Implementation Document

for

store.it

Version 0.2

Prepared by

Group #: 3 **Group Name:** localghosts

Name	Roll No.	Email
Akanksha Singh	200070	akankshas20@iitk.ac.in
Antreev Singh Brar	190163	antreev@iitk.ac.in
Bhuvan Singla	180199	bhuvans@iitk.ac.in
Deepankur Kansal	180226	deepank@iitk.ac.in
Dipanshu Garg	190306	dipanshu@iitk.ac.in
Harshit Raj	200433	harshitr20@iitk.ac.in
Hitesh Anand	200449	ahitesh20@iitk.ac.in
Manas Gupta	200554	manasg20@iitk.ac.in
Priya Gole	200727	priyagole20@iitk.ac.in
Tushar	190915	tusharb@iitk.ac.in

Course: CS253A

Mentor TA: Mr. Swastik Maiti

Date: 20th March, 2022

Contents

Contents I

Revisions I

1 Implementation Details 1

1.1 Presentation Tier 2
1.2 Logic Tier 2
1.3 Data Tier 3
1.4 Development and version control environment 3
1.5 Hosting Service 4
1.6 CI/CD Pipeline 4
1.7 Authentication and Authorization 4
1.8 Rest Endpoints 5

2 Codebase 16

2.1 store-it-frontend 16
2.2 store-it-backend 19

3 Completeness 22

Appendix A - Group Log 23

Revisions

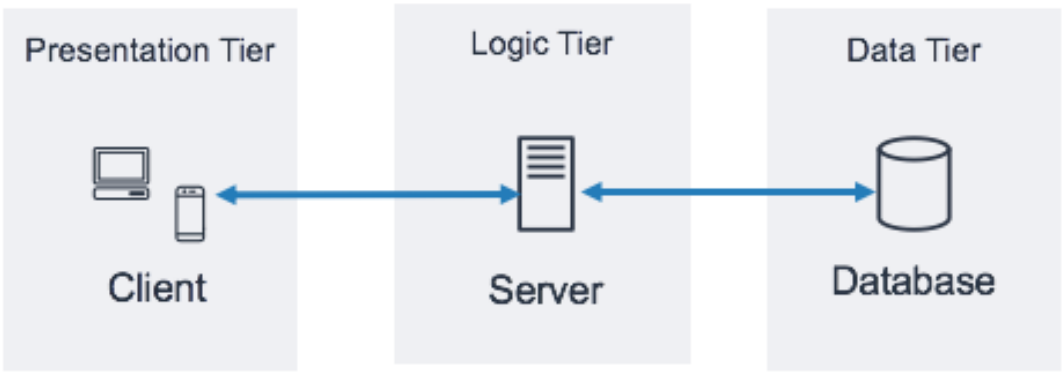
Version	Primary Author(s)	Description of Version	Date Completed
0.1	Akanksha Singh Antreev Singh Brar Bhuvan Singla Deepankur Kansal Dipanshu Garg Harshit Raj Hitesh Anand Manas Gupta Priya Gole Tushar	First draft	20/03/22

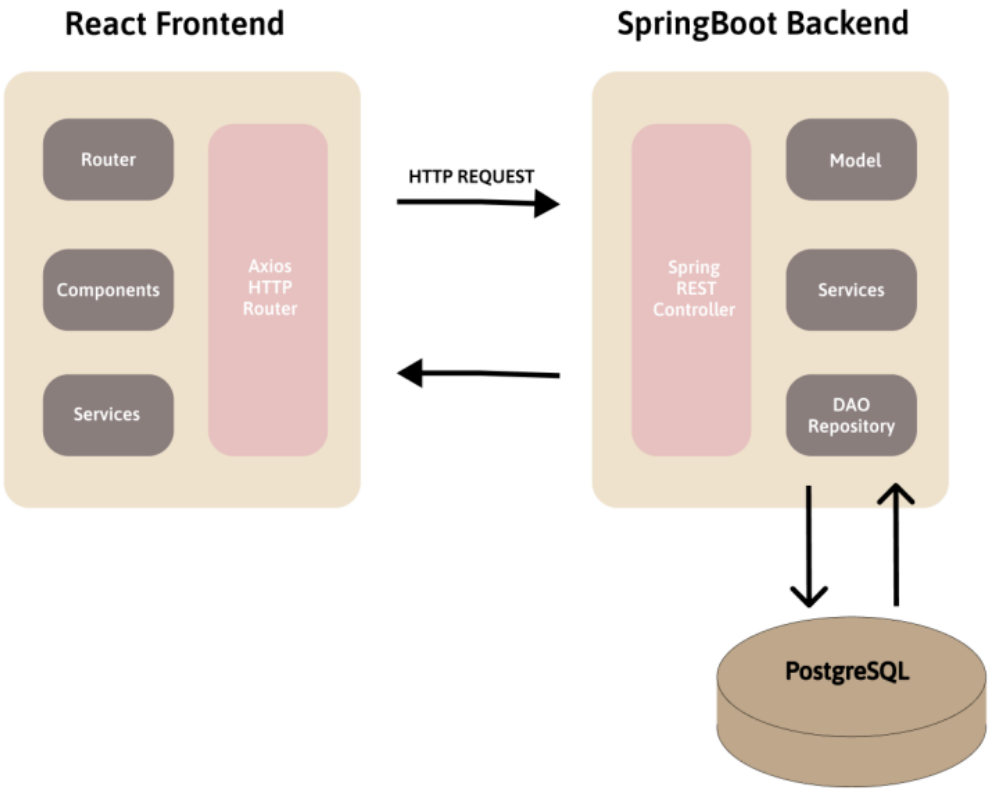
Version	Primary Author(s)	Description of Version	Date Completed
0.2	Akanksha Singh Antreev Singh Brar Bhuvan Singla Deepankur Kansal Dipanshu Garg Harshit Raj Hitesh Anand Manas Gupta Priya Gole Tushar	Final Draft	27/04/2022

1 Implementation Details

store-it implements a three-tier architecture, a popular pattern for user-facing applications. The tiers that comprise this architecture includes:

- **Presentation Tier**: This represents the components users directly interact with. In our case, this is the React based web application that runs inside a browser.
- **Logic Tier**: This contains the code required to translate user actions to the functionality, written in Java, that drives the application's behavior at the presentation tier.
- **Data Tier**: This consists of databases that hold the data relevant to the application.





1.1 Presentation Tier

The presentation layer is accessible to users via a browser and consists of user interface components that enable interaction with the system.

It is developed using three core technologies: **HTML, CSS, and JavaScript**. While HTML is the code that determines what the website will contain, CSS controls how it will look. JavaScript and its frameworks make the website interactive and responsive to a user's actions.

React, a JavaScript framework was used to make the content on the page dynamic. ReactJS is preferred as compared to other frameworks due to its

- **Speed**: React allows developers to utilize individual parts of their application which ultimately boosts the speed of the development process.
- **Flexibility**: Compared to other frontend frameworks, the React code is easier to maintain and is flexible due to its modular structure.
- **Performance**: The core of the framework offers a virtual DOM program and server-side rendering, which makes complex apps run extremely fast.
- **Usability**: Deploying React is fairly easy.
- **Reusable Components**: It saves time as we don't have to write various codes for the same features.

The library we used on top of React is **Material-UI**. MUI is a massive library of UI components designers and developers can use to build React applications. The open-source project follows Google's guidelines for creating components, giving us a customizable library of foundational and advanced UI elements. The reasons why the MUI component library is used:

- **Faster Time-to-Market**: We can spend more time designing customer experiences rather than spending time building and testing UI components from scratch–a process that increases time-to-market significantly.
- **Design Consistency**: Consistency is vital for user experience.
- **Scalability**: With MUI's comprehensive UI library, designers can search for the components they need to prototype and scale right away. Engineers can copy/paste the identical React components from MUI and customize them to the designer's specifications.

1.2 Logic Tier

The Logic Tier accepts user requests from the browser, processes them and determines the routes through which the data will be accessed. The workflows by which the data and requests travel through the backend are encoded in this layer.

The logic tier is implemented in **Java**, one of the most widely used programming languages in the world.
The advantages of using Java is as follows:

- **Simplicity**: Java has been used by developers for more than 20 years and is considered to be one of the simplest languages to learn due to its less ambiguous syntax terminology.
- **Cross-platform**: Being an object-oriented compiled language, Java allows you to write the code once and run it anywhere on any platform (Windows, Mac OS, and Linux)

- **Multi-threading**: Java uses a multi-threaded web server that processes each request in a separate thread. That enables it to perform several tasks simultaneously without querying the events.
- **Robust & scalable**: The automatic memory management and garbage collection make Java highly scalable and speed up the development of web applications.

The framework used on the top of Java is **Spring Boot**. Spring Boot is an open-source Java-based framework used to create web services, especially microservices.

Spring Boot offers the following advantages to its developers −

- Easy to understand and develop spring applications
- It provides a flexible way to configure Java Beans, XML configurations, and Database Transactions.
- It provides powerful batch processing and manages REST endpoints.
- In Spring Boot, everything is auto-configured; no manual configurations are needed.
- It offers annotation-based spring application
- Eases dependency management
- It includes Embedded Servlet Container

1.3 Data Tier

The Data Tier, sometimes called Database Tier, is where the information processed by the application is stored and managed. We have used **PostgreSQL**, an open-source object-relational database system with over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance. We have hosted the database on **AWS.**

Below are the main advantages/benefits of PostgreSQL:

- PostgreSQL source code is freely available under an open-source license. This allows us the freedom to use, modify, and implement it as per our business needs.
- To learn Postgres, you don’t need much training as it's easy to use.
- Its write-ahead logging makes it a highly fault-tolerant database.
- Low maintenance.
- Smooth integration with Java using JPA.

1.4 Development and version control environment

We have used **Git** as our version control system. It is the most commonly used version control system that is used for software development and other version control tasks.
It tracks the changes you make to files, so you have a record of what has been done, and you can revert to specific versions should you ever need to. It also makes collaboration easier, allowing changes by multiple people to all to be merged into one source.

We used **GitHub**, a web-based Git repository hosting service to manage our repositories and collaborate amongst ourselves.
Jira is a proprietary issue tracking product developed by Atlassian that allows bug tracking and agile project management

We have used **Jira** for issue/tickets tracking which helped us in bug tracking and agile project management.

1.5 Hosting Service

We used Heroku, a platform as a service (PaaS) that enables developers to build, run, and operate applications entirely in the cloud. All three tiers of our application, the frontend, the backend and the database run on the Heroku cloud.

Here are the pros/benefits of using Heroku:

- Allows the developer to focus on code instead of infrastructure
- Generous free-tier is useful for students.
- Enhance the productivity of cloud app development team
- Simple Horizontal & Vertical Scalability
- Offers a powerful dashboard and CLI
- Integrates with familiar developer workflows
- Beginner and startup-friendly

1.6 CI/CD Pipeline

CI/CD is a method to frequently deliver apps to customers by introducing automation into the stages of app development.

Development: We have added actions that run on every pull request and detect any possible linting errors and code smells.

Deployment: We have set up actions on our repositories that continuously deploy the main branch of our applications' repositories to the Heroku cloud.

1.7 Authentication and Authorization

- To implement **authentication** we have stored the email and password in our database. The password uses **cryptographic techniques** of **hashing and salting** to prevent the data being exposed in case of data leaks.
- To implement **authorization**, we have made use of JWT (Json
- Web Token)

1.8 Rest endpoints

POST otp

```
http://localhost:8080/otp
```

Headers

Content-Type	application/json
Authorization	Bearer null

Body json

```
{
  "email": "hello@bhuvansingla.com"
}
```

POST seller singup

```
http://localhost:8080/seller/signup
```

Headers

Content-Type	application/json

Body json

```
{
  "email": "harshit9304@gmail.com",
  "password": "123",
  "otp": "gcoCPy",
  "name": "Harshit",
  "storename": "Something",
  "storelogo": "https://images.unsplash.com/photo-1603966474815-85d21585ffb9?
ixlib=rb-
1.2.1&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=18
85&q=80",
  "storebanner": "https://images.unsplash.com/photo-1603966474815-85d21585ffb9?
ixlib=rb-
1.2.1&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=18
85&q=80"
}
```

POST login seller

```
http://localhost:8080/seller/login
```

Headers

Content-Type	application/json

Body json

```
{
  "email": "harshit9304@gmail.com",
  "password": "123"
}
```

POST buyer singup

```
http://localhost:8080/buyer/signup
```

Headers

Content-Type	application/json

Body json

```
{
  "email": "harshit9304@yahoo.com",
  "password": "123",
  "otp": "rcxlN6",
  "name": "Someone"
}
```

POST login buyer

```
http://localhost:8080/buyer/login
```

Headers

Content-Type	application/json

Body json

```
{
  "email": "harshit9304@yahoo.com",
  "password": "123"
}
```

GET get stores

```
http://localhost:8080/stores
```

Headers

Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20iLCJleHAiOjE2NDc4Nzk5NzEsImlhdCI6MTY0NzM3OTk3MSwidXNlcm5hbWUiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20ifQ.j3tqP023LEuMu7cmMo5gWgmFEt6tFK2C3ZBHEa-_nPHXPpcEzDaP1_27FLQJLffMiAQOniVcLq0Mq7xs4dyy1g

DELETE delete category

```
http://localhost:8080/store/lazyfoods/category/21
```

Headers

Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20iLCJleHAiOjE2NDc4Nzk5NzEsImlhdCI6MTY0NzM3OTk3MSwidXNlcm5hbWUiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20ifQ.j3tqP023LEuMu7cmMo5gWgmFEt6tFK2C3ZBHEa-_nPHXPpcEzDaP1_27FLQJLffMiAQOniVcLq0Mq7xs4dyy1g

POST add category

```
http://localhost:8080/store/lazyfoods/category
```

Headers

Content-Type	application/json
Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20iLCJleHAiOjE2NDc4Nzk5NzEsImlhdCI6MTY0NzM3OTk3MSwidXNlcm5hbWUiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20ifQ.j3tqP023LEuMu7cmMo5gWgmFEt6tFK2C3ZBHEa-_nPHXPpcEzDaP1_27FLQJLffMiAQOniVcLq0Mq7xs4dyy1g

Body json

```
{
  "name": "chooooco",
  "image": "linktosomething"
}
```

GET search

```
http://localhost:8080/products
```

Parameters

name	b

Headers

Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20iLCJleHAiOjE2NDc4Nzk5NzEsImlhdCI6MTY0NzM3OTk3MSwidXNlcm5hbWUiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20ifQ.j3tqP023LEuMu7cmMo5gWgmFEt6tFK2C3ZBHEa-_nPHXPpcEzDaP1_27FLQJLffMiAQOniVcLq0Mq7xs4dyy1g

GET get prod

```
http://localhost:8080/store/lazyfoods/1
```

Headers

Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20iLCJleHAiOjE2NDc4Nzk5NzEsImlhdCI6MTY0NzM3OTk3MSwidXNlcm5hbWUiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20ifQ.j3tqP023LEuMu7cmMo5gWgmFEt6tFK2C3ZBHEa-_nPHXPpcEzDaP1_27FLQJLffMiAQOniVcLq0Mq7xs4dyy1g

PUT toggle prod

```
http://localhost:8080/store/lazyfoods/product/3/toggle
```

Headers

Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20iLCJleHAiOjE2NDc4Nzk5NzEsImlhdCI6MTY0NzM3OTk3MSwidXNlcm5hbWUiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20ifQ.j3tqP023LEuMu7cmMo5gWgmFEt6tFK2C3ZBHEa-_nPHXPpcEzDaP1_27FLQJLffMiAQOniVcLq0Mq7xs4dyy1g

DELETE del prod

```
http://localhost:8080/store/lazyfoods/product/3
```

Headers

Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20iLCJleHAiOjE2NDc4Nzk5NzEsImlhdCI6MTY0NzM3OTk3MSwidXNlcm5hbWUiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20ifQ.j3tqP023LEuMu7cmMo5gWgmFEt6tFK2C3ZBHEa-_nPHXPpcEzDaP1_27FLQJLffMiAQOniVcLq0Mq7xs4dyy1g

POST add prod

```
http://localhost:8080/store/lazyfoods/1
```

Headers

Content-Type	application/json
Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20iLCJleHAiOjE2NDc4Nzk5NzEsImlhdCI6MTY0NzM3OTk3MSwidXNlcm5hbWUiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20ifQ.j3tqP023LEuMu7cmMo5gWgmFEt6tFK2C3ZBHEa-_nPHXPpcEzDaP1_27FLQJLffMiAQOniVcLq0Mq7xs4dyy1g

---

Body json

```
{
  "name": "Vanilaaaaaaaaaaaa icecream",
  "price": 123
}
```

POST add cart

```
http://localhost:8080/store/lazyfoods/cart/9/2
```

Headers

Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20iLCJleHAiOjE2NDc4Nzk5NzEsImlhdCI6MTY0NzM3OTk3MSwidXNlcm5hbWUiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20ifQ.j3tqP023LEuMu7cmMo5gWgmFEt6tFK2C3ZBHEa-_nPHXPpcEzDaP1_27FLQJLffMiAQOniVcLq0Mq7xs4dyy1g

GET get cart

```
http://localhost:8080/store/lazyfoods/cart
```

Headers

Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20iLCJleHAiOjE2NDc4Nzk5NzEsImlhdCI6MTY0NzM3OTk3MSwidXNlcm5hbWUiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20ifQ.j3tqP023LEuMu7cmMo5gWgmFEt6tFK2C3ZBHEa-_nPHXPpcEzDaP1_27FLQJLffMiAQOniVcLq0Mq7xs4dyy1g

POST checkout

```
http://localhost:8080/store/lazyfoods/checkout
```

Headers

Content-Type	application/json
Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20iLCJleHAiOjE2NDc4Nzk5NzEsImlhdCI6MTY0NzM3OTk3MSwidXNlcm5hbWUiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20ifQ.j3tqP023LEuMu7cmMo5gWgmFEt6tFK2C3ZBHEa-_nPHXPpcEzDaP1_27FLQJLffMiAQOniVcLq0Mq7xs4dyy1g

Body json

```
{
  "address": "H2/C233"
}
```

GET All orders

```
http://localhost:8080/orders
```

Headers

Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20iLCJleHAiOjE2NDc4Nzk5NzEsImlhdCI6MTY0NzM3OTk3MSwidXNlcm5hbWUiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20ifQ.j3tqP023LEuMu7cmMo5gWgmFEt6tFK2C3ZBHEa-_nPHXPpcEzDaP1_27FLQJLffMiAQOniVcLq0Mq7xs4dyy1g

GET store orders

```
http://localhost:8080/store/lazyfoods/orders
```

Headers

Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20iLCJleHAiOjE2NDc4Nzk5NzEsImlhdCI6MTY0NzM3OTk3MSwidXNlcm5hbWUiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20ifQ.j3tqP023LEuMu7cmMo5gWgmFEt6tFK2C3ZBHEa-_nPHXPpcEzDaP1_27FLQJLffMiAQOniVcLq0Mq7xs4dyy1g

PUT store orders update

```
http://localhost:8080/store/lazyfoods/order/42
```

Headers

Content-Type	application/json
Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20iLCJleHAiOjE2NDc4Nzk5NzEsImlhdCI6MTY0NzM3OTk3MSwidXNlcm5hbWUiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20ifQ.j3tqP023LEuMu7cmMo5gWgmFEt6tFK2C3ZBHEa-_nPHXPpcEzDaP1_27FLQJLffMiAQOniVcLq0Mq7xs4dyy1g

Body json

```
{
  "status": "ACCEPTED"
}
```

GET get category

```
http://localhost:8080/store/lazyfoods/category
```

Headers

Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20iLCJleHAiOjE2NDc4Nzk5NzEsImlhdCI6MTY0NzM3OTk3MSwidXNlcm5hbWUiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20ifQ.j3tqP023LEuMu7cmMo5gWgmFEt6tFK2C3ZBHEa-_nPHXPpcEzDaP1_27FLQJLffMiAQOniVcLq0Mq7xs4dyy1g

GET /

```
http://localhost:8080/
```

Headers

Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20iLCJleHAiOjE2NDc4Nzk5NzEsImlhdCI6MTY0NzM3OTk3MSwidXNlcm5hbWUiOiJoYXJzaGl0OTMwNEB5YWhvby5jb20ifQ.j3tqP023LEuMu7cmMo5gWgmFEt6tFK2C3ZBHEa-_nPHXPpcEzDaP1_27FLQJLffMiAQOniVcLq0Mq7xs4dyy1g

2 Codebase

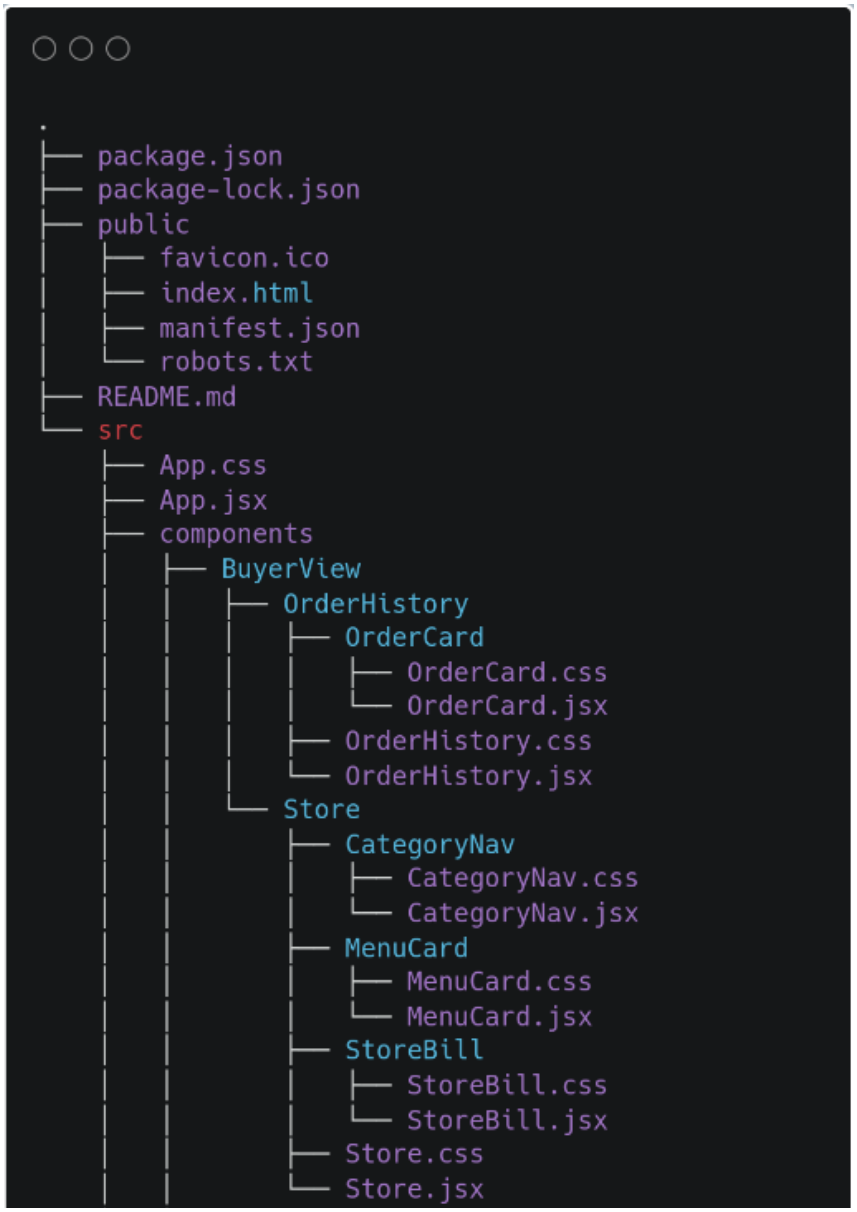
An organization has been set up on GitHub for storing and collaborating on the source code of this project store-it-localghosts.

The organization hosts **four public repositories**:

- store-it-frontend
- store-it-backend
- store-it
- store-it-test-logs

2.1 store-it-frontend

Repository contains the following files:

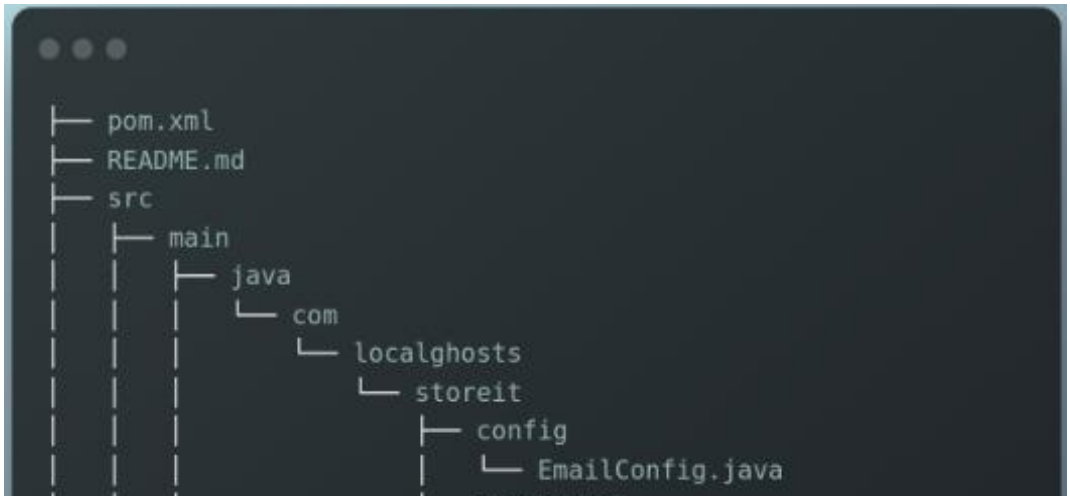

.
├── package.json
├── package-lock.json
├── public
│   ├── favicon.ico
│   ├── index.html
│   ├── manifest.json
│   └── robots.txt
├── README.md
└── src
    ├── App.css
    ├── App.jsx
    ├── components
    │   ├── BuyerView
    │   │   ├── OrderHistory
    │   │   │   ├── OrderCard
    │   │   │   │   ├── OrderCard.css
    │   │   │   │   └── OrderCard.jsx
    │   │   │   ├── OrderHistory.css
    │   │   │   └── OrderHistory.jsx
    │   │   └── Store
    │   │   ├── CategoryNav
    │   │   │   ├── CategoryNav.css
    │   │   │   └── CategoryNav.jsx
    │   │   ├── MenuCard
    │   │   │   ├── MenuCard.css
    │   │   │   └── MenuCard.jsx
    │   │   ├── StoreBill
    │   │   │   ├── StoreBill.css
    │   │   │   └── StoreBill.jsx
    │   │   ├── Store.css
    │   │   └── Store.jsx

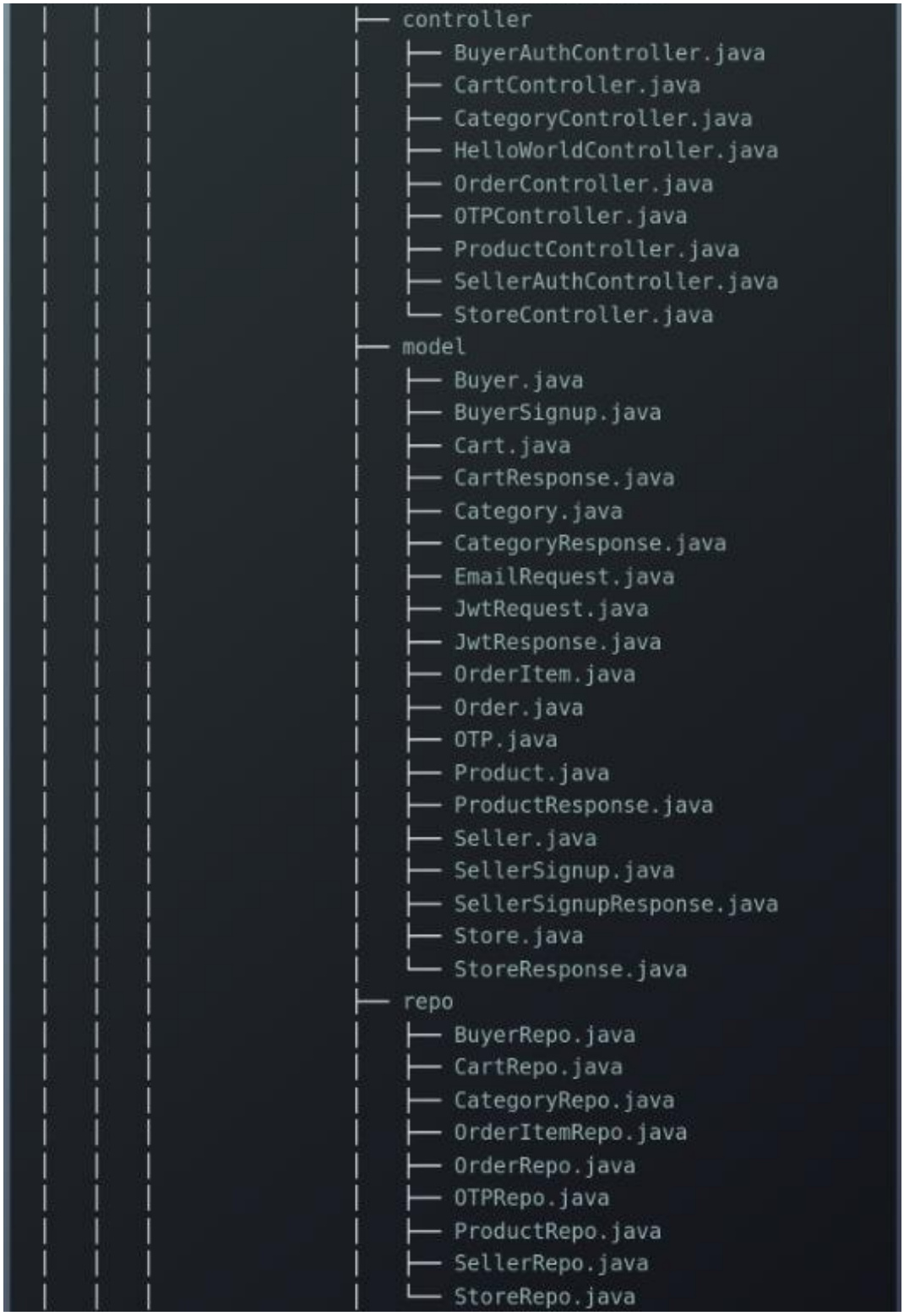

Catalog
CatalogItem.jsx
Catalog.jsx
Home
Home.css
Home.jsx
Login-SignUp
Buyer
BuyerLogin.jsx
BuyerSignUp.jsx
Login.css
Login.jsx
Seller
SellerLogin.jsx
SellerSignUp.jsx
Navbar
Card.jsx
Display.jsx
Navbar.jsx
Nav.css
Topbar.jsx
SellerView
Dashboard
Categories
AddCategory
AddCategory.css
AddCategory.jsx
Categories.css
Categories.jsx
CategoryLog
CategoryLog.css
CategoryLog.jsx
Dashboard.css
Dashboard.jsx
Navigation
Navigation.css
Navigation.jsx
Orders
OrderCard
OrderCard.css
OrderCard.jsx
OrderStatus
AcceptOrReject.jsx
AfterAccepted.jsx
AfterRejected.jsx
Orders.css
Orders.jsx

- **README.md**: The steps on how to run the app is given
- **package.json**: contains human-readable metadata about the project (like the project name and description) as well as functional metadata like the package version number and a list of dependencies required by the application.
- **src/index.jsx**: works as an entry point in the application
- **src/components** folder: contains all the components of the frontend (Login-SignUp, Home, Navbar, Catalog, BuyerView, SellerView/Dashboard). All these sub-folders contain the implementation of respective components.

2.2 store-it-backend

Repository contains the following files:




controller
BuyerAuthController.java
CartController.java
CategoryController.java
HelloWorldController.java
OrderController.java
OTPController.java
ProductController.java
SellerAuthController.java
StoreController.java
model
Buyer.java
BuyerSignup.java
Cart.java
CartResponse.java
Category.java
CategoryResponse.java
EmailRequest.java
JwtRequest.java
JwtResponse.java
OrderItem.java
Order.java
OTP.java
Product.java
ProductResponse.java
Seller.java
SellerSignup.java
SellerSignupResponse.java
Store.java
StoreResponse.java
repo
BuyerRepo.java
CartRepo.java
CategoryRepo.java
OrderItemRepo.java
OrderRepo.java
OTPRepo.java
ProductRepo.java
SellerRepo.java
StoreRepo.java

- **pom.xml**: contains information of project and configuration information for the maven to build the project such as dependencies, build directory, source directory, test source directory, plugin, goals, etc.
- **src/main/resources/application.properties**: defines our application configuration
- **src/main/java/com/localghosts/storeit/** folder:
    - **controller** : contains the end points of the application
    - **model** : models the entities
    - **repo** : interface of the model with database
    - **security** : takes care of jwt
    - **StoeritBackendApplication.java** : works as an entry point

3 Completeness

The “Section 3: Specific Requirements” of Software Requirements Specification Document lists all the desired product functionality.

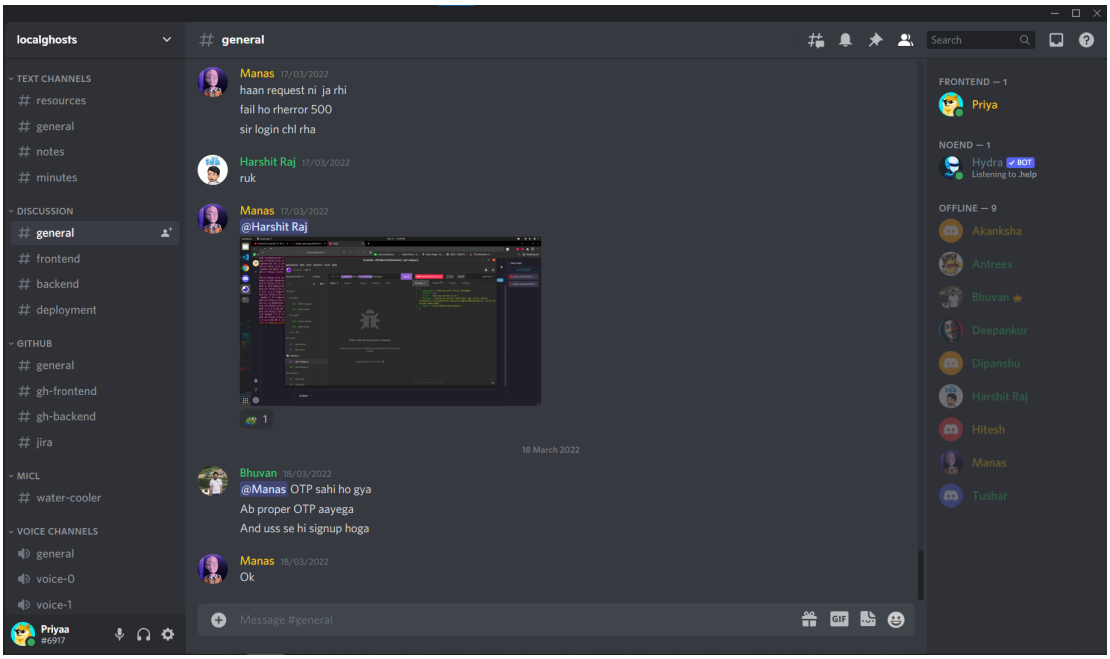
- “Section 3.1: External Interface Requirements” is implemented in **store-it-frontend**. Different user interfaces displayed in section 3.1 of the SRS document are implemented using React. The different views include - Home page, catalog, seller dashboard views (products, orders, categories), signup/login view, buyer’s view.
- All the features listed in “Section 3.2: Functional Requirements” except for 3.2.7, and 3.2.18 are implemented in **store-it-backend**.

The application - “store-it” is aimed at digitizing various physical stores and service providers at the IIT Kanpur campus. Currently, the implementation is done to provide the basic features for both the sellers and buyers, i.e., placing orders(for buyers), managing orders(for sellers), checking order history, browsing through store catalogs, etc. In addition to these features, we have a future development plan to improve the functionality of our product by adding more features which include:

- Adding an online payment option via credit/debit cards, net banking, UPI to increase the ease of transactions.
- Deploying a corresponding mobile app for enhancing the ease of use and accessibility. Most people prefer mobile apps over web browsers as it suits their convenience.
- Possibility of using other languages in the future for users belonging to different regional backgrounds.
- The user feedback system for each store to improvise and adapt according to the changing customer needs. This feedback will include a rating for the current transaction as well as subjective suggestions to improve the store services as well as the services provided by our application.
- Ratings for each store will be displayed along with their names so that the customer can prefer the stores with better services first. These ratings will be based upon the feedback and individual ratings provided by the user after the completion of each transaction. This will help the users to make a sound choice while selecting among different stores providing similar services.
- An order tracking mechanism might be added, which will enable the users to track their order in real-time as it reaches them. This might enhance the transparency in the whole mechanism as the user will be able to track the order and complain in case of any unnecessary delay made by the seller.

Appendix A - Group Log

- The group activities during implementation were asynchronous, where we kept constant communication via our discord server and WhatsApp group.
- After people started to return to campus, we also worked offline inside the KD Lab of the Dept. of Computer Science and Engineering.



- 5:00 pm, 5th March 2022: Meeting with TA: Updated TA with our progress and discussed the issues we were facing.
- 5:00 - 7:00, 27th April 2022: Made final Changes to the Documentation