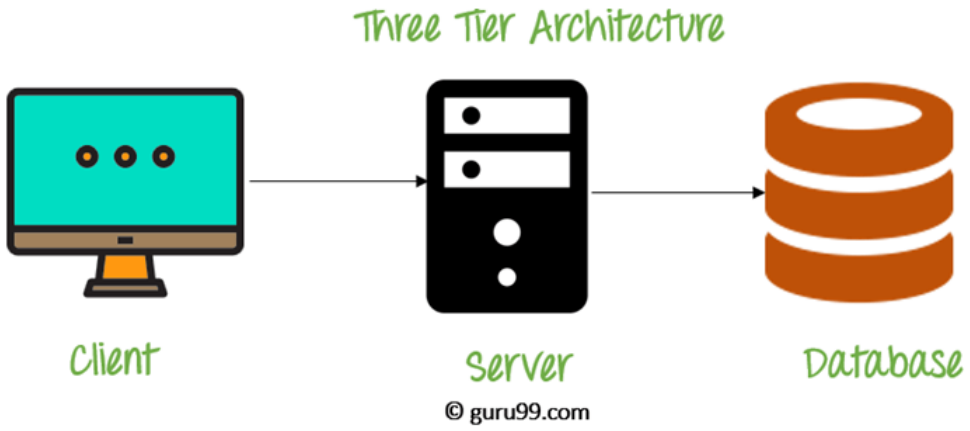
Implementation:

Architecture:

Three Tier Architecture:

A three-tier architecture is a client server architecture that develops, maintains, independently of a platform, a functional process logic, data access, computer storage and the user interface.[1] Three level architecture is a model for software and a well-established architecture of software.



[2]

Presentation Tier:

It is also defined as the Client Tier where users get direct interreact with it. It basically contains the UI and element like it. The main objective of ts is to display information for the user and to collect it. For example, this tier of the top level can operate on a web browser, as an application for the Desktop or on a GUI. Using HTML, CSS and JavaScript web presentation stages are usually developed. Depending on the platform desktop applications can be written in various languages.

Business Logic Tier:

It is the main Logic Tier and contains the Server or the Main application. It contains the code which connects with the database. The application level, also known as the logic level, is written in a programming language like Java, containing the business logic that supports the core functions of the application. Depending on how much processing power the application requires, the application type can be hosted on distributed cloud servers or on a dedicated internal server. The presentation level is derived from this level, which can also be called the middle level, the logic level, business logic or logic level.

Data Tier:

It is the Data Tier or the tier contains the database or data storing related components. Houses database servers that store and retrieve information. Data are kept independent from application servers or corporate logic in this tier, and are managed and available with programs such as MongoDB, Oracle, MySQL and Microsoft SQL Server.

Language and Technology:

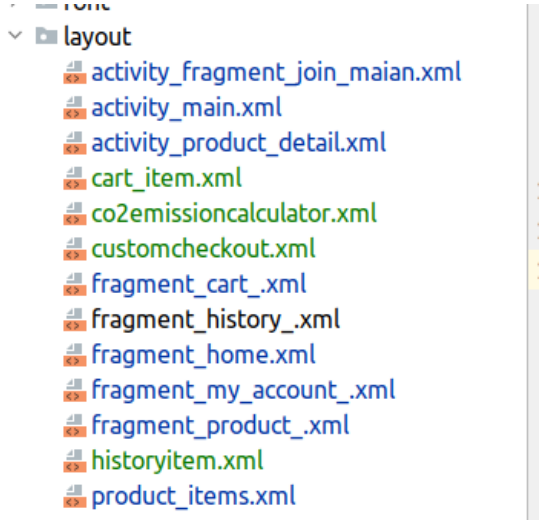
Java:

Java is a powerful programming language for all purposes. It is used for desktop and mobile applications, large-scale processing, embedded systems, etc. According to Oracle, which owns Java, Java is running on 3 billion devices around the world, making Java one of the most popular languages for programming. It is used for the development, large data processing, embedded systems, etc. of mobile and desktop apps. In our Case it is main backbone language and it is used on both android application and backend server Development. Java files format is .java and Some samples of Java Code are:

```java
package com.dao.wethemany.services;

import java.util.List;...

@Service
public class Product_Services {

    @Autowired
    private ProductRepository productRepository;

    // Method For Geting All Product
    public MessageResponse getAllProductInfo() {

        MessageResponse messageResponse=new MessageResponse();

        List<Product> returnValue=productRepository.findAll();
        if(returnValue !=null && !returnValue.isEmpty()) {
            messageResponse.setHttpStatus(HttpStatus.OK);
            messageResponse.setReturnValueList(returnValue);
            messageResponse.setReturnStatus(1);
            messageResponse.setMessage("Sucessfully Retrieved Data");
        }else {


            messageResponse.setHttpStatus(HttpStatus.BAD_REQUEST);
            messageResponse.setReturnStatus(0);
            messageResponse.setReturnValueList(null);
            messageResponse.setMessage("Sucessfully Retrieved Data");
        }

        return messageResponse;
    }

```

XML:

XML full form is extensible Markup Language. It works like markup language. It was designed in such a way to store and transport the data. Its main asset is that it is language and platform independent. The Main benefit of xml is that you can use this to capture data from a program such as Microsoft SQL and convert it into XML.Two platforms, that are generally very difficult, can communicate [8]. The XML files is in .xml format. It is used to design and develop UI Parts in the Android Application.




```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">

    <gradient
        android:startColor="@color/dark_blue"
        android:endColor="@color/light_blue"
        android:angle="0" />

    <corners android:radius="16dp" />

</shape>
```

Maven:

It is an open-source Project Management or Dependency management tools. Maven is a project management tool that offers us the ability to create various software in this life cycle. The focus of this tool is on standardizing, i.e., software development within a short time period in a standard layout. This allows us to build Java projects, but is also compatible with other languages. For structuring applications, Maven uses Extensible Markup (XML) language. It is used in Java especially in Java Spring or Spring MVC. It is known aspom.xml

WeTheManApp/pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.5.3</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.dao</groupId>
    <artifactId>WeTheManApp</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>
    <name>WeTheManApp</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-mongodb</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-mail</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>

        <!-- https://mvnrepository.com/artifact/javax.validation/validation-api -->
        <dependency>
```

```
41            <groupId>javax.validation</groupId>
42            <artifactId>validation-api</artifactId>
43            <version>2.0.1.Final</version>
44        </dependency>
45
46        <!-- https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt -->
47        <dependency>
48            <groupId>io.jsonwebtoken</groupId>
49            <artifactId>jjwt</artifactId>
50            <version>0.9.1</version>
51        </dependency>
52
53        <dependency>
54            <groupId>javax.xml.bind</groupId>
55            <artifactId>jaxb-api</artifactId>
56            <version>2.3.1</version>
57        </dependency>
58
59        <dependency>
60          <groupId>com.stripe</groupId>
61          <artifactId>stripe-java</artifactId>
62          <version>20.67.0</version>
63        </dependency>
64
65        <dependency>
66            <groupId>org.springframework.boot</groupId>
67            <artifactId>spring-boot-devtools</artifactId>
68            <scope>runtime</scope>
69            <optional>true</optional>
70        </dependency>
71        <dependency>
72            <groupId>org.projectlombok</groupId>
73            <artifactId>lombok</artifactId>
74            <optional>true</optional>
75        </dependency>
76        <dependency>
77            <groupId>org.springframework.boot</groupId>
78            <artifactId>spring-boot-starter-tomcat</artifactId>
79            <scope>provided</scope>
80        </dependency>
```

```xml
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

        <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>5.8.0-M1</version>
            <scope>test</scope>
        </dependency>


    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

Gradle:

It is open-source Project Management or Dependency Management tools just like Maven. Gradle is an open-source tool to help us develop mechanized software. Because of its high performance, this tool is used to generate different types of software. The project structure is developed in Java and a DSL based in Groovy. It develops the project structure. Gradle supports the development and deployment on different platforms of mobile and web applications. It is preferred as an official tool for the development of Android applications with its functionality [9]. It is used in the android application and known as .gradle files

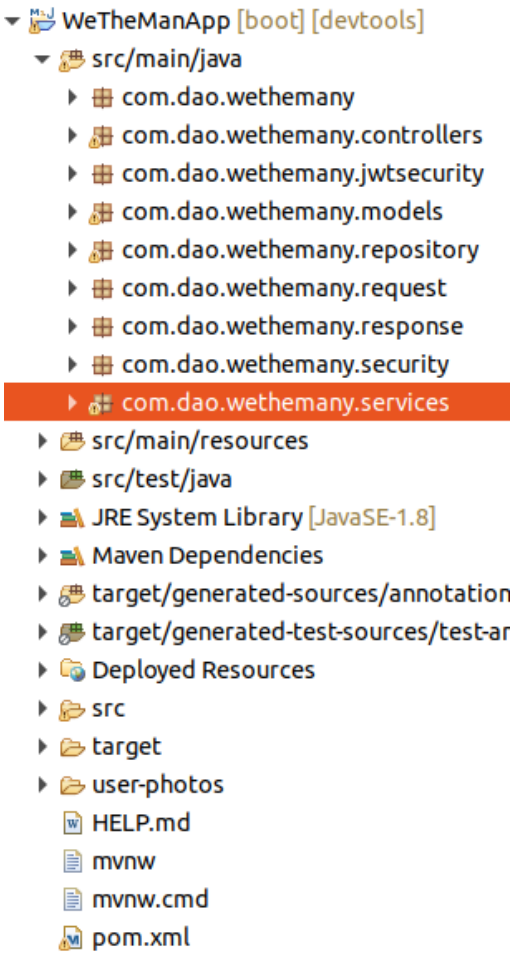
```
plugins {
    id 'com.android.application'
}
android {
    compileSdkVersion 30
    buildToolsVersion "30.0.3"

    defaultConfig {
        applicationId "com.example.wethemanyapp"
        minSdkVersion 21
        targetSdkVersion 30
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

dependencies {

    implementation 'androidx.appcompat:appcompat:1.3.0'
    implementation 'com.google.android.material:material:1.3.0'
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation 'com.github.bumptech.glide:glide:4.12.0'
    annotationProcessor 'com.github.bumptech.glide:compiler:4.12.0'
//    implementation "com.stripe:stripe-java:20.67.0"
    implementation 'com.stripe:stripe-android:17.1.0'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
}
```

Applications:

Spring Boot App:

Spring Boot is a micro-framework open source maintained by Pivotal. It provides Java developers with a platform for a Spring application that can be automatically configured for production. It is an extension of Spring MVC used for creating an API's or Microservices. Restful Api will be used for this purpose so that it can be used by any other system like android, iOS or even web applications. Spring boot app is an advanced and simplified version of spring and much e effective in now a day.

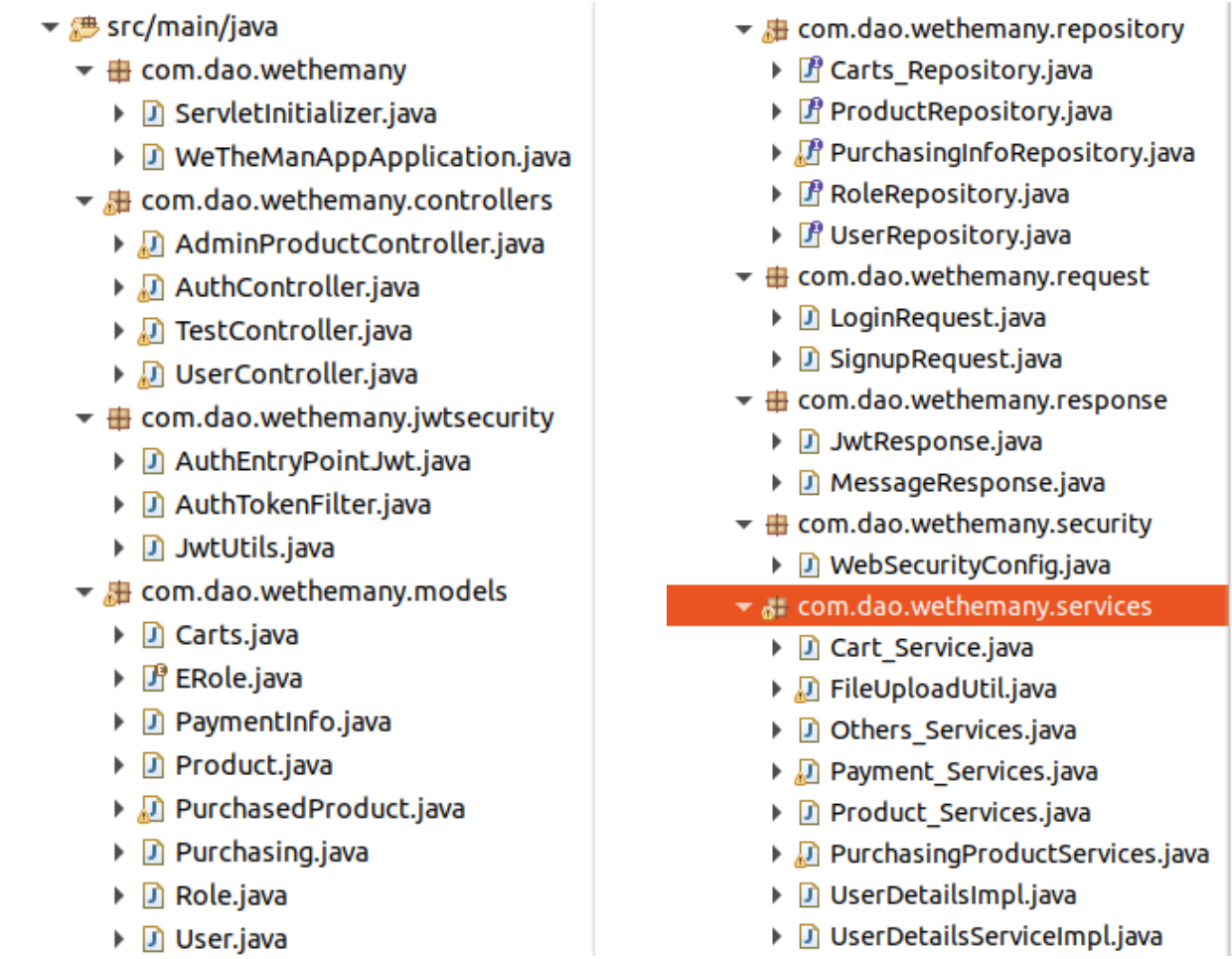
Spring boot Structure:



Src/main/java:

It is the structure in which the real Java Code of the Backend are present and every coding related item is done in this section. Inside it there are various Package a represent start with base package name com.dao.wethemany.

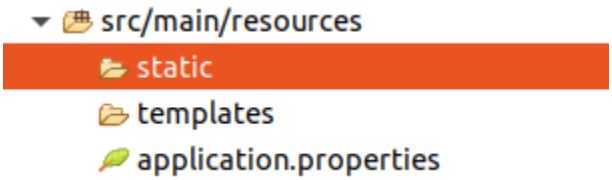
- **com.dao.wethemany** contains the application starting property.
- **com.dao.wethemany.controller** contains all the controller or Routes related Work.
- **Com.dao.wethemany.jwtecurity** contains all the items related to security or JWT.
- **Com.dao.wethemany.models** contains all the Basics business and defining logic.
- **Com.dao.wethemany.reposistory** contains all the interface that interreact with database.
- **Com.dao.wethemany.request** contains some of the request DTO like login, signup and etc.
- **Com.dao.wethemany.response** contains all the DTO or item that will be used for throwing response.
- **Com.dao.wethemany.security** contains all the security related items like: filtering, access control and etc.



Src/main/resource:

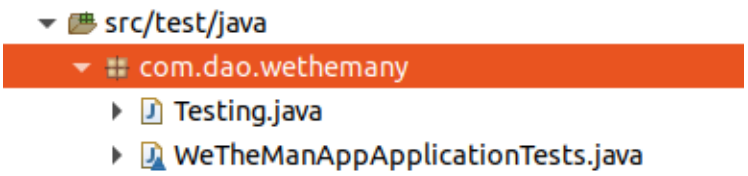
It contains all the element regarding resource component like static. Templates, application. Properties and etc.

- Static contains static element like: js, CSS and static component
- Templates contain all the templates like: Html, react, angular frontend and etc.
- Application.properties contain all the database setup and attributes or variables defining ones.



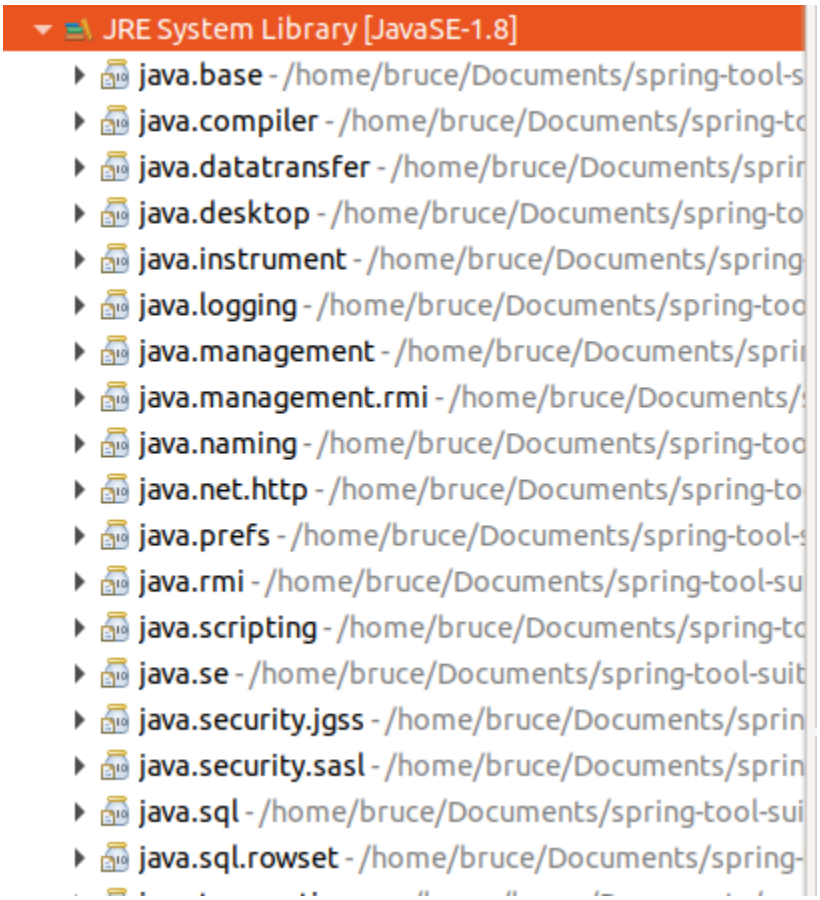
Src/test/java:

It contains the testing files and test related attributes. Java has created an inbuilt structure for testing the program.



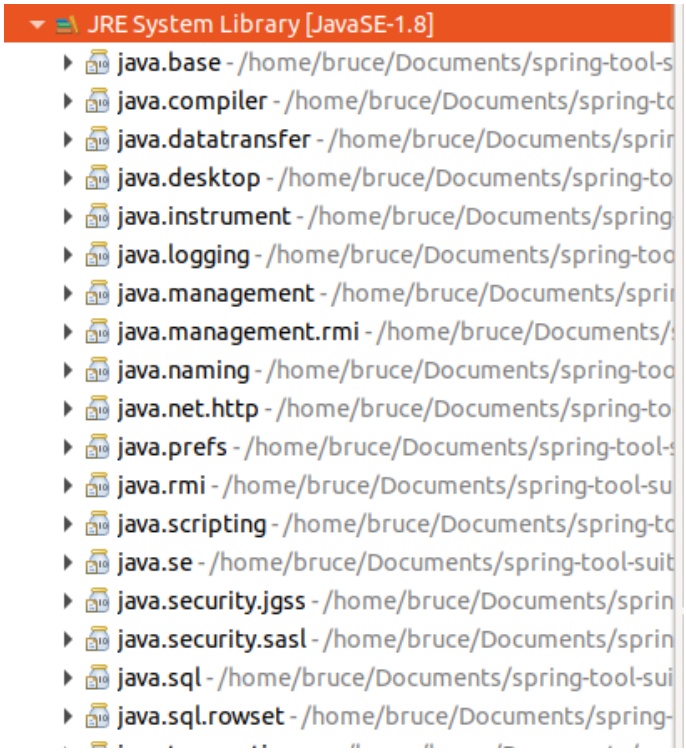
JRE System Library:

It contains all the library and jar files containing JRE, java and spring boot project. All the jdk or jre related items are found here.



Maven Dependencies:

It contains all the library and jar files related to Maven dependency or POM.XML. POM.XML will automatically download the files according to its mentioned dependencies and store in this structure.



target/ generated-sources/annotations:

User-photos:

It is the folder in the spring boot app where it contains all the images or files related component. Whenever user uploads the product then it's photo will be stored here and its file name will be stored in the database.

Some Examples of Spring boot Coding:

```java
package com.dao.wethemany.controllers;

import java.io.File;



@CrossOrigin(origins = "*", maxAge = 3600)
@RestController
@RequestMapping("/api/auth")
public class AuthController {
    @Autowired
    AuthenticationManager authenticationManager;

    @Autowired
    UserRepository userRepository;

    @Autowired
    RoleRepository roleRepository;

    @Autowired
    PasswordEncoder encoder;

    @Autowired
    JwtUtils jwtUtils;

    @Autowired
    PurchasingProductServices purchasingProductServices;

    @Autowired
    Payment_Services payment_Services;

    @Autowired
    Others_Services others_Services;



    @RequestMapping(value = "/getImages/{Images}", method = RequestMethod.GET,
            produces = MediaType.IMAGE_JPEG_VALUE)

    public byte[] getImage(HttpServletResponse response,@PathVariable(name = "Images") String Images) throws IOE
//      File file =new File("user-photos/"+Images);
        byte[] array = Files.readAllBytes(Paths.get("user-photos/"+Images));

        return array;

    }





    @PostMapping("/signin")
    public ResponseEntity<?> authenticateUser(@Valid @RequestBody LoginRequest loginRequest) {

        Authentication authentication = authenticationManager.authenticate(
                new UsernamePasswordAuthenticationToken(loginRequest.getEmail(), loginRequest.getPassword()));

        SecurityContextHolder.getContext().setAuthentication(authentication);
        String jwt = jwtUtils.generateJwtToken(authentication);

        UserDetailsImpl userDetails = (UserDetailsImpl) authentication.getPrincipal();
        List<String> roles = userDetails.getAuthorities().stream()
                .map(item -> item.getAuthority())
                .collect(Collectors.toList());

        return ResponseEntity.ok(new JwtResponse(jwt,
                                                 userDetails.getId(),
                                                 userDetails.getEmail(),
                                                 roles));
    }


```

```
125     @PostMapping("/signup")
126     public ResponseEntity<?> registerUser(@Valid @RequestBody SignupRequest signUpRequest) {
127 //      if (userRepository.existsByEmail(signUpRequest.getEmail())) {
128 //          return ResponseEntity
129 //                  .badRequest()
130 //                  .body(new MessageResponse("Error: Username is already taken!"));
131 //      }
132
133         if (userRepository.existsByEmail(signUpRequest.getEmail())) {
134             MessageResponse messageResponse=new MessageResponse();
135             messageResponse.setMessage("Error: Email is already in use!");
136             messageResponse.setReturnStatus(0);
137             messageResponse.setHttpStatus(HttpStatus.ALREADY_REPORTED);
138 //          messageResponse.set
139             return ResponseEntity
140                     .badRequest()
141                     .body(messageResponse);
142         }
143
144         // Create new user's account
145         User user = new User(
146                             signUpRequest.getEmail(),
147                             encoder.encode(signUpRequest.getPassword()));
148
149         Set<String> strRoles = signUpRequest.getRoles();
150         Set<Role> roles = new HashSet<>();
151
152         if (strRoles == null) {
153             Role userRole = roleRepository.findByName(ERole.ROLE_USER)
154                     .orElseThrow(() -> new RuntimeException("Error: Role is not found."));
155             roles.add(userRole);
156         } else {
157             strRoles.forEach(role -> {
158                 switch (role) {
159                 case "admin":
160                     Role adminRole = roleRepository.findByName(ERole.ROLE_ADMIN)
161                             .orElseThrow(() -> new RuntimeException("Error: Role is not found."));
162                     roles.add(adminRole);
163
164                     break;
165                 case "mod":
```

```
                    break;
                case "mod":
                    Role modRole = roleRepository.findByName(ERole.ROLE_MODERATOR)
                            .orElseThrow(() -> new RuntimeException("Error: Role is not found."));
                    roles.add(modRole);

                    break;
                default:
                    Role userRole = roleRepository.findByName(ERole.ROLE_USER)
                            .orElseThrow(() -> new RuntimeException("Error: Role is not found."));
                    roles.add(userRole);
                }
            });
        }

        user.setRoles(roles);
        userRepository.save(user);
        MessageResponse messageResponse=new MessageResponse();
        messageResponse.setMessage("User registered successfully!");

        return ResponseEntity.ok(messageResponse);
    }


    @PostMapping("/getpaymentWork")
    public ResponseEntity<?>  getAllPurchasedProduct(@RequestBody Purchasing purchasisng) {

        return ResponseEntity.ok(payment_Services.createCharge(purchasisng));
    }


    @PostMapping("/calculateTheCo02/{productvalue}")
    public ResponseEntity<?>  calculateTheCo02(@PathVariable(name = "productvalue") double productvalue) {


        return ResponseEntity.ok(others_Services.calculateC02Emission(productvalue));
    }


}
```

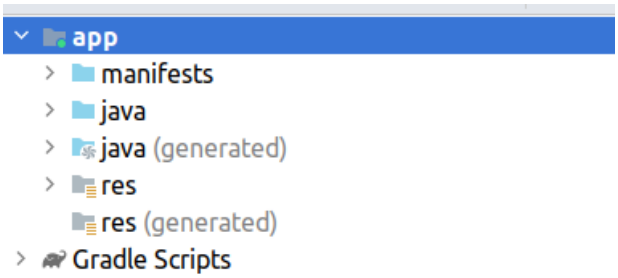
Front-End:

Android Application:

The Android app is a platform-based software application. Because the Android platform is for mobile devices, a typical Android app has been developed to work on the Android OS for a smartphone or tablet. We are going to build the android application as a project so that any android system users can use the system. Android app are most popular now a days. Android application is managed by the android package manager and its file known by apk files. It is developed in Java language with helping framework like: Gradle and XML.

Android Application Structure:

Android Application Structure consist of various Structure like: Manifest, Java, res and Gradle Scripts.



Manifest Structure:

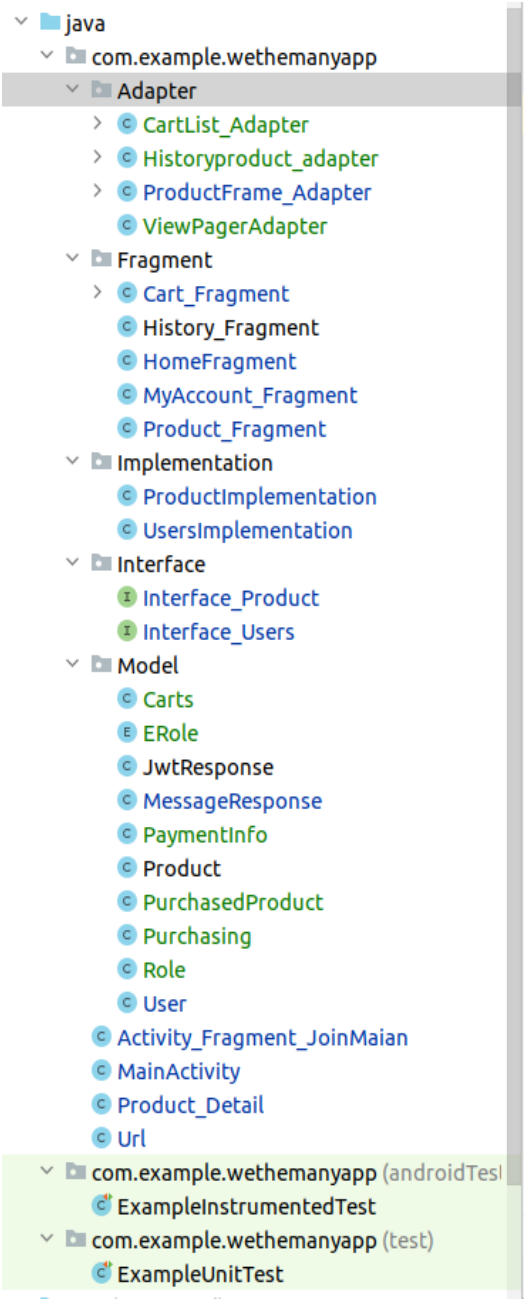
It consists of Manifest file which contains the basic setting regarding the app like: App definition, Launcher UI Sections, Permission and etc.

AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.wethemanyapp">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="WeTheManyApp"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:usesCleartextTraffic="true"
        android:theme="@style/Theme.WeTheManyApp">
        <activity android:name=".Product_Detail"></activity>
        <activity android:name=".Activity_Fragment_JoinMaian" />
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

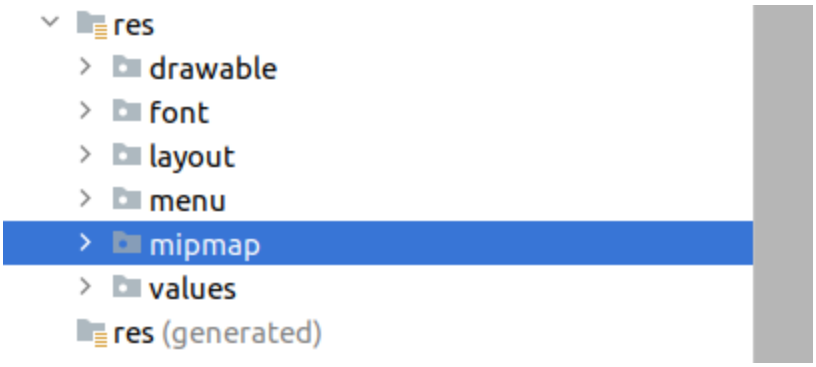
Java Structure:

It Contains the Structure or the materials regarding the Java Code in the android app. All the Java Code for android is done on these sections. It contains Packages regarding Interface, Implementation, UI and etc. It contains the elements like: Activity, Fragment, Recycle View and etc.



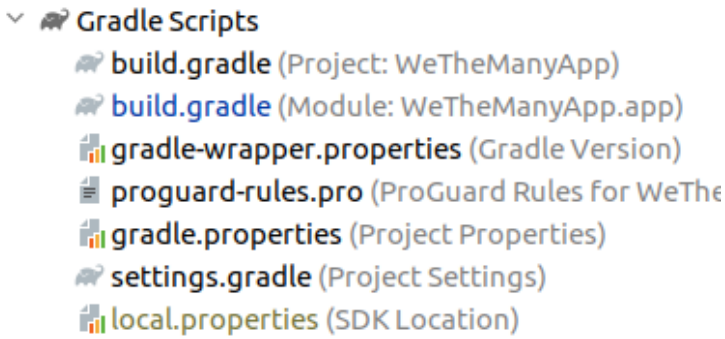
Res Structure:

It is the structure which contains all the resource files regarding the UI. All the designed and developed UI elements are present in this structure. Layouts, Menu, Icon and etc. Drawable contain the files regarding icon, vector, shape and etc., font package contains all the font related things, layout contains all the UI elements, values contain color, String defining aspects.



Gradle Scripts:

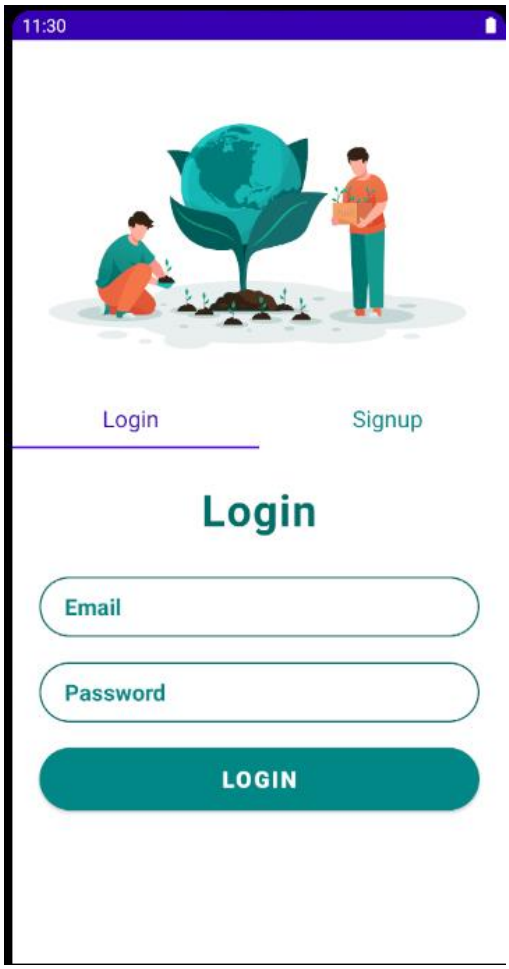
It contains all the Gradle related Scripts where all the defining and project management and dependency management items are kept.



UI Elements:

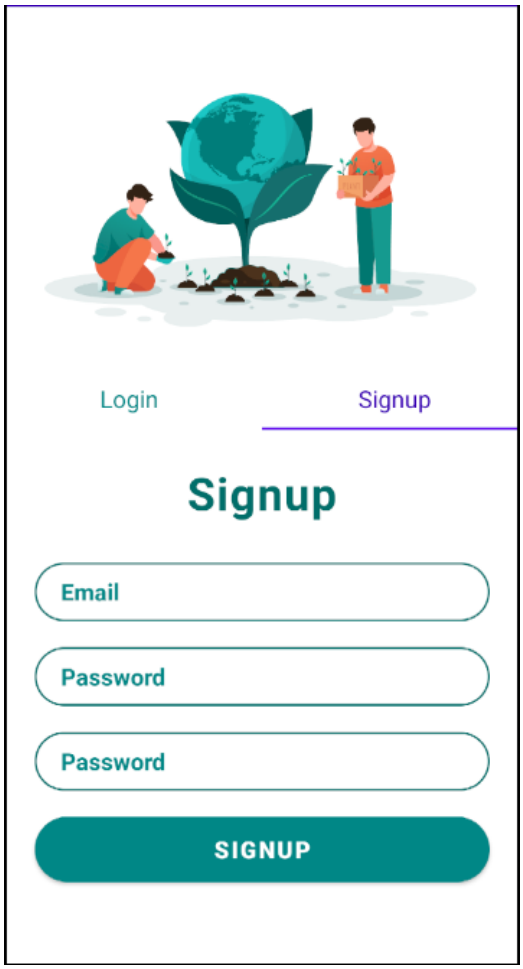
Login:

It is a Login Section where user can enter their credentials and enter into the system. User need to enter email and password. User need to enter data, if not enter then error will throw and also in case when the email and password do not match to database



Signup:

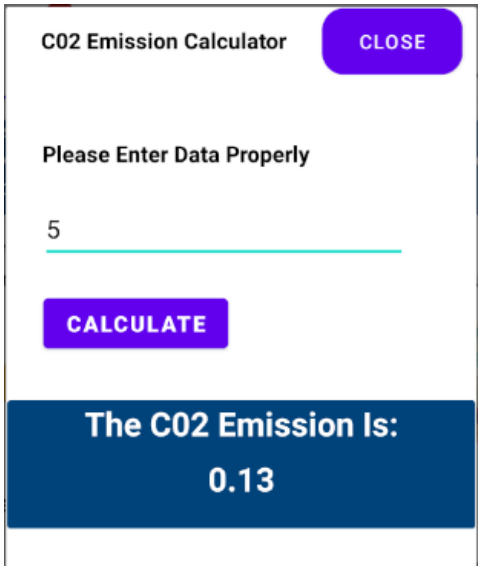
User Need to enter proper email address which is not used in this system, password and confirm password. If any field is blank then it will throw error message, will also throw error message is password and confirm don't and also throw if this email is already existed in System.



Home:

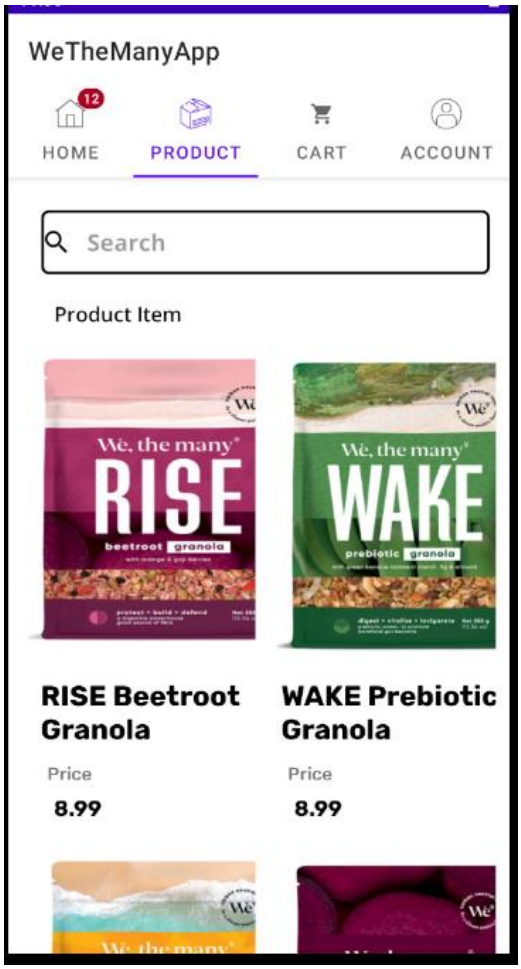
It is a section or Fragment in android App that represent the overall application or the Business. It starts with the representing login user person following by company information. It also describes about sectors in which it working on. At last, it contains social site links follows by C02 emission Calculator.

On Clicking the + Button it will open the alert for calculating c02.After entering data it will show c02 emission.



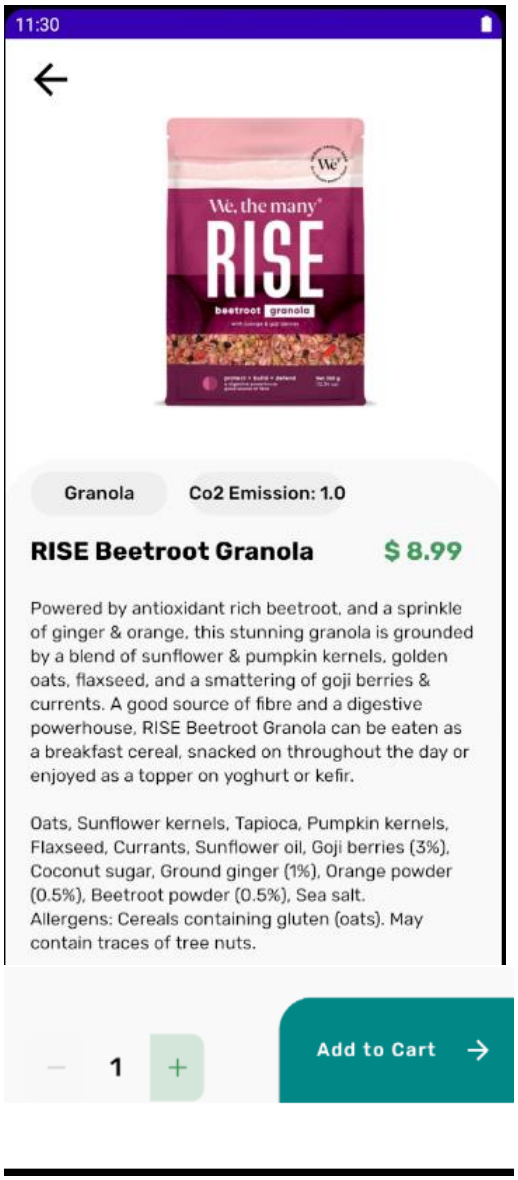
Product:

Product Fragment consist all the details regarding available Product with the option to search product. User can Search the Product and get that product on it.



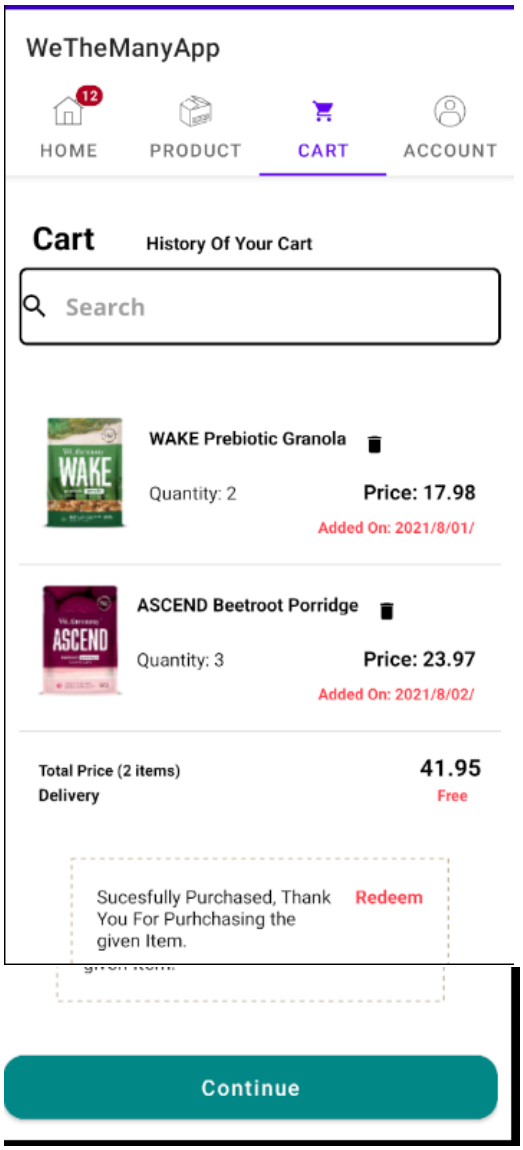
Product Details:

It is the Activity that contains all the Details regarding the Specific Product. Use can also select the quantity of product and cart that product in the cart list.



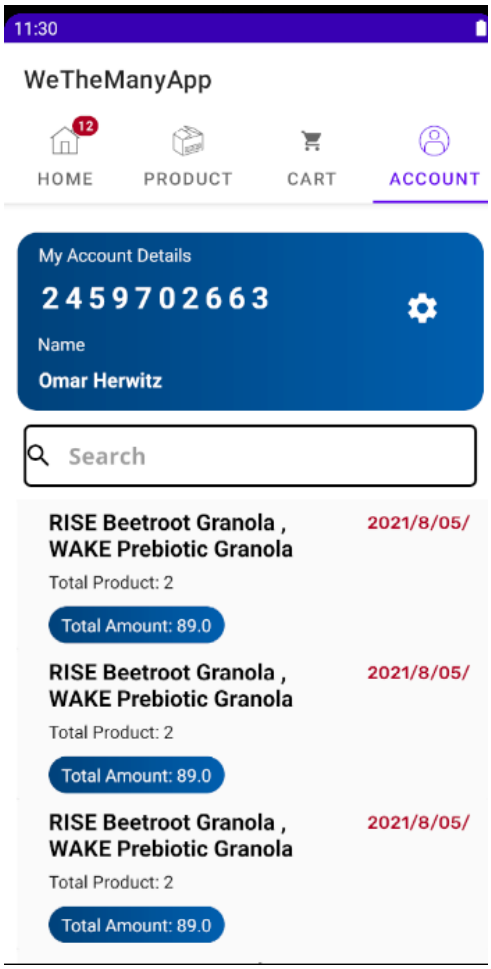
Cart Fragment:

It is a Fragment that contain all the product and cart details that users have carted or added. It also contains the search field to search the data. On User wish they can delete the data and even buy or purchased it.



MyAccount Fragment:

My Account Fragment contain the details about Login Users and details of the product that user have buyed. They can also search the purchased product based on the various Criteria.



Database:

A database is a systematic collection of data. They support electronic storage and manipulation of data. Databases make data management easy [1]. There are various types of database type: Relational and Non-Relational Database. Relational Database is a type of database that maintain the data in the form of database and each of them is related with each other. It is one of famous type of database and some of them are: MySQL, MySQL Server, Oracle and etc. NoSQL database is another type of databases which means not only database. It is used for large sets of distributed data.

Reason to Choose NoSQL I.e., Mongo dB instead of Relational Database:

- Mongo dB provides the flexibility in schemas as it provides dynamic in structure
- It helps to store the object or even the data in the JSON Format
- We do not need to maintain any relationship between tables.
- Lots of documentation and community help is available for Mongo db.
- It provides a 500 MB online storage on free trial which is the main asset

NoSQL:

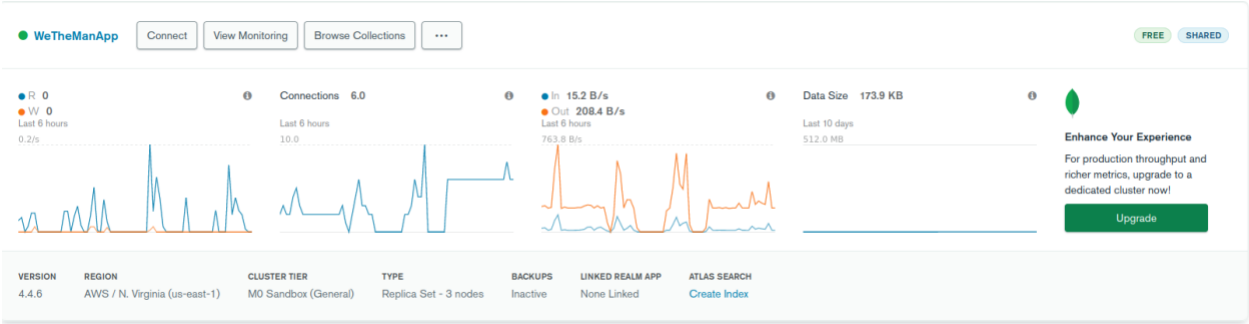
Non-tabular NoSQL databases (also known as "not merely SQL") store data differently from relational tables. NoSQL databases are classified according to their data model. Document, key-value, wide-

column, and graph are the most common types. Hey have adaptable schemas and can handle big amounts of data and high user loads with ease.[3]

Mongo dB:

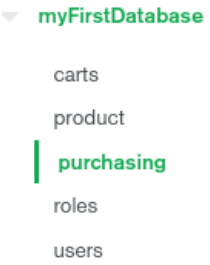
MongoDB is a cross-platform, document-oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document [2]. MongoDB's document data model supports JSON by default, and its expressive query language is simple to understand and use for developers. Automatic failover, horizontal scaling, and the ability to assign data to a specific place are all built-in features [4].

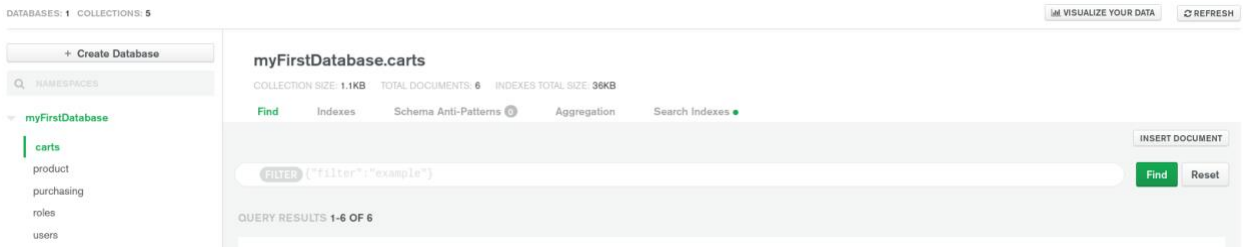
In our Project, we have used the Mongo dB Atlas which is an online database which is integrated with the backend. Cloud Mongo dB is used.



Collections:

Collection is just like table was in relational database. It stores the data in the form of documents in the collections. In our database the collections are:





Documents:

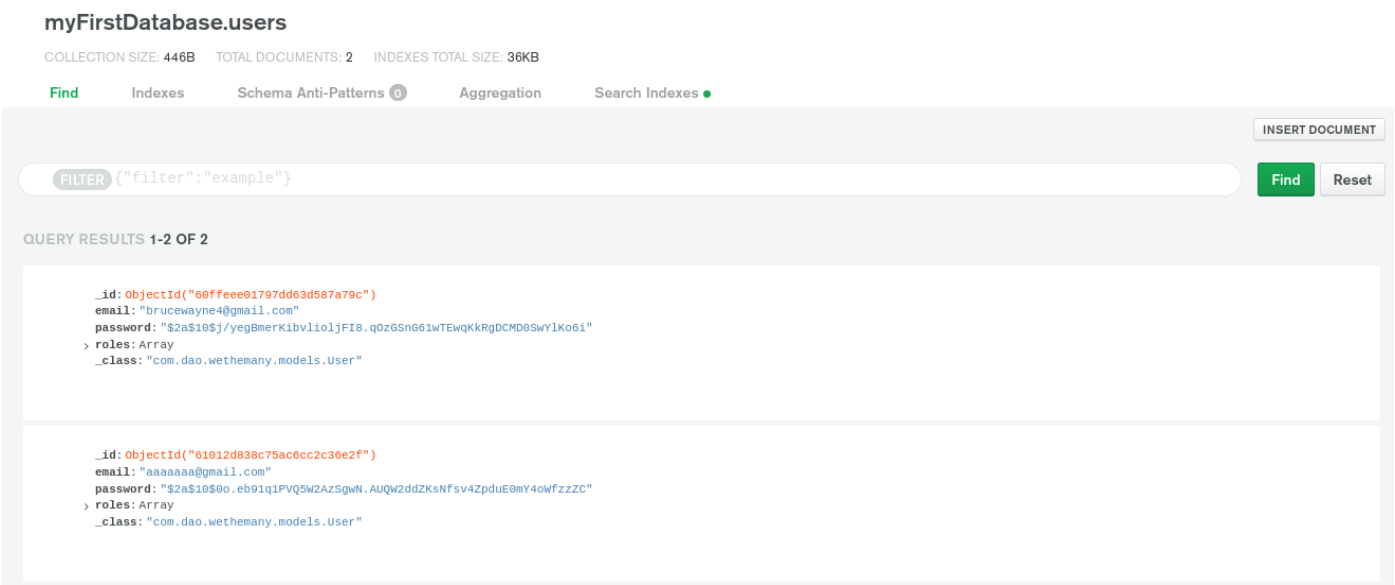
Document are just like a column or table heading or also can be defined as set of key value pairs. It has dynamic schema and will store dynamically. Documents in the same collection don't have to have the same set of fields or structure, and common fields in a collection's documents can contain different types of data.

```
  _id: ObjectId("610138088c75ac6cc2c36e50")
> purchasedproduct: Array
  purchasedDate: 2021-07-28T10:57:12.303+00:00
> paymentInfo: Object
  userId: "61012d838c75ac6cc2c36e2f"
  status: true
  userEmail: "aaaaaaa@gmail.com"
  _class: "com.dao.wethemany.models.Purchasing"
```

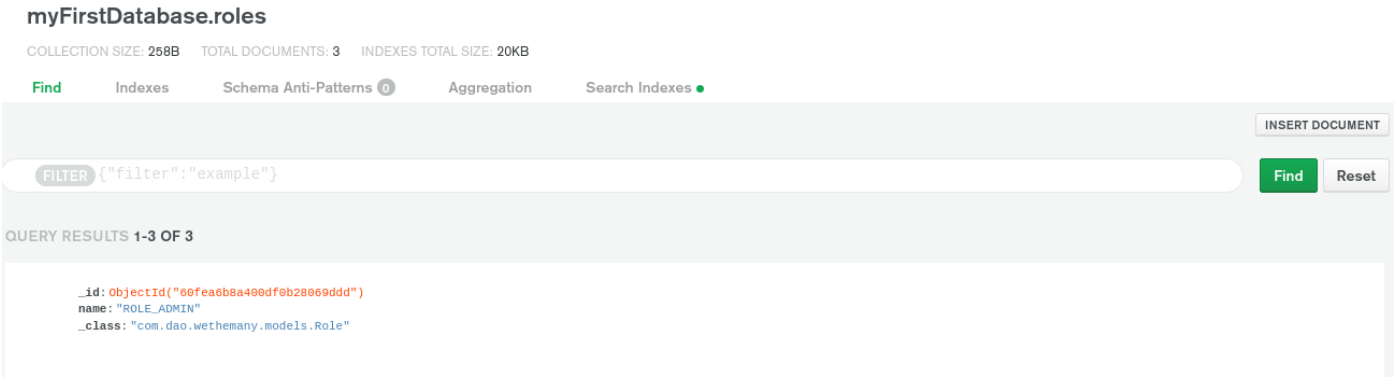
.

Collections and Document in the Project:

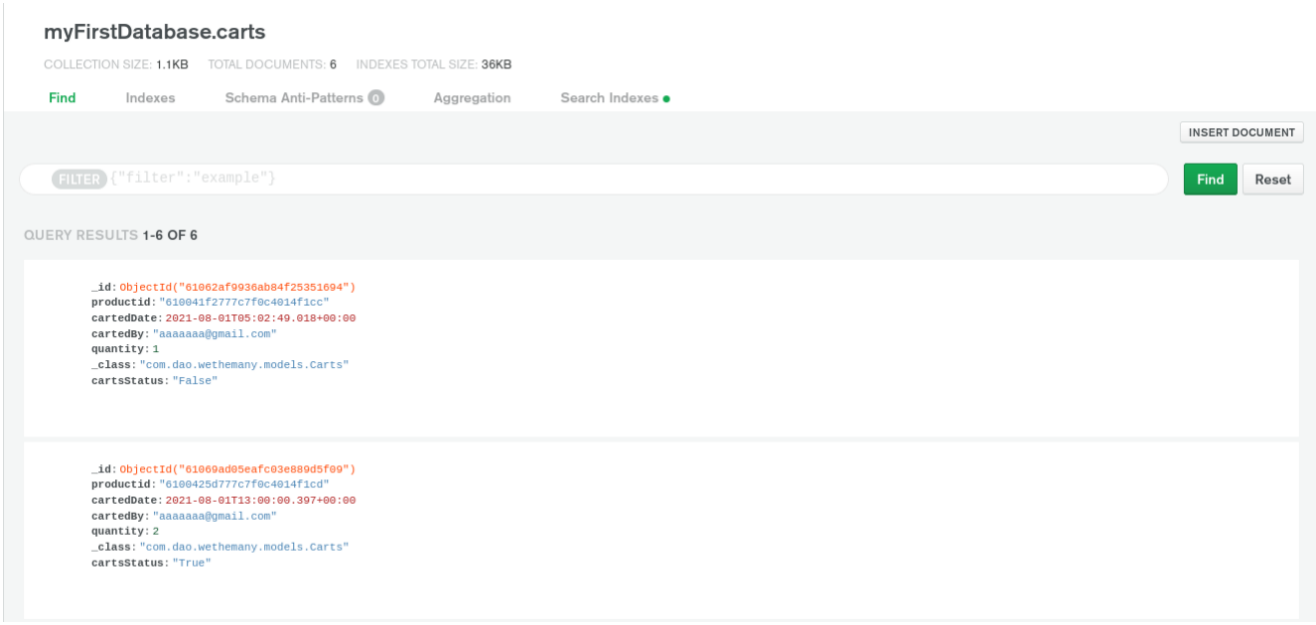
Users: User is a Collection where the details regarding the User are kept. In this the document or details regarding User like: userid, email, hashed password, roles detail and etc.



Roles: It is the Collection that contain document or value regarding roles are kept. There are three roles like: ADMIN, USER and etc.



Carts: It is the Collection that contains the document or values regarding the Carts like: productid, date, cartedby, quantity and others are kept.



Product: It is the Collection that contain the document or values regarding product are kept. Product information like: Name, category, description, price, c02emission, images and others are kept.

myFirstDatabase.product

COLLECTION SIZE: 3.95KB TOTAL DOCUMENTS: 5 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns 0 Aggregation Search Indexes

INSERT DOCUMENT

FILTER {"filter":"example"} Find Reset

QUERY RESULTS 1-5 OF 5


```
_id: ObjectId("610041f2777c7f0c4014f1cc")
name: "RISE Beetroot Granola"
category: "Granola"
description: "Powered by antioxidant rich beetroot, and a sprinkle of ginger & orang..."
price: 8.99
co2Emission: 1
images: "1627406834775Screenshot2021-07-27at23-09-57RISEBeetrootGranola.png"
_class: "com.dao.wethemany.models.Product"
```

```
_id: ObjectId("6100425d777c7f0c4014f1cd")
name: "WAKE Prebiotic Granola"
category: "Granola"
description: "Kickstart your day and feed your gut with one of nature's best prebiot..."
price: 8.99
co2Emission: 1
images: "1627406941485Screenshot2021-07-27at23-12-46WAKEPrebioticGranola.png"
_class: "com.dao.wethemany.models.Product"
```


Purchasing: Purchasing is an collection that contains the document or values regarding the Purchased product are kept. Some of it's information like: purchased product, payment info, purchaseduser info and etc are kept

myFirstDatabase.purchasing

COLLECTION SIZE: 4.14KB TOTAL DOCUMENTS: 8 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns 0 Aggregation Search Indexes

INSERT DOCUMENT

FILTER {"filter":"example"} Find Reset

QUERY RESULTS 1-8 OF 8


```
_id: ObjectId("610138088c75ac6cc2c36e50")
> purchasedproduct: Array
purchasedDate: 2021-07-28T10:57:12.303+00:00
> paymentInfo: Object
userId: "61012d838c75ac6cc2c36e2f"
status: true
userEmail: "aaaaaaa@gmail.com"
_class: "com.dao.wethemany.models.Purchasing"
```

```
_id: ObjectId("61081490556cf940933dd84d")
> purchasedproduct: Array
purchasedDate: 2021-08-02T15:51:43.726+00:00
> paymentInfo: Object
userId: "61012d838c75ac6cc2c36e2f"
status: true
userEmail: "aaaaaaa@gmail.com"
_class: "com.dao.wethemany.models.Purchasing"
```


Database Connection Done in Project:

We have connected the database in our Backend through the Application. Properties and through definite structure like as Belows.

```
# Mongo Connection
spring.data.mongodb.uri=mongodb+srv://admin123:admin123@wethemanapp.eokvk.mongodb.net/myFirstDatabase?retryWrites=true&w=majority
```

Framework Used in Backend:

spring-boot-starter-data-mongo dB is a dependency or inner framework used in the project. Spring Data for MongoDB is part of the larger Spring Data project, which attempts to give new datastores with a familiar and consistent Spring-based programming model while keeping store-specific features and capabilities [5]. Some Features of this are:

- It provides a medium to connect to mongo dB database and make it runnable.
- It provides an inbuilt function which is used for the multiple CRUD Operations and etc.
- It provides dynamins in function as well in the custom-made function

We need to import the dependency in dependency management file I.e pom.xml

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-mongodb</artifactId>
</dependency>
```

We need to extends its property in the repository section or can use its class and its inbuilt function as per our requirements.

```
@Repository
public interface Carts_Repository extends MongoRepository<Carts, String>{
```

Testing:

Software Testing is an approach to test whether the software or product is working properly or not and whether it matches up to expectation or not and to ensure that the product is defect or bug free. It includes execution of software/system components using manual or automated tools to evaluate one or more properties of internet's purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements [6]. There are usually Two types testing we categories Normally which are White Box and Black Box Testing.

Black Box Testing	White Box Testing
It is the testing approach in which no prior testing skills is required and can test system without the knowledge of internal working structure of Program	It is a testing approach in which prior knowledge of testing skills is required and internal structure knowledge is known to tester.
It is the function Test or Behavior Test of System	It is the structure Test or Logic Test of System
Common People Can test and no testing knowledge is required	Skilled People can test and testing knowledge is required

Benefits of black box is it is best suited and efficient for large code or system	It's benefits is that it removes extra lines of code which can bring the defect code
Some of this testing examples are: System Testing and Acceptance Testing	Some of this testing examples are Unit and Integrated Testing

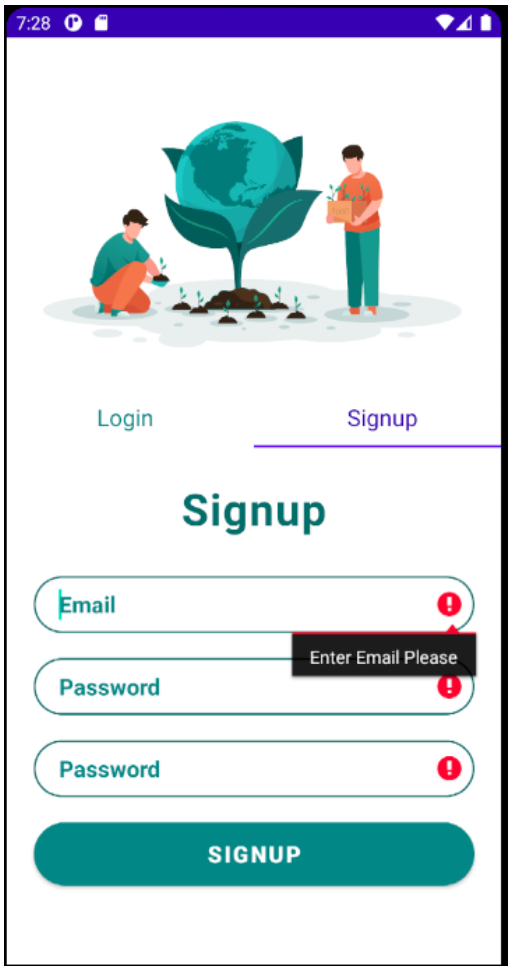
Black Box Testing:

System Testing:

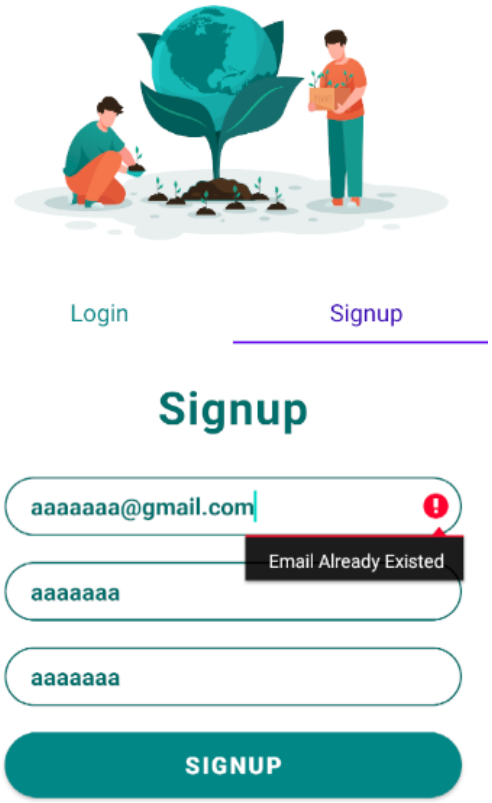
This black box testing type is connected to the functional requirements of a system; it is accomplished by software testers. It comes under the Functional Testing

Signup Check:

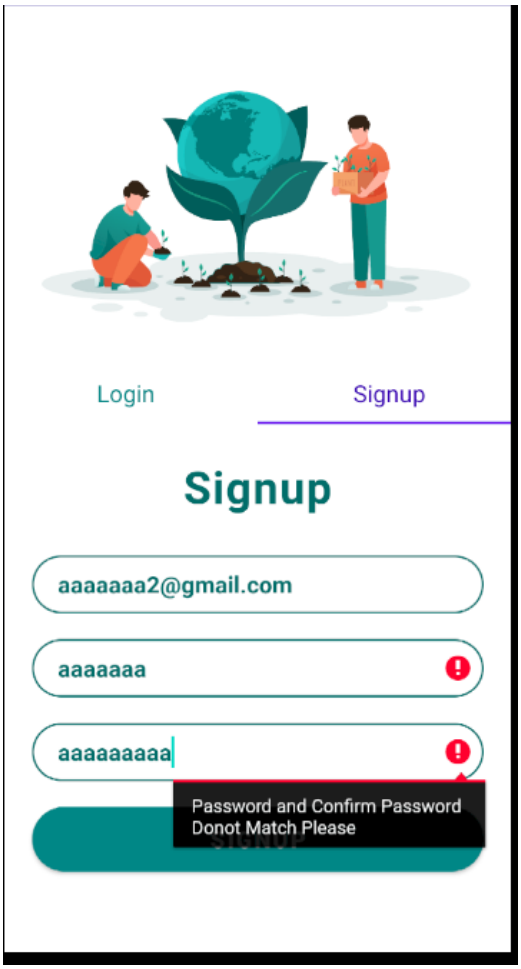
With Null Values: Testing the Signup with the Null Value, it will throw the error with message Enter the specific Edit Text data in red format



Using already Existed Email: Testing by data with already used email address, it will show the red error message with email already existed.

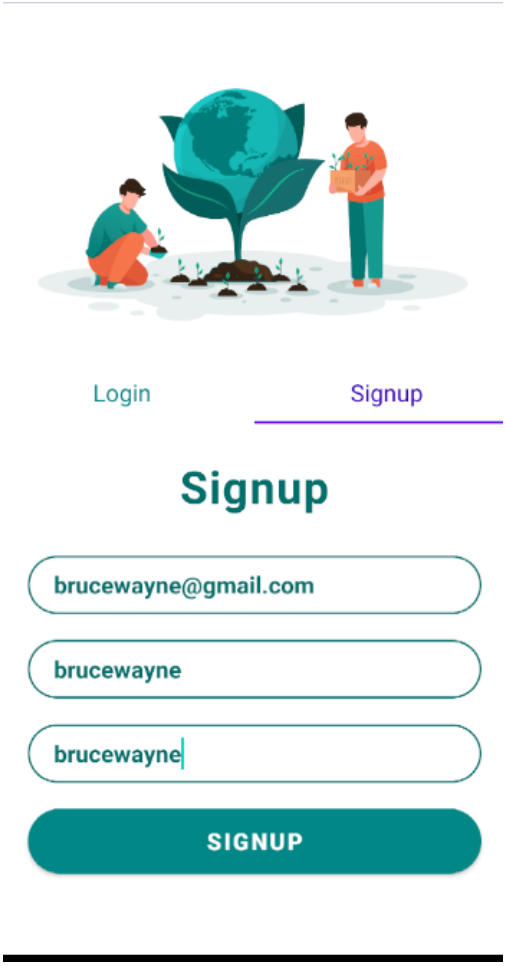


On Using different Password on Password and Re-password section: By entering the data with different data on Password and confirm-password, it will throw the error message as validation is done to make user to know and enter password components.



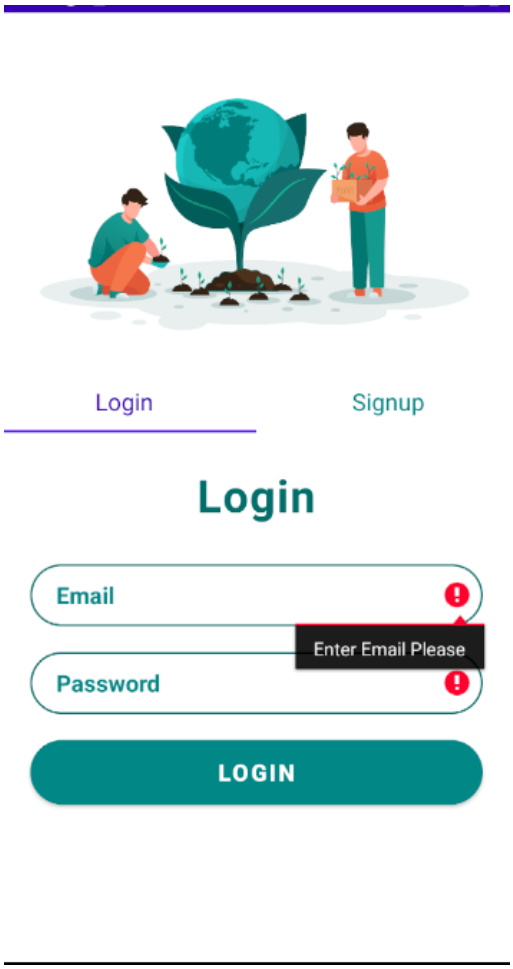
.

With Correct Data: By entering all the correct data like as in below one will successfully create an account and will show message in the Toast Section.

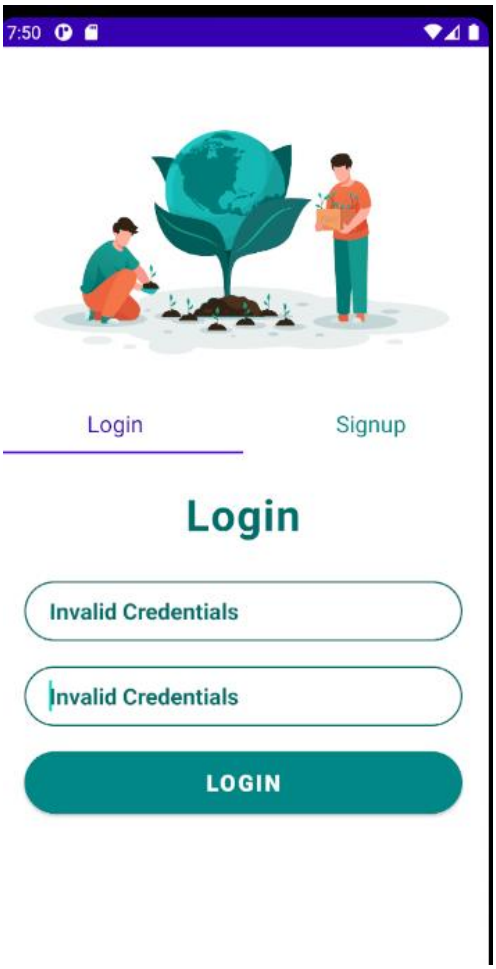


Login Check:

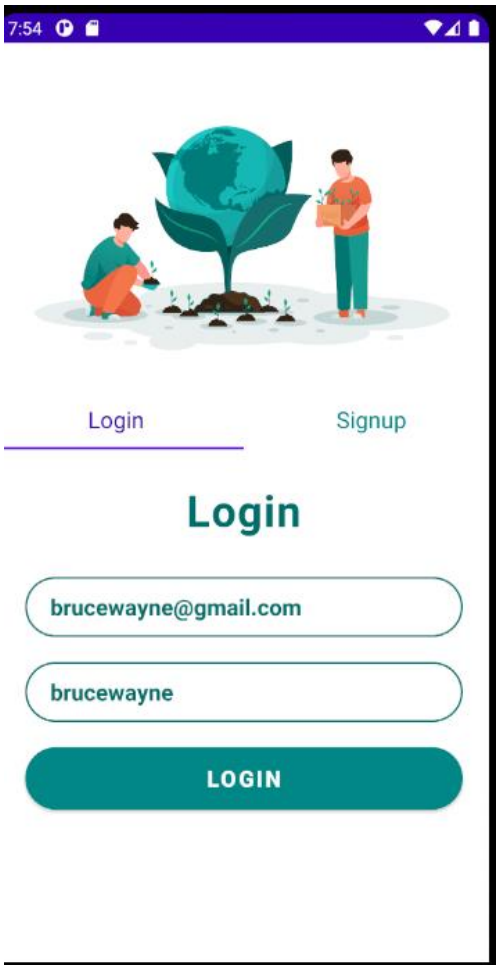
Checking With Null Values: On checking with the Null Values, it will



With Incorrect Data: By Entering incorrect data it will show the error as shown as in below image. User e natter the correct data in order to Login.



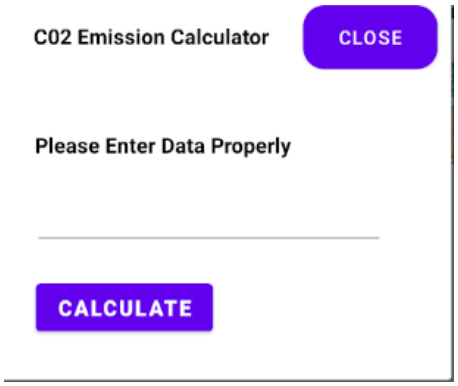
With Correct Data: With the correct data entered, it will show no error and directly take to the Home Sections.



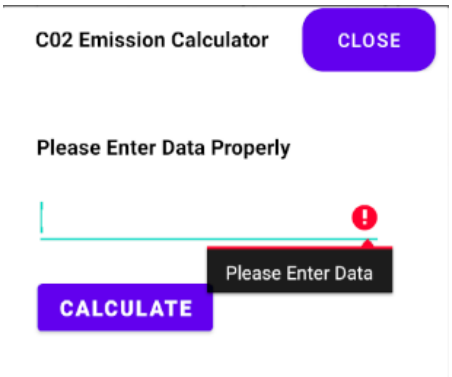
Calculating C02 Emission:

If we click the plus button or the floating button in the home section, the following alert button to enter the data arises to calculate c02.

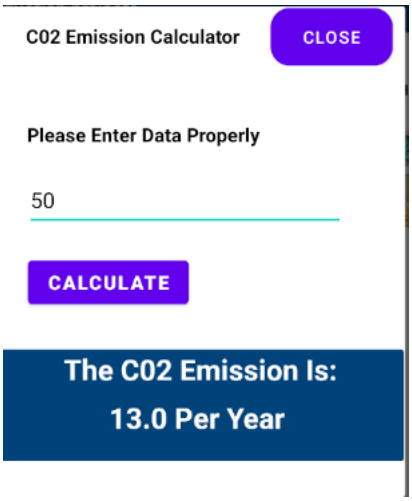
The Appear Dialog box is



If We enter the nothing in the text field then the warning message will be shown saying, “Please Enter Data”.

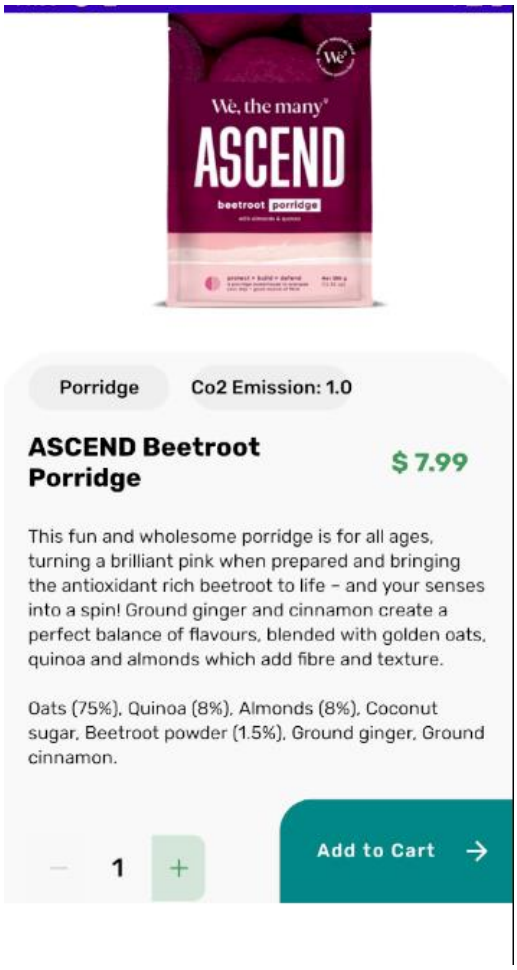


If the Product value in weight is shown then it will show the result of c02 emission occurred in a year,

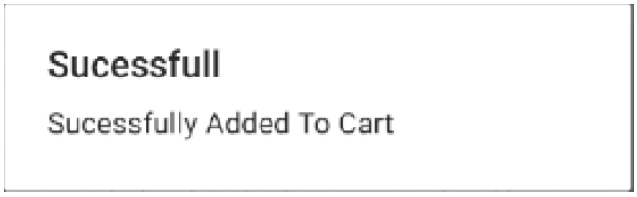


Testing Add to Cart:

In the Product Detail Section, the details of product is shown as in below

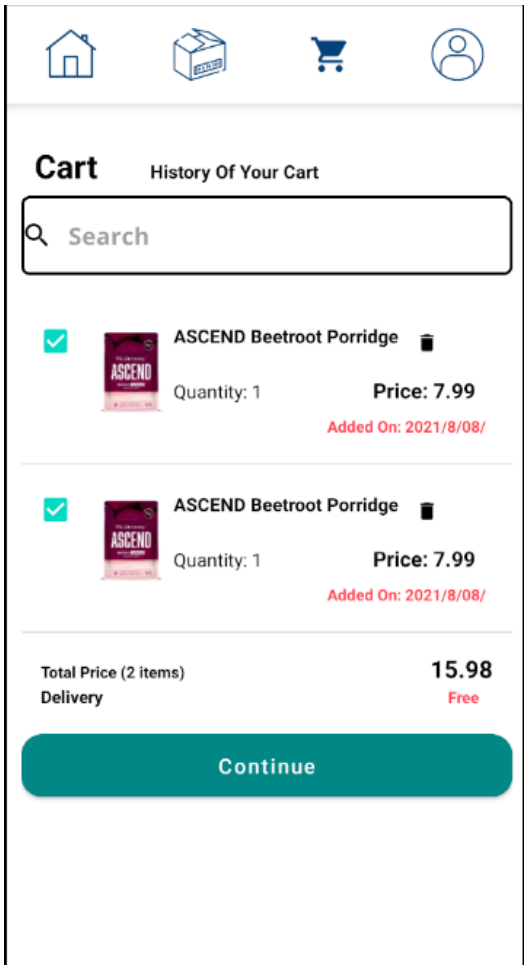


After Clicking the add to cart data the following success alert dialog will be open

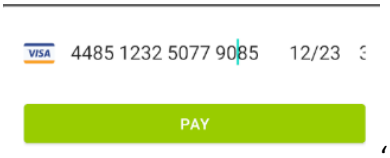


Purchase Product:

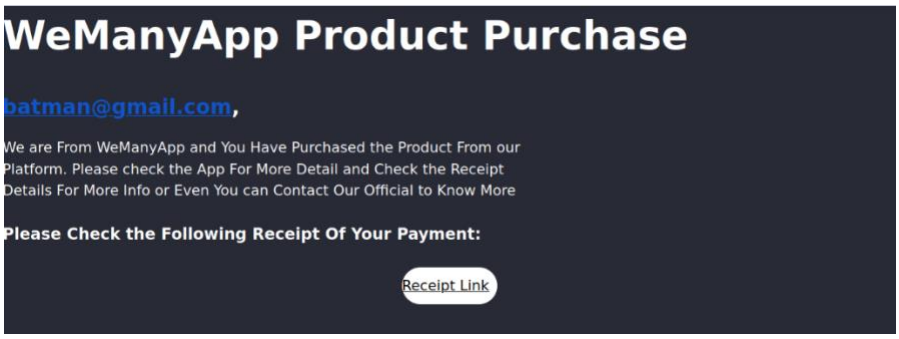
In order to purchase the Product, please check the carted items and click on the continue button. Remember that Continue button appears only if you click on the check field.



After clicking button then the Credit card open Box will appear.



After Clicking on the Pay Button, the product payment is successfully done, you can see the purchased history Fromm Account Section or can get product receipt and purchasing notification through email.



White Box Testing:

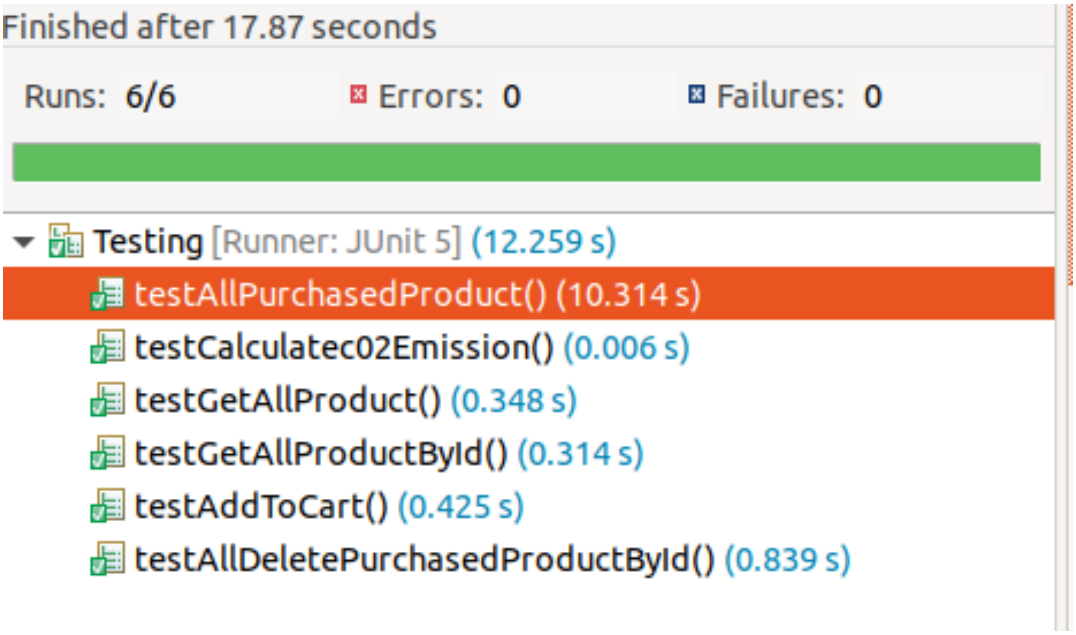
JUnit Testing:

Junit Testing in a testing approach that comes under the Unit testing. It is called Junit because Unit testing is done on Java. JUnit 5 is JUnit's next generation. The aim is to establish an up-to-date basis for developer-side JVM testing. This includes Java 8 and higher and allows a wide range of test styles [7].

Test Case:

S.N	Test Name	Test Data	Expected Result	Actual Result	Status
1	testCalculatec02Emission	Productvalue=5	0.13	0.13	Pass
2	testGetAllProduct	none	1	1	Pass
3	testGetAllProductById	Id= 61069ad05eafc03e889d5f09	1	1	Pass
4	testAllPurchasedProduct	Useremail:Aaaaaaa@gmail.com	1	1	Pass
5	testAllDeletePurchasedProductById	Id= 61081490556cf940933dd84d	1	1	Pass
6	testAddToCart	CartedBy = aaaaaaa@gmail.com Productid= “61062af9936ab84f25351694” Quantity=2	1	1	Pass

Test Result:



Testing Code:

```java
package com.dao.wethemany;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertNotNull;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;

import com.dao.wethemany.models.Carts;
import com.dao.wethemany.response.MessageResponse;
import com.dao.wethemany.services.Cart_Service;
import com.dao.wethemany.services.Others_Services;
import com.dao.wethemany.services.Product_Services;
import com.dao.wethemany.services.PurchasingProductServices;

@SpringBootTest

public class Testing {

    @Autowired
    private Others_Services others_Services;

    @Autowired
    private Cart_Service cart_Service;

    @Autowired
    private Product_Services product_Services;

    @Autowired
    private PurchasingProductServices purchasingProductServices;

    //This Test WIll calculate and check whether the c02 emission generated value is correct or not
      @Test
      public void testCalculatec02Emission() {
         assertEquals(0.13, others_Services.calculateC02Emission(5));
      }

      // It Will Test whether the getAllProduct Work or Not
      @Test
      public void testGetAllProduct() {

          assertEquals(1,product_Services.getAllProductInfo().getReturnStatus());


      }

      // It Will Test whether the getAllProductById Work or Not
      @Test
      public void testGetAllProductById() {

          assertEquals(1,product_Services.getAllProductById("610041f2777c7f0c4014f1cc").getReturnStatus());

      }

      // It Will Test whether the getAllPurchasedProduct Work or Not
      @Test
      public void testAllPurchasedProduct() {

          assertEquals(1,purchasingProductServices.getAllPurchaseHistory("aaaaaaa@gmail.com").getReturnStatus());

      }

      // It Will Test whether the testAlldeletePurchasedProductById Work or Not
```

```java
    @Test
    public void testAllDeletePurchasedProductById() {

        assertEquals(1, purchasingProductServices.deletePurchaseHistoryById("610b77213452f74594bdc067").getReturnStatus());

    }

    // It will test whether the addtocart WOrk Or Not
    @Test
    public void testAddToCart() {

        Carts cart=new Carts();
        cart.setCartedBy("aaaaaaa@gmail.com");
        cart.setProductid("61062af9936ab84f25351694");
        cart.setQuantity(2);
        MessageResponse messageResponse=cart_Service.addToCartService(cart, "aaaaaaa@gmail.com");
        assertEquals(1, messageResponse.getReturnStatus());

    }

}
```

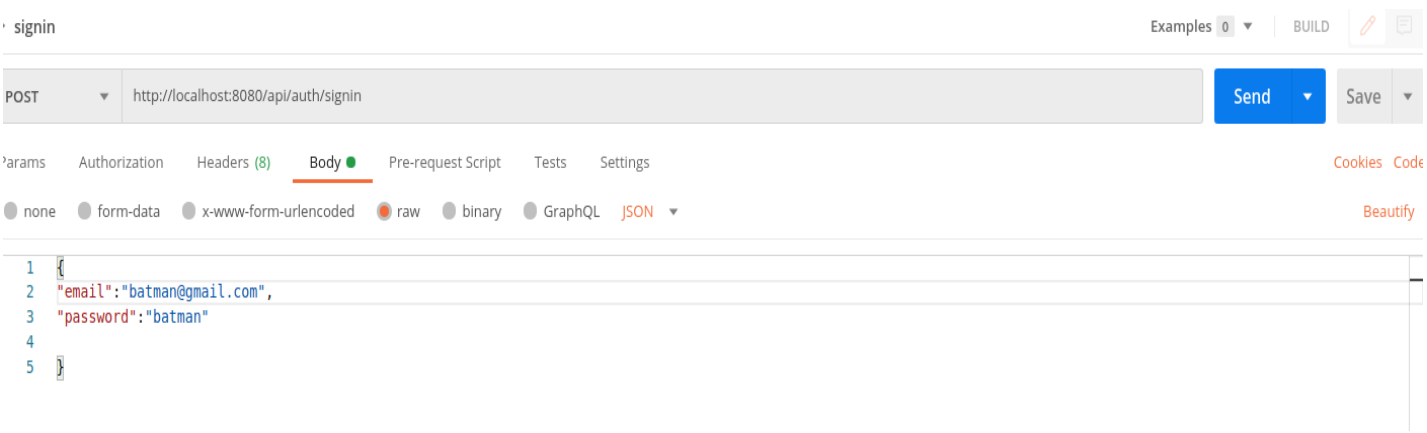
Integrated API Testing:

The Integrated testing of Api Is done by the usage of Postman. Postman is a build-up and API platform. Postman simplifies every step of the API lifecycle and streamlines collaboration so that improved APIs can be created more quickly. It will test the functions or Api routes by testing whether it works or not.

Login:

Testing of it done by entering the routes and entering the body or data in the form of json to test the whether it working or not.

Request:



Response:

In the Response we can see it's status like: response of the Request, its status, time taken by it to response it along with the size.

Body Cookies Headers (14) Test Results Status: 200 OK Time: 665 ms Size: 743 B

Pretty Raw Preview Visualize JSON


```
{
    "id": "610f6c9cc685b835434b4693",
    "email": "batman@gmail.com",
    "roles": [
        "ROLE_USER"
    ],
    "accessToken": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJiYXRtYW5AZ21haWwuY29tIiwiaWF0IjoxNjI4NDQ0MzM2LCJleHAiOjE2Mjg1MzA3MzZ9.
        gbN9Cu6XzFcj7WFmAEToJdDGHBpei70bJQdm5-2zxXTOO5gD8Qu8UVTlLNe873RAEjN_sxmJET1pE31lwbl2Og",
    "tokenType": "Bearer"
}
```


Signup Testing:

In the Signup Testing, the data in entered in the body in the form of json with the signup URL and Request is send to the server

signup

POST http://localhost:8080/api/auth/signup

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
"email":"rupeshevil617@gmail.com",
"username":"rupeshevil617",
"roles":["user"],
"password":"aaaaaaa"

}
```

Response:

In the Response of the API Hits, the following response comes with the successfully account created with the status, time and size of it.

Body Cookies Headers (14) Test Results Status: 200 OK Time: 953 ms Size: 642 B

Pretty Raw Preview Visualize JSON


```
{
    "message": "User registered successfully!",
    "responseType": null,
    "httpStatus": null,
    "returnStatus": 0,
    "returnValue": null,
    "returnValueList": null,
    "productList": null,
    "cartsValueList": null,
    "purchasedValueList": null
}
```


Trying to Register with already used Email:

Trying to register the email with already used email and send the request to the server

Request:

signup

POST http://localhost:8080/api/auth/signup

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
"email":"rupeshevil617@gmail.com",
"username":"rupeshevil617",
"roles":["user"],
"password":"aaaaaaa"

}
```

Response:

In the response we can see the message with already uses one and the status, time and size one.

Body Cookies Headers (13) Test Results Status: 400 Bad Request Time: 269 ms Size: 638 B

Pretty Raw Preview Visualize JSON

```
{
    "message": "Error: Email is already in use!",
    "responseType": null,
    "httpStatus": "ALREADY_REPORTED",
    "returnStatus": 0,
    "returnValue": null,
    "returnValueList": null,
    "productList": null,
    "cartsValueList": null,
    "purchasedValueList": null
}
```

Testing GetAllPurchased Product:

Request:

In the Request the token or bearer token generated in login is used and Request is hits on the following routes

http://localhost:8080/api/auth/user/getAllPurchasedProduct Examples 0

GET http://localhost:8080/api/auth/user/getAllPurchasedProduct Sen

Params Authorization Headers (7) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Response:

In the Response we can see the data, status, time, and size of the data.

Body Cookies Headers (14) Test Results Status: 200 OK Time: 11.72 s Size: 47.65 KB

Pretty Raw Preview Visualize JSON


```
1  {
2      "message": "Sucessfully Retrieved Data",
3      "responseType": null,
4      "httpStatus": "OK",
5      "returnStatus": 1,
6      "returnValue": null,
7      "returnValueList": null,
8      "productList": null,
9      "cartsValueList": null,
10     "purchasedValueList": [
11         {
12             "id": "610f7b82c685b835434b4696",
13             "purchasedproduct": [
14                 {
15                     "productId": "610041f2777c7f0c4014f1cc",
16                     "productQuantity": 1.0,
```


Testing the C02 Emission Method:

Request:

Request is hits to the server with the data in the URL to generate the carbon Emission

Untitled Request BUILD

POST http://localhost:8080/api/auth/calculateTheCo02/5 Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Response:

In The Response Section, we can see the c02 emission generated with the status, time and size of the data

Body Cookies Headers (14) Test Results Status: 200 OK Time: 19 ms Size: 439 B

Pretty Raw Preview Visualize JSON


```
1  0.13
```


Flow Chart:

Flowchart can be defined as the pictorial form of algorithm or step by step procedure. A flow chart represents a process graphically or symbolically. In each step of the process, a different

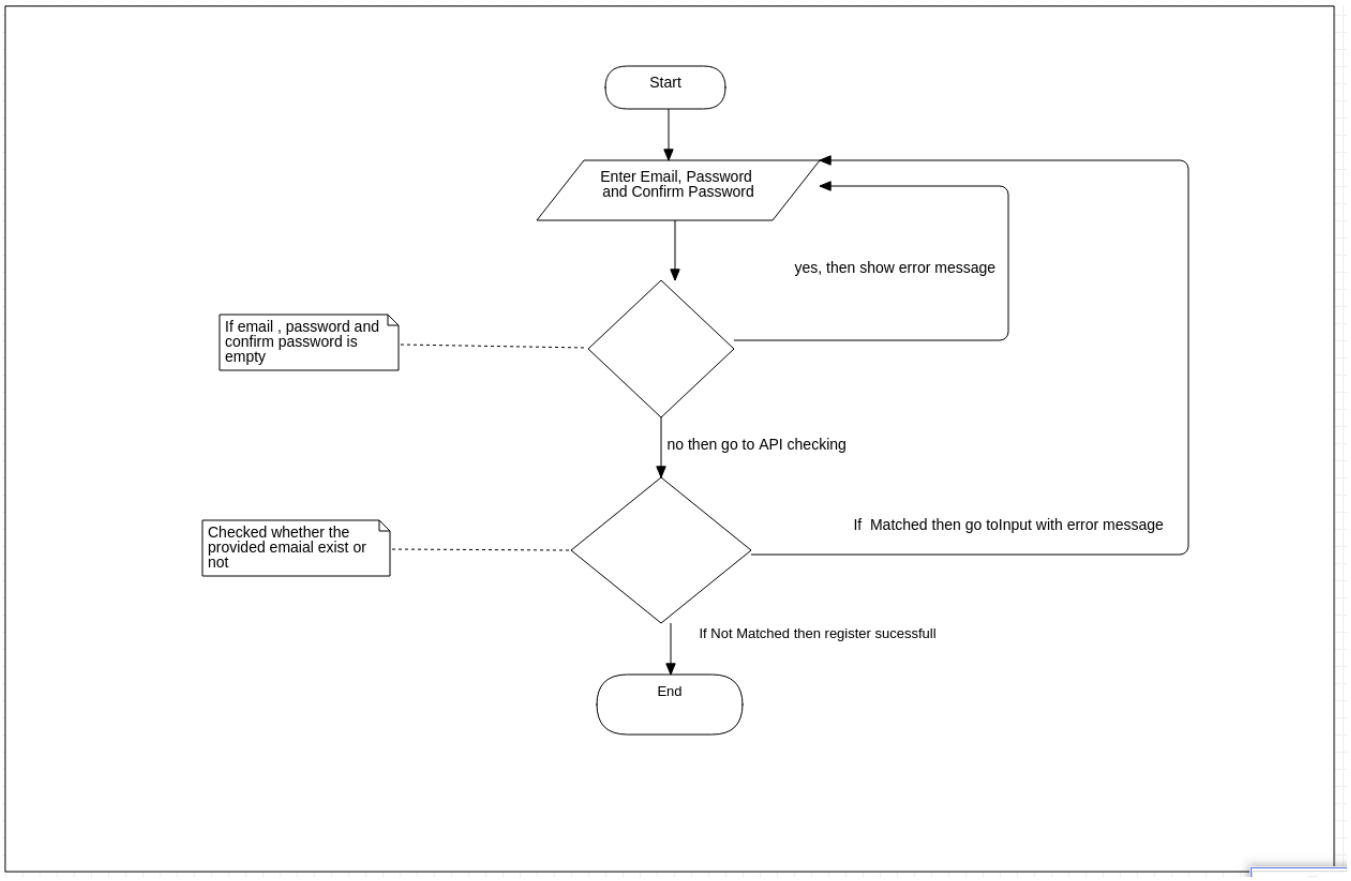
symbol is displayed and the process step is briefly described. The flow charts are connected to files showing the direction of the process flow direction [10].

Advantage of Using Flow Chart:

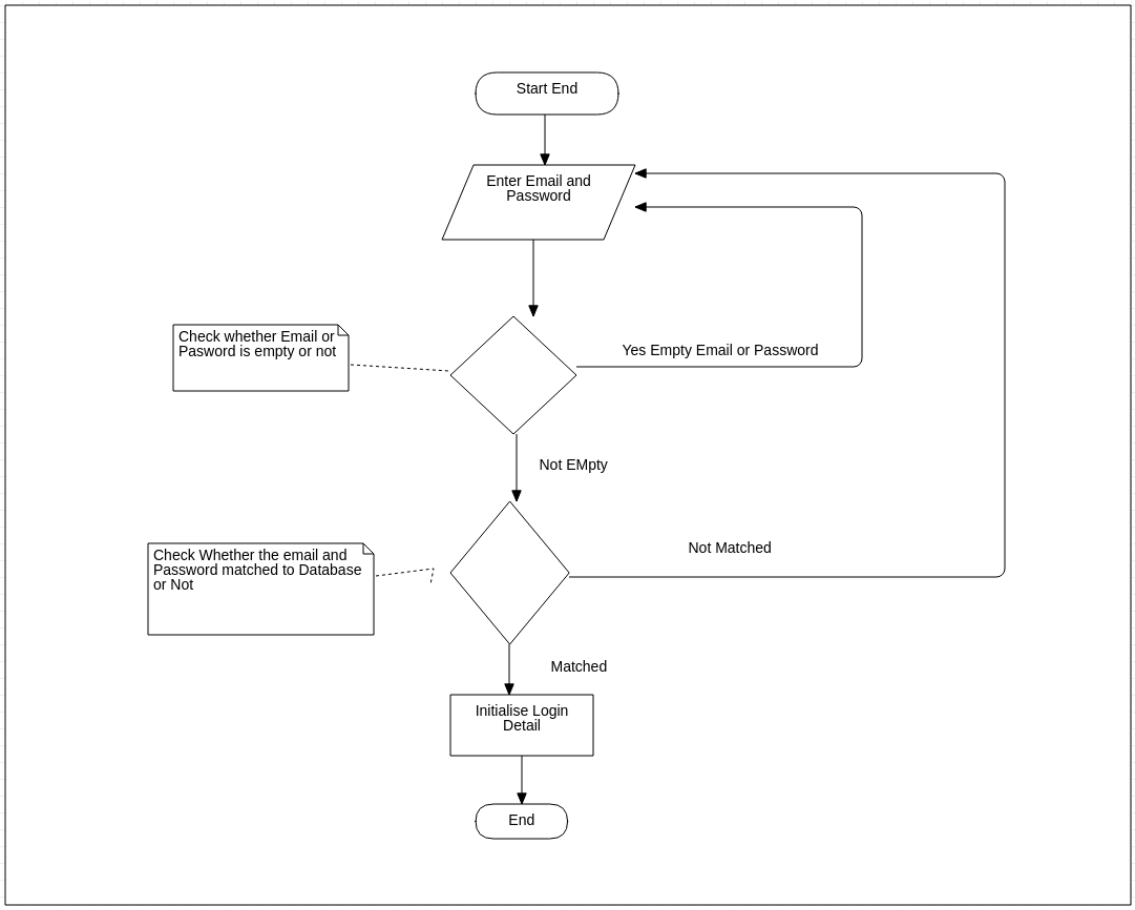
- It helps to maintain the effective communication as it helps to represent and define the logic of system or application.
- It helps do the effective analysis as the problem can identified quickly.
- It helps to do the easy debugging efficient testing.
- It will make coding efficient and effective coding s it helps to understand the code or function easily.
- It acts as a proper documentation which can be used for further process. [11]

Flow Charts of Function:

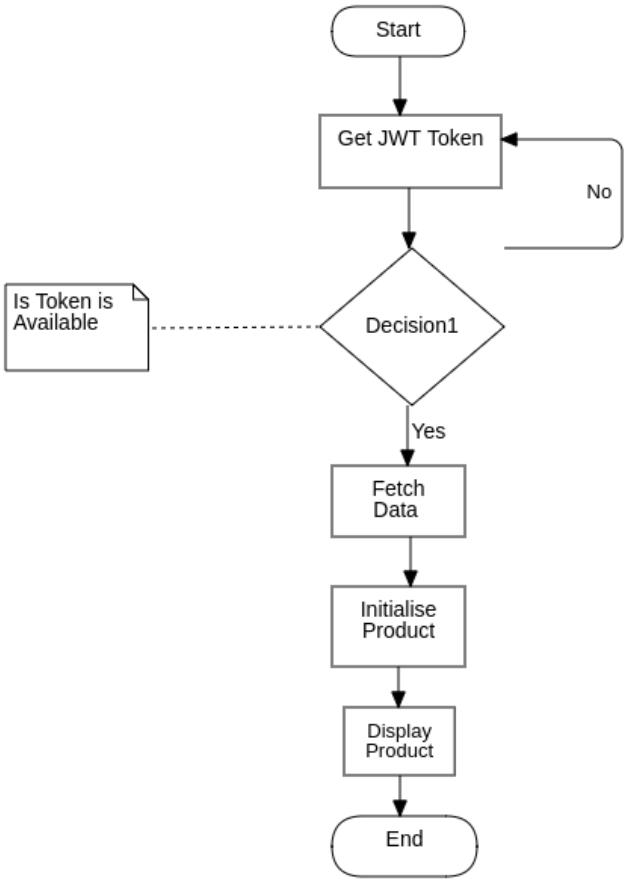
Register:



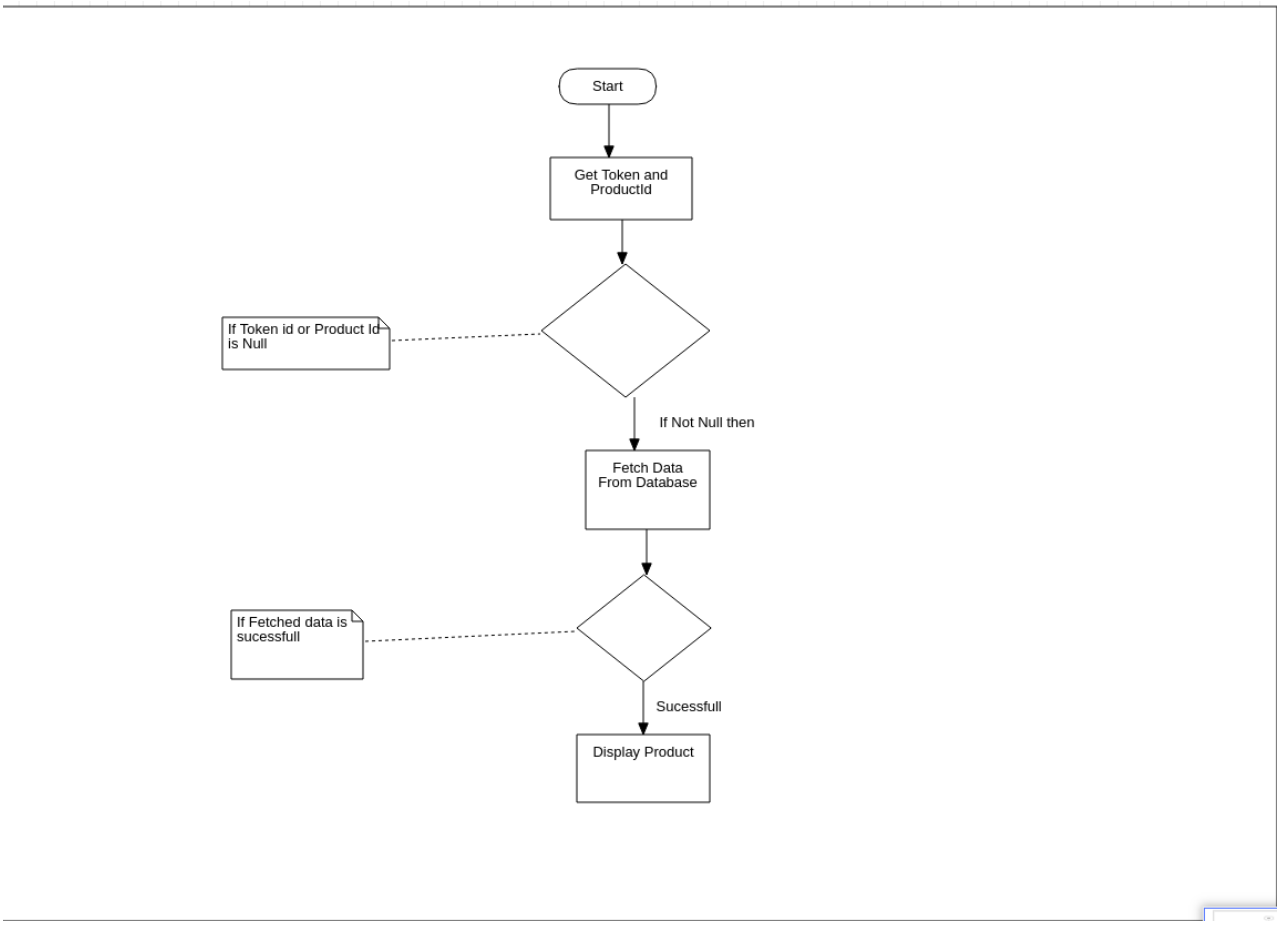
Login:



Get All Product:



Get All Product By Id:



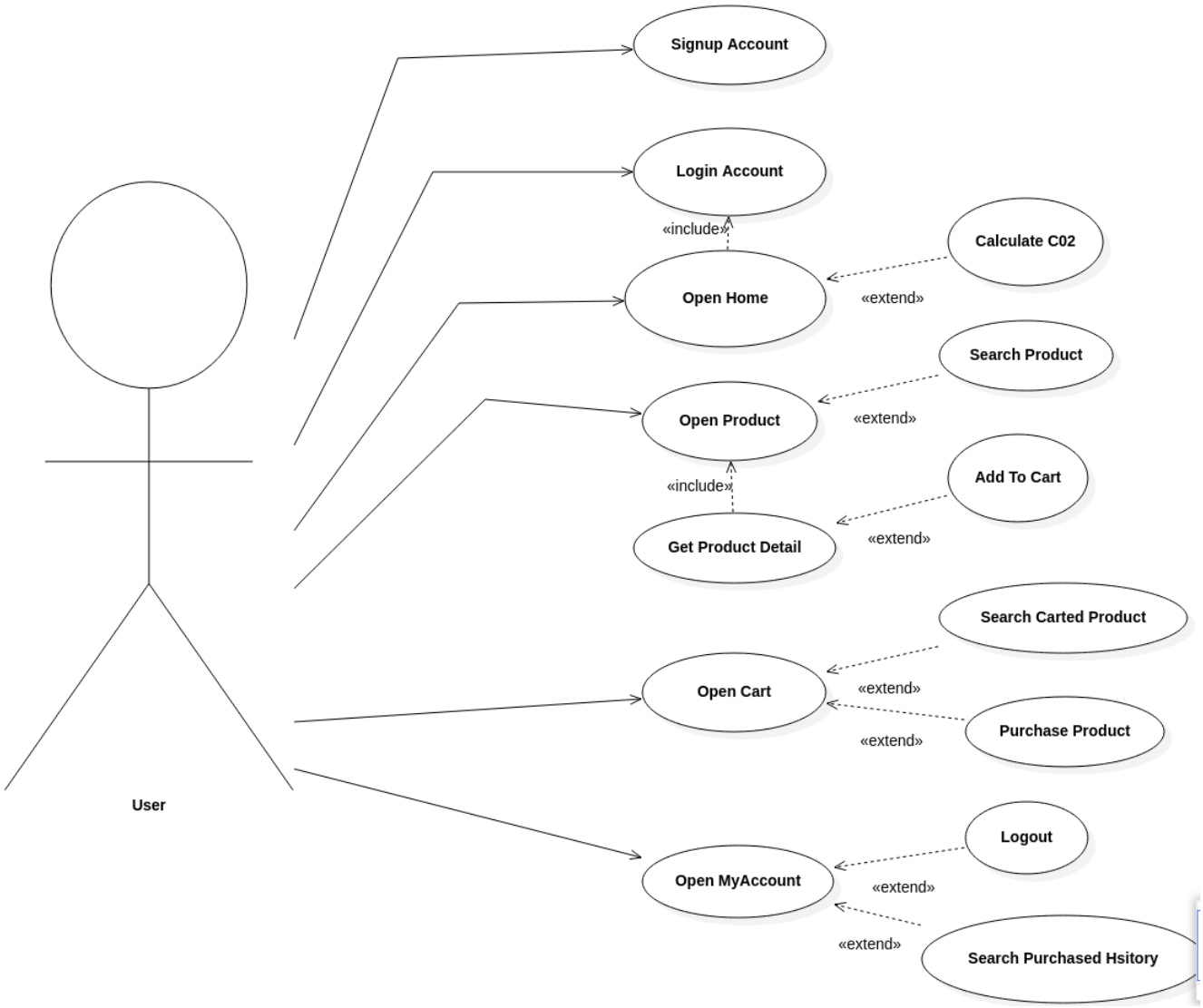
User-Case:

Use case diagram is such type of documentation Process of Diagram that helps to find or describe the interactions between the actors. Usually, a case diagram is straightforward. The details of the use cases are not shown. The development team can identify and understand where errors may occur during a transaction to resolve the errors through a use case document. A key idea for case modeling is that it helps us to design a system from the point of view of end users.

It is a powerful method to communicate system conduct in the terms of the user, by specifying all system behavior visible externally. It main uses and assets are:

- It summarizes only some relations between applications, actors and systems.
- The order in which the steps are taken to achieve each application case's goals is not shown.
- It helps to define the Scope of the System

Use Case of System from User Point of View:



References:

1. Guru99.com. 2021. *What is a Database? Definition, Meaning, Types, Example*. [online] Available at: <https://www.guru99.com/introduction-to-database-sql.html> [Accessed 5 August 2021].
2. Tutorialspoint.com. 2021. *MongoDB - Overview - Tutorialspoint*. [online] Available at: <https://www.tutorialspoint.com/mongodb/mongodb_overview.htm> [Accessed 5 August 2021].
3. MongoDB. 2021. *What is NoSQL? NoSQL Databases Explained*. [online] Available at: <https://www.mongodb.com/nosql-explained> [Accessed 5 August 2021].
4. MongoDB. 2021. *The most popular database for modern apps*. [online] Available at: <https://www.mongodb.com/> [Accessed 5 August 2021].
5. Spring.io. 2021. *Spring Data MongoDB*. [online] Available at: <https://spring.io/projects/spring-data-mongodb> [Accessed 5 August 2021].

6. Guru99.com. 2021. *What is Software Testing? Definition, Basics & Types in Software Engineering*. [online] Available at: <https://www.guru99.com/software-testing-introduction-importance.html> [Accessed 5 August 2021].
7. Junit.org. 2021. *JUnit 5*. [online] Available at: <https://junit.org/junit5/> [Accessed 5 August 2021].
8. www.javatpoint.com. 2021. *What is XML - javatpoint*. [online] Available at: <https://www.javatpoint.com/what-is-xml> [Accessed 5 August 2021].
9. GeeksforGeeks. 2021. *Difference between Gradle and Maven - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/difference-between-gradle-and-maven/> [Accessed 5 August 2021].
10. Breezetree.com. 2021. *What is a Flow Chart? | BreezeTree*. [online] Available at: <https://www.breezetree.com/articles/what-is-a-flow-chart> [Accessed 6 August 2021].
11. Codesansar. 2021. *Flowcharts (Guidelines, Advantages & Disadvantages)*. [online] Available at: <https://www.codesansar.com/computer-basics/flowcharts.htm> [Accessed 6 August 2021].