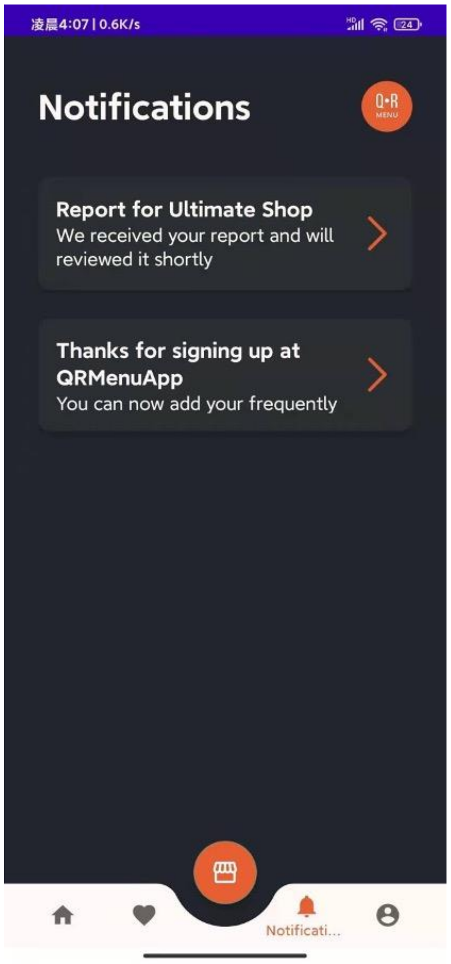
Bulatan MITC, Melaka International Trade Centre, 75350 Melaka, Malaysia

Figures 6 Favorite Fragment

Figures 6 shows the screenshot of the Favorite Fragment. In this fragment, the list of menus which are tagged as favorite by the user will be displayed. Clicking onto one of the menus will redirect users to the Merchant Item Activity which shows the menu items and merchant detail.


Notifications

Thanks for signing up at QRMenuApp
You can now add your frequently visited stores as favourite

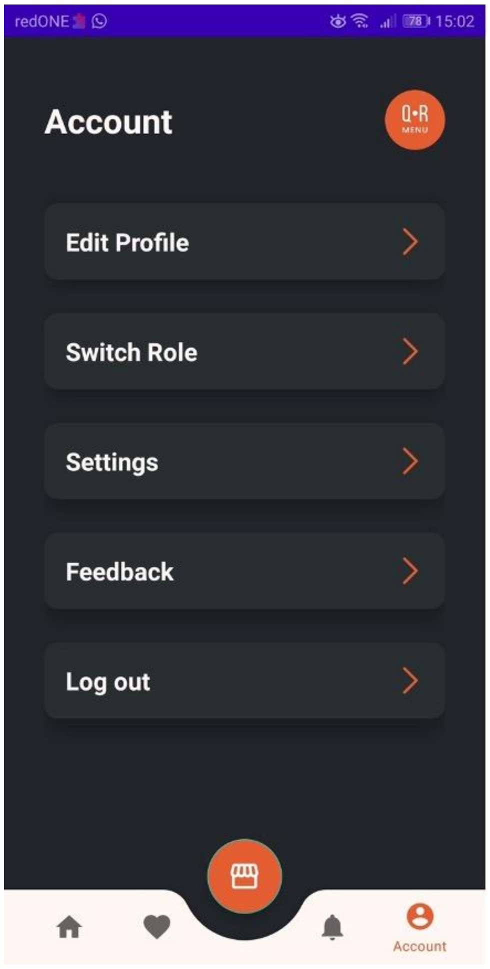

Notifications

Report for Ultimate Shop
We received your report and will reviewed it shortly

Thanks for signing up at QRMenuApp
You can now add your frequently

Figures 7 Notification Fragment

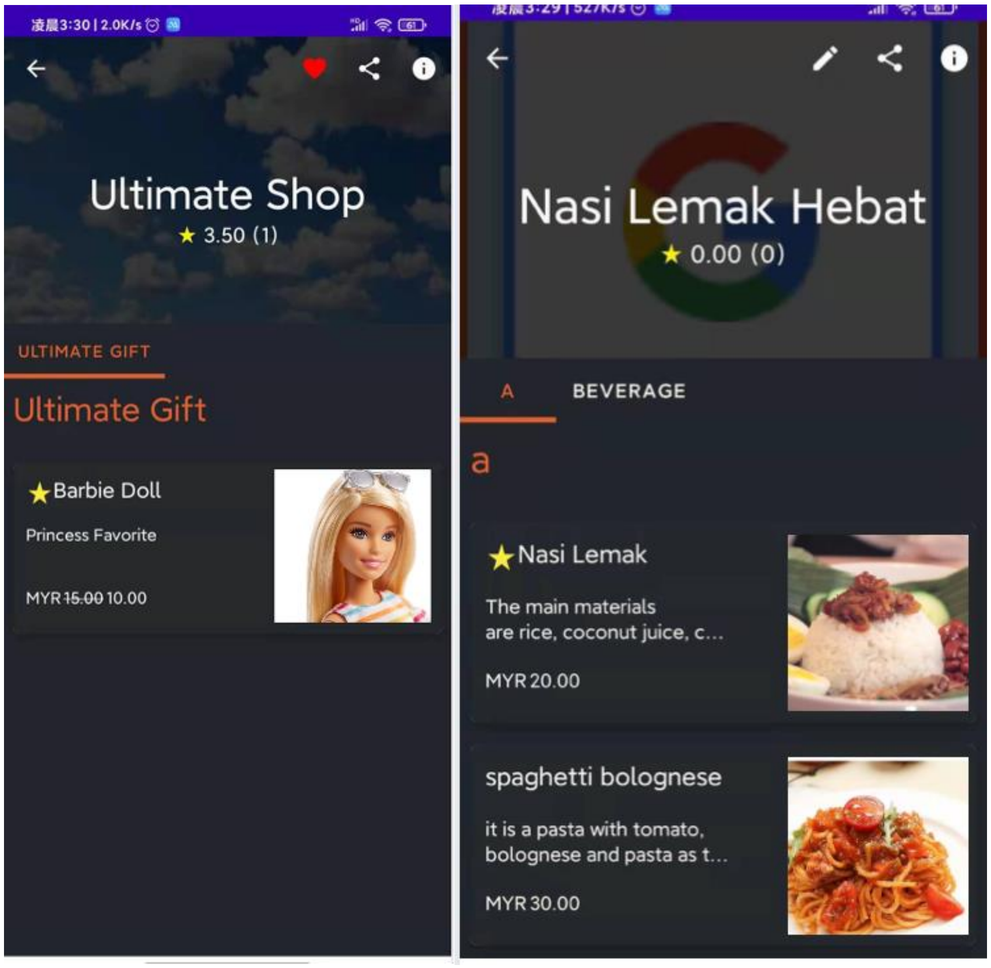
Figures 7 shows the page of the Notification Fragment. In this fragment, the list of notifications is shown. Unread notifications are displayed with the orange background whereas the read notifications have a gray background. Pressing into the notifications, users will be shown more details about the notification that cannot be shown in the excerpt.



Figures 8 Account Fragment

Figures 8 is the screenshot of the Account Fragment. In this fragment, users are be able to manage personal information and application setting.

‘Edit profile’ button allows user to edit personal info, profile picture etc. ‘Switch role’ button enable the switching of customer to merchant and vice versa. ‘Setting’ button allows users to manage notification settings and permission settings. ‘Feedbacks’ button allows user to share feedback about the application in Google Play Store as the button will be linked to Google Play. 'Log out' button allows user to log out of their account.

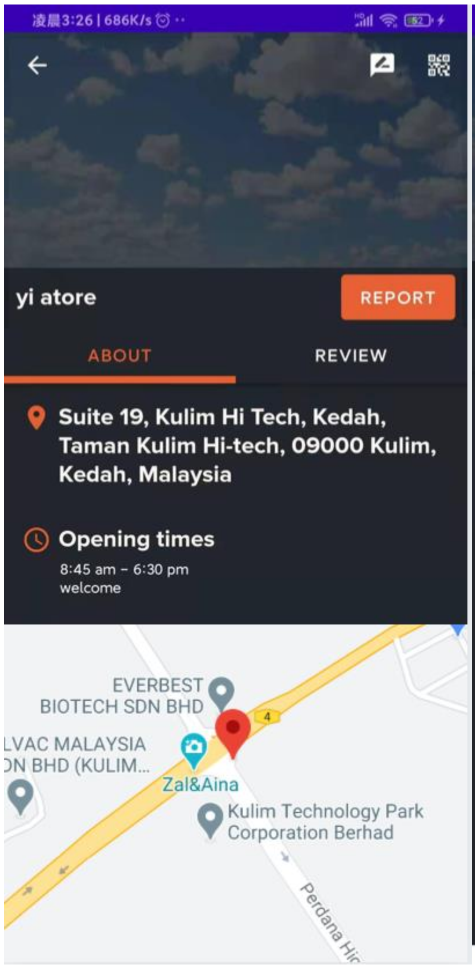


Figures 9 Merchant Item Activity

Figures 9 is the screenshot of the Merchant Item Activity. Merchant Item Activity shows the items in the menu customized by the merchant. The first figure shows the customer view of the Merchant Item Activity whereas the second figure shows the merchant view of the Merchant Item Activity.

In the customer view of this activity, in the top right corner, there are 3 buttons of 'Favorite', 'Share' and 'Info'. 'Favorite' button allows user to save the respective menu to the favorite list, 'Share' button allows user to share the menu through email or social media whereas the 'Info' button directs the user to the Merchant Info Activity.

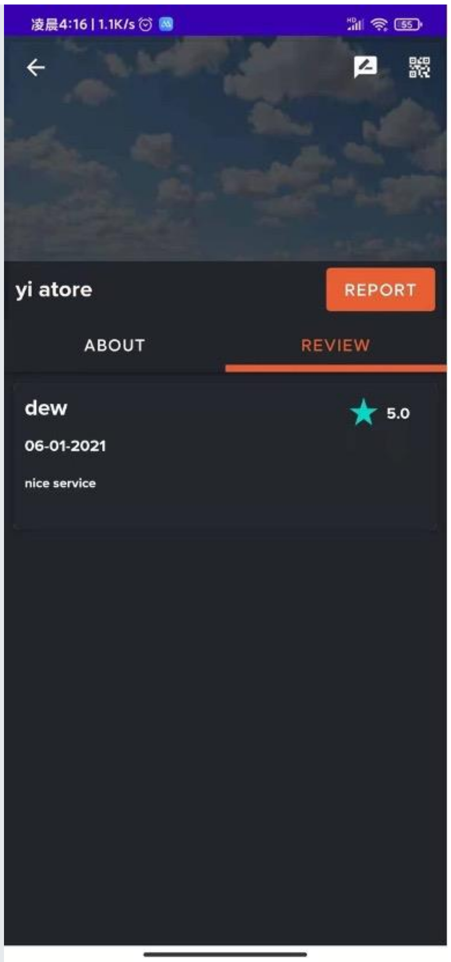
Three buttons are available in the top right corner for Merchant which are 'Edit Menu' , 'Share' and 'Info'. In the merchant view of this activity, merchant can edit the menu by clicking on the 'Edit Menu' button to the left of the 'share' button.



Suite 19, Kulim Hi Tech, Kedah, Taman Kulim Hi-tech, 09000 Kulim, Kedah, Malaysia

Opening times

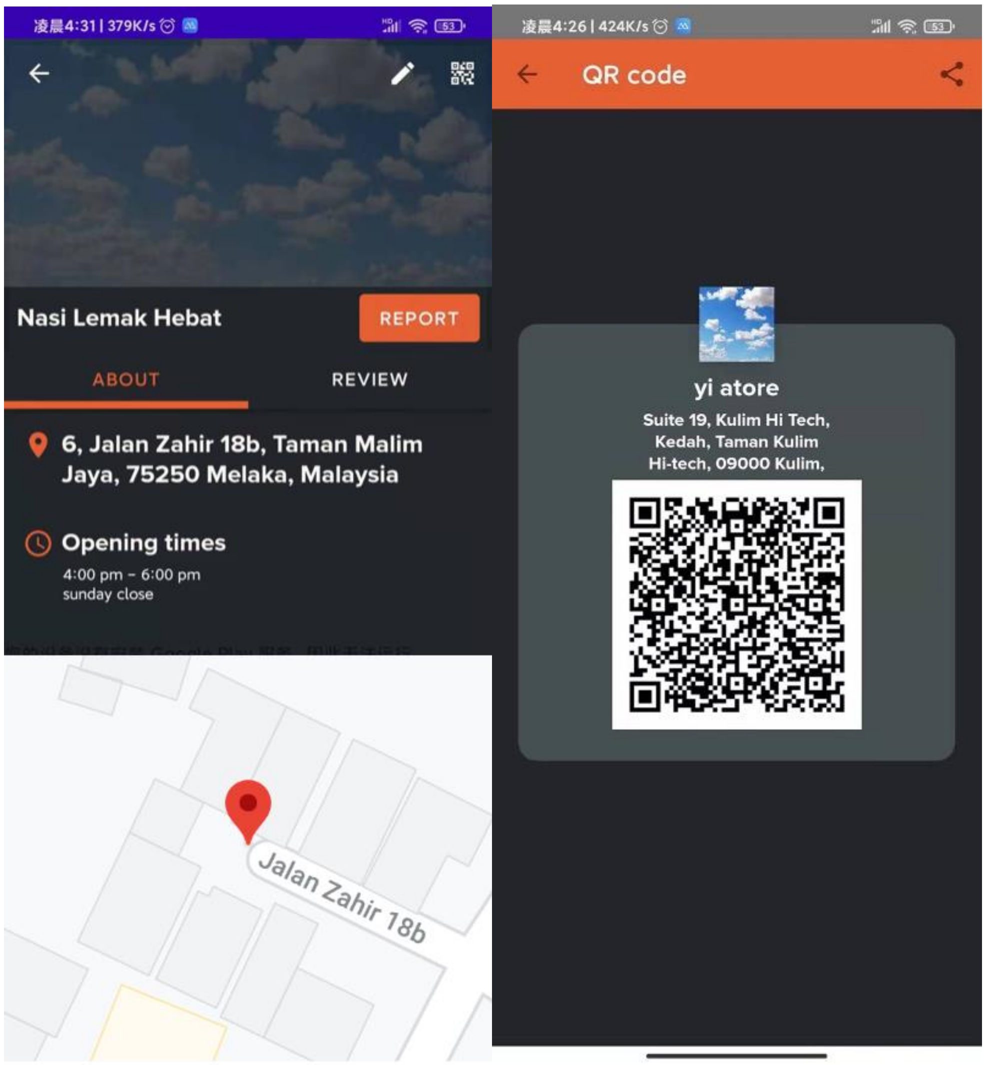
8:45 am - 6:30 pm
welcome



dew ★ 5.0

06-01-2021

nice service



Nasi Lemak Hebat

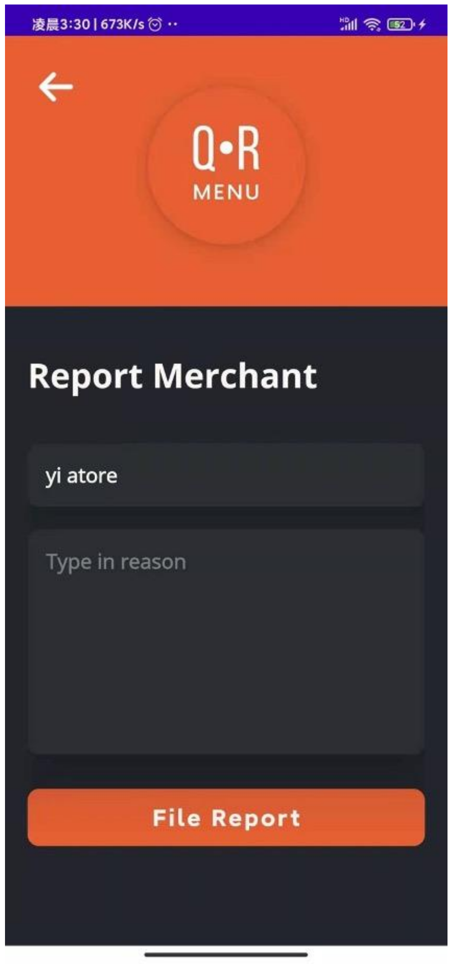
6, Jalan Zahir 18b, Taman Malim Jaya, 75250 Melaka, Malaysia

Opening times
4:00 pm - 6:00 pm
sunday close

yi atore
Suite 19, Kulim Hi Tech, Kedah, Taman Kulim Hi-tech, 09000 Kulim,

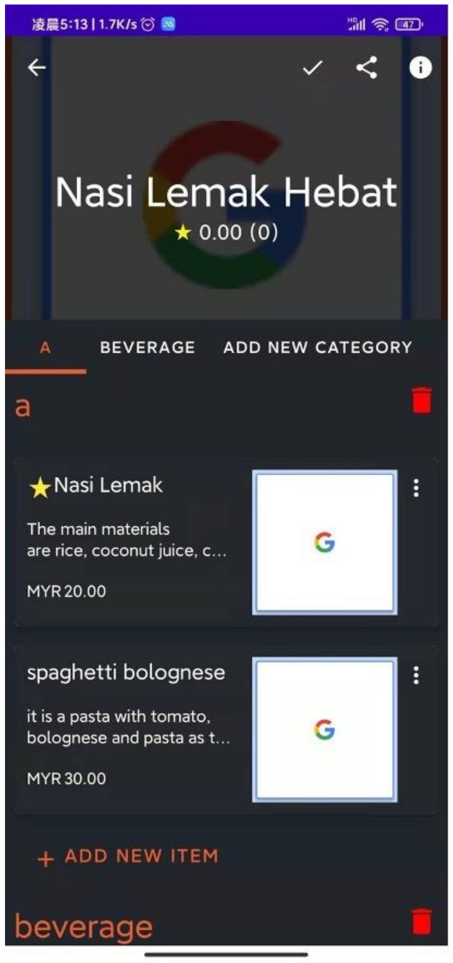
Figures 10 Merchant Info Activity

Figures 10 show the Merchant Info Activity. First and second figures show the customer view of this activity whereas fourth figure shows the merchant view. In the first figure, merchant information such as shop name, shop address, Google Map location will be shown. In the second figure, reviews about the menu are shown. Also, there is a ‘Report’ button beside the shop name which allows customer to report the merchant for the occurrence of inappropriate items appearing in the menu. In the top right corner, there are 2 buttons of 'Review' and 'QR code'. 'Review' button allows users to rate the store and write reviews. Clicking the 'QR code' will generate the QR code of the store. While in the merchant view, the two buttons available in the top right corner are 'Edit Menu' and 'QR code'. Merchant can edit the menu by clicking on the' Edit Menu' button.



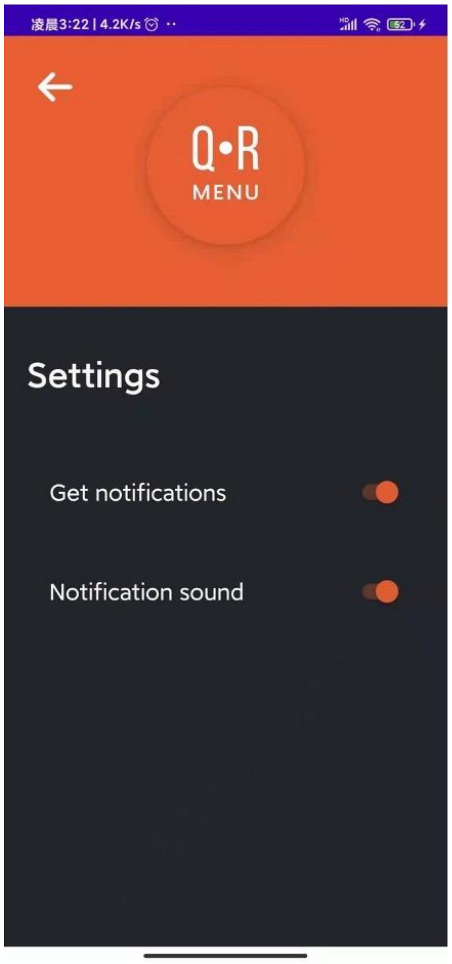
Figures 11 Report Activity

Figures 11 shows the Report Activity. Users are able to file a report against a merchant by typing in the reason for filing the report. The report will then be emailed to the mobile app admin.



Figures 12 New Item Form Activity

Figures 12 shows the New Item Form Activity. The New Item Form Activity is only available to users logged in as 'Merchant' role. Merchant can add or delete items or category from the menu in this activity. After merchant added the menu items and clicked on the confirm button, the menu will be generated.



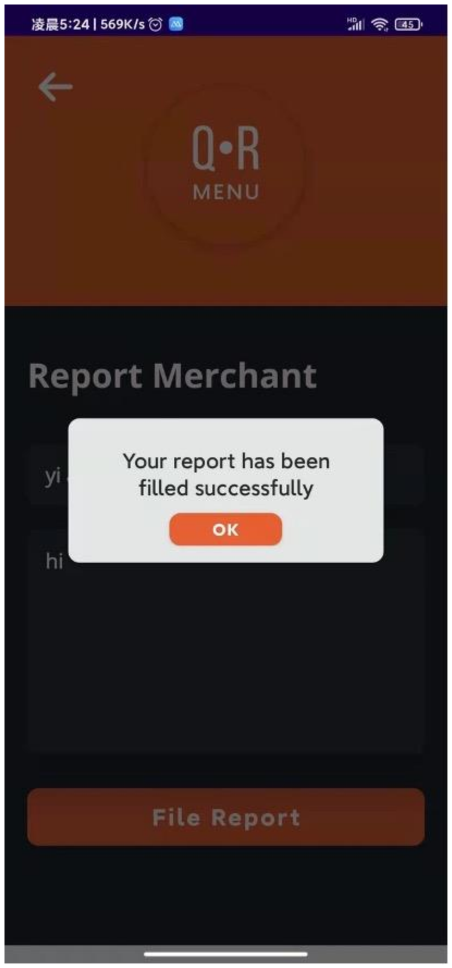
Settings

Get notifications

Notification sound

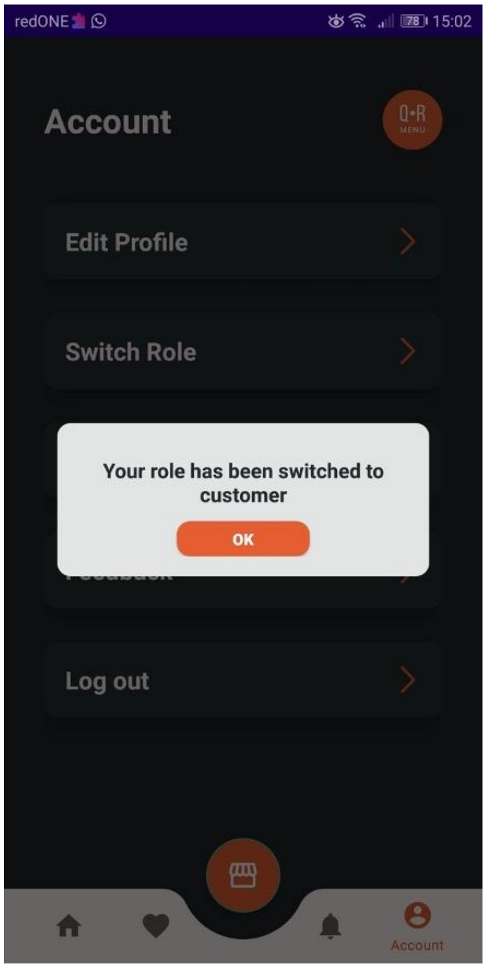
Figures 13 Setting Activity

In the Setting activity, users can manage notification settings, and manage application permission settings.



Your report has been filled successfully

OK



Your role has been switched to customer

OK

Figures 14 Result Dialog

Figures 14 shows the Result Dialog. Result Dialog displays whether the form submission of filing a report or role switching is successful or failed.

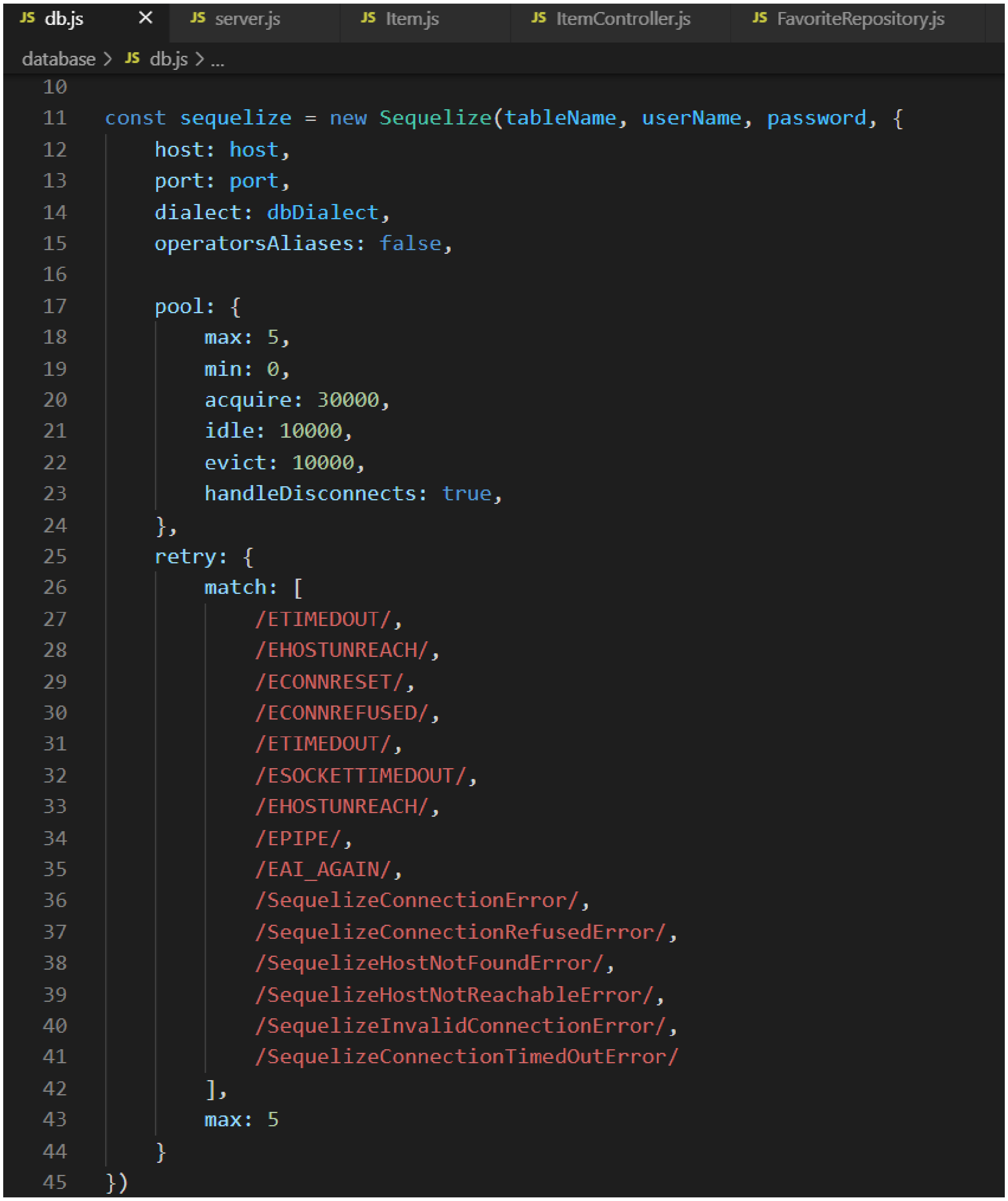
d. Proof of successful implementation for evaluation criteria

Criteria 1: Database

As for the database, we are using Sequelize, an Object Relational Mapping Library for Express JS, where it will handle the connection open and will store the connection info when another query is triggered. It mitigates the issue where the connection is not close and causing the memory leak issue. Sequelize also helps in table creation, if we migrate our app to a new environment, we will not require to generate the tables ourselves in the new environment as the table creation can be done by Sequelize. Also, Sequelize is database independent, if we change from MySQL to Postgres SQL, we just need to change the db Dialect in the setup function.

We have chosen MySQL rather than SQL Lite because SQL Lite only runs on mobile and it is impossible for a customer to retrieve data from SQL Lite that resides in other people mobile phones.

The following code is used to initialize the connection. We need to provide the host, port and dialect for the project, the host, port and the dialect is currently provided by the environment file. The environment file is used to store sensitive api data and it shouldn't be uploaded to any version control system, eg. Github



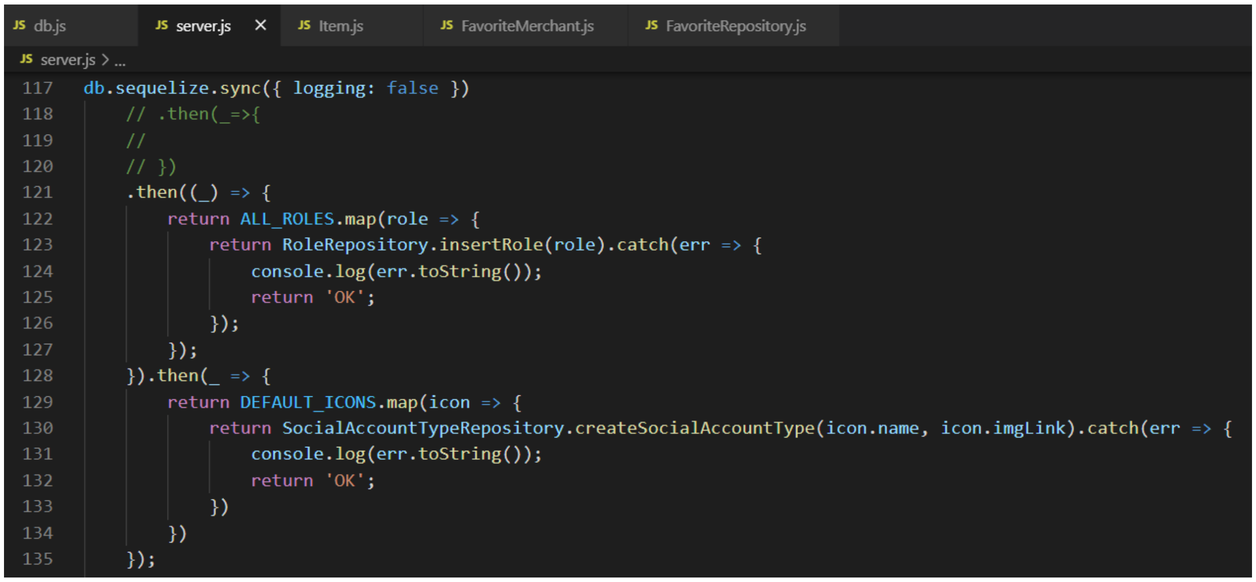
```

const sequelize = new Sequelize(tableName, userName, password, {
    host: host,
    port: port,
    dialect: dbDialect,
    operatorsAliases: false,

    pool: {
        max: 5,
        min: 0,
        acquire: 30000,
        idle: 10000,
        evict: 10000,
        handleDisconnects: true,
    },
    retry: {
        match: [
            /ETIMEDOUT/,
            /EHOSTUNREACH/,
            /ECONNRESET/,
            /ECONNREFUSED/,
            /ETIMEDOUT/,
            /ESOCKETTIMEDOUT/,
            /EHOSTUNREACH/,
            /EPIPE/,
            /EAI_AGAIN/,
            /SequelizeConnectionError/,
            /SequelizeConnectionRefusedError/,
            /SequelizeHostNotFoundError/,
            /SequelizeHostNotReachableError/,
            /SequelizeInvalidConnectionError/,
            /SequelizeConnectionTimedOutError/
        ],
        max: 5
    }
})
```

Figure d.1.1 Screenshot of db.js

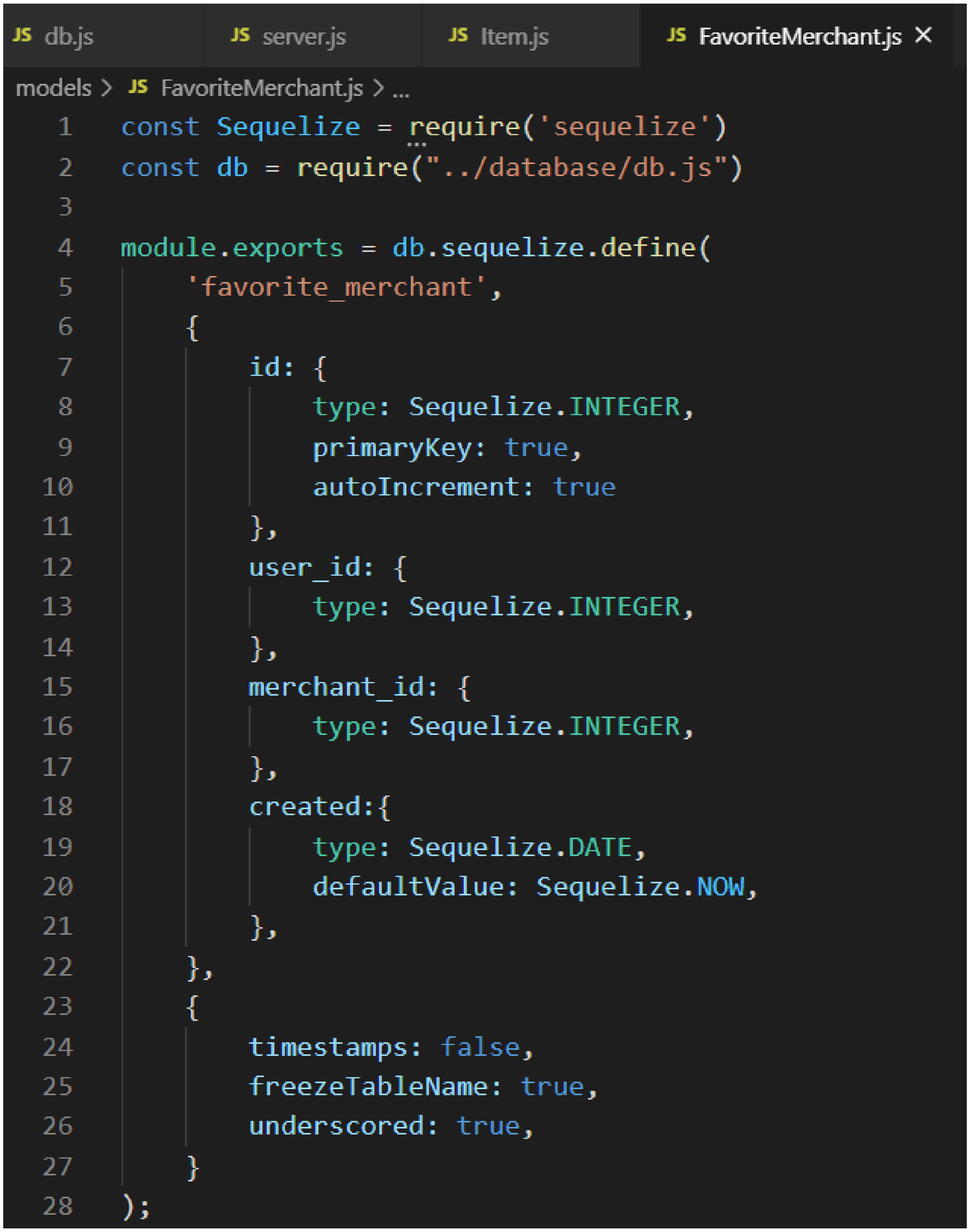
This particular code is to start the database connection, the db.sequelize.sync() function will start the connection and sync the database with the local sequelize model. The then function after the sync function will be triggered every time the server.



```javascript
db.sequelize.sync({ logging: false })
    // .then(_=>{
    //
    // })
    .then((_) => {
        return ALL_ROLES.map(role => {
            return RoleRepository.insertRole(role).catch(err => {
                console.log(err.toString());
                return 'OK';
            });
        });
    }).then(_ => {
        return DEFAULT_ICONS.map(icon => {
            return SocialAccountTypeRepository.createSocialAccountType(icon.name, icon.imgLink).catch(err => {
                console.log(err.toString());
                return 'OK';
            })
        })
    });
```

Figure d.1.2 Screenshot of server.js

The following is the sample model file for Sequelize, the database field will be mapped to their respective model class.



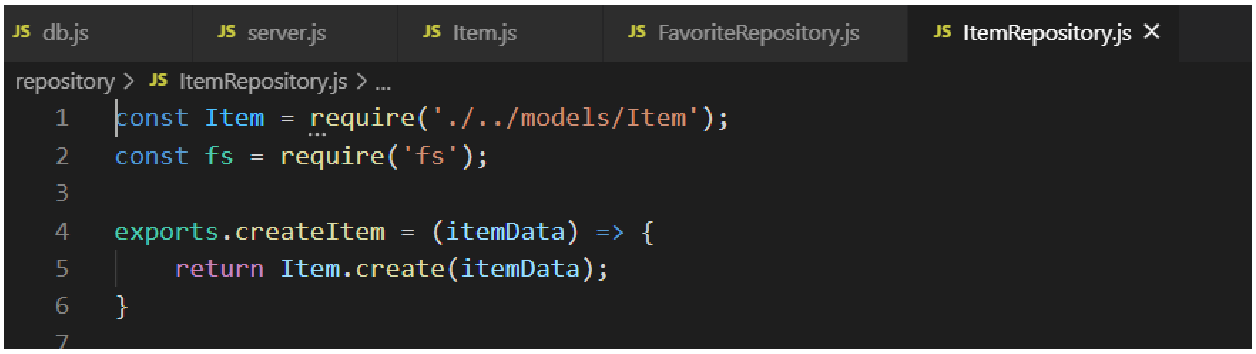
```javascript
const Sequelize = require('sequelize')
const db = require("../database/db.js")

module.exports = db.sequelize.define(
    'favorite_merchant',
    {
        id: {
            type: Sequelize.INTEGER,
            primaryKey: true,
            autoIncrement: true
        },
        user_id: {
            type: Sequelize.INTEGER,
        },
        merchant_id: {
            type: Sequelize.INTEGER,
        },
        created:{
            type: Sequelize.DATE,
            defaultValue: Sequelize.NOW,
        },
    },
    {
        timestamps: false,
        freezeTableName: true,
        underscored: true,
    }
);
```

Figure d.1.3 Screenshot of FavoriteMerchant.js

This is the sample repository class where the database query will be generated by the Sequelize.

The following is the code for creating item using Sequelize. The create function will trigger a create sql command.



db.js | server.js | Item.js | FavoriteRepository.js | ItemRepository.js

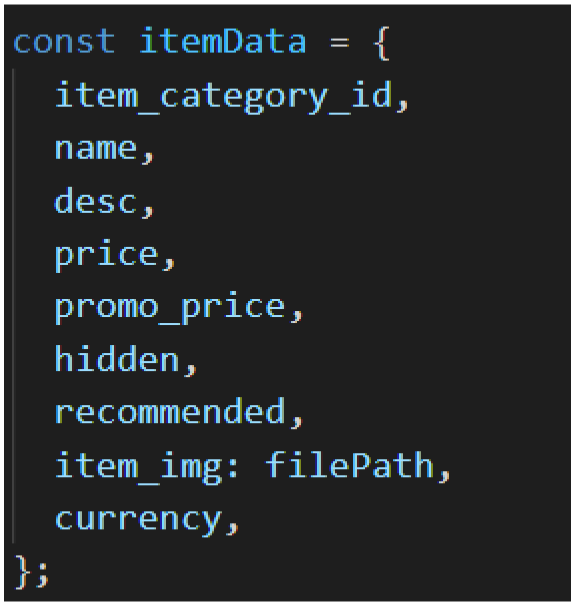
repository > ItemRepository.js > ...

```
const Item = require('./../models/Item');
const fs = require('fs');

exports.createItem = (itemData) => {
    return Item.create(itemData);
}
```

Figure d.1.4 Screenshot of ItemRepository.js

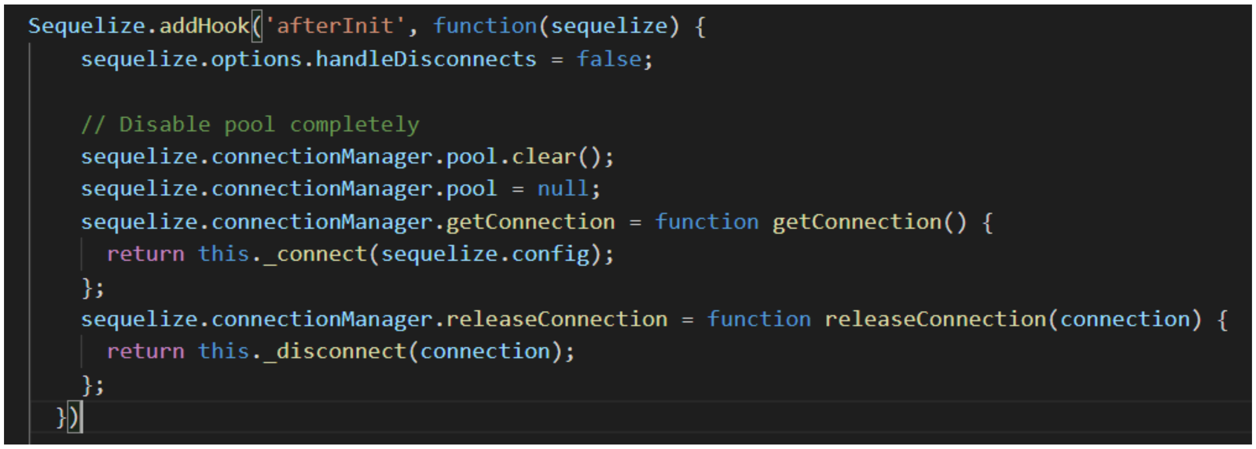
The following is the item data field



```
const itemData = {
  item_category_id,
  name,
  desc,
  price,
  promo_price,
  hidden,
  recommended,
  item_img: filePath,
  currency,
};
```

Figure d.1.5 Screenshot of itemData field in ItemRepository.js

By default, Sequelize will handle the connection for us. Since closing of the connection is part of our assessment, we have added a hook in the sequelize config that allows us to handle the connection manually. The this._connect function corresponds to the code to startup the connection with custom configuration. And the releaseConnection function is triggered if the Sequelize is disconnected. If the Sequelize connection is disconnected, the disconnect() function will help to close the mysql connection with the database.

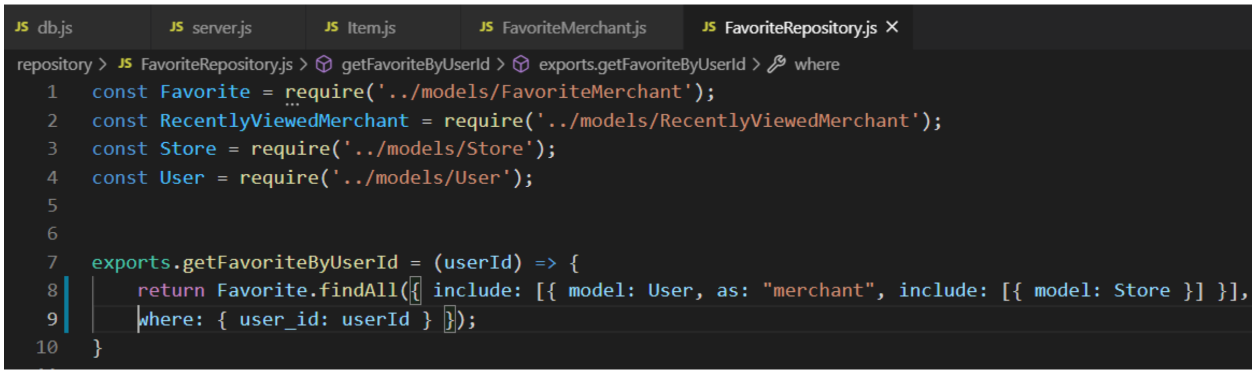


```
Sequelize.addHook('afterInit', function(sequelize) {
    sequelize.options.handleDisconnects = false;

    // Disable pool completely
    sequelize.connectionManager.pool.clear();
    sequelize.connectionManager.pool = null;
    sequelize.connectionManager.getConnection = function getConnection() {
      return this._connect(sequelize.config);
    };
    sequelize.connectionManager.releaseConnection = function releaseConnection(connection) {
      return this._disconnect(connection);
    };
  })
```

Figure d.1.6 Screenshot of Sequelize.addHook

Code to retrieve all favorite store list. In this case, it is to carry out the select * function using the sequelize library.



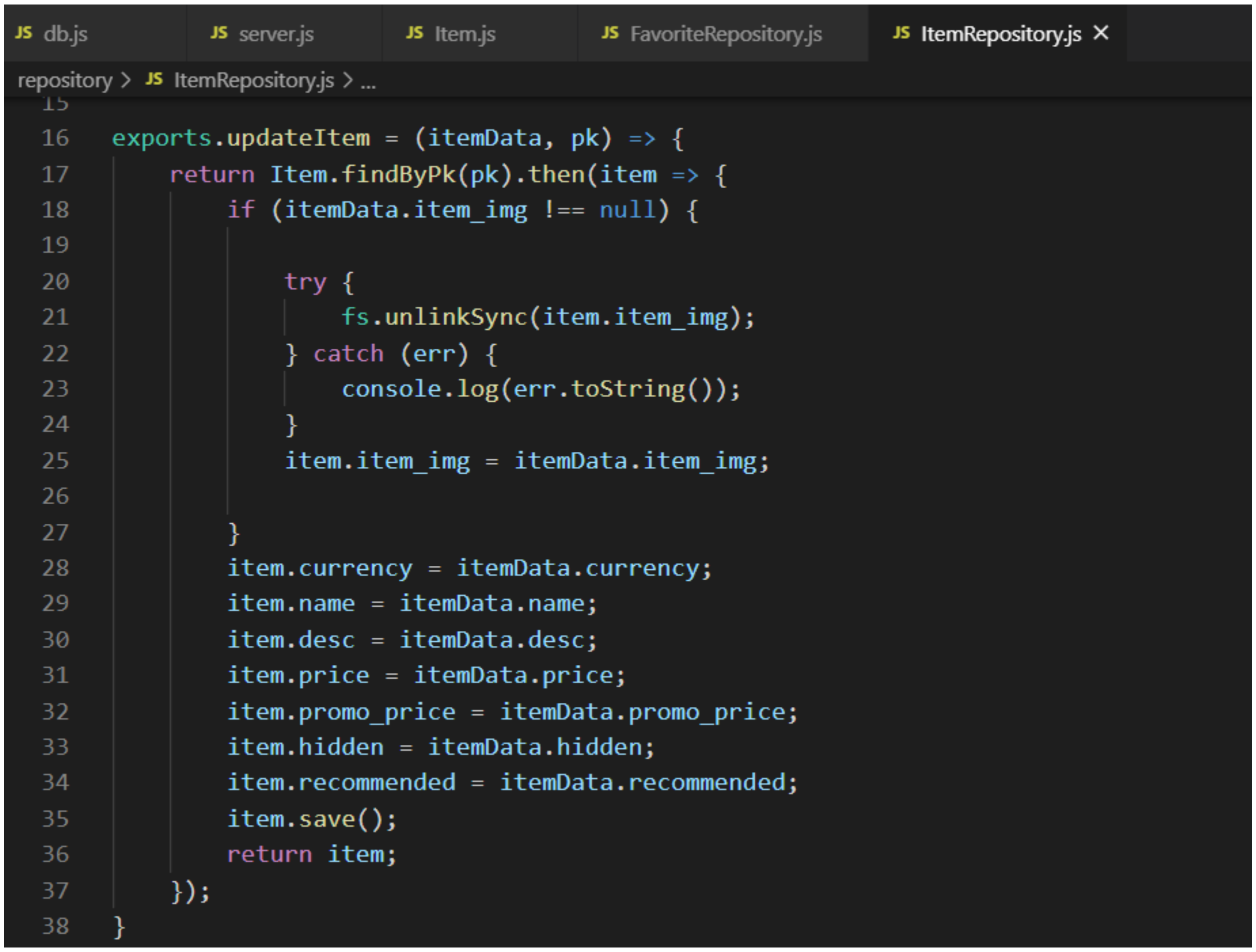
db.js
server.js
Item.js
FavoriteMerchant.js
FavoriteRepository.js
repository > FavoriteRepository.js > getFavoriteByUserId > exports.getFavoriteByUserId > where

```
const Favorite = require('../models/FavoriteMerchant');
const RecentlyViewedMerchant = require('../models/RecentlyViewedMerchant');
const Store = require('../models/Store');
const User = require('../models/User');


exports.getFavoriteByUserId = (userId) => {
    return Favorite.findAll({ include: [{ model: User, as: "merchant", include: [{ model: Store }] }],
    where: { user_id: userId } });
}
```

Figure d.1.7 Screenshot of getFavoriteByUserId in FavoriteRepository.js

The following is the code to update a row. To update a row, we will need to obtain the current row reference, update the object reference and call save(). The save function will trigger an update sql command to update the fields.



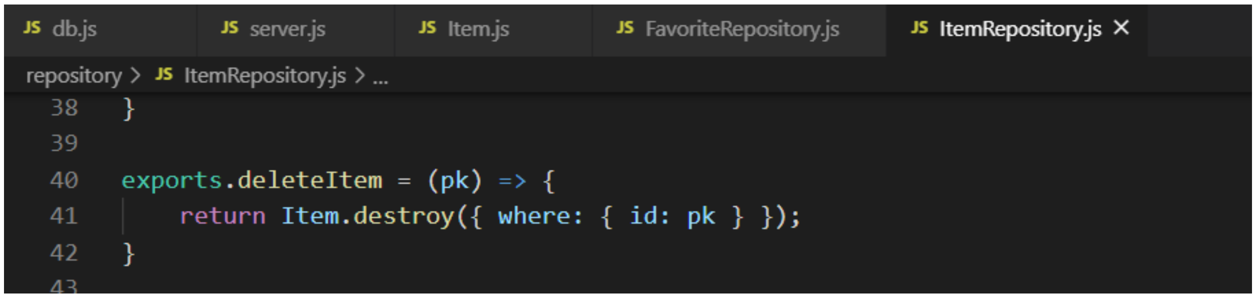
```
exports.updateItem = (itemData, pk) => {
    return Item.findByPk(pk).then(item => {
        if (itemData.item_img !== null) {

            try {
                fs.unlinkSync(item.item_img);
            } catch (err) {
                console.log(err.toString());
            }
            item.item_img = itemData.item_img;

        }
        item.currency = itemData.currency;
        item.name = itemData.name;
        item.desc = itemData.desc;
        item.price = itemData.price;
        item.promo_price = itemData.promo_price;
        item.hidden = itemData.hidden;
        item.recommended = itemData.recommended;
        item.save();
        return item;
    });
}
```

Figure d.1.8 Screenshot of updateItem in ItemRepository.js

The following is the code to delete an item from the database. The destroy function is built into the Sequelize model class, by calling it, it will execute a “Delete” sql command to the database.



```
}

exports.deleteItem = (pk) => {
    return Item.destroy({ where: { id: pk } });
}
```

Figure d.1.9 Screenshot of deleteItem in ItemRepository.js

Criteria 2: Different views and view groups

View groups/Layouts

1. LinearLayout

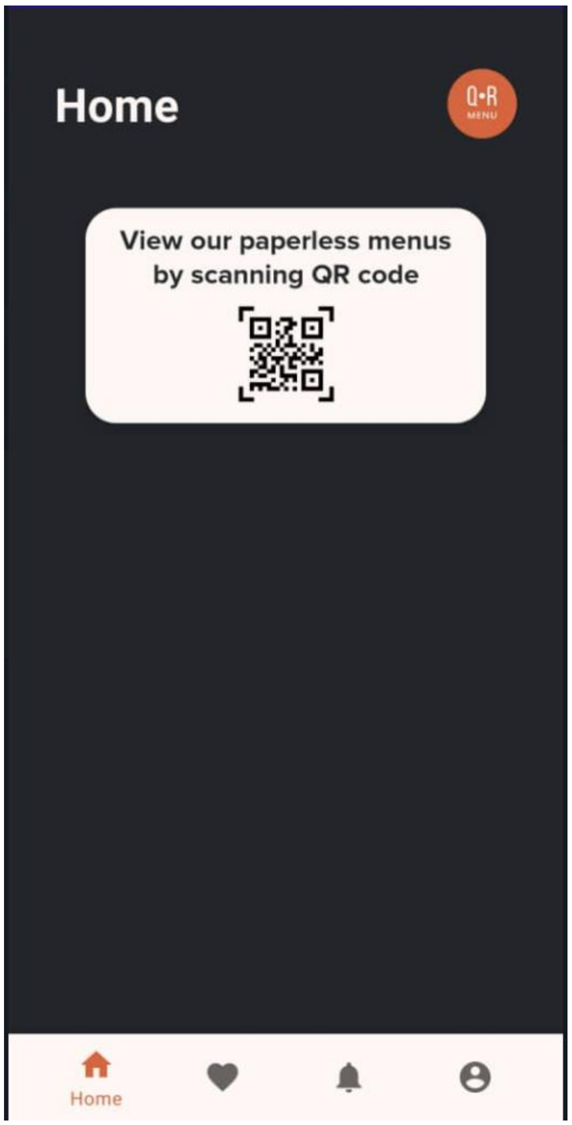


Figure d.2.1 Home Fragment

Figure d.2.1 shows an example of a fragment (HomeFragment) developed using the LinearLayout. A LinearLayout with "vertical" orientation is used here with 2 components in it, which are the top component (title and logo using ContraintLayout), and a CardView.

2. RelativeLayout

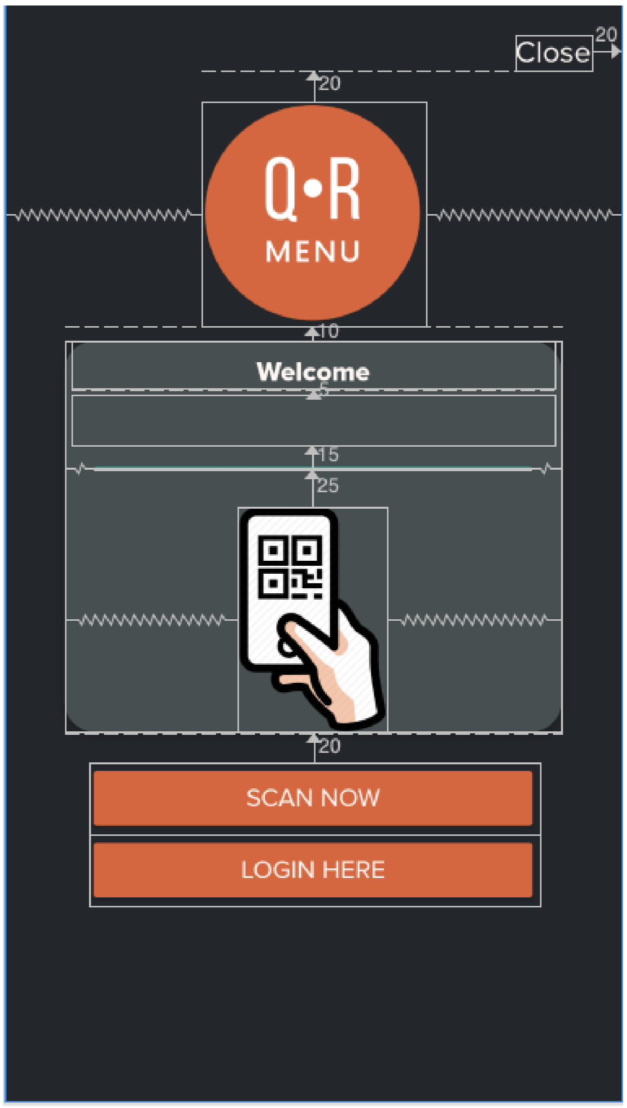
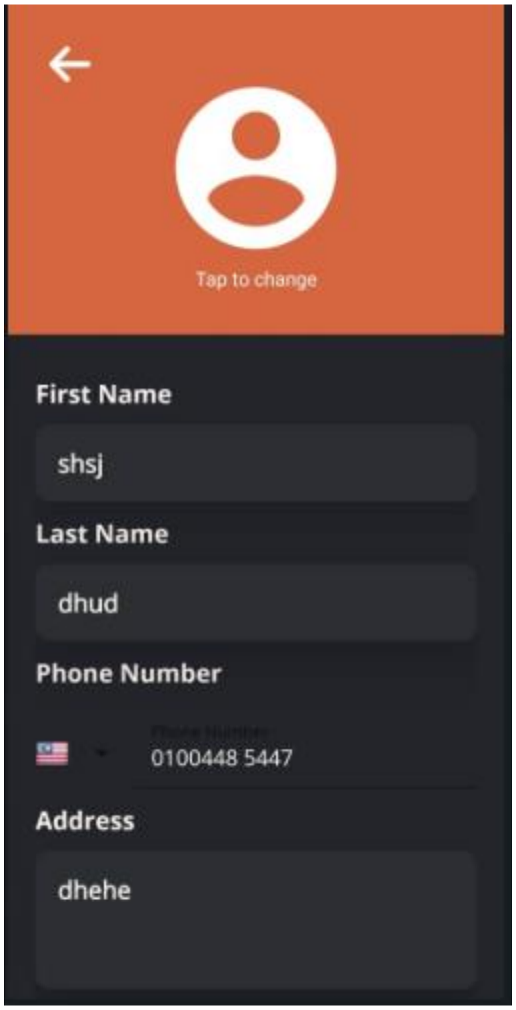


Figure d.2.2 Layout of QR Scan Activity

Figure d.2.2 shows an example of a page in the QR Scan Activity developed using the RelativeLayout. All the child views are displayed in relative to other views or the parent.

3. ConstraintLayout



shsj

dhud

0100448 5447

dhehe

Figure d.2.3 EditProfileActivity

Figure d.2.3 shows the EditProfileActivity which consists of a ConstraintLayout. In the ConstraintLayout, each view must contain at least one horizontal and one vertical constraint.

4. CoordinatorLayout

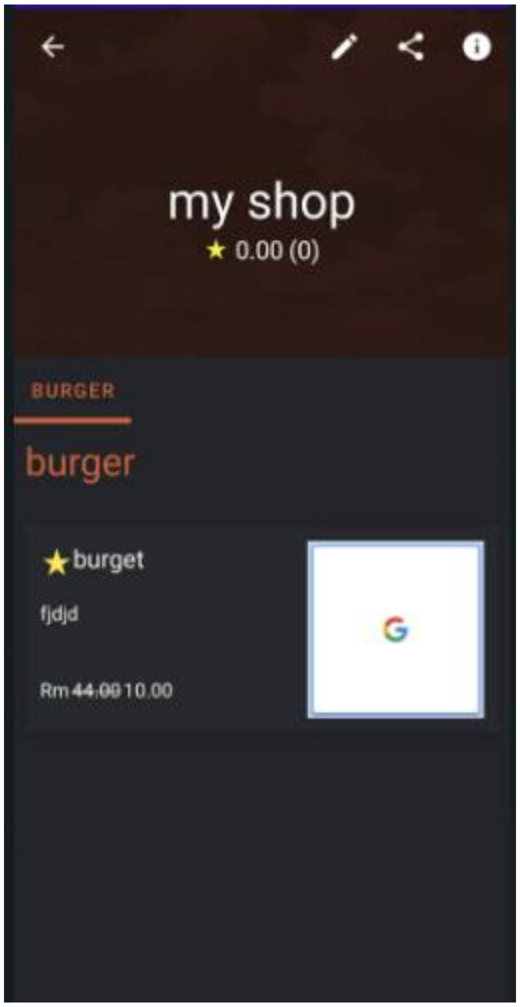


Figure d.2.4 Merchant Activity

Figure d.2.4 shows the MerchantActivity developed using the CoordinatorLayout. It is a super-powered FrameLayout intended for acting as a container for specific interaction with one or more child views.

5. RecyclerView

RecyclerView is a ViewGroup that contains the views corresponding to our data. It is used in the QR Menu mobile app to efficiently display sets of data. We supply the data and define how each item looks, and the RecyclerView library dynamically creates the elements when they're needed.

6. ScrollView

ScrollView is a ViewGroup that allows the view hierarchy placed within it to be scrolled. It is used in the QR Menu mobile app whenever an activity contains many elements and needs a scrolling functionality to view all the elements.

Complex/Special Views

1. CustomPhoneInput

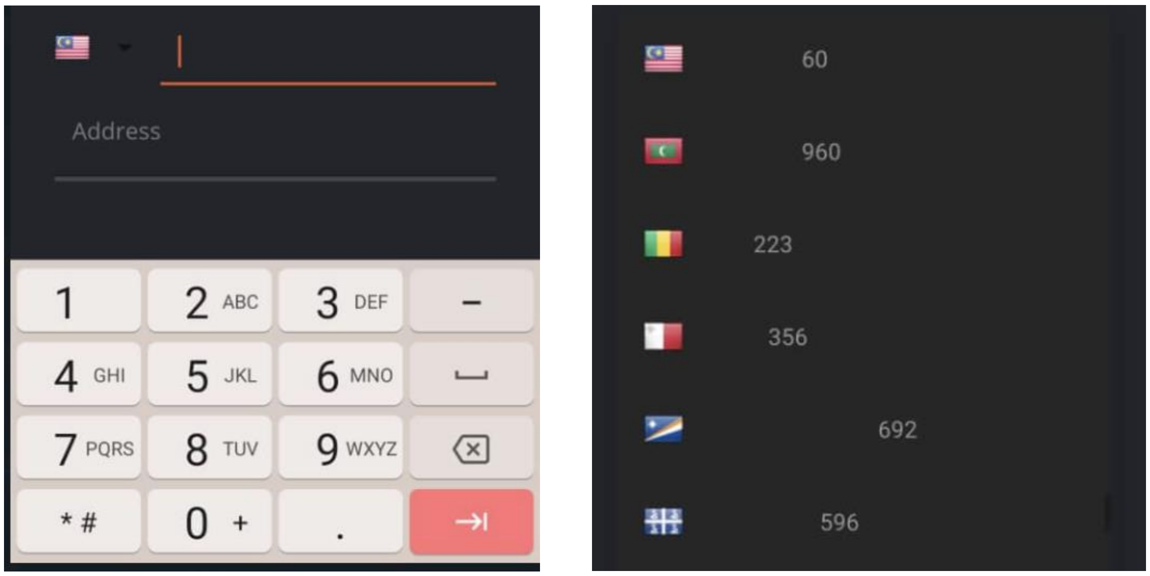


Figure d.2.5 Screenshot of CustomPhoneInput

Figure d.2.2 shows the CustomPhoneInput which is implemented in both the edit profile and merchant registration page. It is a custom component developed by us. We implemented this feature because we believe that not everyone will remember their country phone code and if we need to internationalize this application, we must identify the country phone code.

We developed this custom component as an android library, deploy it on Github and publish it on JitPack. JitPack is package repository alternative for JVM and android libraries. Besides the functionality of a normal phone number field, we include an extra functionality for users to choose their country, and the specific phone number prefix for the chosen country will be added in front.

2. LottieAnimationView

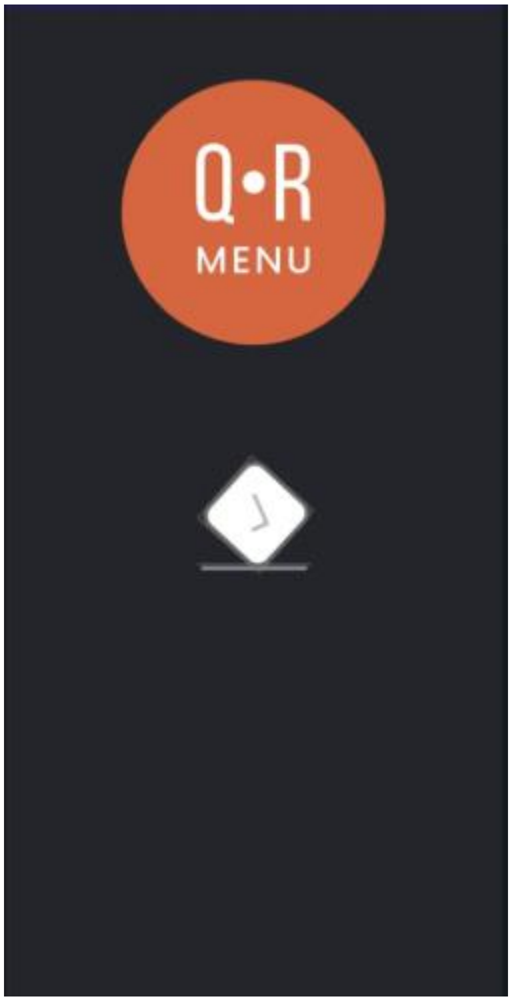
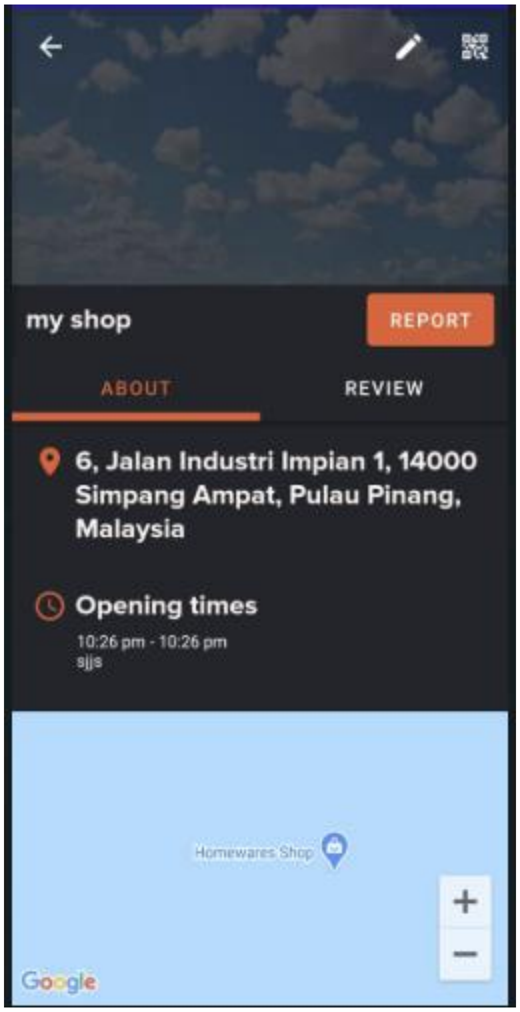


Figure d.2.6 SplashScreen Activity

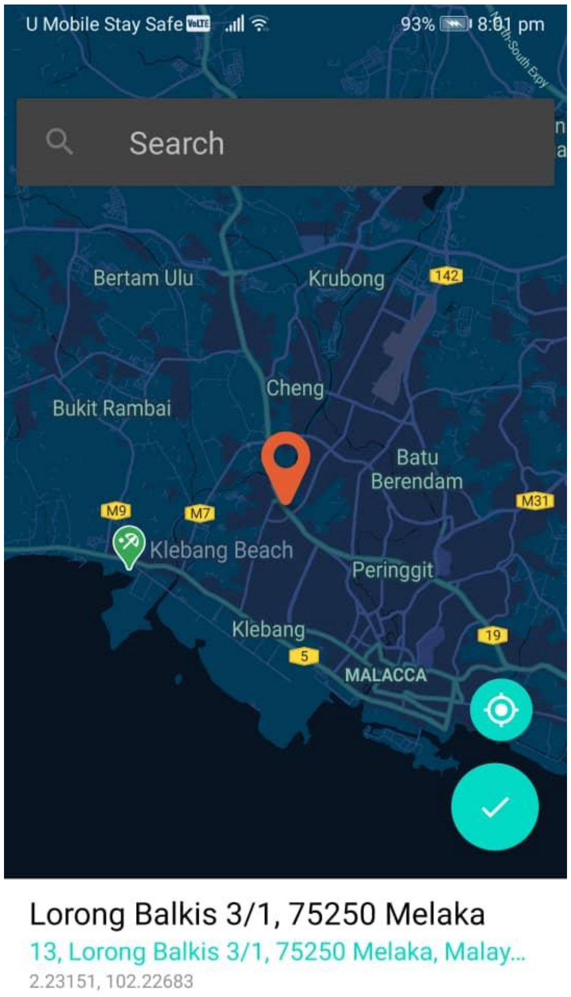
The LottieAnimationView is a third-party library created by AirBnB that allows developer to add complex animation to both mobile and web application by using single json file. This LottieAnimationView is used in the SplashScreenActivity to display the animation for the qr code.

3. MapView



6, Jalan Industri Impian 1, 14000 Simpang Ampat, Pulau Pinang, Malaysia

10:26 pm - 10:26 pm
sjjs



Lorong Balkis 3/1, 75250 Melaka
13, Lorong Balkis 3/1, 75250 Melaka, Malay...
2.23151, 102.22683

Figure d.2.7 Mapview Screenshot

The MapView is used in the MerchantInfoFragment to display the location of the merchant store and allow users to drag and drop the pin to locate their store/home location. The MapView component is a part of the Google Maps library for Android SDK. To activate the Google Maps API, we will need to generate an API key from the Google Developer Console first.

4. ShapeableImageView

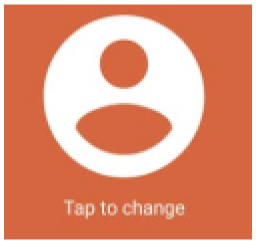
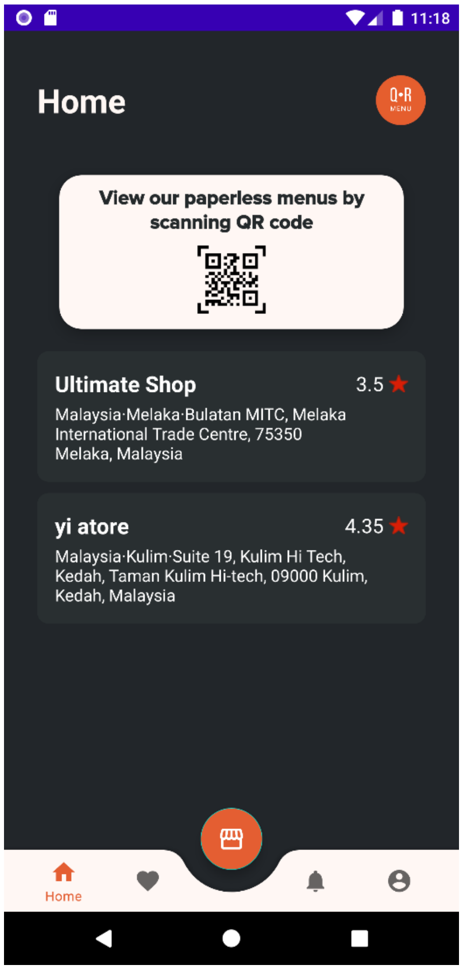


Figure d.2.8 Screenshot of Avatar Profile Image Component

The Avatar profile image component in the EditProfileActivity is implemented using the ShapeableImageView. By using the ShapableImageView, we change the shape of the image to circle.

5. BottomNavigationBar with FloatingActionButton



View our paperless menus by scanning QR code

Ultimate Shop 3.5
Malaysia·Melaka·Bulatan MITC, Melaka International Trade Centre, 75350 Melaka, Malaysia

yi atore 4.35
Malaysia·Kulim·Suite 19, Kulim Hi Tech, Kedah, Taman Kulim Hi-tech, 09000 Kulim, Kedah, Malaysia

Figure d.2.9 Home Fragment

The BottomNavigationBar is used in the main activity and it serve as access area that allows users to navigate to different pages on the main activity. The users can navigate by tapping the buttons in the navigation bar and by swiping. The swiping action is enabled by ViewPager2 library that we implemented along with the implementation of a FragmentStateAdapter abstract class that store a list of fragments available on that activity. This FragmentStateAdapter is a successor to the FragmentPagerAdapter, both were created by the developer teams at Google to solve the performance issue that will arise when we have multiple complex view on a single activity. These adapters will destroy the old view as we navigate them through the ViewPager and creates new one each time. It works just like a recyclerview and it is also an implementation of the RecyclerView.Adapter library. This approach reduces the memory usage significantly.

The center floating button is a floating action button. By adding round radius, some margins and offset on the BottomNavigationBar we will be able to design the layout design as shown in figure d2.9. We choose the BottomNavigationBar over the traditional drawer design because we want to make it easy for users to explore and switch between the fragments in a single tap. Also, we have notice that several applications from big companies such as Youtube, Instagram and Tiktok, have migrated their drawer design to bottom navigation design.

Criteria 3: Accessibility

Our target user is the shop owners (Merchant) and customers visiting the shops.
Our accessibility targets include:

1. **Ease of navigation**
   For example, we utilize the bottom navigation bar to give users confidence in knowing where they are in the mobile app.
2. **Ease of understanding the app flow**
   The flow of our mobile application is easy to understand, we also included an onboarding screens for first-time users so that they can understand our app flow easily.
3. **Robust design to accommodate both types of users**
   Our design is intended to provide a feel that accommodates both types of our target users (Merchant and Customers)
4. **Reinforcement of important information**
   We reinforce the important information using visual and textual cues such as Bold text, larger font size, high contrast colors.

Criteria 4: UI Design

Before the development phase of our mobile application, our designer first created a high-fidelity design prototype based on our app flow. Then the development is carried out in reference to the design prototype to ensure the consistency of the design throughout the mobile app. The multimedia contents such as the App Logo, icons are placed by the designer in a shared folder for developers to access.

The color palette and font family are first defined by the designer and is followed by the developers to ensure consistency of the UI design.

This combination of colors portrays a feel of minimalist and professional with sufficient contrast, accommodating both the Merchants and Customers.

Figure d.4.1 Application Logo

QRMenu
Go paperless

Figure d.4.2 Feature Graphic for Play Store

Color Palette:

1. Primary: #E45E32 🟧
2. Background: #22262A ⬛
3. Secondary gray: #24292B ⬛
4. Text: #FFF7F4 ⬜

Font family:

Open Sans(for titles), Montserrat (for subtitles and description)

Criteria 5: Extra functionalities

Major Extra Functionalities

1. Location

An extra major functionality of allowing merchant to choose his/her store location conveniently is added into the QR Menu mobile app.

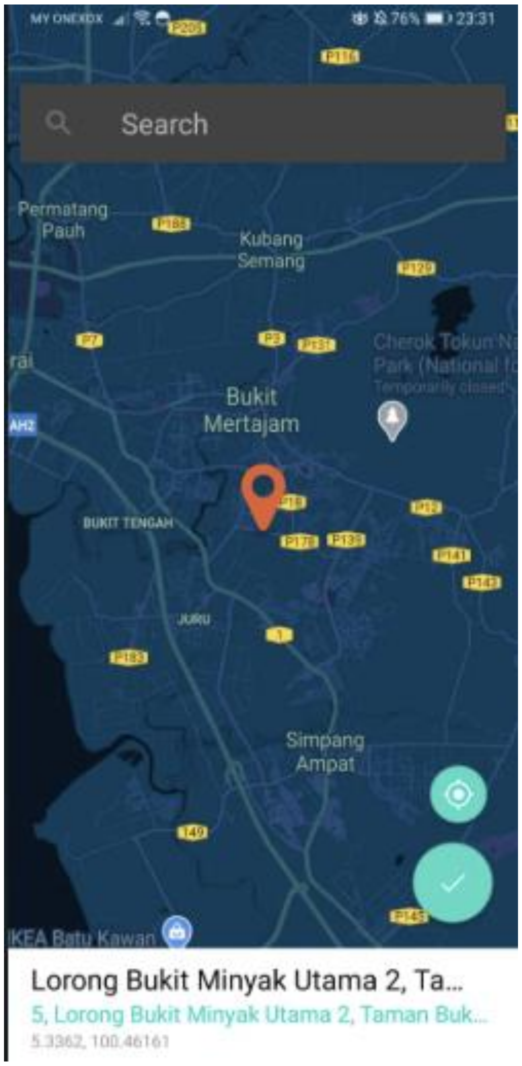


Figure d.5.1 Mapview Screenshot

Figure d.5.1 shows that merchant can choose the store location by moving and dropping the pin the MapView. This eliminates the extra step of entering the store location manually. Also, we can ensure that the location in the merchant detail page will be display correctly and the address is valid.

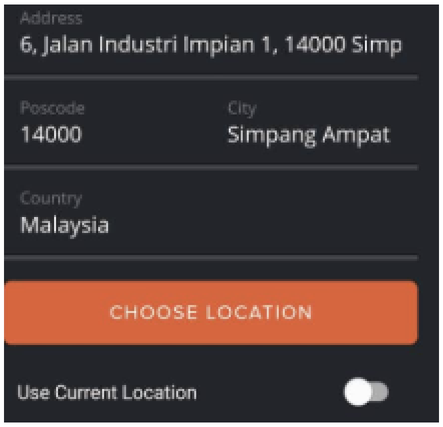
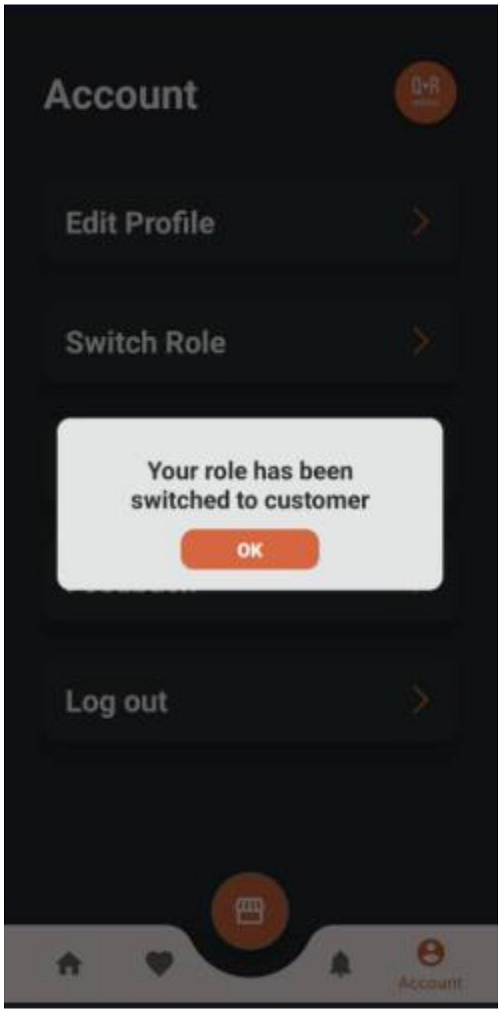

Address
6, Jalan Industri Impian 1, 14000 Simp
Poscode
14000
City
Simpang Ampat
Country
Malaysia

Figure d.5.2 Use Current Location Functionality Screenshot

Figure d.5.2 shows that the merchant can choose the store location by enabling the "Use Current Location" function and his current location will be retrieved and set up as the store location automatically. This eliminates the extra step of entering the store location manually.

2. Role Switching

An extra major functionality of allowing the registered users to switch role between Customer and Merchant is added to the QR Menu mobile app. This functionality allows users to utilize the functionalities he needs that are provided for different user roles from the mobile app. If a Merchant no longer owns a store and would like to access the functionalities provided for the Customer role, he can switch his role. The same case applies for the Customer owning a store in the future switching to Merchant role.



Your role has been switched to customer

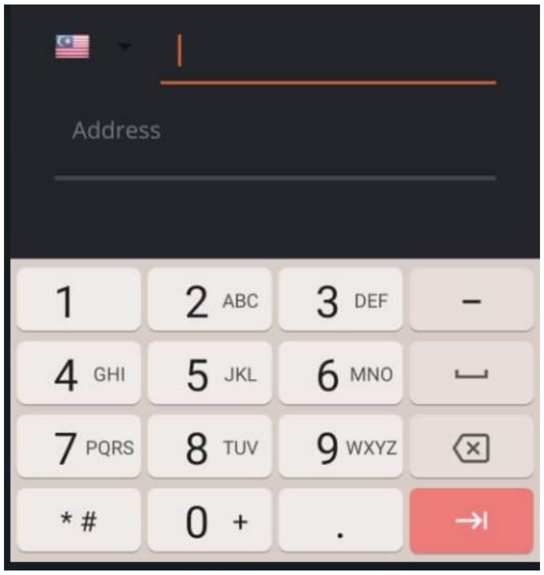
OK

Figure d.5.3 Account Fragment

Figure d.5.3 shows an example of a user registered as Merchant switching his role to Customer.

Minor Extra Functionalities

1. Custom Phone Number Input



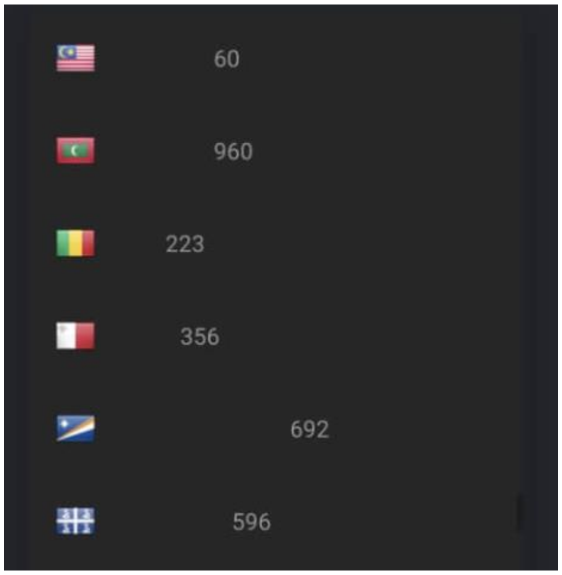


Figure d.5.4 Screenshot of CustomPhoneInput

Figure d.5.4 shows a minor extra functionality of the Custom Phone Number Input. It allows users to choose the mobile country code with a spinner. The GitHub repository link for the component is dylansalim3/android-phone-number-input (github.com))

2. User details encryption

The user details entered by the users are encrypted by our system before transferring the details through HTTP requests. The encryption is done using JWT Token with specific algorithm and secret salt value. By doing so, we can reduce the risk of data breaches, since the eavesdropper will need to identify both algorithm used and secret salt value to decode the message.

3. Password hashing

The newly created passwords of the users are hashed before being stored into the database. In case of any database attack, attacker will not able to obtain user password. Since we cannot reverse a hashing function (because data is lost during hashing), the password remains confidential even after attack.

4. Hierarchical folder structure

We have implemented a hierarchical folder structure for both frontend and backend, each class is split into different folders with specific functions. This is a minor extra functionality implemented for the developers instead of the users of mobile app.

5. MVP Architecture

We have implemented the MVP architecture in our application. We chose MVP architecture over the more popular MVVM architecture as it is easier to learn and it is one to one mapping from the presenter to the activity class. As compared to the MVVM architecture, one ViewModel can be used by multiple activity classes. This functionality is major to the development but not the users.

6.CI/CD and Publish on Google Play Store

We have developed a CI/CD pipeline in Github Actions to deploy the backend code to a VPS every time we make a push/pull request on our Github project master branch. The backend is hosted in our own VPS hosted by Aliyun. We also published our application in Play Store as a beta application. The application repository is available on dylansalim3/WIX3004_QR_Menu_App (github.com).

SCAN ME

Figure d.5.5 QR Code to our application on Google Play

Criteria 6: Error handling

Error handling in our mobile applications is implemented in both frontend and backend. Below are show examples.

Frontend – Form validation and Error occurrence

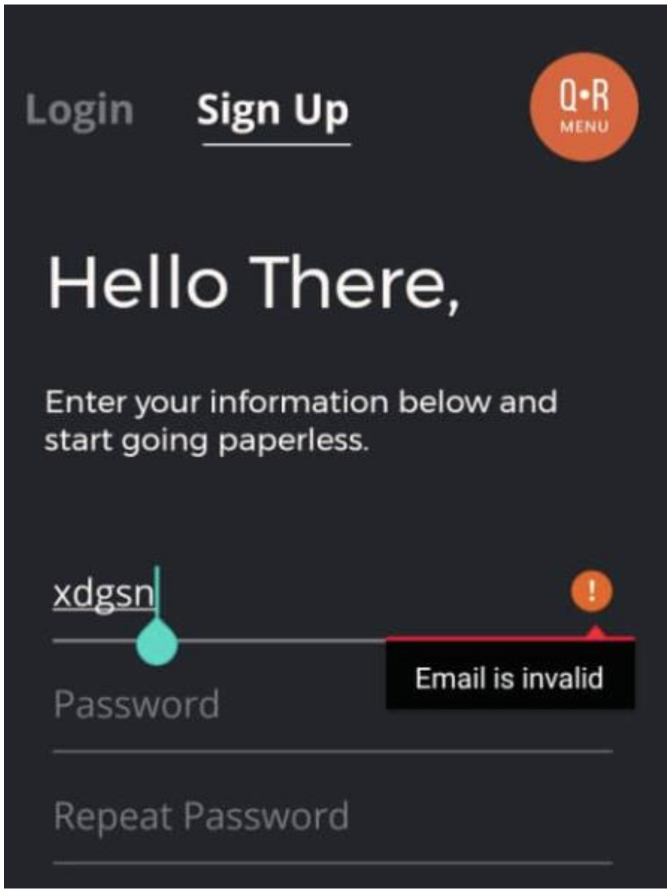
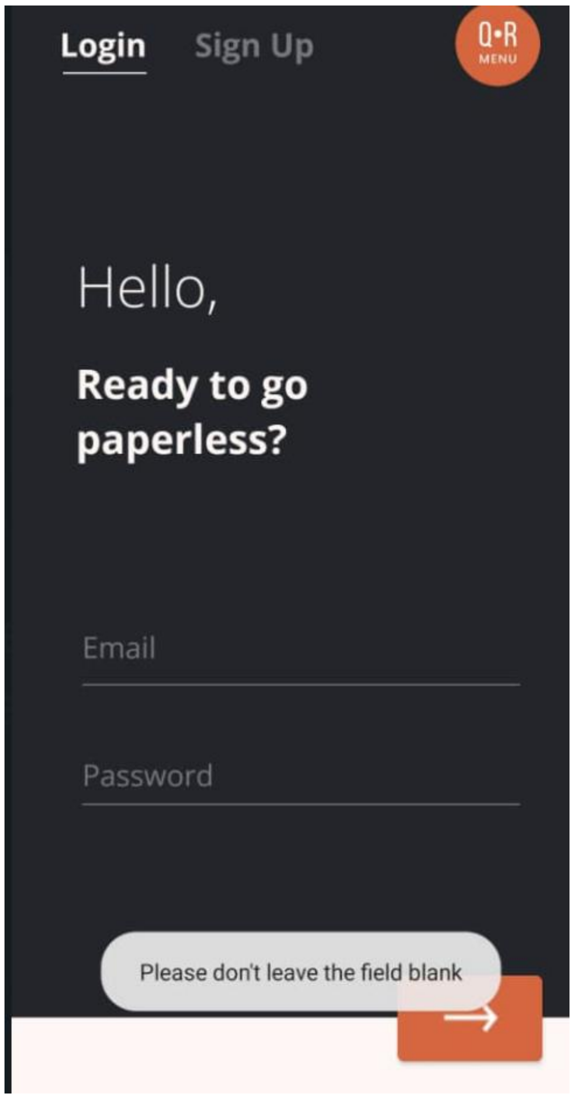


Figure d.6.1 Screenshot of Registration Fragment

Figure d.6.1 shows an example of error handling in the mobile application frontend, which is the form fields validation. Whenever user keys in an invalid email in the email field of the Sign Up page, the message of "Email is invalid" is shown, prompting user to enter a valid email format.



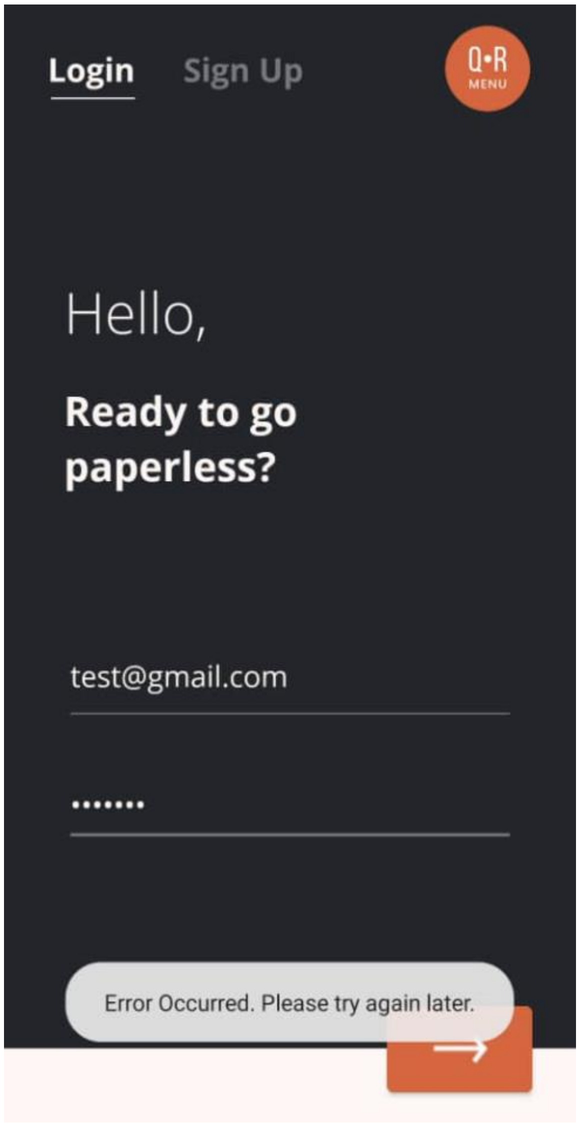
Hello,

Ready to go paperless?

Please don't leave the field blank

Figure d.6.2 Login Fragment

Figure d.6.2 shows the form validation handling, whenever the user tries to login without filling in the required fields, our mobile application will display the message of “Please don’t leave the field blank”, prompting user to fill in the required fields.

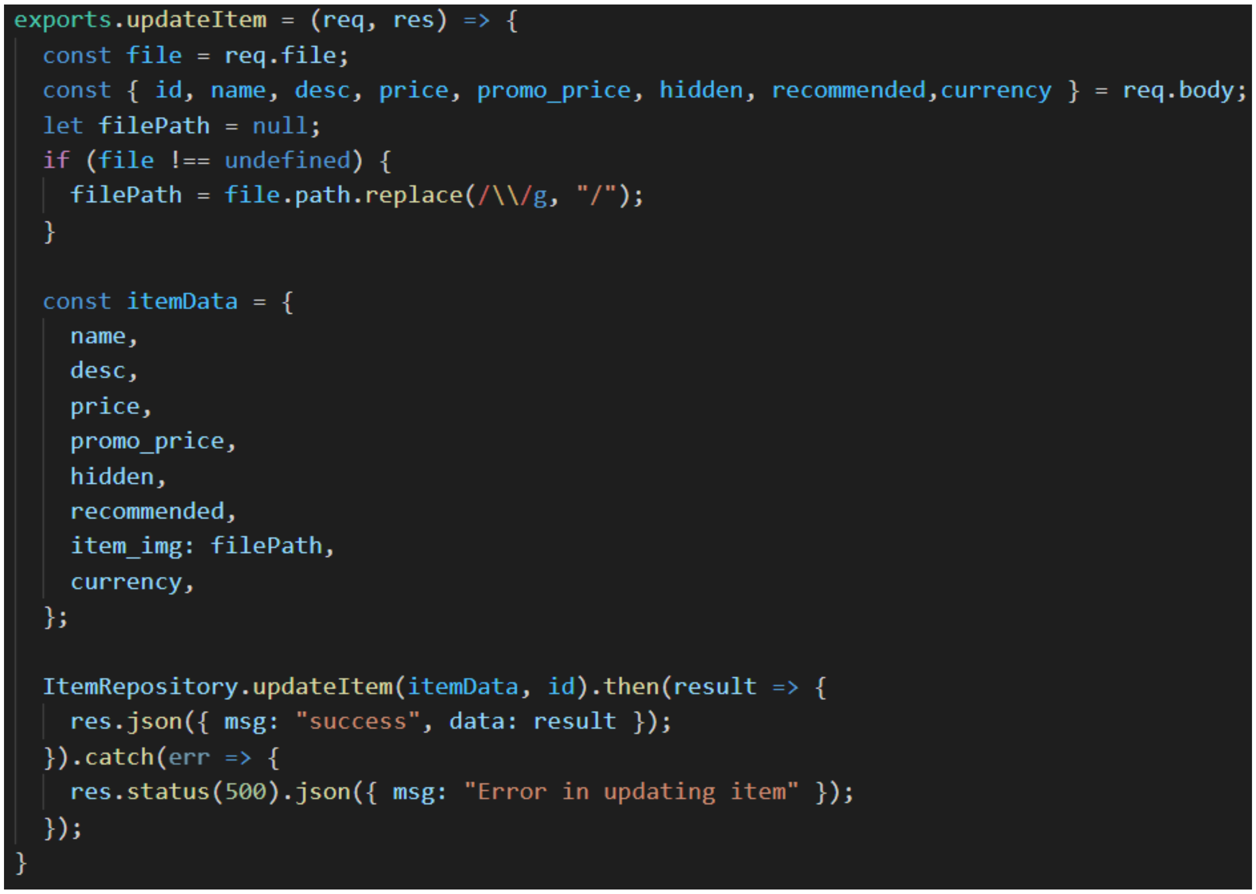


Error Occurred. Please try again later.

Figure d.6.3 Login Fragment

Figure d.6.3 shows error handling in our mobile application frontend whenever the is an error occurring, for example in the case of network fail. A toast message of error occurring will be shown to provide context to the users.

Backend error handling



```
exports.updateItem = (req, res) => {
  const file = req.file;
  const { id, name, desc, price, promo_price, hidden, recommended,currency } = req.body;
  let filePath = null;
  if (file !== undefined) {
    filePath = file.path.replace(/\\/g, "/");
  }

  const itemData = {
    name,
    desc,
    price,
    promo_price,
    hidden,
    recommended,
    item_img: filePath,
    currency,
  };

  ItemRepository.updateItem(itemData, id).then(result => {
    res.json({ msg: "success", data: result });
  }).catch(err => {
    res.status(500).json({ msg: "Error in updating item" });
  });
}
```

Figure d.6.4 Screenshot of updateItem in ItemRepository.js

Figure d.6.4 shows an example of error handling in the backend code. Whenever an error occurs in processing an API in the backend, the error response status of '500' couple with the error message will be passed to the frontend for debugging.

e. Tasks distribution table

Name	Tasks
DYLAN SALIM	• Allocate tasks to members and keep track of members’ progress • Design activity flow including allocation of functions to each activity/fragment • Define application and folder structure, common coding practices for both frontend and backend • Maintain git repository, merge pull requests • Application development – Login and registration, Merchant Item, New Item Form Activity, Merchant Info activity • Server-side development – API development, Authentication and deployment • Deploy the Node JS backend application to Heroku • Setup MySQL server • Provision the Google Play Console release status
NG JIH YANN	• UI/UX design and create graphical resources like app logo, icons, pictures. • Define color palette and font family • Create high-fidelity UI prototype using Adobe XD • Upload design resources to shared folder for teammates to access • Application development – Splash screen activity, Application styling and touchup • Server-side development – Email utility
FANG WAI NAM	• Application development – Notification fragment, Report suspicious merchant by submitting form and showing the result, Account Activity

	• Server-side development – Send push notifications and related API for reporting suspicious merchant, notification and account settings • Firebase – Notification • Publish application to Google Play
DONG YIRUI	• Application development – Favorite and recently visited merchant • Server-side development – Related API and code to support favorite merchant functionality
TAN QING LIN	• Application development – Onboarding activity, QR scanning activity, camera access • Server-side development – Related API for QR code generation when merchant create new catalog