

International Institute of Information Technology (IIIT Bangalore)

Project Report iGlobe

In the Guidance of Prof. B. Thangaraju

Teaching Assistant: Neha Kothari



Nikunj Khakhkar :MT2021084

Pragya Khare :MT2021094

TABLE OF CONTENTS

1.Abstract	3
1.1 Features	
2. Introduction	4
2.1 Overview	
3. System Configuration	4
4. Software development life cycle	
a) Frontend	
b) Backend	
5. Experimental Setup	5
a. Functional Requirements	
b. Non Functional Requirements	
c. Exploring Controllers	
d. Future Pipeline Optimization	
6. Result and Discussion	14
a. User	
b. Buy Stuff	
7. Challenges Faced	16
a. Cors	
b. Storing picture in container	
8. Monitoring	17
9.Future Work	17
a.Progressive Web Application:	
b.User Authentication:	
c. Adding option of google maps	
10.API Calls List	18
11. Conclusion	19
12.References	19

1. Abstract:

When the pockets are empty and you look at your parents for all your needs the words 'DONATION' and 'second hand things' make real sense.

iGOBLE provides A platform to donate and sell your old stuff to the welfare of college students.

Being a student of IIITB and living in hostel got me a chance to come across multiple needs that require a permanent and user-friendly solution.

To understand cause of problem one need to be on others shoe and igoble provides a medium where the alumni of college can help the current students from donating to finding vacate rooms in different cities when students are living campus for their internships and work.

Iglobe provides a lost and found feature which reduces the use of telegram where we put all the lost things we find.

There had been different standalone projects for these same problems but iglobe combines all this in one go web application with an attractive and easy to use GUI and it has an immense scope in upcoming years.

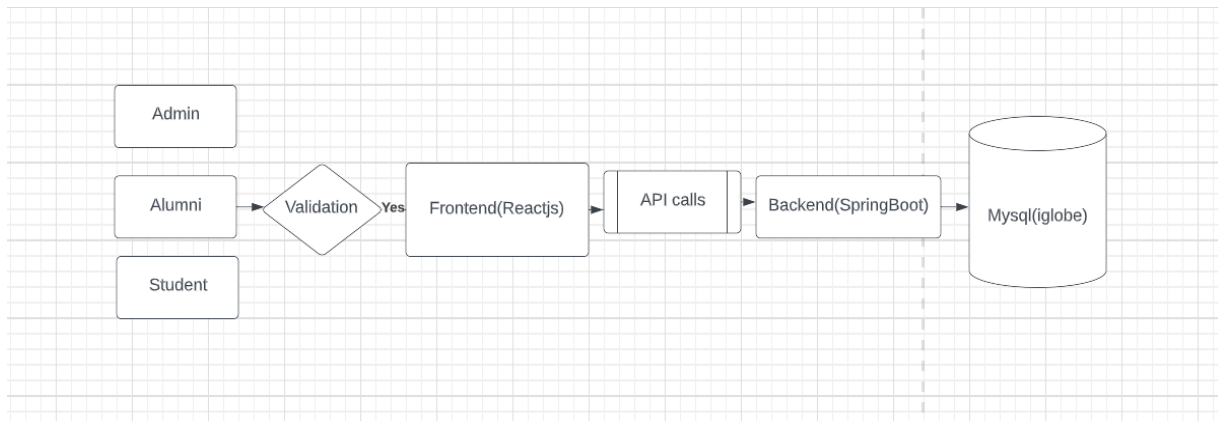
1.1 Features:

- Buy and Sell: Students can sell their old stuff and help other students.
- Donate: Alumni can donate stuff.
- Lost and Found: To find lost things in side campus.
- Find Rooms: To find vacate rooms or PG .

User:

- Student
- Admin
- Alumni

2.Introduction:



- Iglobe **Front-end** is developed and designed on React-js and Vs code is used as IDE.
- Iglobe **Back-end** is developed using java and spring boot frame work ,on Intelj IDE .
- The data base used for this application is my-sql.
- Iglobe backend and frontend is dockerized and deployed on local host.

3. SYSTEM CONFIGURATION:

3.1 Project Technology Stack:

- Frontend: React (HTML, CSS, Javascript)
- Backend: Springboot (Java)
- Database: Mysql

3.2 DevOps Tools

- Source Control Management: GitHub
- Build tools: Maven
- Containerization: Docker
- Continuous Integration (CI): Jenkins
- Continuous Deployment (CD): Ansible
- Monitoring: ELK stack

4. SOFTWARE DEVELOPMENT LIFE CYCLE (iGlobe frontend):

4.1.1 Installations:

React js : React makes it painless to create interactive UIs. Design simple views for each state in your application, and react will efficiently update and render just the right components when your data changes. React is a free and open-source front-end JavaScript library for building user interfaces based on UI components.

Update local before installing:

Keep the local packages and software updated

```
sudo apt-get update
```

Install NodeJS and NPM: In order to run React, node environment shall be installed before starting React

```
sudo apt-get install nodejs
```

```
sudo apt-get install npm
```



The screenshot shows a terminal window with four tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is active. The prompt is 'pragya@pragya-hp:~/iiitb_handshake\$'. The first command is 'node -v', which outputs 'v16.13.2'. The second command is 'npm -v', which outputs '8.1.2'. The third command is an empty prompt, showing a cursor.

```
pragya@pragya-hp:~/iiitb_handshake$ node -v
v16.13.2
pragya@pragya-hp:~/iiitb_handshake$ npm -v
8.1.2
pragya@pragya-hp:~/iiitb_handshake$
```

Creating new project

```
npx create-react-app iiitb-handshake
```

```
cd iiitb-handshake
```

```
npm start
```

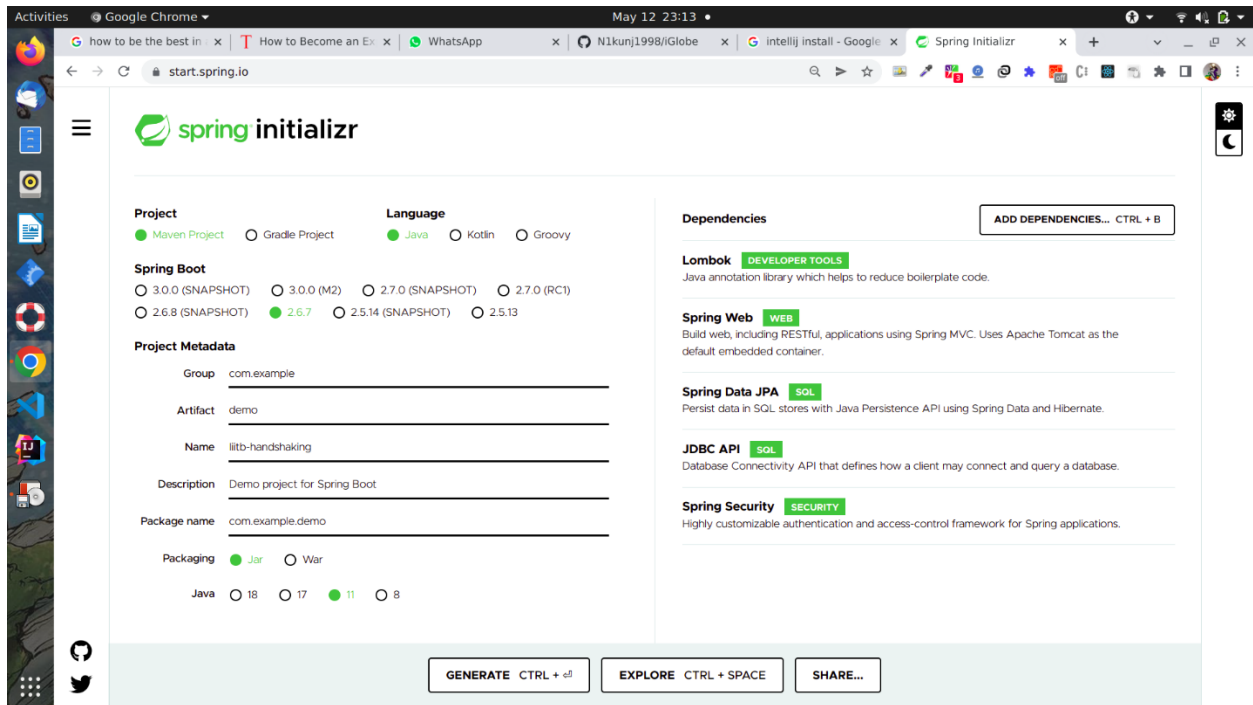
Output after npm start:

Index.js Run

Spring Boot:

Spring Boot is a Spring module that provides the RAD (Rapid Application Development) feature to the Spring framework.

We used spring initializer to create our project.



4.1.2 Source Control Management

Source control Management is used for tracking the file changes history, source code etc. It helps us in many ways in keeping the running project in a structured and organized way. In this project, we'll be using Github as SCM Tool which is a cloud based version control system. Here we'll be focusing on the front-end part and the SCM setup, steps, workflow for iglobe-backend is in the backend section under SCM.

Repository Link:

https://github.com/pragyak26/IITB_Handshaking

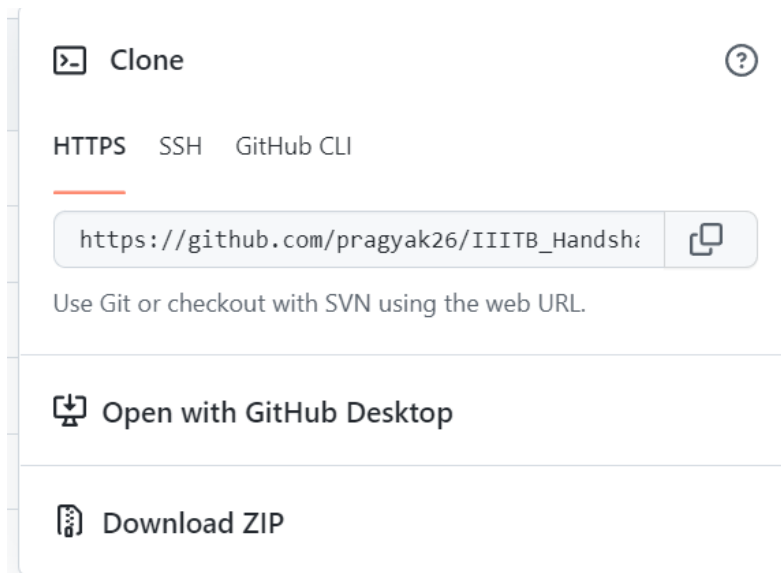
<https://github.com/N1kunj1998/iGlobe>

Initializing the project with git:

```
git init
```

```
git remote add origin https://github.com/pragyak26/IIITB\_Handshaking.git
```

Cloning the Project:



```
Git clone https://github.com/pragyak26/IIITB\_Handshaking.git
```

Workflow:

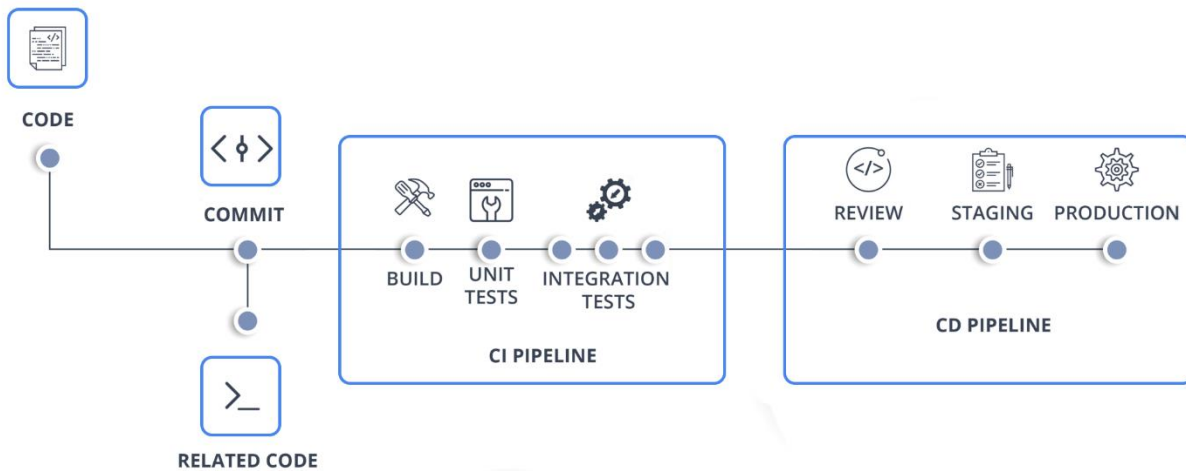
```
git add
```

```
git commit -m "commit message"
```

```
git pull origin master git push origin master
```

The code is first pulled before making a push to make sure that our project is in the latest stage and to avoid merge conflicts. For the above, the pull and push is done on the “master” branch.

4.1.3 Continuous Integration with Jenkins:



Continuous Integration is a development practice where workflows like build, test and deploy are automated. Continuous Integration assures low integration risks, low down time, quality tests and automations, easy understanding between the teams and less time for deployment. Few Popular tools for CI are Jenkins, Travis CI, GitLab CI, Github actions and many more.

What is Jenkins:

Jenkins is an open-source server-based application written in Java used to achieve continuous integration in an automated fashion. It is popular because it makes monitoring the repeated task which arises during the development phase. While working on the project, Jenkins automation will continuously test the builds and recognize the errors in the early stages of the development. Jenkins speeds up the software development process by automating the build and test quickly.

Getting Started with Jenkins:

1. Create pipeline:

Enter an item name

» Required field

Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project

Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder separate namespace, so you can have multiple things of the same name as long as they are in different folders.

2. Writing pipeline script:

```

25 }
26 stage('Push Front End Docker Image') {
27     steps {
28         script{
29             docker.withRegistry('', 'docker-jenkins'){
30                 dockerIng_frontend.push()
31             }
32         }
33     }
34 }
35 stage('Maven Build') {
36     steps {
37         dir('Iiitb-Handshake-Backend'){
38             sh 'mvn clean install'
39         }
40     }
41 }
42 stage('Build backend image') {
43     steps {
44         dir('Iiitb-Handshake-Backend'){
45             script {
46                 dockerIng_app-docker.build(inagename_app)
47             }
48         }
49     }
50 }
51 stage('Push backend image') {
52     steps {
53         script{
54             docker.withRegistry('', 'docker-jenkins'){
55                 dockerIng_app.push()
56             }
57         }
58     }
59 }
60 stage('Pull Image on Deployment server') {
61     steps {
62         ansiblePlaybook colored: true, disableHostKeyChecking: true, installation: 'Ansible', inventory: 'inventory', playbook: 'playb
63     }
64 }
65 }
66 }

```

3.Stage View:

Pipeline iglobe-pipeline
just a simple pipeline

[Edit description](#)
[Disable Project](#)

[Recent Changes](#)

Stage View

Average stage times:
(Average full run time: ~1min 37s)

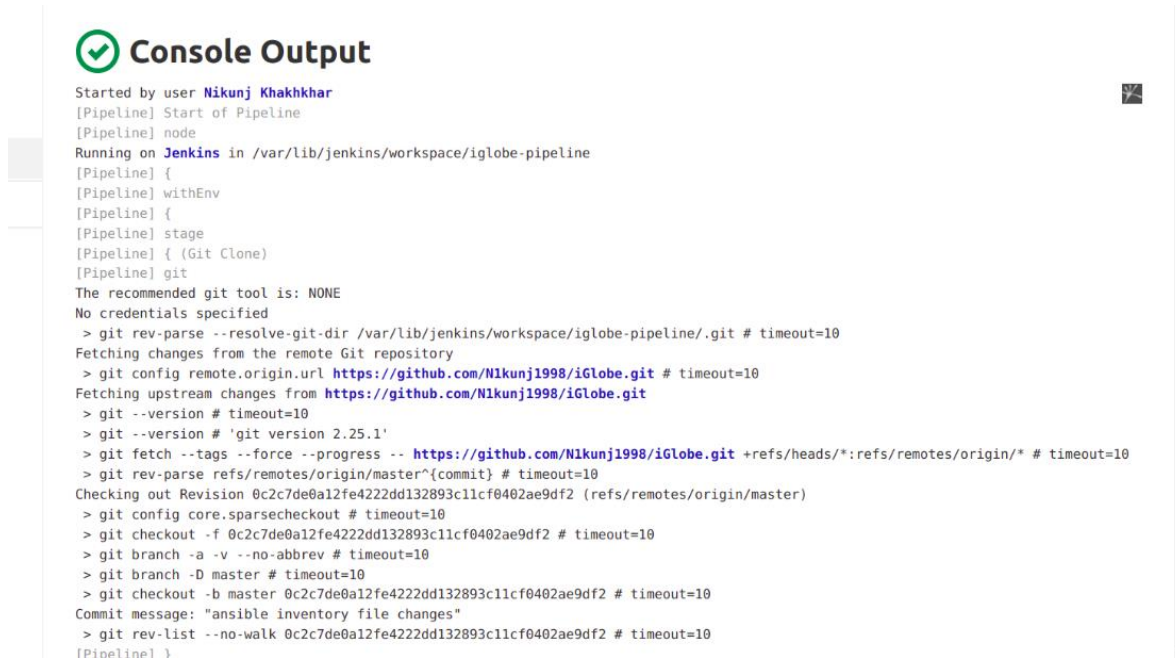
	Git Clone	Build frontend image	Push Front End Docker Image	Maven Build	Build backend image	Push backend image	Pull Image on Deployment server
#50 May 12 14:22 No Changes	1s	2s	12s	25s	2s	39s	10s
#49 May 12 11:13 1 commit	1s	719ms	13s	29s	3s	41s	11s
#48 May 12 No	1s	728ms	12s	28s	3s	40s	1s

Build History

Filter builds...

- #50 12 May 2022, 14:22
- #49 12 May 2022, 11:13
- #48 12 May 2022, 11:08
- #47 12 May 2022, 11:03
- #46 12 May 2022, 11:00

4. Console output:



```

✓ Console Output
Started by user Nikunj Khakhkhar
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/iglobe-pipeline
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Git Clone)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/iglobe-pipeline/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/N1kunj1998/iGlobe.git # timeout=10
Fetching upstream changes from https://github.com/N1kunj1998/iGlobe.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'
> git fetch --tags --force --progress -- https://github.com/N1kunj1998/iGlobe.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 0c2c7de0a12fe4222dd132893c11cf0402ae9df2 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 0c2c7de0a12fe4222dd132893c11cf0402ae9df2 # timeout=10
> git branch -a -v --no-abbrev # timeout=10
> git branch -D master # timeout=10
> git checkout -b master 0c2c7de0a12fe4222dd132893c11cf0402ae9df2 # timeout=10
Commit message: "ansible inventory file changes"
> git rev-list --no-walk 0c2c7de0a12fe4222dd132893c11cf0402ae9df2 # timeout=10
[Pipeline] }

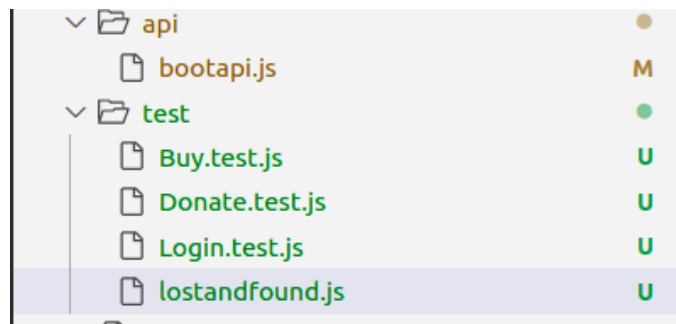
```

4.1.4 Building React Application:

Through the pipeline, testing of the front end, the React application, can be automatically triggered after the build process. The tests are run using Jest javascript test framework through npm test runner.

React has the following files for each component:

- 1.component.js - a component file
- 2.component.test.js - a testing specification file



3. component.html - a template file

4. component.css - a CSS file

4.2.4 Testing with Jest:

Snapshot Testing:

```

Login.test.js > describe("Login component test") callback > it("render div") callback
import {getByTestId, render} from "@testing-library/react";
import {act} from "react-dom/test-utils";
import Login from "../Login";

describe("Login component test", () => {
  it("render input", () => {
    const {getByTestId} = render(<Login/>);
    const login = getByTestId("login_button");
    expect(login).toBeTruthy();
  });

  it("render div", () => {
    const {getByTestId} = render(<Login/>);
    const div = getByTestId("form1");
    expect(div).toBeTruthy();
  });
});

```

Sample of a test case:

```
at Object.<anonymous> (src/test/Donate.test.js:3:1)
```

```

PASS src/Buy.test.js
PASS src/Signup.test.js (5.078 s)
PASS src/Login.test.js
PASS src/Donate.test.js

```

Test Suites: 3 failed, 4 passed, 7 total

Tests: 8 passed, 8 total

Snapshots: 0 total

Time: 7.178 s

Ran all test suites related to changed files

4.1.6 Docker Management:

Docker helps in making the software building process faster by the idea of images and containers. Containers are isolated within the environment. Docker has also the flexibility of

```
---
- name: "pull images and start docker compose"
  hosts: localhost
  tasks:
    - name: "Pull docker images"
      command: "docker-compose -f docker-compose.yml pull"
    - name: "Start docker compose"
      command: "docker-compose -f docker-compose.yml up -d"
```

building and pushing the Images to Docker Hub. Docker file is a simple text file that consists of a set of instructions to build Docker images.

```
mysqlldb:
  container_name: mysqlldb
  image: mysql
  volumes:
    - ./db:/var/lib/mysql
  ports:
    - 3307:3306
  environment:
    - MYSQL_DATABASE=iiitb-handshake
    - MYSQL_ROOT_PASSWORD=root
```

5. Experimental Setup:

5.1 Functional Requirements:

- User can signup and login an account.
- User can buy and sell stuff.
- User can donate.
- User can use lost and found.

5.2 Non-Functional Requirements:

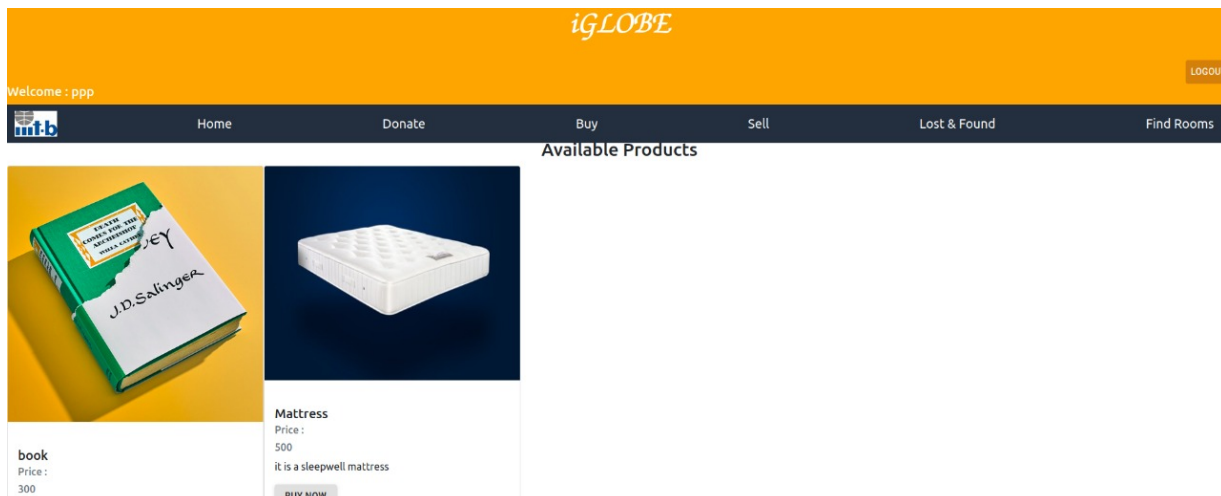
- Usability: The website is very user-friendly and any user should be able to familiarize themselves with the flow very easily.
- The system has scalability and will perform well even during high load.
- The WebApp is highly portable as it is based on React and NodeJS.

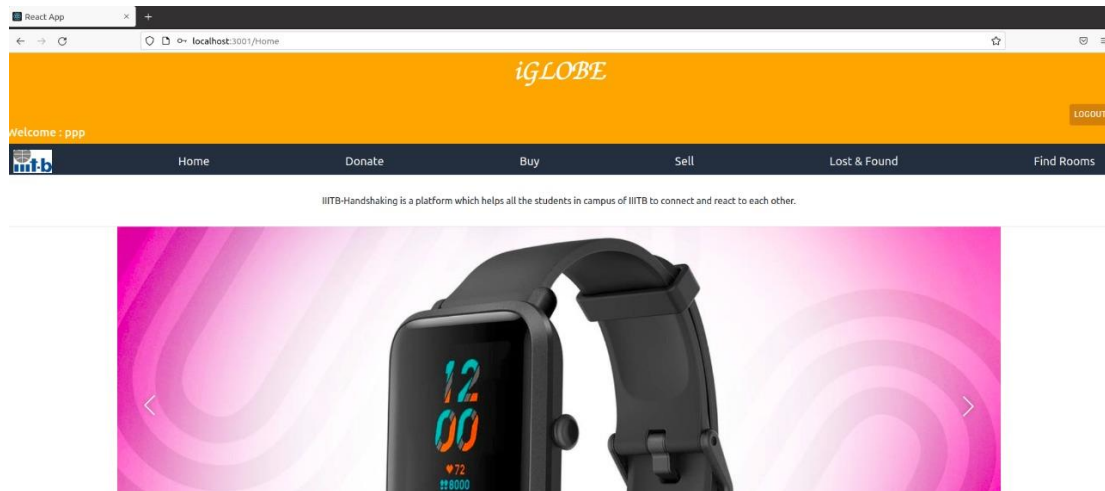
6. Results and Discussion:

6.1.1 Login Page:

Initially when we start the application, the login page is displayed. After successful login

User will able to see home page



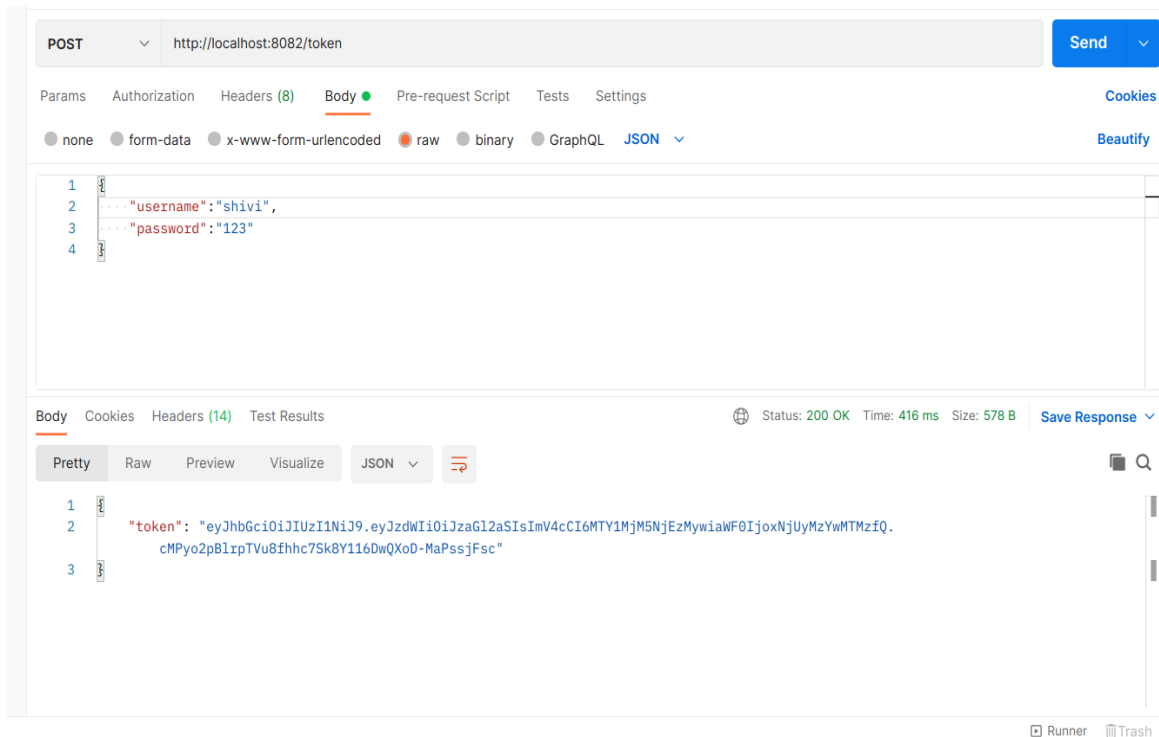


The screenshot shows the 'Sell' page of the iGLOBE website. The browser's address bar displays 'localhost:3001/sell'. The website has the same orange header and dark blue navigation bar as the home page. The 'LOGOUT' button is on the right, and the 'Sell' link is highlighted in the navigation bar. Below the navigation bar, a text line states: 'Fill the details of item you want to Sell.' The form contains the following fields: 'Product Name' with the value 'product', 'Details' with the value 'Condition of product', 'Price', and 'File' with a 'Browse...' button and the text 'No file selected.' A blue 'SUBMIT' button is at the bottom left of the form.

7.Challenges Faced:

1.CORS: This error was faced many a times:

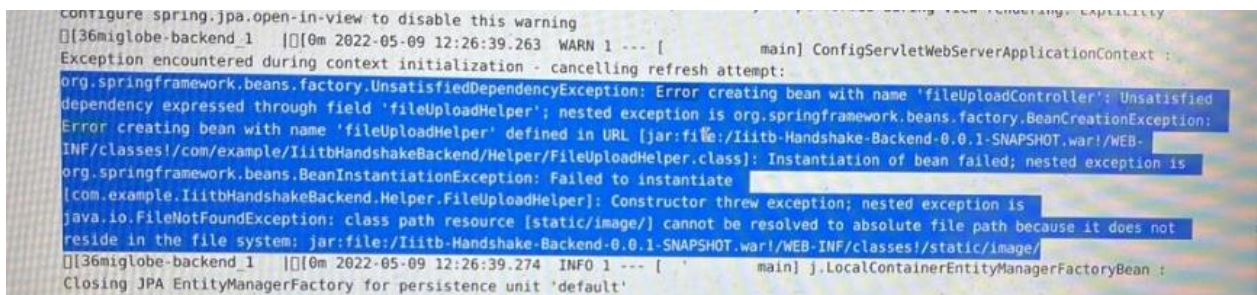
- While using ngrok to connect front-end and back-end code.
- While implementing **JWT** : Code was working fine when we tested it through Postman but later when we integrated the frontend code it was showing CORS error.



The **solution** that we tried :

1. Downloaded CORS plugin.
2. we used this annotation **@CrossOrigin("*")** for all controller classes.

2.Storing pictures inside docker container:



The **solution** that worked:

We removed **getFile** Method , which was being used while getting the class path resource,

In place of that we used "Paths" class to get the location of file storage.

8. Monitoring:

It was done by log file and ELK.

```

2022-05-12 21:07:43.777 INFO 10913 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-05-12 21:07:43.778 INFO 10913 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.60]
2022-05-12 21:07:43.830 INFO 10913 --- [main] o.s.c.e.c.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-05-12 21:07:43.830 INFO 10913 --- [main] o.s.s.s.ServletContextWebApplicationContext : Root WebApplicationContext: initialization completed in 1117 ms
2022-05-12 21:07:43.948 INFO 10913 --- [main] o.hibernate.jpa.internal.util.LogHelper : HH0000204: Processing PersistenceUnitInfo [name: default]
2022-05-12 21:07:44.970 INFO 10913 --- [main] org.hibernate.Version : HH0000412: Hibernate ORM core version 5.6.7.Final
2022-05-12 21:07:44.057 INFO 10913 --- [main] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
2022-05-12 21:07:44.125 INFO 10913 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2022-05-12 21:07:44.398 INFO 10913 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2022-05-12 21:07:44.408 INFO 10913 --- [main] org.hibernate.dialect.Dialect : HH0000400: Using dialect: org.hibernate.dialect.MySQL57Dialect
2022-05-12 21:07:44.773 INFO 10913 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HH0000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.j
2022-05-12 21:07:44.778 INFO 10913 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2022-05-12 21:07:44.812 WARN 10913 --- [main] JpaBaseConfigurationJpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may b
2022-05-12 21:07:45.074 INFO 10913 --- [main] c.e.i.controller.UserController : Logs are getting generated
2022-05-12 21:07:45.154 INFO 10913 --- [main] o.s.s.web.DefaultSecurityFilterChain : Will not secure any request
2022-05-12 21:07:45.390 INFO 10913 --- [main] o.s.s.w.embedded.TomcatWebServer : Tomcat started on port(s): 8082 (http) with context path ''
2022-05-12 21:07:45.400 INFO 10913 --- [main] c.e.i.IltbHandshakeBackendApplication : Started IltbHandshakeBackendApplication in 3.29 seconds (JVM running for 4.884)
2022-05-12 21:07:52.045 INFO 10913 --- [http-nio-8082-exec-2] o.s.c.e.c.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-05-12 21:07:52.046 INFO 10913 --- [http-nio-8082-exec-2] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-05-12 21:07:52.047 INFO 10913 --- [http-nio-8082-exec-2] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
2022-05-12 21:07:52.085 INFO 10913 --- [http-nio-8082-exec-3] c.e.i.controller.SellController : Get Details : 166
2022-05-12 21:07:56.030 INFO 10913 --- [http-nio-8082-exec-3] c.e.i.controller.SellController : Get Details : 167
2022-05-12 21:07:59.632 INFO 10913 --- [http-nio-8082-exec-4] c.e.i.controller.SellController : Get Details : 168
2022-05-12 21:08:03.533 INFO 10913 --- [http-nio-8082-exec-5] c.e.i.controller.SellController : Get Details : 169
2022-05-12 21:08:07.260 INFO 10913 --- [http-nio-8082-exec-6] c.e.i.controller.SellController : Get Details : 170
2022-05-12 21:08:11.025 INFO 10913 --- [http-nio-8082-exec-7] c.e.i.controller.SellController : Get Details : 171
2022-05-12 21:08:14.952 INFO 10913 --- [http-nio-8082-exec-8] c.e.i.controller.SellController : Get Details : 172
2022-05-12 21:08:18.812 INFO 10913 --- [http-nio-8082-exec-9] c.e.i.controller.SellController : Get Details : 173
2022-05-12 21:08:31.125 INFO 10913 --- [SpringApplicationShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit
2022-05-12 21:08:31.126 INFO 10913 --- [SpringApplicationShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...
2022-05-12 21:08:31.126 INFO 10913 --- [SpringApplicationShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.
  
```

9. Future work:

9.1 Progressive Web Application:

The application will be able to run on any desktop and it was a future scope in the form of mobile application.

9.2 User Authentication:

The user will be able to login to their own accounts which can be selected from drop down list.

Users can also choose the option according to their need like sell, buy, donate etc.

9.3 Adding option of google maps: This application could also has an option of finding rooms with the help of google maps.

10.API calls list:

Endpoint	Method	Description
/getDonateProducts	GET	Get the list of donated products
/getSellProducts	GET	Get the list of sold products
/buyDonateProduct/{donateId}	POST	Get a Donated product
/buySellProduct/{sId}/{uId}	POST	Buy a New Product
/donate/addItem	POST	Add a new item in Donate table
/donate/getDetails/{did}	GET	To get Details of a Donated Product
/addRoom	POST	Add the details of Room
/getRooms	GET	Get the Details of available rooms
/sell/addItem	POST	Add a new item in Sell table
/sell/getDetails/{sid}	GET	To get Details of a Product in sell table
/addUser	POST	For a new user to Signup
/validate	POST	For a user to Sign in
/reset	POST	For user to reset his password

11.Conclusion:

We build an application that can help college students to get things easily and in cheap rates.

Students can sell their stuff and alumni can help current students to find new rooms and even donate the old stuff

This entire application was automated using DevOps tools like Jenkins for continuous integration, Ansible for configuration management and deployment, Docker for containerization, Jest , junit and mockito for testing, localhost and container for deployment, and finally ELK for monitoring.

12. References:

<https://www.youtube.com/c/LearnCodeWithDurgesh>

<https://medium.com/adessoturkey/introduction-to-spring-boot-458cb814ec14>

<https://www.baeldung.com/>

<https://www.javatpoint.com/reactjs-tutorial>

<https://www.youtube.com/c/TechWorldwithNana>

<https://reactjs.org/>

<https://spring.io/guides/gs/spring-boot/>

