BRAC UNIVERSITY

Inspiring Excellence

CSE471 : System Analysis and Design

Project Report

Project Title : InventoryPro

Group Name: Codebell-2 CSE471 Lab Section : 04 Summer 2023	
ID	**Name**
20101478	Tasfia Ayesha
20101409	Tirthankar Saha Dibbya

Introduction

Introducing the simplest method for effectively tracking and controlling rice stocks—our cutting-edge Rice Inventory Management Web App. Our software enables rice farmers, distributors, and retailers to easily monitor inventory levels, foresee restocking needs, and manage supply chains thanks to easy user interfaces and real-time updates. Reject manual record-keeping and adopt a cutting-edge, digital strategy for managing rice inventory that will save you time, cut down on mistakes, and guarantee a continuous supply of this necessary product.

Functional requirements

1. Plot Tab

a. User can search Plot by product Name or ID

b. User can see Plot's general information

c. User can see Plot's location and what is planted and when to reap

d. User can create new plot

e. User can update existing plot information

d. User can remove Plots when needed

2 . Warehouses Tab

a. User can see the warehouses information which has goods

b. Can peek at warehouse contents and search for a single content

c. User can add/remove Warehouse (accessible by Owner only)

3. Login

a. It will have two types of user; regular and owner

b. User will have three buttons after login

c. User tab will be blocked for regular users

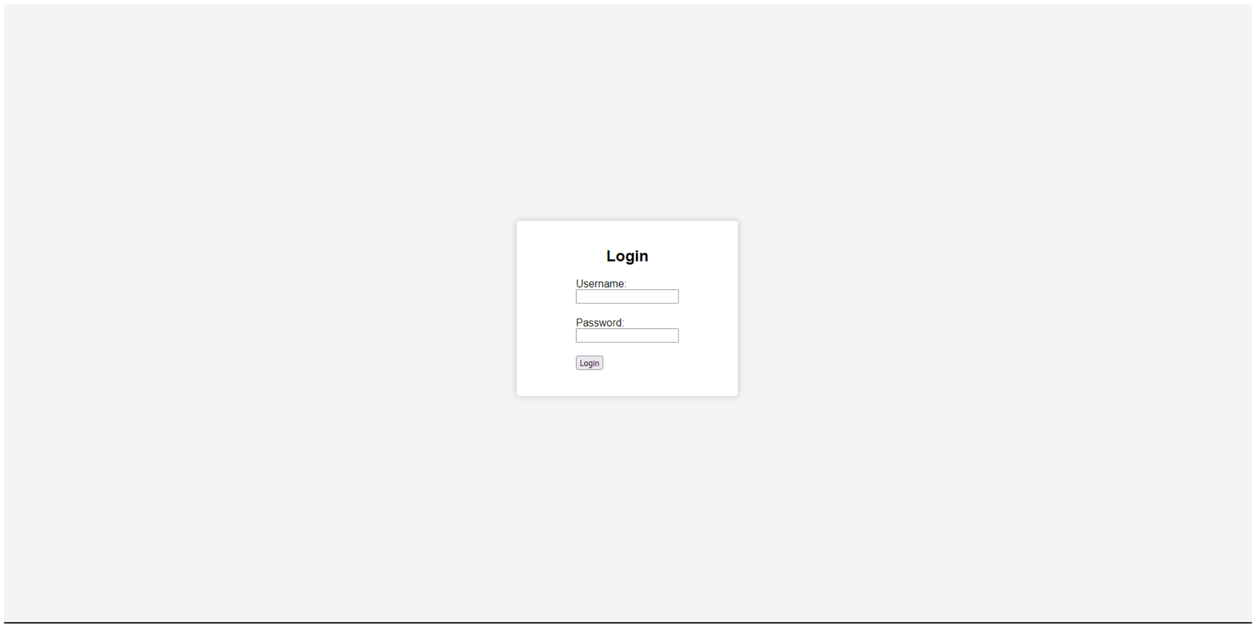
4. User Tab

a. The owner can see all details of existing users

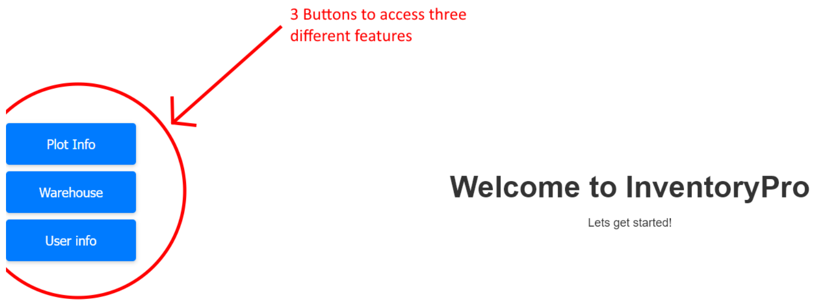
b.Owner can create and delete as many users as want

User Manual

At first, the user needs to login with the username and password that have been set by the owner.

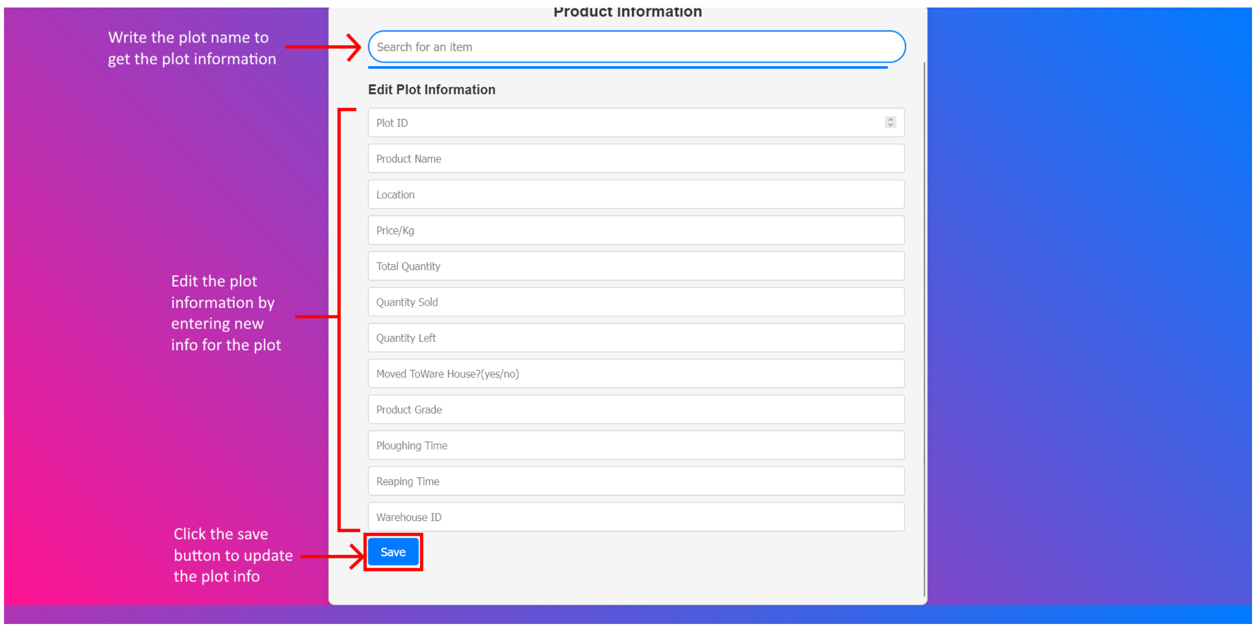


After login, the user will see a welcome screen and 3 buttons on the left: Plot Info, Warehouse, User Info.



Plot Info:

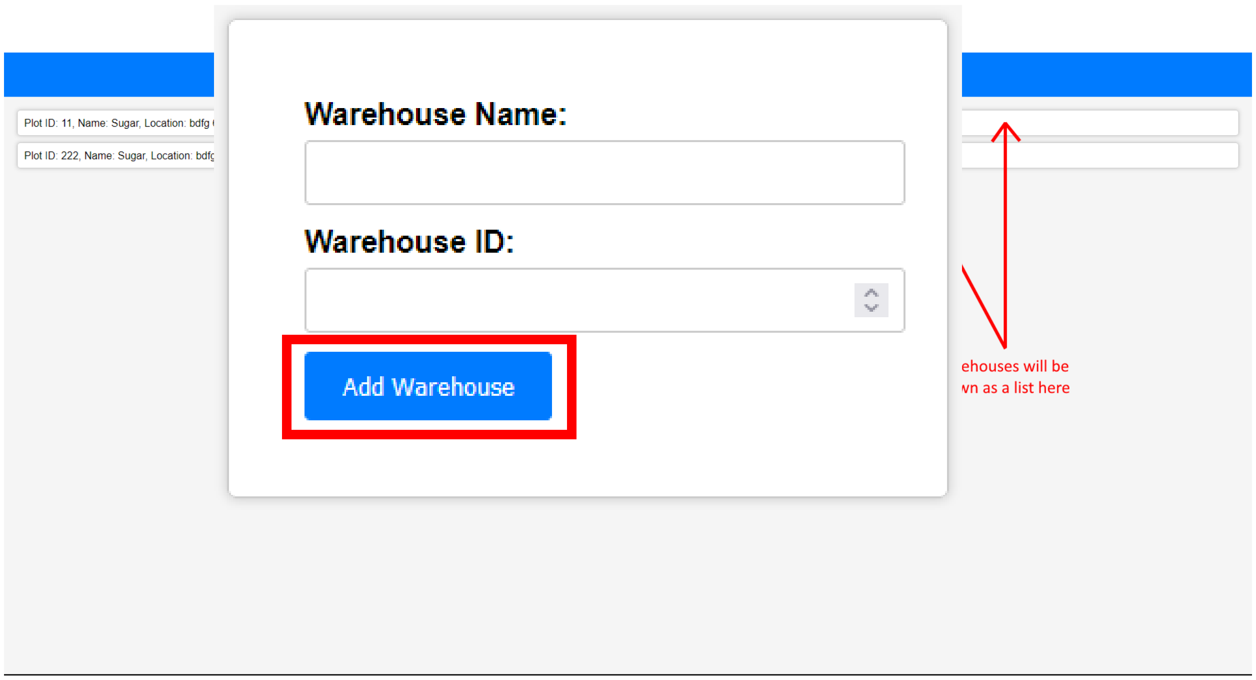
The user can add or edit the plot information in this page. The user can search for an item for edit.



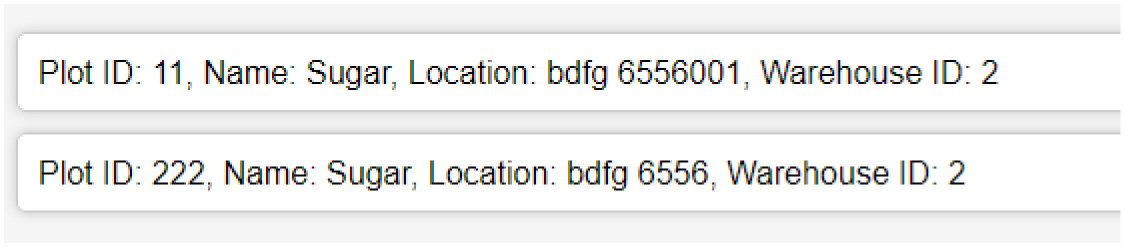
The user can edit plot information by entering the necessary information and press the save button to update the plot information.

Warehouse:

In this page, the user can add warehouse by filling the warehouse name and Warehouse ID.


Plot ID: 11, Name: Sugar, Location: bdfg
Plot ID: 222, Name: Sugar, Location: bdfg
Warehouse Name:
Warehouse ID:
Add Warehouse
ehouses will be
vn as a list here

The list of the warehouse items will be shown in the page right above the warehouse entry block. This will show the Plot ID, Name of the Item, Location and Warehouse ID.

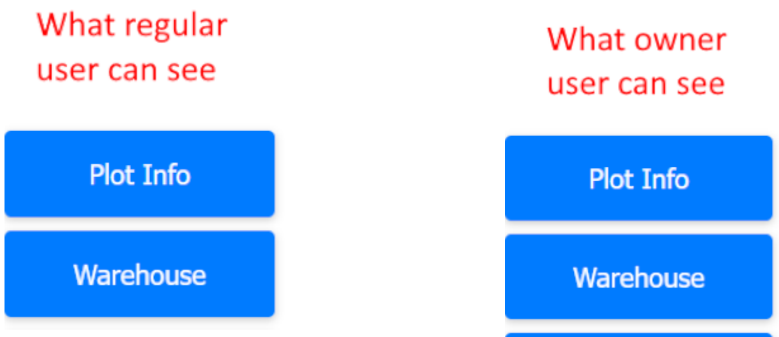


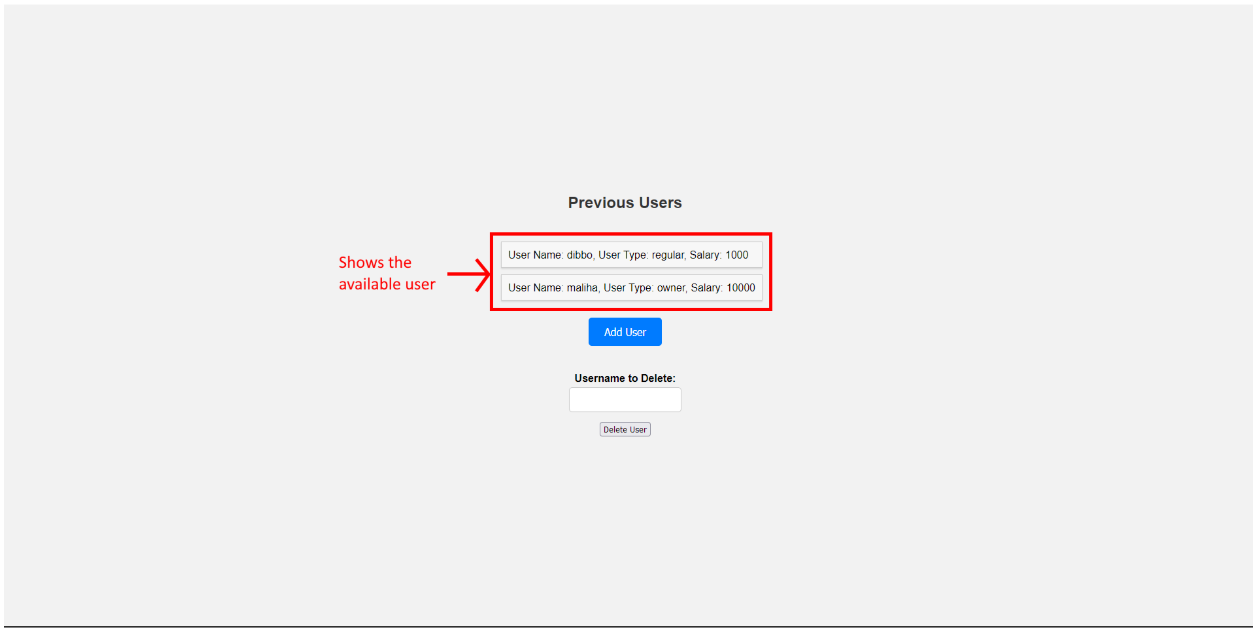
Plot ID: 11, Name: Sugar, Location: bdfg 6556001, Warehouse ID: 2

Plot ID: 222, Name: Sugar, Location: bdfg 6556, Warehouse ID: 2

User Info:

Note: The User Info page can only be accessible by the user type: Owner.

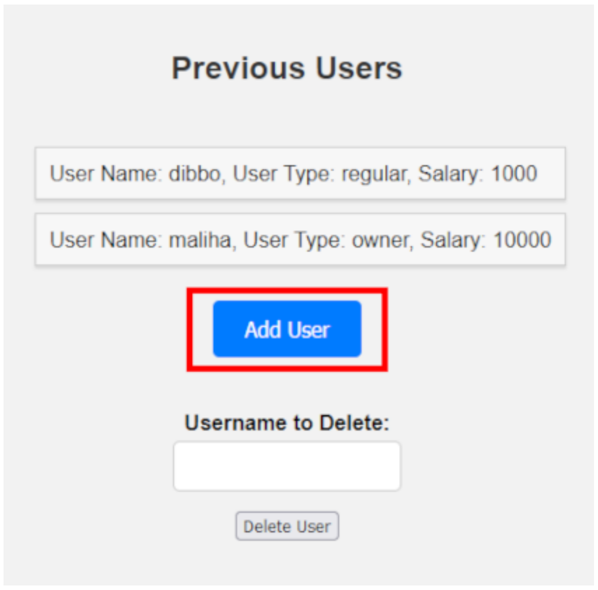



Previous Users

User Name: dibbo, User Type: regular, Salary: 1000

User Name: maliha, User Type: owner, Salary: 10000

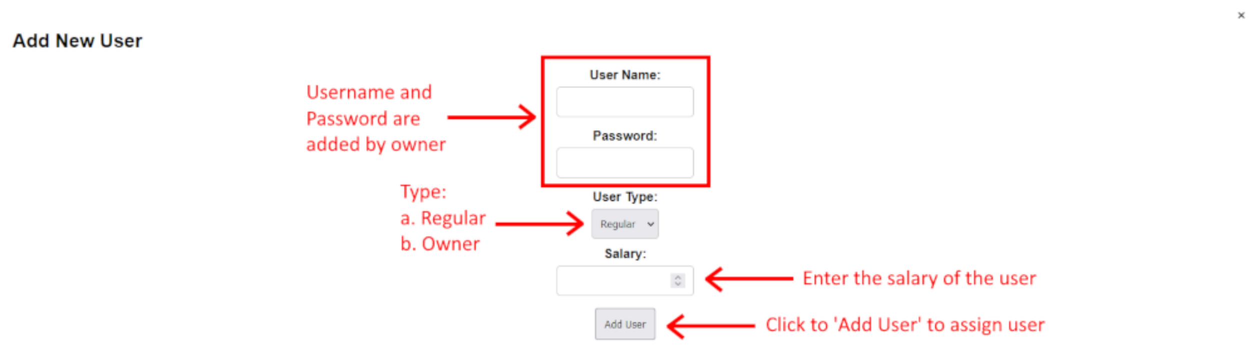
The owner can add user by pressing Add User button.


Previous Users

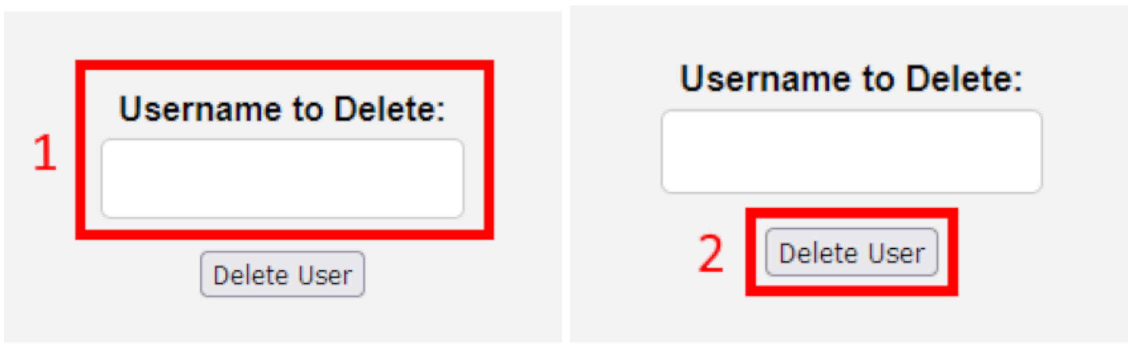
User Name: dibbo, User Type: regular, Salary: 1000

User Name: maliha, User Type: owner, Salary: 10000

The owner will add the username and password and set the user type: Regular and Owner and fix the salary for the user.


Add New User
User Name:
Password:
User Type:
Regular
Salary:
Add User
Username and Password are added by owner
Type:
a. Regular
b. Owner
Enter the salary of the user
Click to 'Add User' to assign user

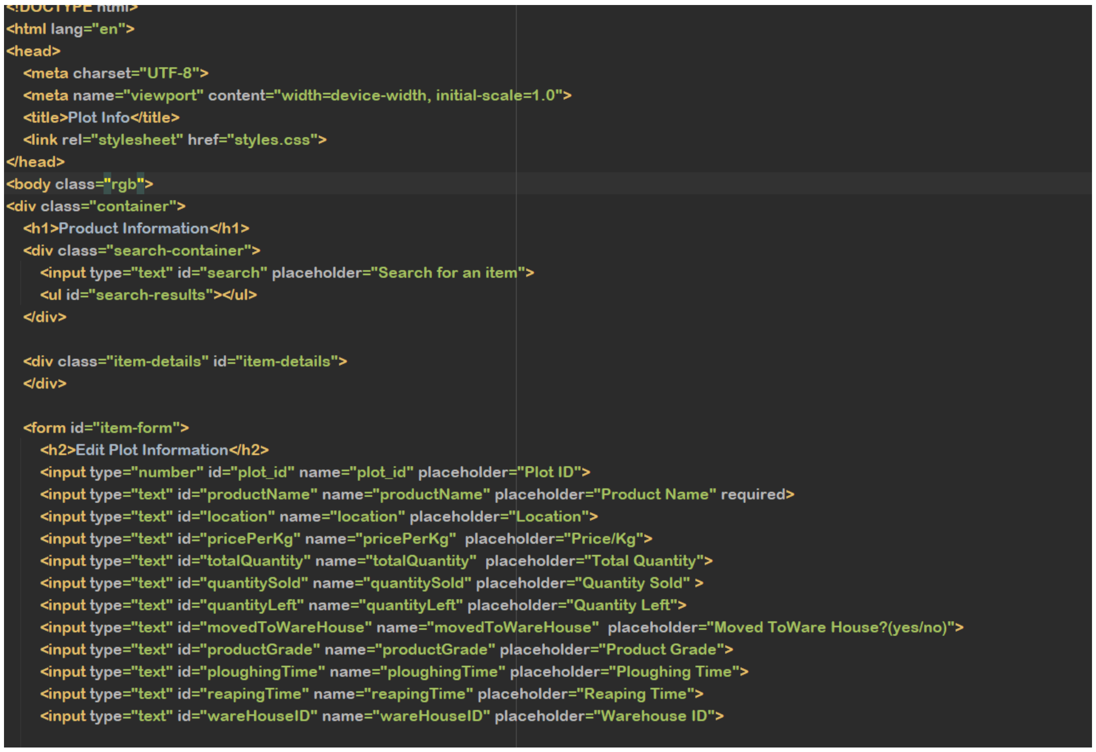
The owner can delete the user by typing the username (for example: dibbo) and press Delete User to delete the user permanently.



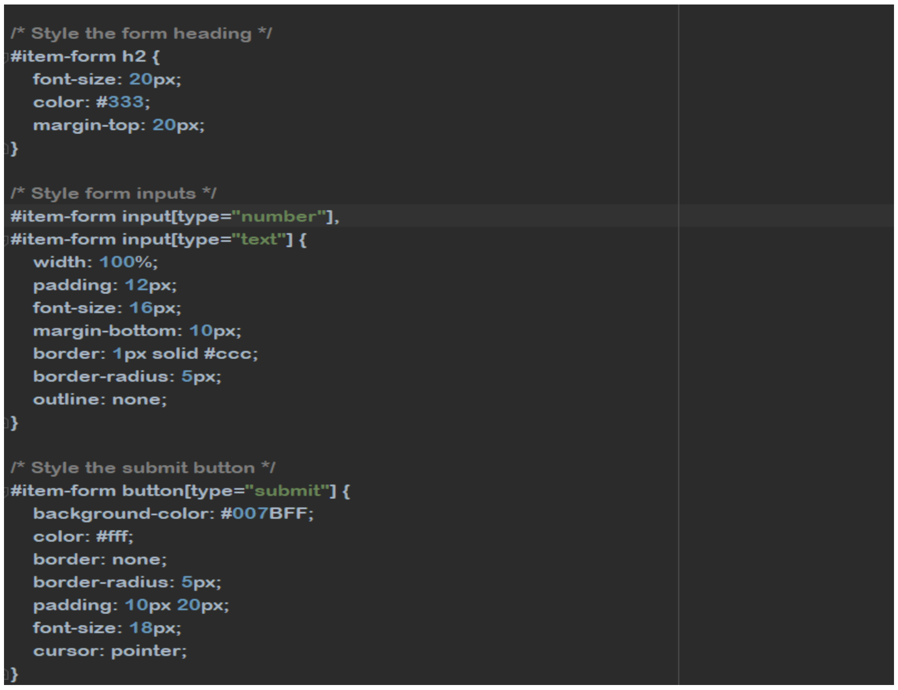
Frontend Development

Since we are following MVC architecture for this project, all of our view related files are under view. Directory → src/main/resources/static/view

We have used html and css for our front end.

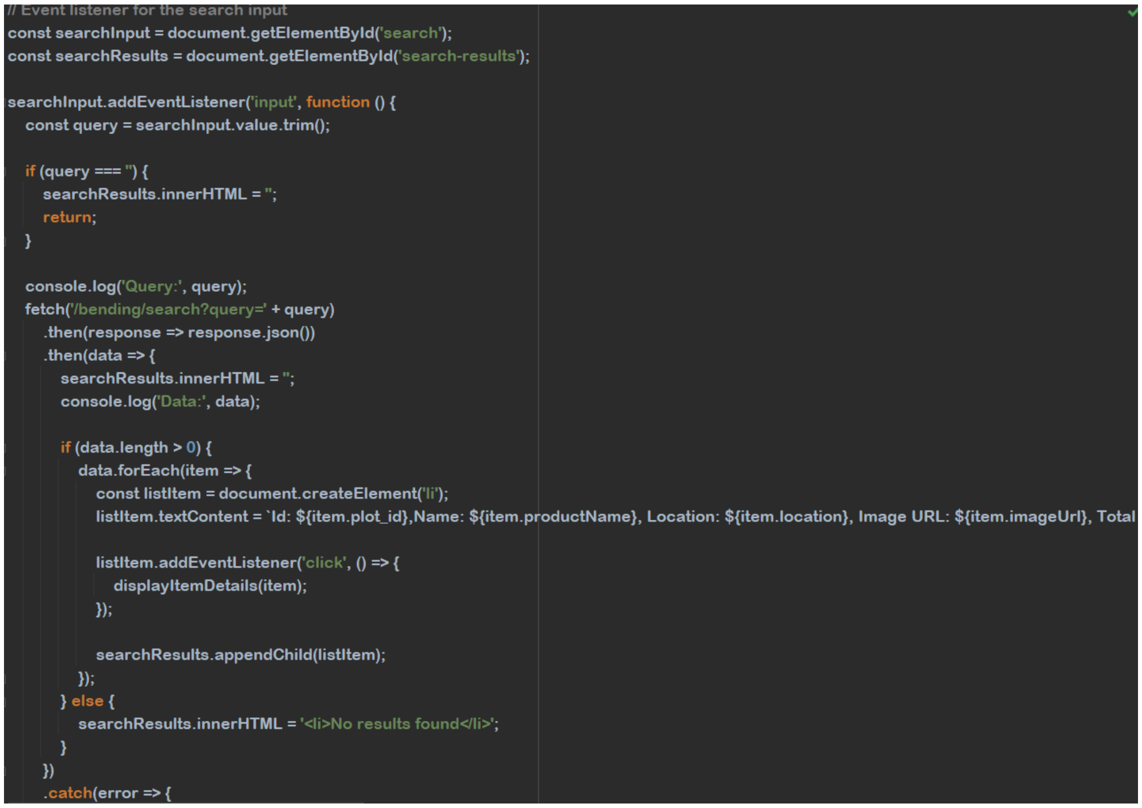


```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Plot Info</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body class="rgb">
<div class="container">
  <h1>Product Information</h1>
  <div class="search-container">
    <input type="text" id="search" placeholder="Search for an item">
    <ul id="search-results"></ul>
  </div>

  <div class="item-details" id="item-details">
  </div>

  <form id="item-form">
    <h2>Edit Plot Information</h2>
    <input type="number" id="plot_id" name="plot_id" placeholder="Plot ID">
    <input type="text" id="productName" name="productName" placeholder="Product Name" required>
    <input type="text" id="location" name="location" placeholder="Location">
    <input type="text" id="pricePerKg" name="pricePerKg"  placeholder="Price/Kg">
    <input type="text" id="totalQuantity" name="totalQuantity"  placeholder="Total Quantity">
    <input type="text" id="quantitySold" name="quantitySold" placeholder="Quantity Sold" >
    <input type="text" id="quantityLeft" name="quantityLeft" placeholder="Quantity Left">
    <input type="text" id="movedToWareHouse" name="movedToWareHouse"  placeholder="Moved ToWare House?(yes/no)">
    <input type="text" id="productGrade" name="productGrade" placeholder="Product Grade">
    <input type="text" id="ploughingTime" name="ploughingTime" placeholder="Ploughing Time">
    <input type="text" id="reapingTime" name="reapingTime" placeholder="Reaping Time">
    <input type="text" id="wareHouseID" name="wareHouseID" placeholder="Warehouse ID">
```

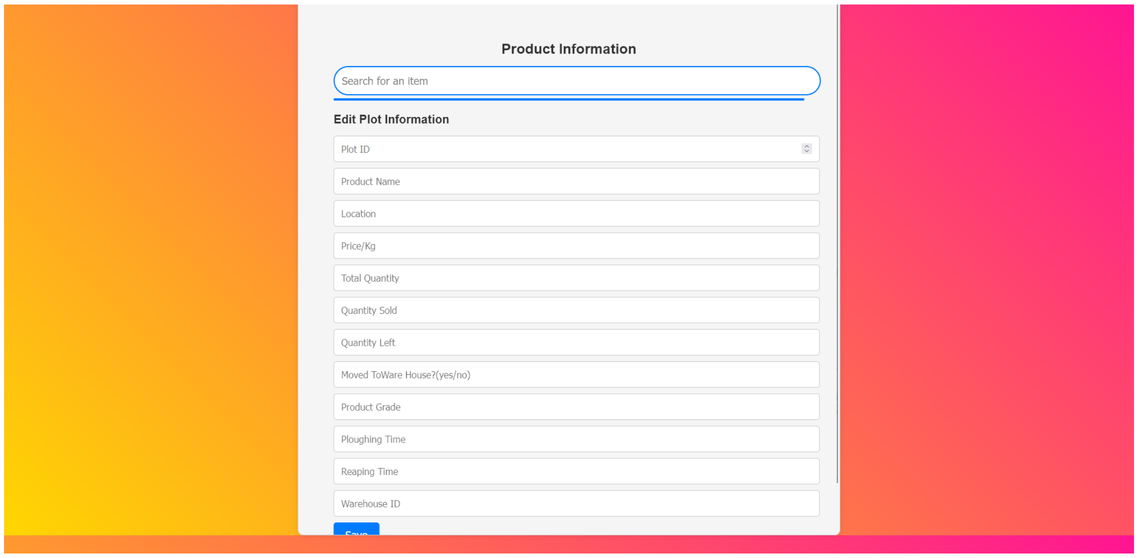


```
/* Style the form heading */
#item-form h2 {
  font-size: 20px;
  color: #333;
  margin-top: 20px;
}

/* Style form inputs */
#item-form input[type="number"],
#item-form input[type="text"] {
  width: 100%;
  padding: 12px;
  font-size: 16px;
  margin-bottom: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
  outline: none;
}

/* Style the submit button */
#item-form button[type="submit"] {
  background-color: #007BFF;
  color: #fff;
  border: none;
  border-radius: 5px;
  padding: 10px 20px;
  font-size: 18px;
  cursor: pointer;
}
```

To communicate between backend and front end without refreshing the page, JavaScript has been used to retrieve the input from the front and set the output to the front end. Here, AJAX has been used to communicate between frontend and backend.



```
// Event listener for the search input
const searchInput = document.getElementById('search');
const searchResults = document.getElementById('search-results');

searchInput.addEventListener('input', function () {
  const query = searchInput.value.trim();

  if (query === '') {
    searchResults.innerHTML = '';
    return;
  }

  console.log('Query:', query);
  fetch('/bending/search?query=' + query)
    .then(response => response.json())
    .then(data => {
      searchResults.innerHTML = '';
      console.log('Data:', data);

      if (data.length > 0) {
        data.forEach(item => {
          const listItem = document.createElement('li');
          listItem.textContent = `Id: ${item.plot_id},Name: ${item.productName}, Location: ${item.location}, Image URL: ${item.imageUrl}, Total

          listItem.addEventListener('click', () => {
            displayItemDetails(item);
          });

          searchResults.appendChild(listItem);
        });
      } else {
        searchResults.innerHTML = '<li>No results found</li>';
      }
    })
    .catch(error => {
```



Product Information

Search for an item

Edit Plot Information

Plot ID

Product Name

Location

Price/Kg

Total Quantity

Quantity Sold

Quantity Left

Moved ToWare House?(yes/no)

Product Grade

Ploughing Time

Reaping Time

Warehouse ID

Save

Backend Development:

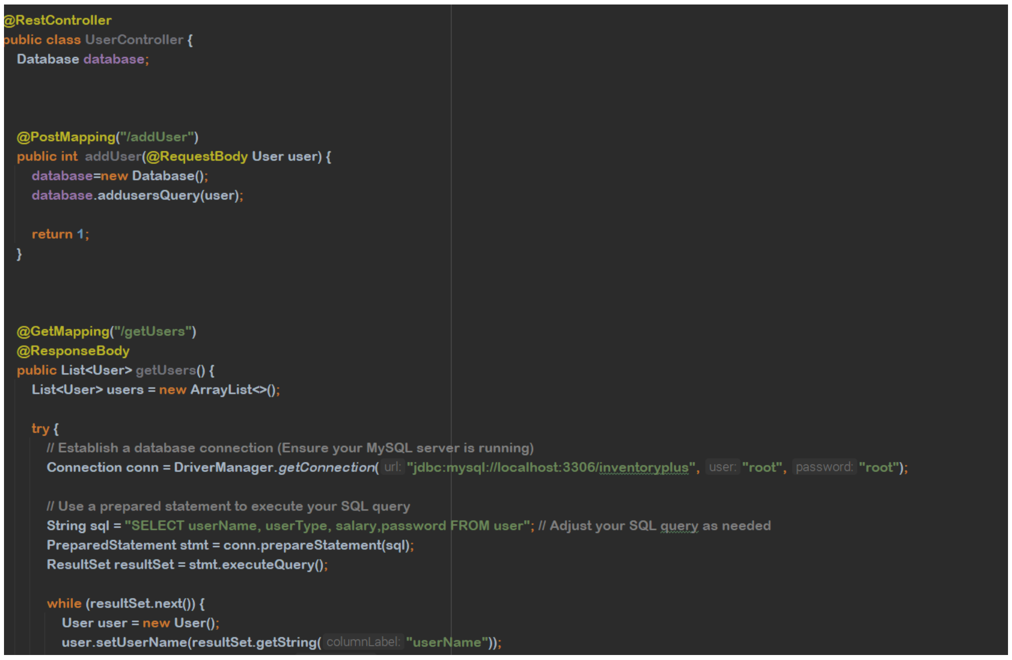
We have completed our whole backend work with api calling; such as post and get.

Whenever a user gives any input, JavaScript files receive that input and make AJAX requests, which are asynchronous HTTP requests to a server from within a web page. In this request, the inputted data is converted into JSON format so that our backend's Java code can Parse it and vice versa.

Since we have to follow MVC architecture, we have put all the controller classes under controller, and we have also put all the blueprint creating classes and DAtabase classes under Model.

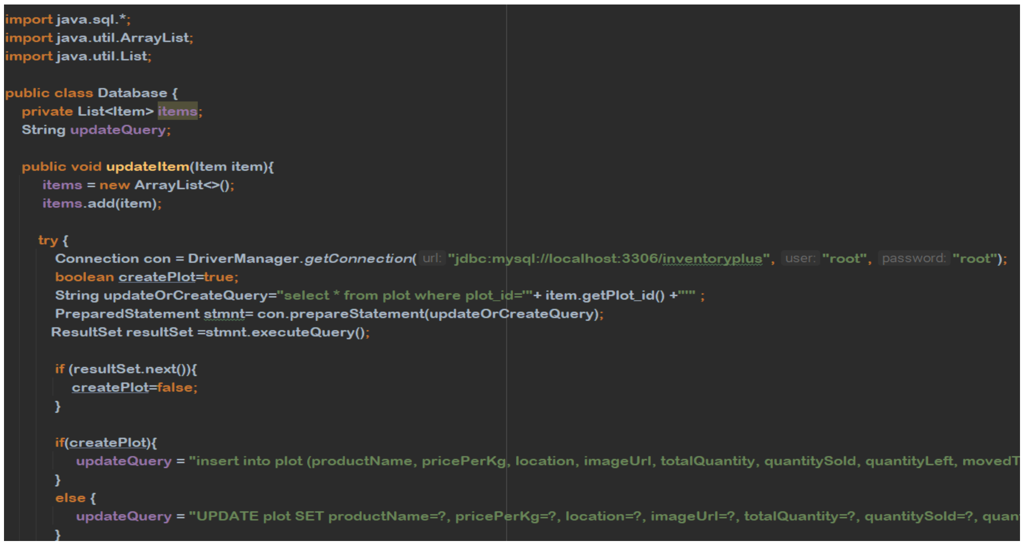
Model directory → src/main/java/tayamee20.github.io/model
Controller directory → src/main/java/tayamee20.github.io/controller

Controller class:



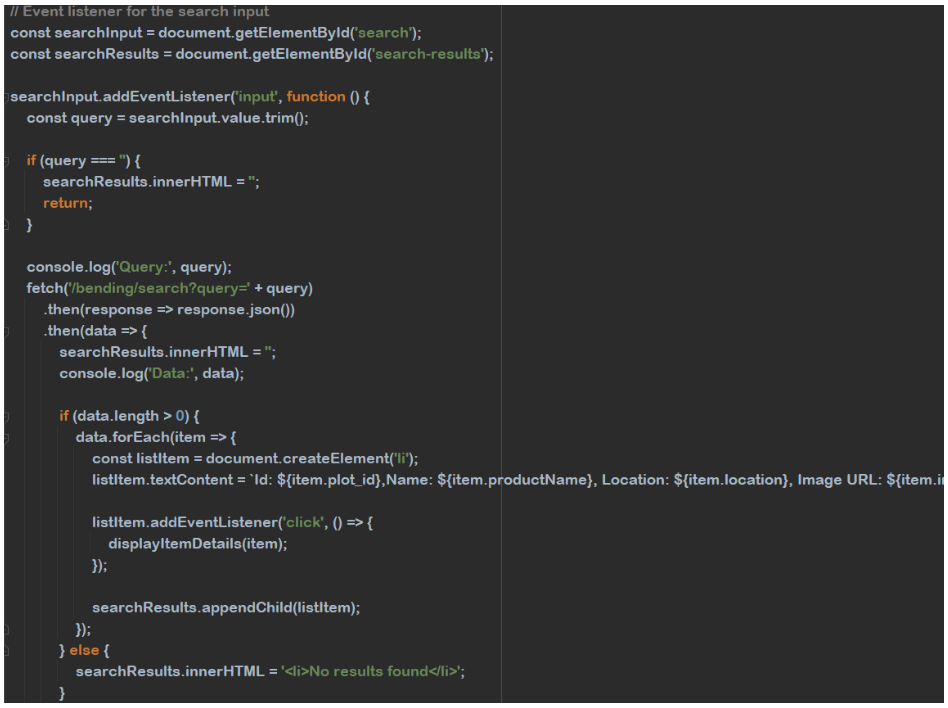
```
@RestController
public class UserController {
  Database database;



  @PostMapping("/addUser")
  public int  addUser(@RequestBody User user) {
    database=new Database();
    database.addusersQuery(user);

    return 1;
  }



  @GetMapping("/getUsers")
  @ResponseBody
  public List<User> getUsers() {
    List<User> users = new ArrayList<>();

    try {
      // Establish a database connection (Ensure your MySQL server is running)
      Connection conn = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/inventoryplus",  user: "root",  password: "root");

      // Use a prepared statement to execute your SQL query
      String sql = "SELECT userName, userType, salary,password FROM user"; // Adjust your SQL query as needed
      PreparedStatement stmt = conn.prepareStatement(sql);
      ResultSet resultSet = stmt.executeQuery();

      while (resultSet.next()) {
        User user = new User();
        user.setUserName(resultSet.getString( columnLabel: "userName"));
```

Database class:



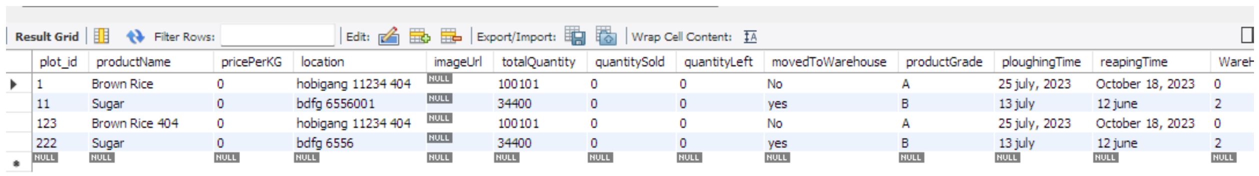
```java
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class Database {
  private List<Item> items;
  String updateQuery;

  public void updateItem(Item item){
     items = new ArrayList<>();
     items.add(item);

    try {
      Connection con = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/inventoryplus", user: "root", password: "root");
      boolean createPlot=true;
      String updateOrCreateQuery="select * from plot where plot_id='"+ item.getPlot_id() +"'" ;
      PreparedStatement stmnt= con.prepareStatement(updateOrCreateQuery);
     ResultSet resultSet =stmnt.executeQuery();

      if (resultSet.next()){
         createPlot=false;
      }

      if(createPlot){
         updateQuery = "insert into plot (productName, pricePerKg, location, imageUrl, totalQuantity, quantitySold, quantityLeft, movedT
      }
      else {
         updateQuery = "UPDATE plot SET productName=?, pricePerKg=?, location=?, imageUrl=?, totalQuantity=?, quantitySold=?, quan
      }
```

JavaScript code:



```javascript
// Event listener for the search input
const searchInput = document.getElementById('search');
const searchResults = document.getElementById('search-results');

searchInput.addEventListener('input', function () {
  const query = searchInput.value.trim();

  if (query === '') {
    searchResults.innerHTML = '';
    return;
  }

  console.log('Query:', query);
  fetch('/bending/search?query=' + query)
    .then(response => response.json())
    .then(data => {
      searchResults.innerHTML = '';
      console.log('Data:', data);

      if (data.length > 0) {
        data.forEach(item => {
          const listItem = document.createElement('li');
          listItem.textContent = `Id: ${item.plot_id},Name: ${item.productName}, Location: ${item.location}, Image URL: ${item.i

          listItem.addEventListener('click', () => {
            displayItemDetails(item);
          });

          searchResults.appendChild(listItem);
        });
      } else {
        searchResults.innerHTML = '<li>No results found</li>';
      }
```

Database:



plot_id	productName	pricePerKG	location	imageUrl	totalQuantity	quantitySold	quantityLeft	movedToWarehouse	productGrade	ploughingTime	reapingTime	WareH
1	Brown Rice	0	hobigang 11234 404	NULL	100101	0	0	No	A	25 july, 2023	October 18, 2023	0
11	Sugar	0	bdfg 6556001	NULL	34400	0	0	yes	B	13 july	12 june	2
123	Brown Rice 404	0	hobigang 11234 404	NULL	100101	0	0	No	A	25 july, 2023	October 18, 2023	0
222	Sugar	0	bdfg 6556	NULL	34400	0	0	yes	B	13 july	12 june	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Conclusion

Finally, our Rice Inventory Management Web App completely transforms how you manage your rice stockpiles. You'll be able to make wise choices and maintain your competitive edge in the ever-changing rice sector because of its user-friendly interface, automatic tracking, and data-driven insights. Our website is your key to eliminating waste, optimizing profitability, and keeping a constant flow of rice to satisfy market needs since it makes inventory management simpler and improves supply chain effectiveness. Accept rice inventory management as it will be in the future and benefit from the shift now.