5308 - Advance Topics in Software Development

Project Report

Comparify

Developed by - Group 15
Developers

- Aman Singh Bhandari (B00910008)
- Chanpreet Singh (B00896766)
- Harsh Shah (B00899805)
- Meghna Rupchandani (B00888479)
- Qiwei Sun (B00780054)

1. Technology Stack

- Backend - Java and SpringBoot
- Frontend - ReactJS and Redux
- Database - MongoDB
- Deploy - Heroku
- Code Quality - DesigniteJAVA, Qscored and SonarQube
- CI/CD - GitLab

2. Dependencies

1. For Server (Backend)

Application uses maven as dependency manager. All the dependencies are configured in pom.xml in the root folder of server. To install dependency prerequisites are Java 1.8 and Maven 3. Run the following command to install all the dependencies:

```
mvn clean install
```

Name	Version	Reason
spring-boot-starter-web	2.6.3	This dependency is used to create the RESTful apis which is served by the server and client consumes it
spring-boot-starter-security	2.6.3	This dependency is used to create a security layer to protect all the important resources and api for unauthorized and unauthenticated access
spring-boot-starter-actuator	2.6.3	This dependency to check the application metrics like CPU usage, memory usage, application health, etc.
spring-boot-starter-data-mongodb	2.6.3	This dependency act as driver between application and database. Create a socket connection to transfer data in and out of the application
spring-boot-starter-websocket	2.6.3	This dependency is used to establish a socket connection between server and client to send the real-time messages and notification
spring-boot-starter-mail	2.6.3	This dependency is used to connect with mailbox to send mail directly from the application.
spring-boot-starter-thymeleaf	2.6.3	This dependency is used as a template engine, those templates are used to sent mail directly from the application

firebase-admin	8.1.0	This dependency is used to connect with Firebase Admin to trigger Cloud Message, those cloud message are used to show Web Notification.
Development Dependencies		
spring-boot-devtools	2.6.3	This dependency is used to create a live server that is reloaded automatically the server during development
Test Dependencies		
spring-boot-starter-test	2.6.3	This dependency is used to write and execute the unit and integration test.
spring-security-test	2.6.3	This dependency is used to write and execute the unit and integration test for the security
mockito-inline	4.3.1	This dependency is used to mock the static class in the unit and integration test
junit	4.13.2	This dependency is used to write and execute the unit and integration test using Junit runner

2. For Client (Frontend)

Application uses npm (node package manager) as dependency manager. All the dependencies are configured in package.json in the root folder of client. To install dependency prerequisites is NodeJS 14 or above. Run the following command to install all the dependencies:

```
npm install
```

Name	Version	Reason
@mui/material	5.5.2	This dependency is used to create a responsive UI component
@mui/system	5.5.1	This dependency is used to design the UI structure
@mui/icons-material	5.4.2	This dependency is used to render icons in the UI
react-redux	7.2.6	This dependency is used to manage the store in the application it can be application or module wide in react application
axios	0.26.1	This dependency is used to call the Rest APIs served by the server

express	4.17.3	This dependency is used to create a server that can serve the html pages, resources, JavaScript files, etc
firebase	9.6.10	This dependency is used connect the client application with the Firebase console and listen the message trigger by the server for Web Push Notification
formik	2.2.9	This dependency is used to create a form dynamically in the react.
highcharts	10.0.0	This dependency is used to create various visualization and chart to show the trends and statistics.
react-toastify	8.2.0	This dependency is used to show the pop-up notification within the application
sockjs-client	1.6.0	This dependency is used to open the socket connection.
stompjs	1.6.0	This dependency consumes the socket connection and subscribe to the topic where message can be pushed by the server,

3. For Database

Name	Version	Reason
MongoDB	5.0.6	MongoDB is used to persist the application data and perform various query on it. MongoDB helps in handling application unstructured data

3. Build documentation

1. Prerequisites:

1. Java version 1.8
2. Maven version 3
3. NodeJS version 14 or higher

2. For Server (Backend)

```
1. cd server
2. mvn clean install -Pprod
3. java -jar target\comparify-server.jar
```

3. For Client (Frontend)

```
1. cd client
2. npm i
3. npm run build
4. npm server/index.js
```

4. CI/CD Pipeline Stages

GitLab pipeline is used for Continuous Integration and Continuous Deployment. All the stages of pipeline uses either **docker** or **linux** runner to execute the commands. Following are the stages of the CI/CD pipeline:

1. Test

This stage will run all the testcases and generate 3 reports and used maven image to execute all the commands:

a. Surefire Report
b. Failsafe Report
c. Code Coverage Report

```
1. cd server
2. mvn clean verify -Ptest
```

Pipeline Script:

```
# ------------------------- Test ------------------------- #

test:
  stage: test
  image: maven:latest
  tags:
    - dalfcs_docker_autoscale
  script:
    - cd server
    - mvn clean verify
  artifacts:
    when: always
    reports:
      junit:
        - server/target/surefire-reports/TEST-*.xml
        - server/target/failsafe-reports/TEST-*.xml
  only:
    changes:
      - server/**/*
```

2. Build

This stage builds the application and generate a deployable artifact. This stage uses maven image to execute all the commands.

a. Build Server

```
1. cd server
2. mvn clean package -DskipTests=true -Pprod
```

Pipeline Script:

```
# ------------------------- Build Server for Production ------------------------- #

build_server_for_prod:
  stage: build
  image: maven:latest
  tags:
    - dalfcs_docker_autoscale
  script:
    - cd server
    - mvn clean package -DskipTests=true -Pprod
  artifacts:
    paths:
       - server/target/*.jar
  only:
    refs:
      - tags
    changes:
      - server/**/*
```

b. Build Client

```
1. cd client
2. npm install
3. npm run build
```

Pipeline Script:

```
# ------------------------- Build Client ------------------------- #

build_client:
  stage: build
  image: node:16
  tags:
    - dalfcs_docker_kvm
  script:
    - cd client
    - npm install
    - npm run build
  only:
    changes:
      - client/**/*
```

3. Code Quality

This stage will check the Code Quality of the application. Code Quality will be assessed by two tools DesigniteJAVA – Qscored and Sonar – SonarCloud. This stage uses linux runner to executes commands for DesigniteJAVA – Qscored and maven image to execute commands for Sonar and SonarCloud

a. Using DesigniteJAVA and Qscored

```
1. wget -O DesigniteJava.jar <DOWNLOAD_DESIGNITE_URL>
2. java -jar DesigniteJava.jar -ci -repo
   <CI_PROJECT_PATH> -pat <PAT> -host "git.cs.dal.ca"
3. eval <UPLOAD_QUALITY_REPORT>
```

Pipeline Script:

```
# ------------------------- Code Quality on Production ------------------------- #

code_quality_for_prod_by_designite:
  stage: code-quality
  tags:
    - ugrad
  variables:
    UPLOAD_QUALITY_REPORT: 'curl -X PUT
        -H "Authorization:Token $QSCORED_API_KEY_HARSH"
        -H "repository-link:$CI_PROJECT_URL"
        -H "username:$QSCORED_USERNAME_HARSH"
        -H "Content-Type:multipart/form-data"
        --url "https://qscored.com/api/upload/file.xml?is_open_access=off&version=$CI_PIPELINE_IID&project_name=$PROJECT_NAME_PROD"
        -F "file=$DESIGNITE_XML_OUTPUT"'
  script:
    - wget -O DesigniteJava.jar $DOWNLOAD_DESIGNITE_URL
    - java -jar DesigniteJava.jar -ci -repo $CI_PROJECT_PATH -pat $PAT -host "git.cs.dal.ca"
    - eval "$UPLOAD_QUALITY_REPORT"
  only:
    refs:
      - tags
    changes:
      - server/**/*
```

b. Using Sonar and SonarCloud

```
1.cd server
2.mvn clean verify sonar:sonar
```

Pipeline Script:

```
# ------------------------- Code Quality by Sonar ------------------------- #

code_quality_by_sonar:
  stage: code-quality
  image: maven:latest
  tags:
    - dalfcs_docker_autoscale
  script:
    - cd server
    - mvn clean verify sonar:sonar
  artifacts:
    when: always
    reports:
      junit:
        - server/target/surefire-reports/TEST-*.xml
        - server/target/failsafe-reports/TEST-*.xml
  only:
    changes:
      - server/**/*
```

4. Deploy

This stage will deploy the application on the Heroku servers by pushing the code to the Heroku Git repository. This stage used ruby image and install dpl tool to push code to Heroku.

a. Deploy Server

```
1. apt-get update -qy
2. apt-get install -y ruby-dev
3. gem install dpl
4. cd client
5. dpl --provider=heroku --app=<HEROKU_APP_NAME> --api-
   key=<HEROKU_API_KEY> --cleanup
```

Pipeline Script:

```yaml
# ------------------------- Deploy on Prod ------------------------- #
deploy_prod_client:
  stage: deploy-production
  image: ruby:latest
  tags:
    - dalfcs_docker_autoscale
  before_script:
    - apt-get update -qy
    - apt-get install -y ruby-dev
    - gem install dpl
  script:
    - cd client
    - dpl --provider=heroku --app=$PRODUCTION_CLIENT_HEROKU_APP_NAME --api-key=$HARSH_HEROKU_API_KEY --cleanup
  only:
    refs:
      - tags
    changes:
      - client/**/*
```

b. Deploy Client

```
1. apt-get update -qy
2. apt-get install -y ruby-dev
3. gem install dpl
4. curl --request PATCH
   "https://api.heroku.com/apps/<HEROKU_APP_NAME>/config-
   vars"
   --data "{\"SPRING_PROFILES_ACTIVE\": \"prod\"}"
   --header "Content-Type: application/json"
   --header "Accept: application/vnd.heroku+json;
   version=3"
   --header "Authorization: Bearer <HEROKU_API_KEY>"
5. cd server
6. dpl --provider=heroku --app=<HEROKU_APP_NAME> --api-
   key=<HEROKU_API_KEY> --cleanup
```

Pipeline Script:

```
deploy_prod_server:
  stage: deploy-production
  image: ruby:latest
  tags:
    - dalfcs_docker_autoscale
  before_script:
    - apt-get update -qy
    - apt-get install -y ruby-dev
    - gem install dpl
  script:
    - >
      curl --request PATCH "https://api.heroku.com/apps/$PRODUCTION_SERVER_HEROKU_APP_NAME/config-vars"
      --data "{\"SPRING_PROFILES_ACTIVE\": \"prod\"}"
      --header "Content-Type: application/json"
      --header "Accept: application/vnd.heroku+json; version=3"
      --header "Authorization: Bearer $HARSH_HEROKU_API_KEY"
    - cd server
    - dpl --provider=heroku --app=$PRODUCTION_SERVER_HEROKU_APP_NAME --api-key=$HARSH_HEROKU_API_KEY --cleanup
  when: delayed
  start_in: 5 minutes
  only:
    refs:
      - tags
    changes:
      - server/**/*
```

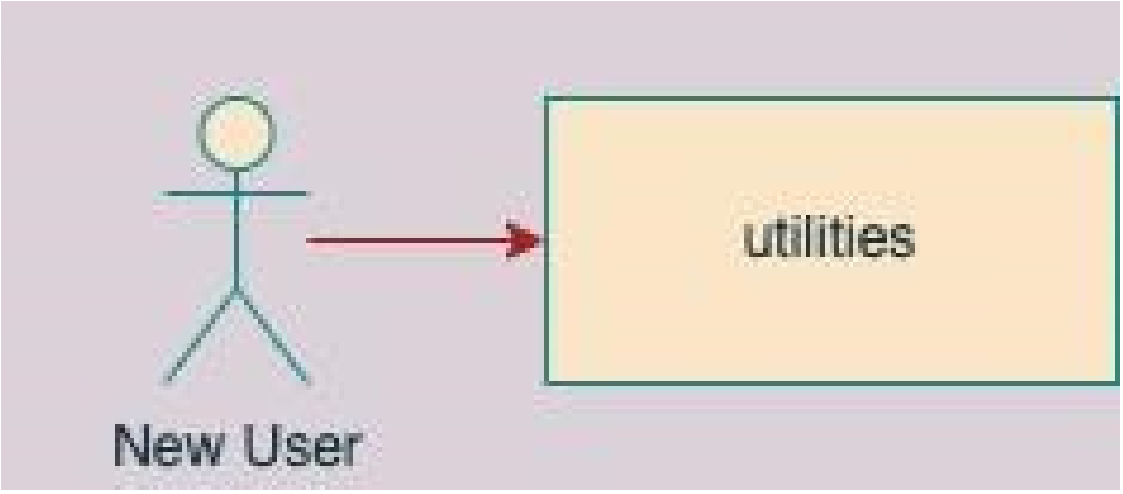
5. User Scenarios and UML

There are 2 types of target users – admin and a general end user.

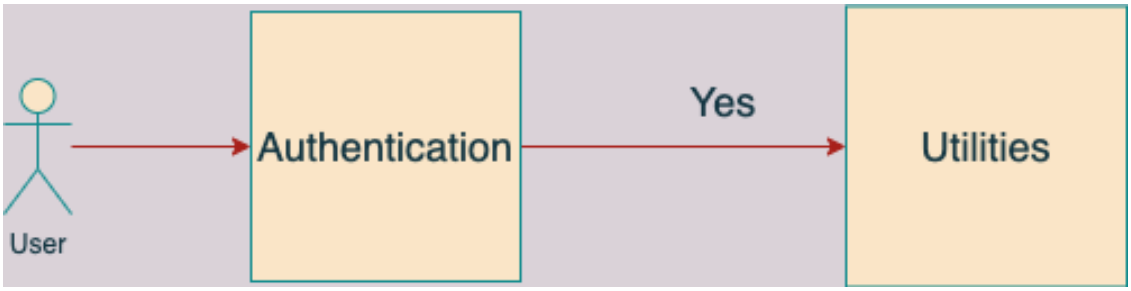
Following are the use cases for

<u>For a general end user</u>

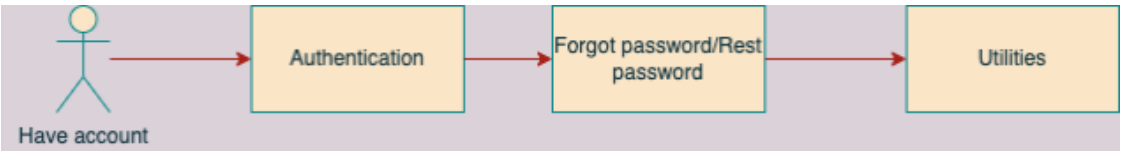
1. **Registration**



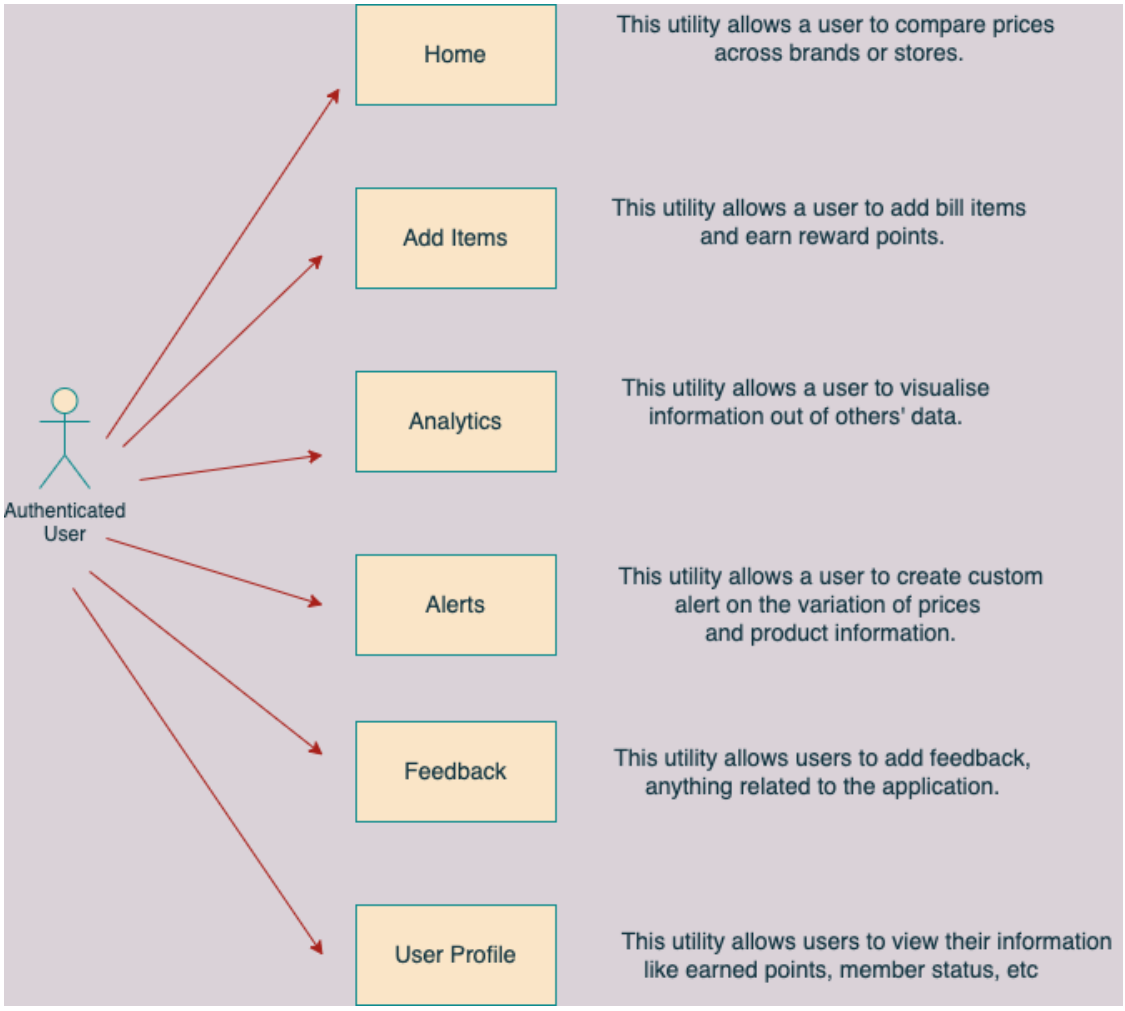
2. **Authentication**



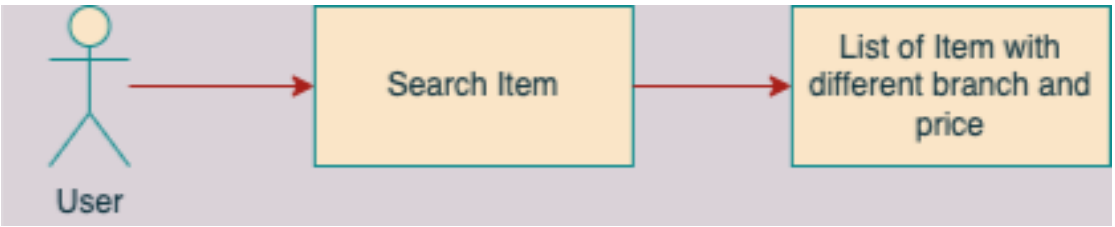
3. **Forgot Password**



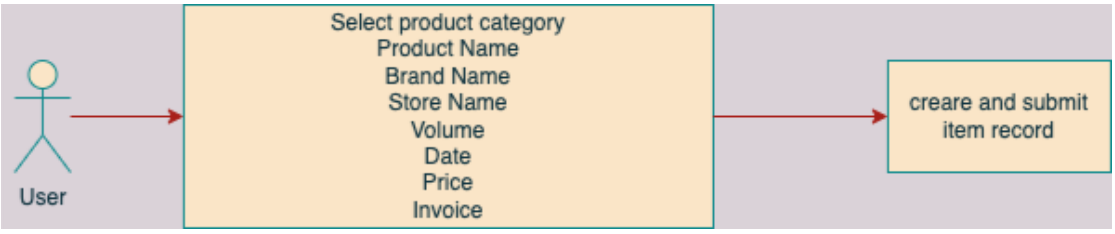
4. **Various utilities provided to an user**



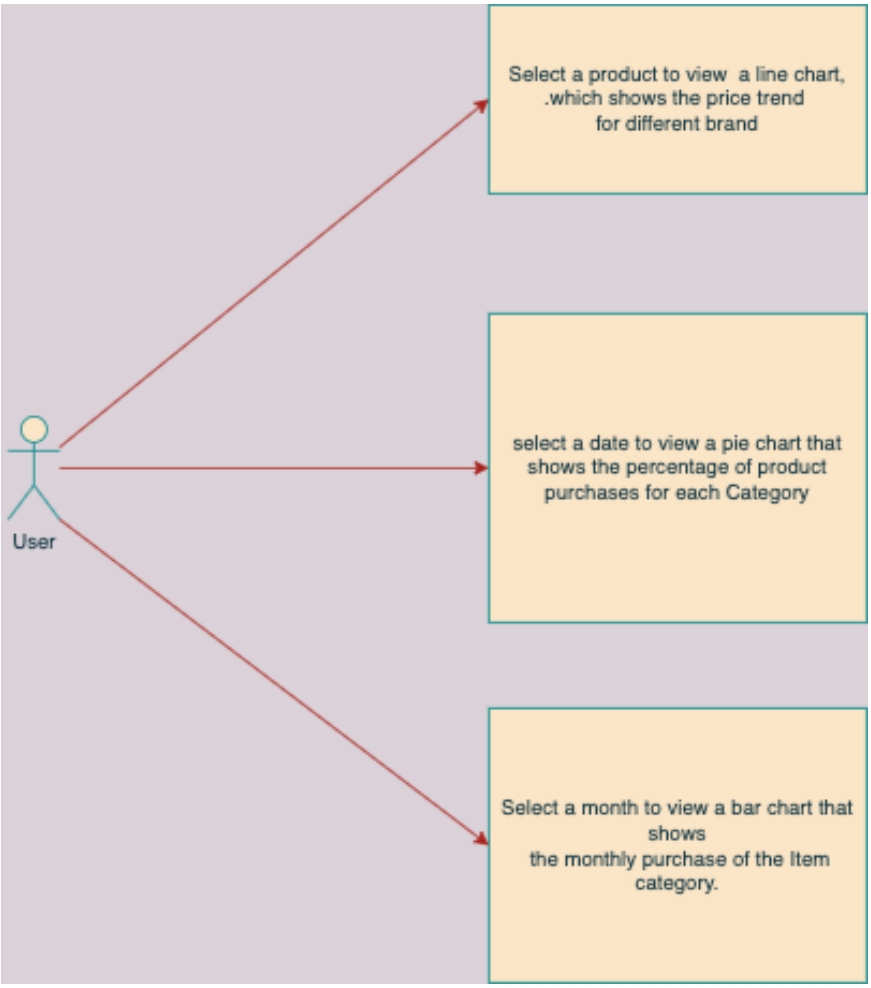
5. **Utility – Comparify module**



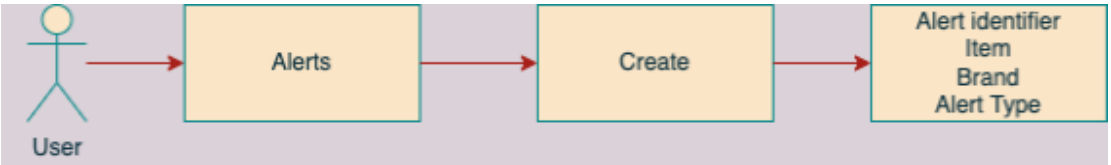
6. **Utility – Add bill items**



7. **Utility – Analytics**



8. **Utility – Create Alert**

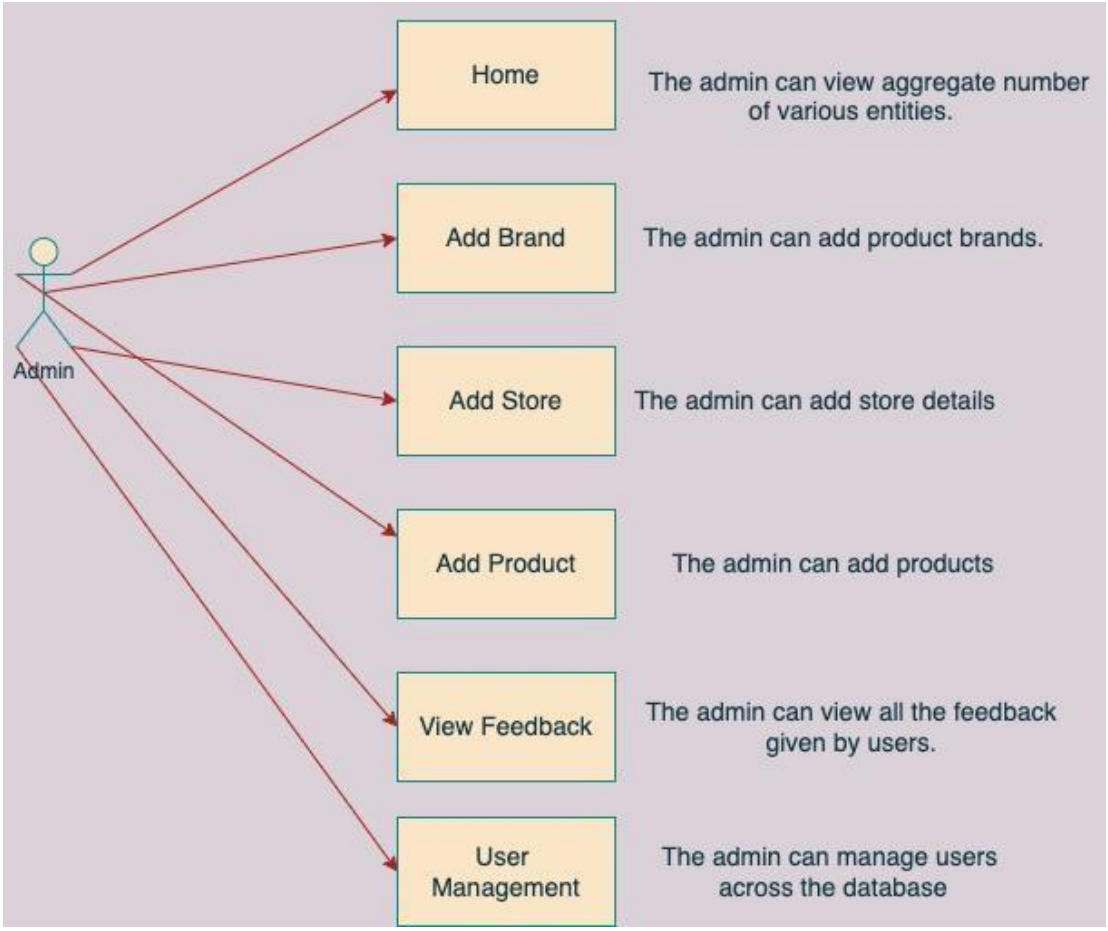


9. **Utility – User Profile**

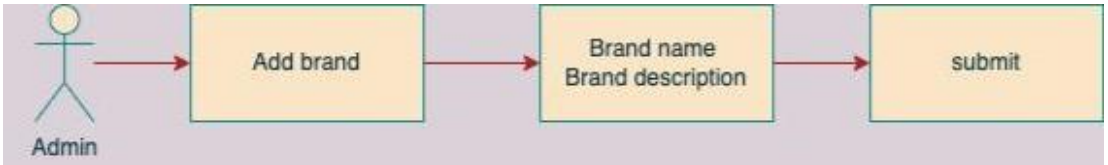
For the admin

As an admin, the person has the following utilities

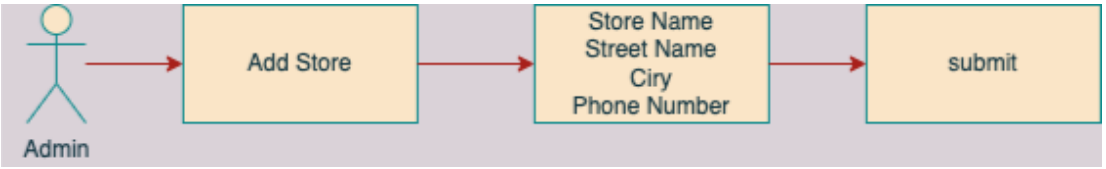
1. **All Utilities**



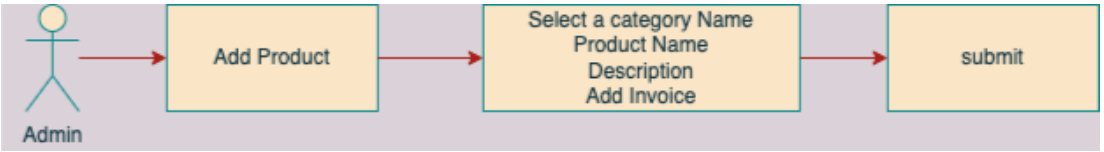
2. **Add Brand**



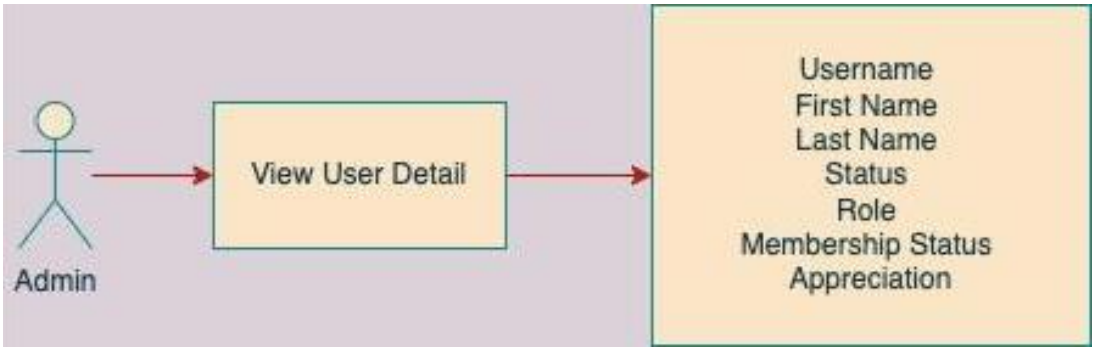
3. **Add Store**



4. **Add Products**



5. **View/Manage Users**



6. **View Feedback**

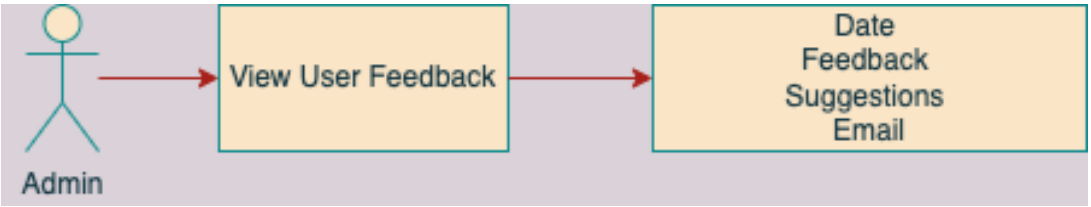
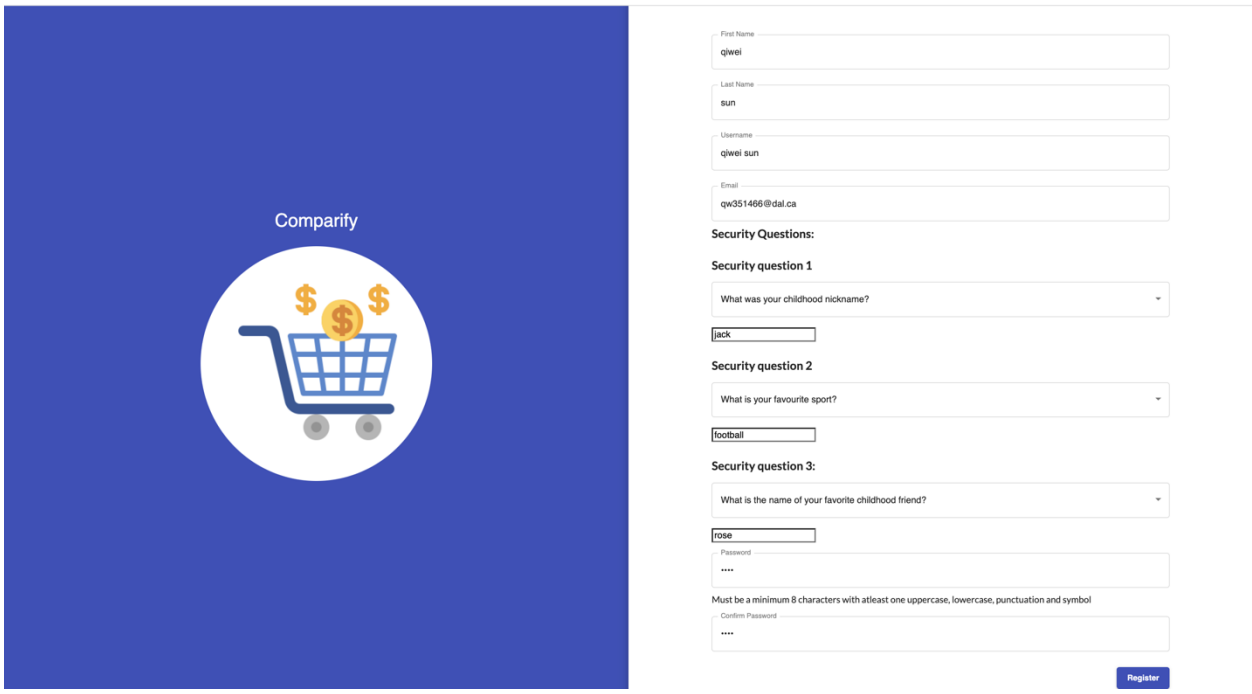


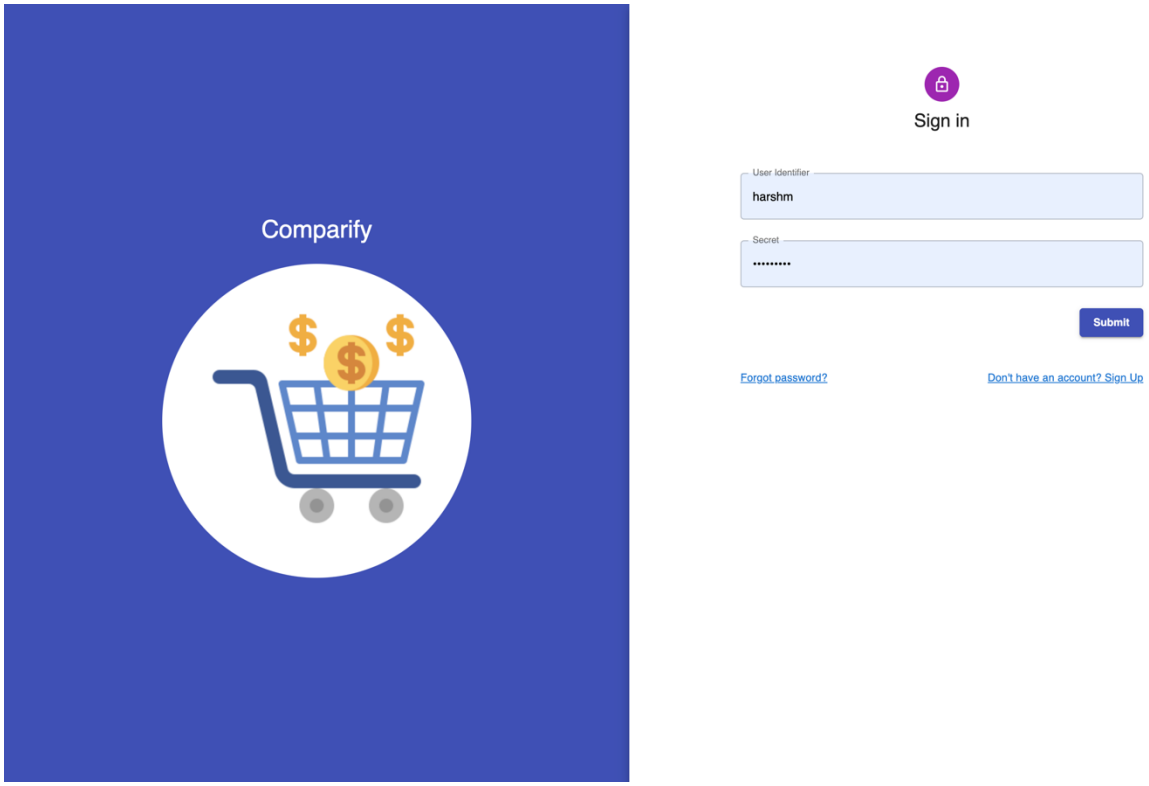
Table one: summary of user scenarios and corresponding feature

User scenarios	Corresponding features
Buy products at relevant low price	[1] Register as a user and [2]login in our product [3] search a product [4] compare similar product with low price.
Get alert when the price of product decrease	[6] getting rewards by submitting the invoice.[7] when the user has enough point, the user will promote from sliver to gold membership. [8][9] As a golden member, he/she can receive an alert from website/ email when the price drop to the threshold.
Users forget/reset the login password	[10] Reset the password by answer correct security question
Want to analysis the trend of the price verses brand and time.	[11] visualized the data reports
Admin wants to have a power to add, delete, verify the information.	[12] Admin can access admin page to verify the user submission and manipulate these data.
Admin want to get some information about the user.	[13] A user profile was built for retrieving
N/A	[14] run the application on three different environment dev, staging and prod
Keep user's account/privacy secure	[15] Failed too many times when login will suspend the account [16] force the user to change the password every month.
N/A	[16] Store and manage the data and data model on mongo dB
Admin want to collet user feedbacks for improving user experience	[17] enable user to submit feedback about their use experience

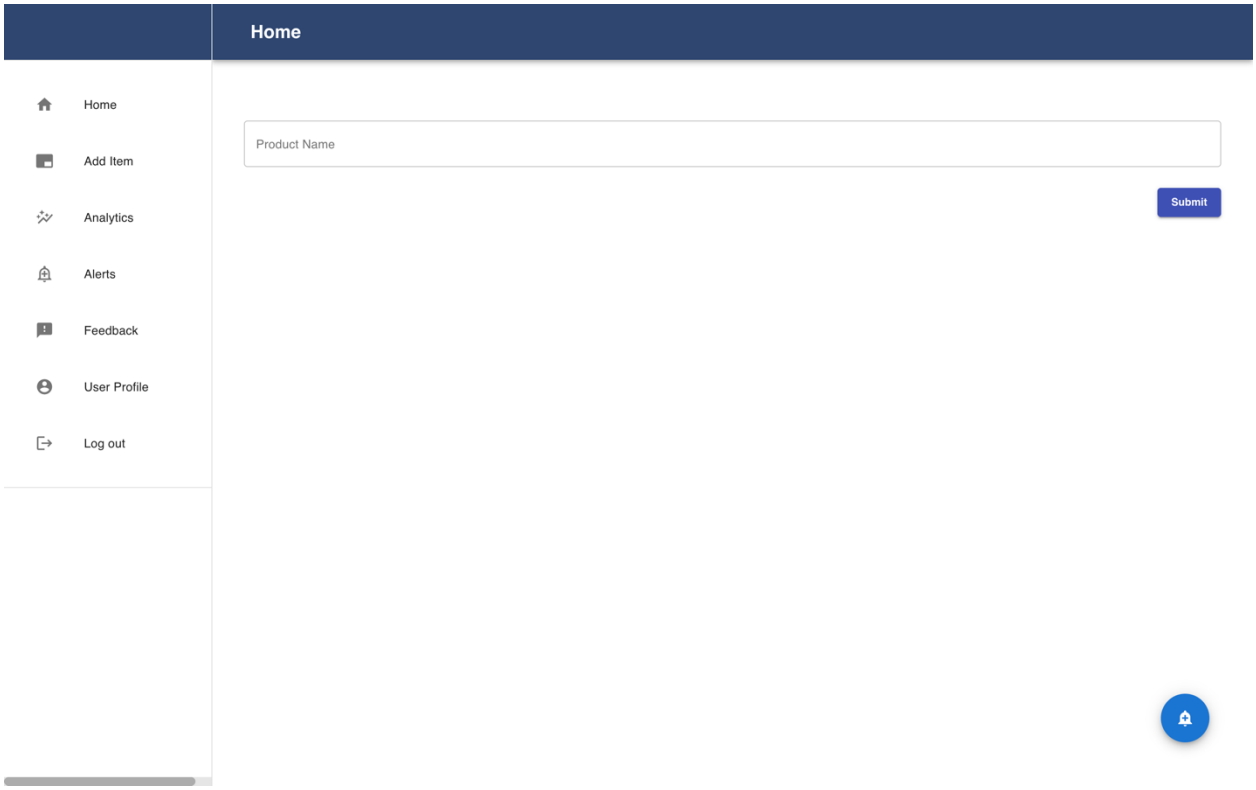
User: registration



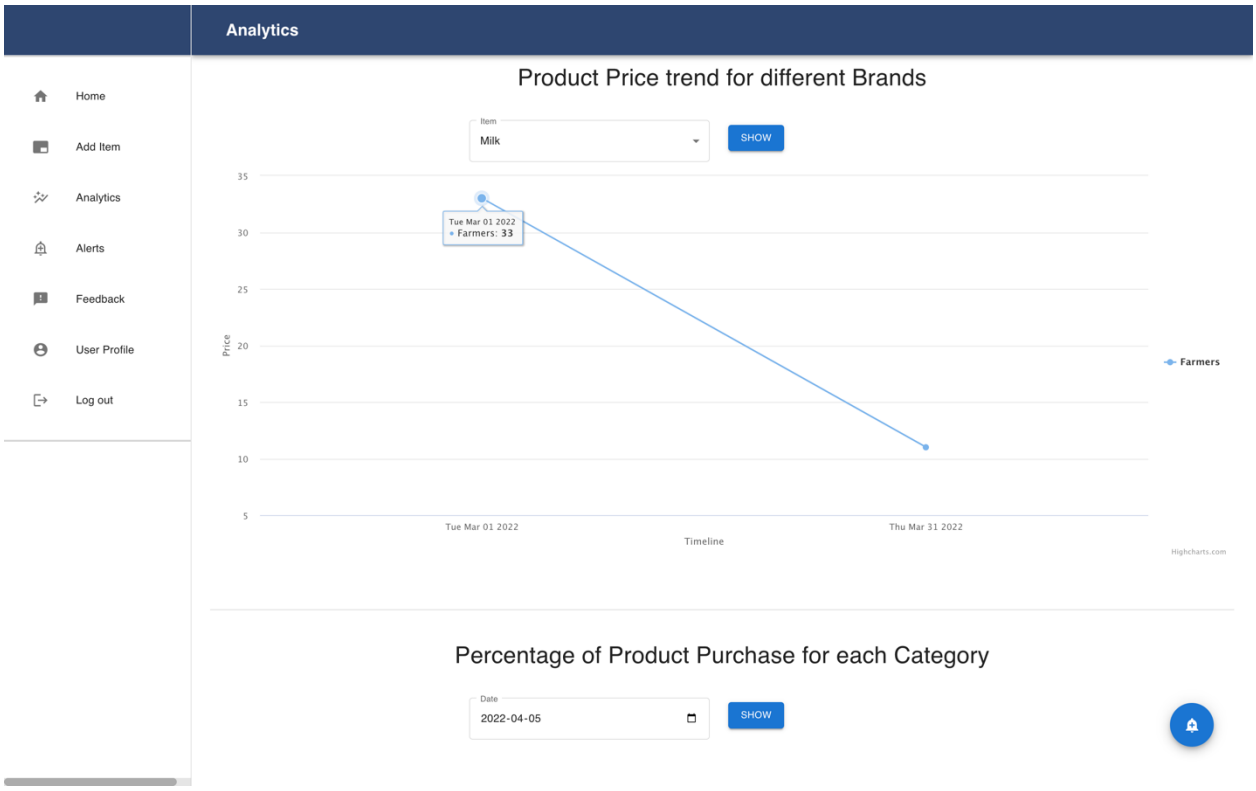
User: Login



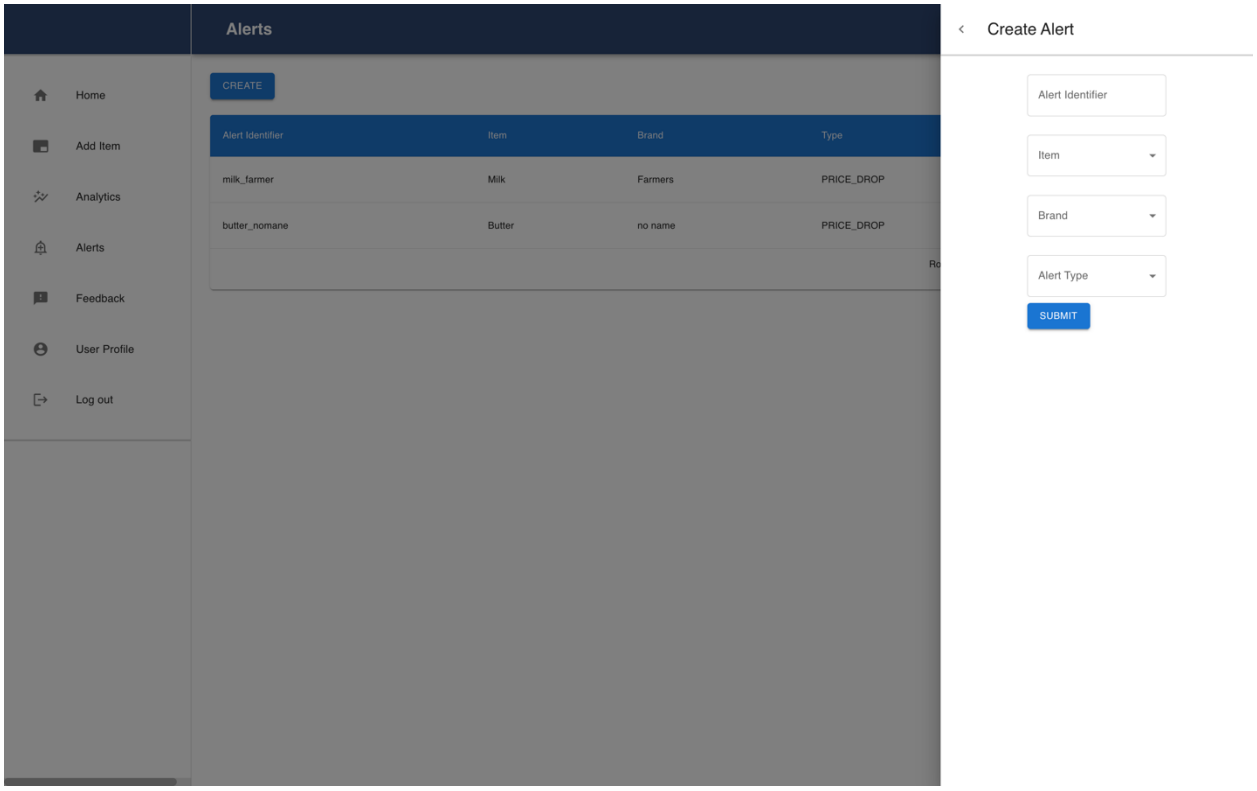
User: Home page



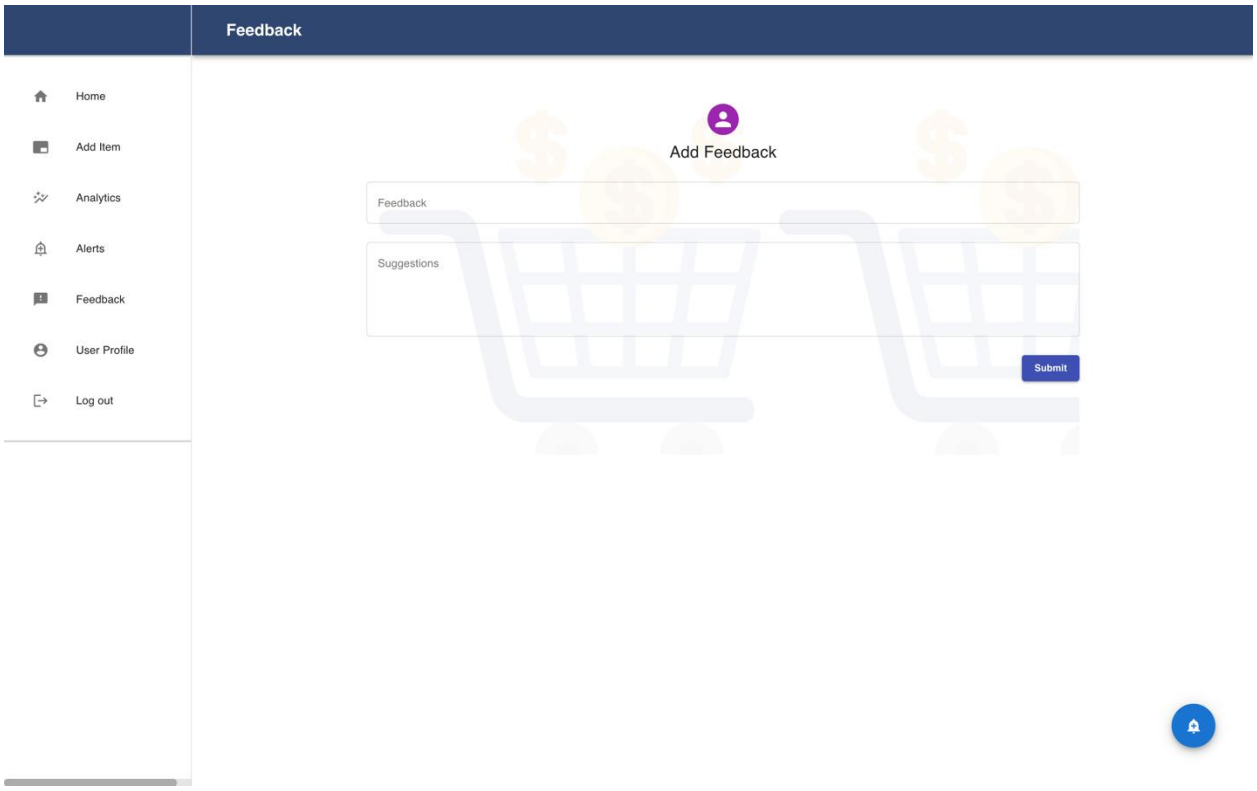
User: View the Price trend over the time



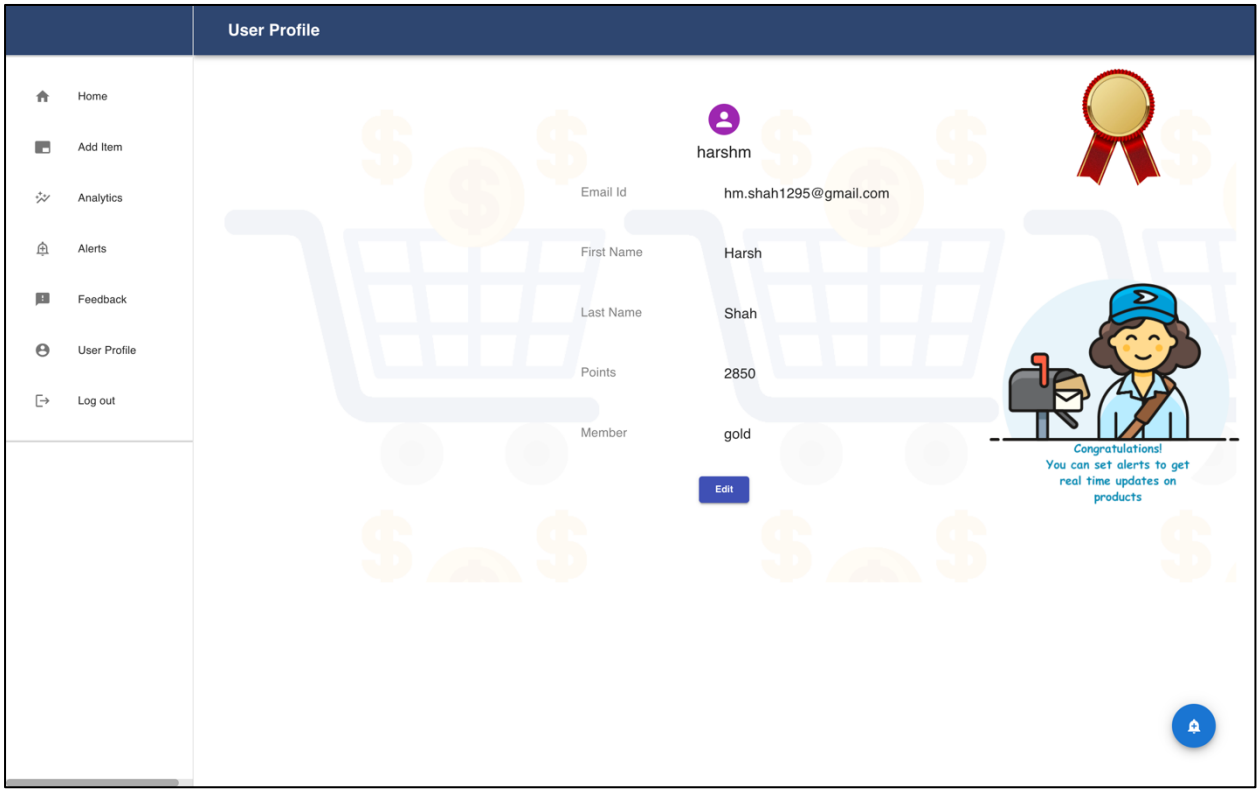
User: Set an alert when the price drop



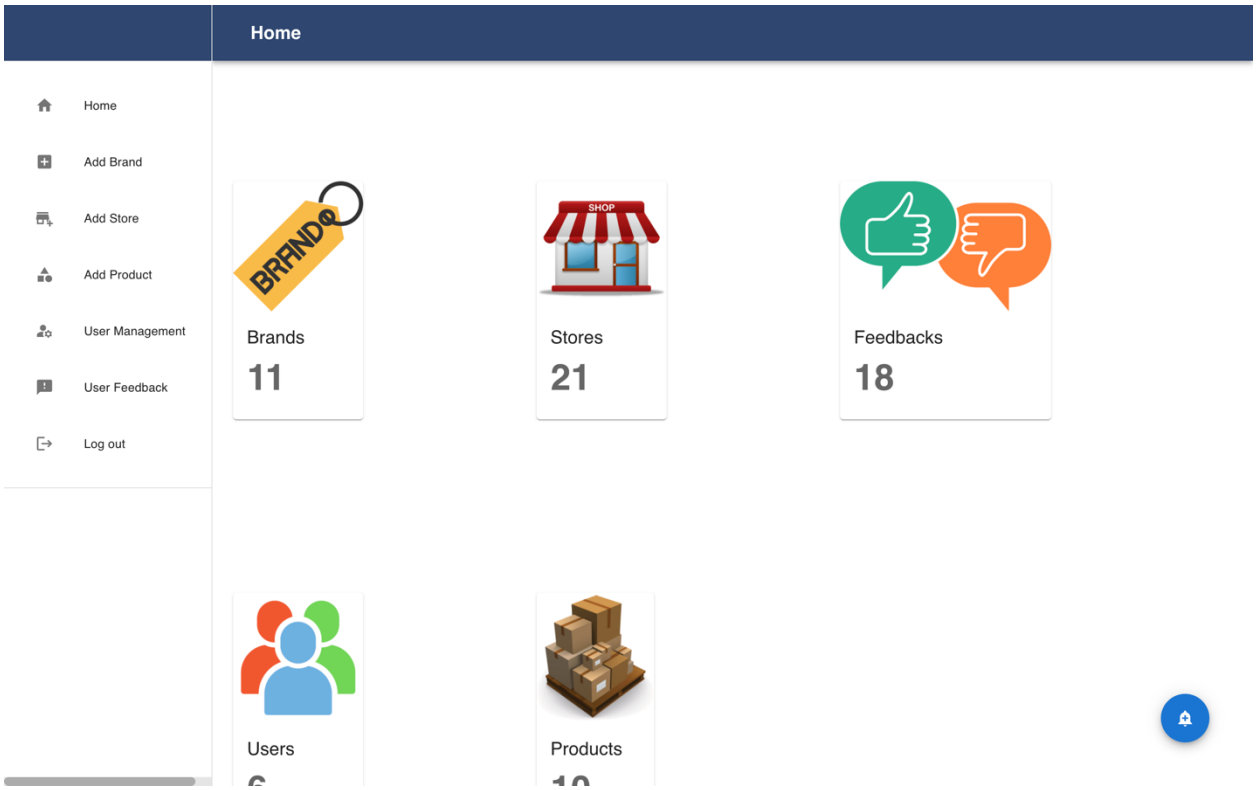
User: Create and submit feedback



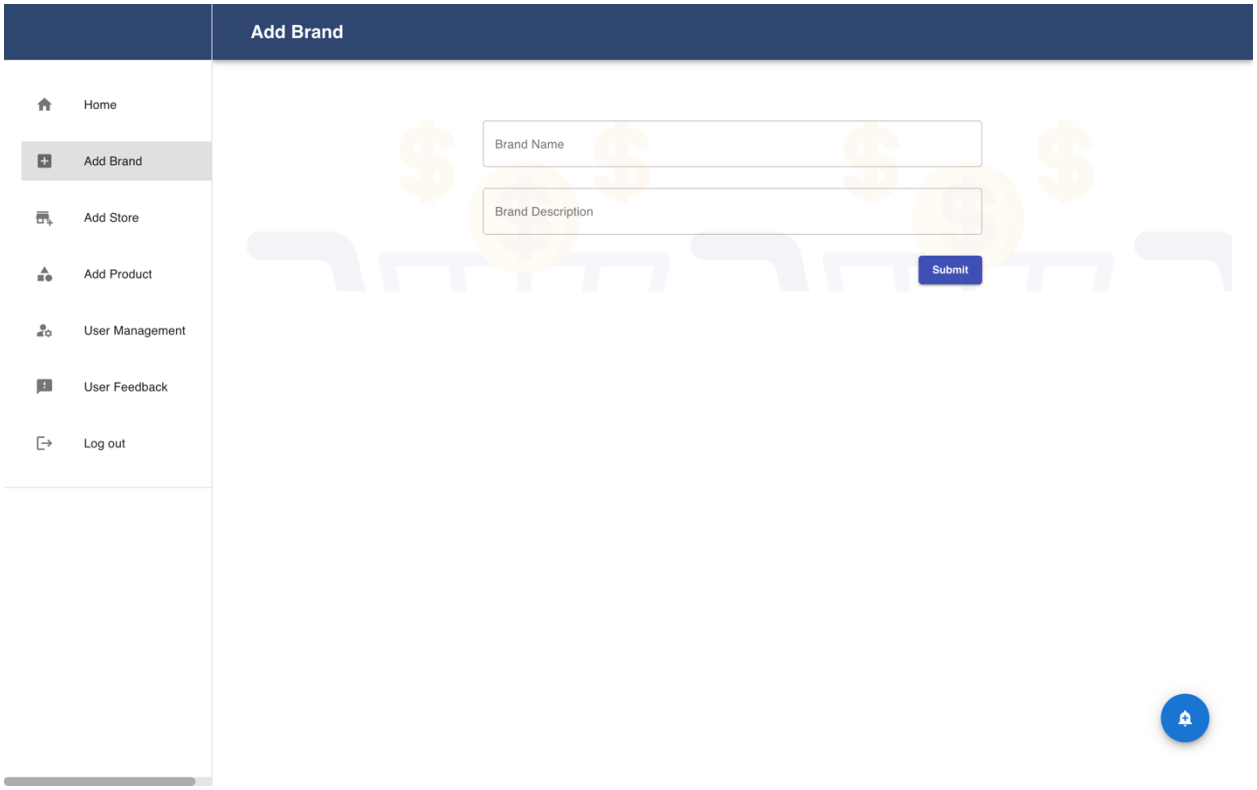
User: View my profile



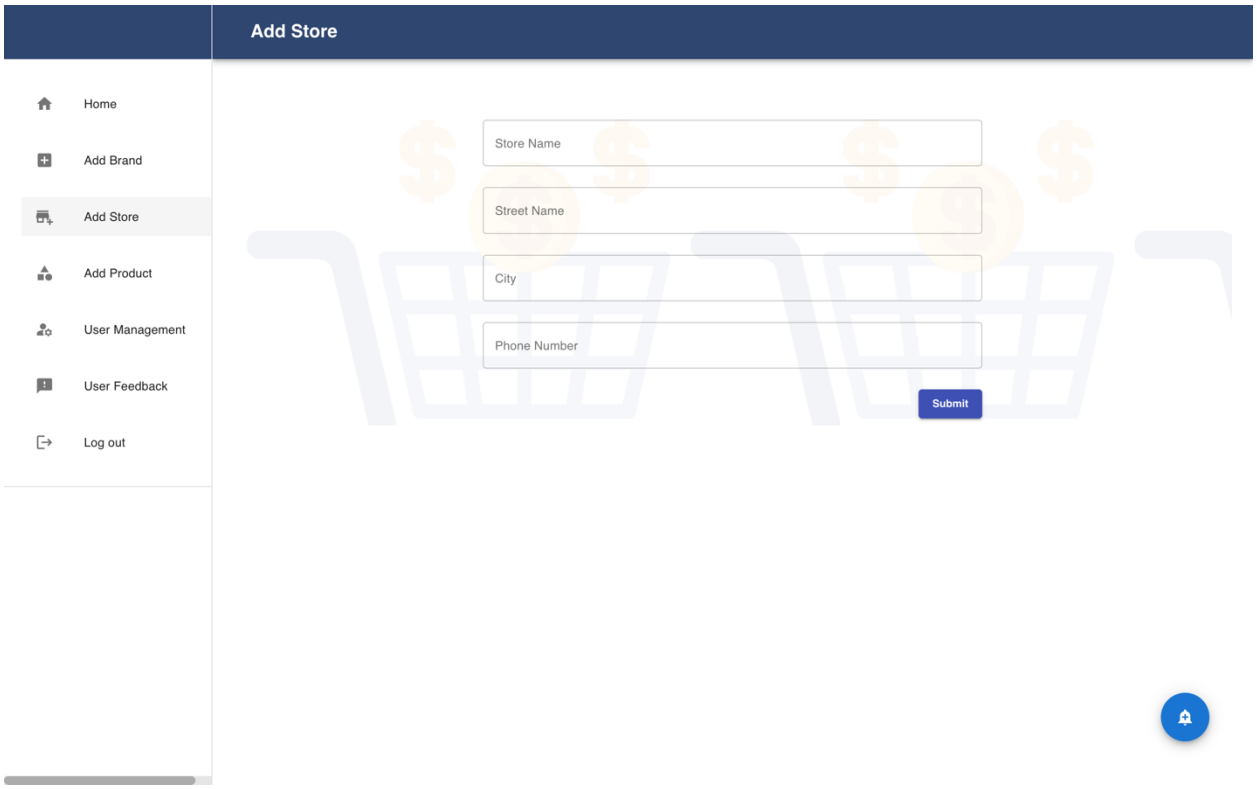
Admin: Home page



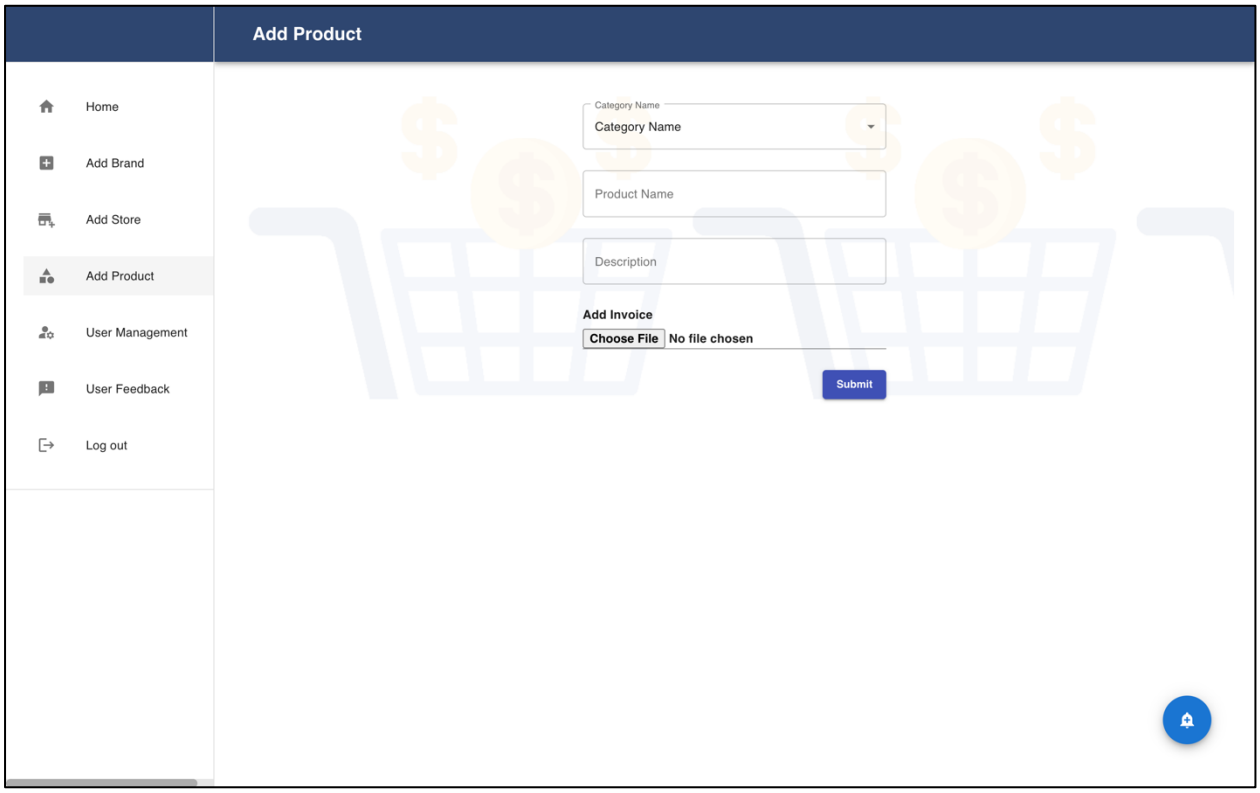
Admin: Add brand record



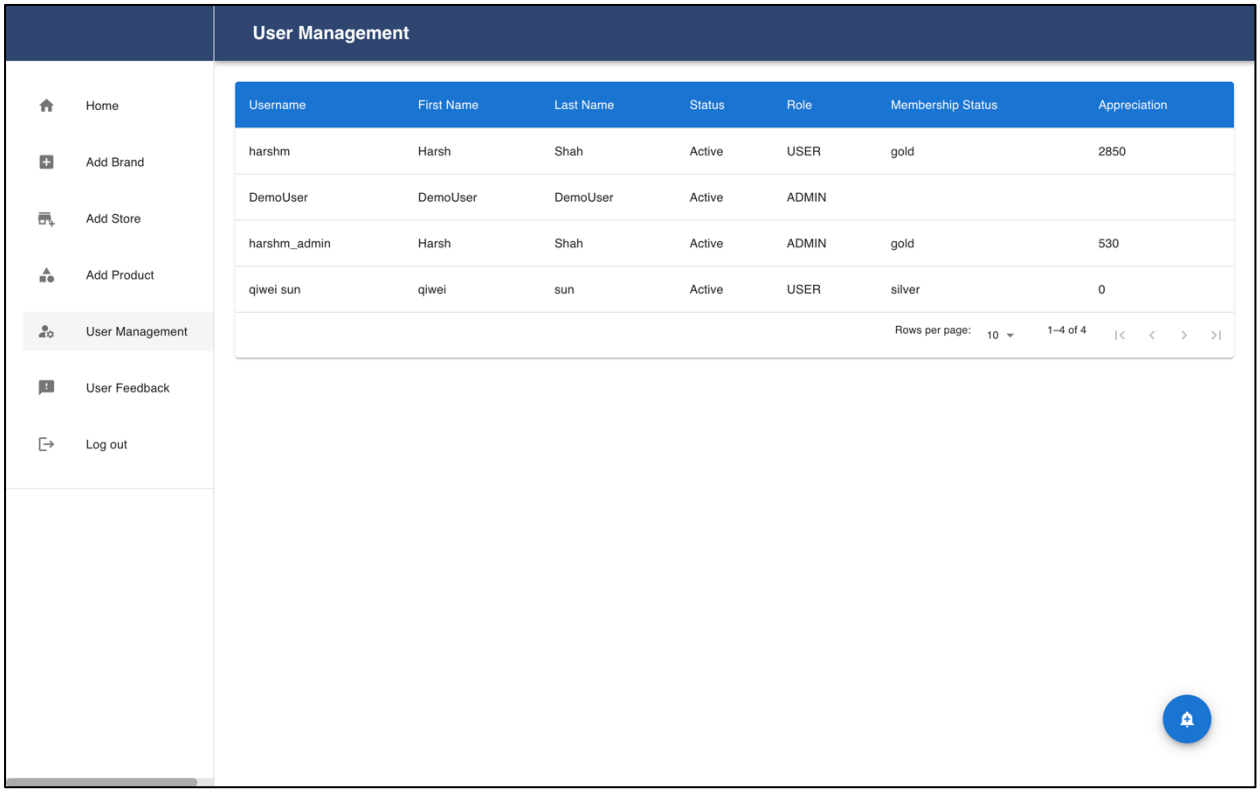
Admin: Add store



Admin: Add product

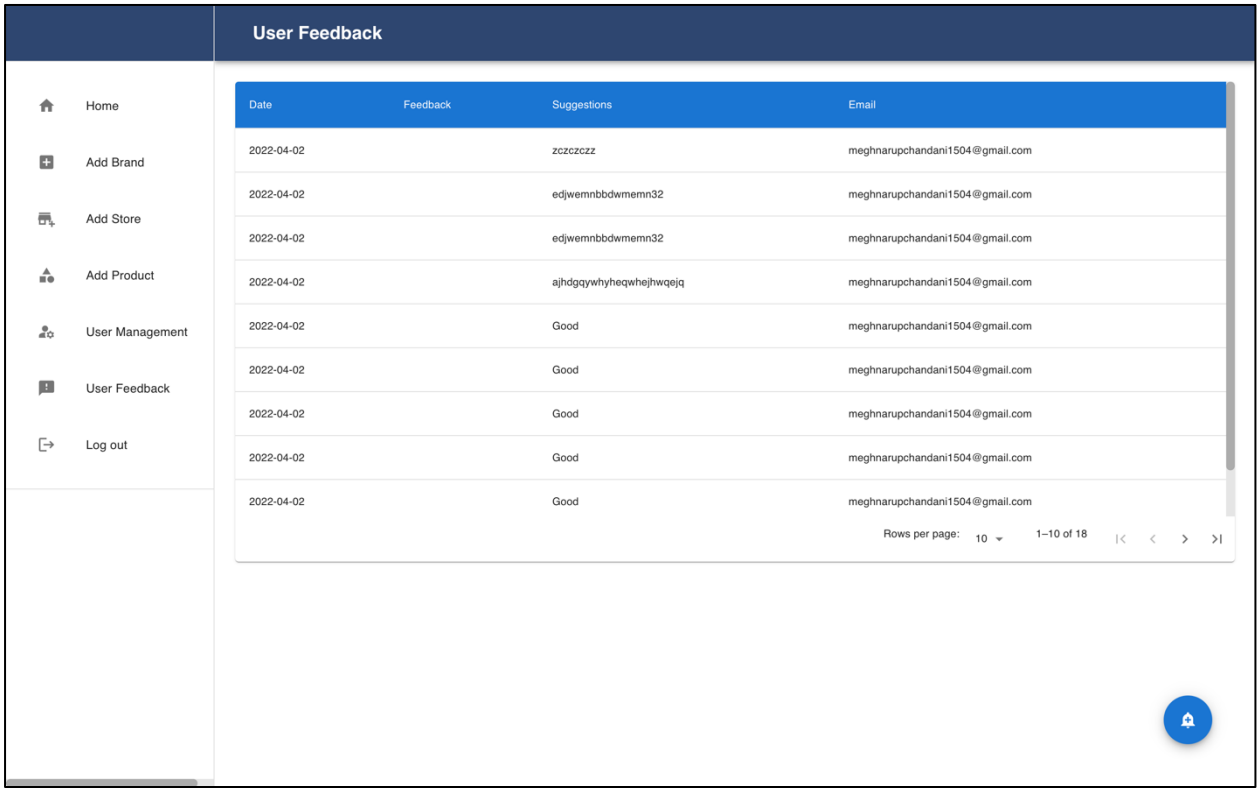


Admin: Manage user



Username	First Name	Last Name	Status	Role	Membership Status	Appreciation
harshm	Harsh	Shah	Active	USER	gold	2850
DemoUser	DemoUser	DemoUser	Active	ADMIN		
harshm_admin	Harsh	Shah	Active	ADMIN	gold	530
qiwei sun	qiwei	sun	Active	USER	silver	0

Admin: View user information



Smell Analysis Summary

See details in attached excel sheet

Member Contribution File

Sr. No	Feature	Weights	Author
1	Products Price Visualisation	4	Harsh Shah
2	Set Alerts	3	
3	Notification#2	4	
4	Notification#1	4	
5	Application Profiling	3	
6	Admin screen: create new role	2	
7	Secure user account by forcing user to change the password after 30 days	2	
8	Secure user account by locking if too many wrong passwords attempted	2	
9	CI/CD Pipeline Setup	5	
10	Implement Sign in Feature	5	
11	User profile feature	10	Aman Singh Bhandari
12	Membership Matrix	10	
13	Appreciation Management	10	
14	Admin: Stats	4	
15	Dataset Creation Feature	10	Chanpreet Singh
16	Admin Screen: Brand	4	
17	Database Model	6	
18	Add Product Items	5	Chanpreet Singh
		8	Meghna Rupchandani
19	Compare Prices	5	Chanpreet Singh,
		8	Meghna Rupchandani
20	Users Feedback	4	Chanpreet Singh
		7	Meghna Rupchandani
21	Implement Sign Up feature	7	Meghna Rupchandani
22	Admin Screen: Brand	4	Meghna Rupchandani
23	Forgot Password feature	20	Qiwei Sun
24	Search Product Item	10	
25	User Management	4	
26	Project Documentation	1	Aman Singh Bhandari
		1	Qiwei Sun
		1	Meghna Rupchandani
		1	Chanpreet Singh
		1	Harsh Shah