
GROUP NAME: FUTURE BUILDERS (9)

MODULE: SOFTWARE ENGINEERING

TASK: PROJECT (SPRING 4)

LECTURER: MR. SAMURA

TITLE: SIMPLE INVENTORY MANAGEMENT (SIMS) FOR SMALL_SCALL FOOD RETAILERS AND DISTRIBUTIORS IN BOMBALI DISTRICT, SIERRA LEONE

# WORKING TASK TABLE (Group 9)

| # | Task Description | Assigned Member | Dead line | Email | Grading Criteria |
|---|---|---|---|---|---|
| 1 | Frontend developer Involved in the Backend Write Report | ALHAJI MALIGIE CONTEH | 5/ 03 / 2025<br><br>13/ 03 / 2025 | maligiemorceray@gmail.com | Task completed |
| 2 | Help the frontend development Involved in preparing the final report | YVONNE OLIVE CONTEH | 4/ 03/2025<br><br>12/03/ 2025 | | Task Completed |

## Perform Design Principles and Patterns

Usability (User-Friendly Design)

Simple Interface: Ensure the system is easy to use for non-tech users (e.g., small retailers).
Clear Labels and Instructions: Use simple-to-read UI/UX to help users manage inventory.
Mobile Responsiveness: As many users will likely be accessing the system on mobile, make sure it's mobile responsive.

Scalability (Handling Future Growth)

Modular Design: Divide the system into separate, manageable modules (e.g., Inventory Management, Sales Tracking).

# Black Box Testing

The Component, the group choose is to perform testing on Registration process in the system. **Black Box Testing** is a **software testing technique** where the internal workings of the system are not known to the tester. Instead, the focus is on testing the **functionality** of the application by providing inputs and checking the expected outputs. Black Box Testing focuses on testing the functionality of the system without knowing its internal structure or implementation. For the **Register Component**, we will analyze the screenshots (**Screenshot 2.png** and **Screenshot 3.png**) to understand how the system behaves during the registration process.

1. **Screenshot.png - OTP Verification**

**What the Screenshot Shows:**

- The page displays a prompt: **"Enter OTP sent to your email/phone."**
- The user's details are pre-filled:
    - **Name**: ALHAJI MALIGIE CONTEH
    - **Email**: maligiemorceray@gmail.com
    - **Username**: @marcel
    - **Password**: Masked with dots (……)
    - **Password Strength**: **Medium**
    - **Role**: **Admin**
- There are two buttons: **OK** and **Cancel**.

**Black Box Testing Explanation:**

- **Test Case**: Verify that the system prompts for OTP verification after the user submits their registration details.
- **Input**: User details (name, email, username, password, role) and an OTP.
- **Expected Output**: The system should prompt the user to enter the OTP sent to their email/phone.
- **Observation**: The system correctly displays the OTP verification prompt.
- **Conclusion**: The **Register Component** correctly implements OTP verification as part of the registration process

**2. Screenshots 3.png - Successful Registration**

**What the Screenshot Shows:**

- The page displays a success message: **"Registration successful! Redirecting to dashboard..."**
- The user's details are pre-filled:
    - **Name**: ALHAJI MALIGIE CONTEH
    - **Email**: maligiemorceray@gmail.com
    - **Username**: @marcel
    - **Password**: Masked with dots (.....)
    - **Password Strength**: **Medium**

    - **Role**: **Admin**
- The system name is displayed: **"SIMPLE"** and **"SKY (51MS)."**

**Black Box Testing Explanation:**

- **Test Case**: Verify that the system successfully registers a user and redirects them to the dashboard after OTP verification.
- **Input**: Valid user details (name, email, username, password, and role) and correct OTP.
- **Expected Output**: The system should display a success message and redirect the user to the dashboard.
- **Observation**: The system correctly displays the success message and indicates that it is redirecting to the dashboard.
- **Conclusion**: The **Register Component** successfully handles valid inputs and completes the registration process.

  **OTP Verification**:
    - The system correctly prompts the user to enter the OTP sent to their email/phone.
    - This ensures an additional layer of security during the registration process.

  **Successful Registration**:
    - The system successfully registers the user and redirects them to the dashboard after OTP verification.
    - The success message confirms that the registration process is complete.

1. **Secure Registration**:
    - The system implements OTP verification, ensuring that only verified users can complete the registration process.
2. **User Feedback**:
    - The system provides clear feedback during the registration process, including:
        - Password strength indicator.
        - OTP verification prompt.
        - Success message upon successful registration.
3. **Navigation**:
    - After successful registration, the system redirects the user to the dashboard, as required.
4. **Data Retention**:
    - The system retains and displays user input during the registration process, ensuring a smooth user experience.

# White Box Testing

Also known as **Clear Box, Open Box, or Structural Testing** is a software testing technique where the **internal structure, logic, and code** of the application are tested. **White Box Testing** requires knowledge of the code and is performed at the **code level**.

```
document.getElementById('login-form').addEventListener('submit', (e) => {
  e.preventDefault();
  const username = document.getElementById('username').value;
  const email = document.getElementById('email').value;
  const password = document.getElementById('password').value;

  // Simulate login validation
  if (username && email && password) {
    alert('Login Successful!');
    document.getElementById('login-dashboard').style.display = 'none';
    document.getElementById('admin-dashboard').style.display = 'block';
  } else {
    alert('Invalid credentials!');
  }
});
```

- **Purpose**: This code handles the login functionality.
- **Functionality**:
  - When the login form is submitted, it prevents the default form submission (e.preventDefault ()).
  - It retrieves the values of the username, email, and password fields.
  - It performs a simple validation to check if all fields are filled.
    - If valid, it displays a success message, hides the **Login Dashboard**, and shows the **Admin Dashboard**.
    - If invalid, it displays an error message.
- **Code Paths**:
  - Valid credentials → Display success message → Hide login-dashboard → Show admin-dashboard.
  - Invalid credentials → Display error message.

## White Box testing output

**Test Case for Login and Register Navigation**

**Test Case 1**: Click "REGISTER HERE" link.

- **Input**: Click event on go-to-register.
- **Expected Output**: Hide login-dashboard, show register-dashboard.
- **Code Path**: go-to-register → Hide login-dashboard → Show register-dashboard.

**Test Case 2**: Click "LOGIN HERE" link.

- **Input**: Click event on go-to-login.
- **Expected Output**: Hide register-dashboard, show login-dashboard.
- **Code Path**: go-to-login → Hide register-dashboard → Show login-dashboard.

## Perform all the test stages for your entire system.

### Unit Testing

Unit testing involves testing individual components or functions in isolation to ensure they work correctly.

**What to Test:**

- **Login Functionality**: Verify that the login form validates inputs and redirects to the Admin Dashboard on successful login.
- **Registration Functionality**: Ensure the registration form validates inputs, checks OTP, and redirects to the Login Dashboard after successful registration.
- **Product Entry**: Test if the product entry form submits data correctly and updates the product table.
- **Sales Entry**: Verify that sales entries are recorded, and profit/loss is calculated correctly.
- **Reports & Analytics**: Check if metrics like total stock, sales, low stock alerts, and expiring soon alerts are updated correctly.
- **Transaction Entry**: Ensure transaction entries are recorded and displayed in the transaction table.

**How to Test:**

- Manually input valid and invalid data into each form and check for correct behavior (e.g., error messages, successful submissions).
- Use `console.log ()` to debug JavaScript functions and verify outputs.

### 2. Integration Testing

Integration testing ensures that different modules or components work together as expected.

**What to Test:**

- **Login → Admin Dashboard**: Verify that successful login redirects to the Admin Dashboard.
- **Registration → Login**: Ensure successful registration redirects to the Login Dashboard.
- **Product Entry → Sales Entry**: Check if products added in the Product Entry section are available for selection in the Sales Entry section.
- **Sales Entry → Reports & Analytics**: Verify that sales data updates the Reports & Analytics section (e.g., total sales, profit/loss).
- **Transaction Entry → Reports & Analytics**: Ensure transactions are reflected in the Reports & Analytics section.

**How to Test:**

- Perform end-to-end workflows (e.g., register → login → add product → record sale → check reports).
- Use browser developer tools to monitor network requests and ensure data flows correctly between components.

---

## 3. System Testing

System testing validates the entire system as a whole to ensure it meets the specified requirements.

**What to Test:**

- **Role-Based Access**: Verify that only authorized users (Admin, Manager, and Salesperson) can access specific features.
- **Automatic Stock Tracking**: Ensure stock levels are updated automatically after sales or product entries.
- **Pop-up Messages**: Confirm that pop-up messages appear for successful actions (e.g., login, registration, product entry).
- **UI Consistency**: Check that the UI design is consistent across all dashboards (e.g., colors, fonts, button styles).

**How to Test:**

- Test the system with different user roles and ensure access control works as expected.
- Simulate real-world scenarios (e.g., adding products, recording sales, checking stock levels).
- Verify that all pop-up messages are displayed correctly.

---

## 4. User Acceptance Testing (UAT)

UAT involves testing the system with end-users to ensure it meets their needs and expectations.

**What to Test:**

- **Ease of Use**: Verify that the system is user-friendly and intuitive.
- **Functionality**: Ensure all features work as expected from the user's perspective.
- **Performance**: Check that the system responds quickly and handles data efficiently.

**How to Test:**

- Provide the system to a group of end-users (e.g., Admin, Manager, and Salesperson).
- Collect feedback on usability, functionality, and performance.
- Address any issues or suggestions raised by users.

## 5. Performance Testing

Performance testing ensures the system performs well under expected workloads.

**What to Test:**

- **Load Handling**: Verify that the system can handle multiple users and large datasets without slowing down.
- **Response Time**: Ensure that pages and forms load quickly.

**How to Test:**

- Simulate multiple users accessing the system simultaneously.
- Use browser developer tools to measure page load times and identify bottlenecks.

## 6. Security Testing

Security testing ensures the system is secure and protects user data.

**What to Test:**

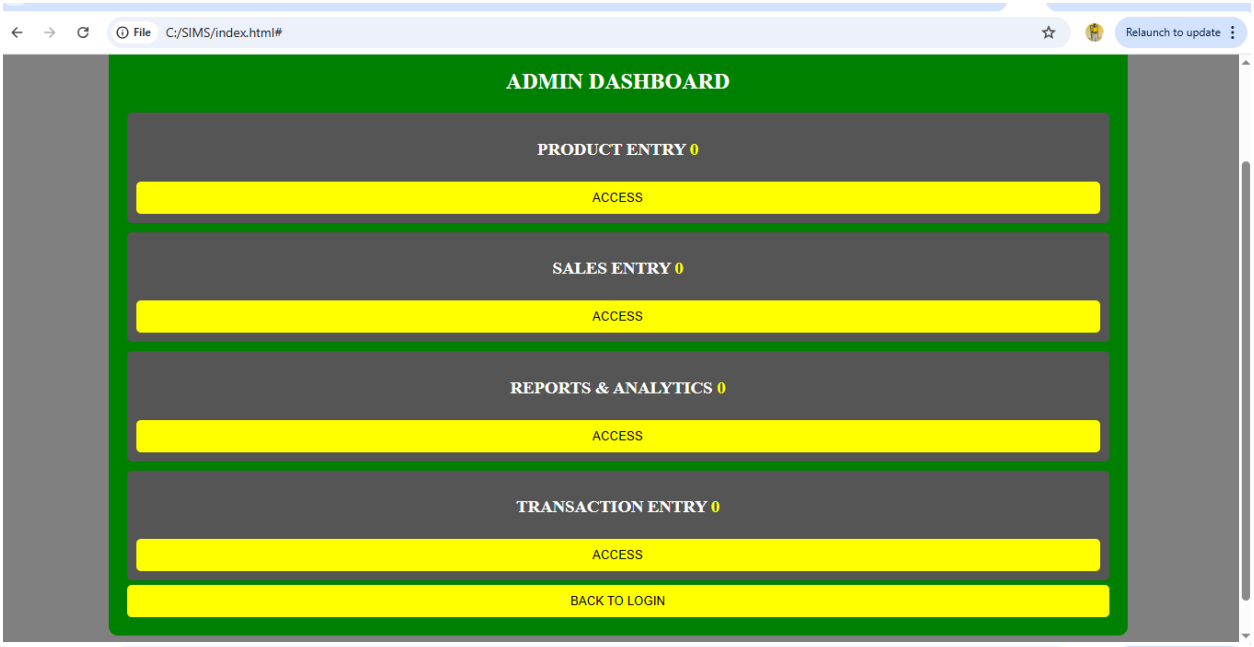
- **Password Encryption**: Verify that passwords are encrypted before storage.
- **Role-Based Access**: Ensure unauthorized users cannot access restricted features.
- **Data Validation**: Check that the system prevents SQL injection, XSS, and other common attacks.

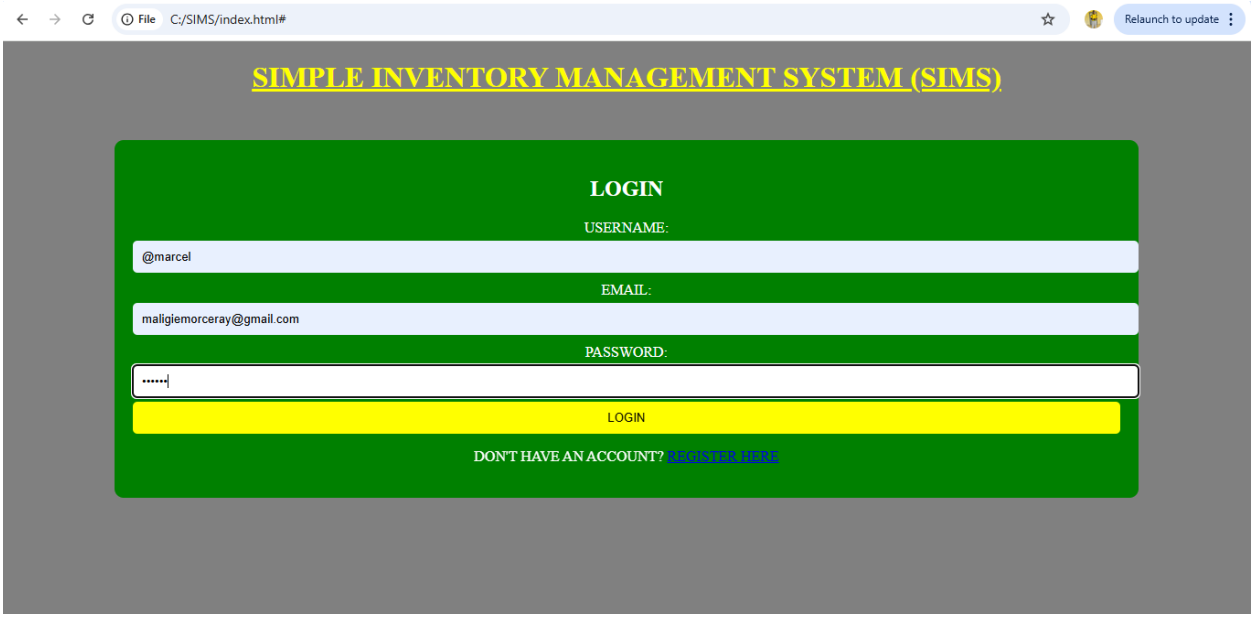
**How to Test:**

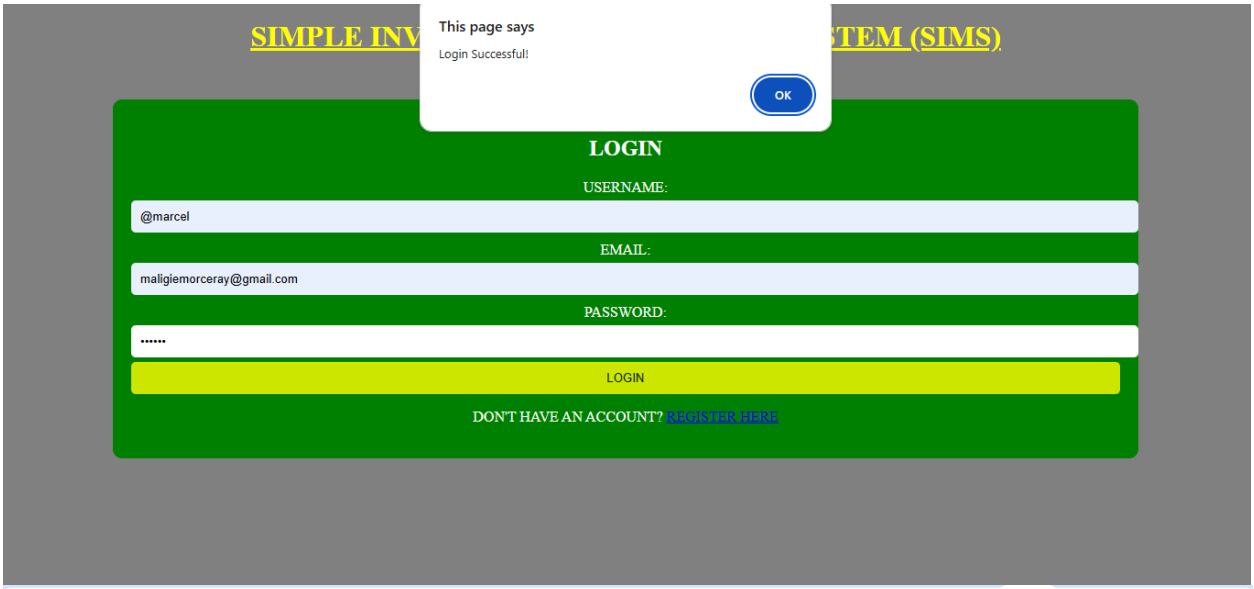
- Attempt to bypass login or access restricted areas without proper credentials.
- Test input fields for vulnerabilities (e.g., entering scripts or SQL queries).

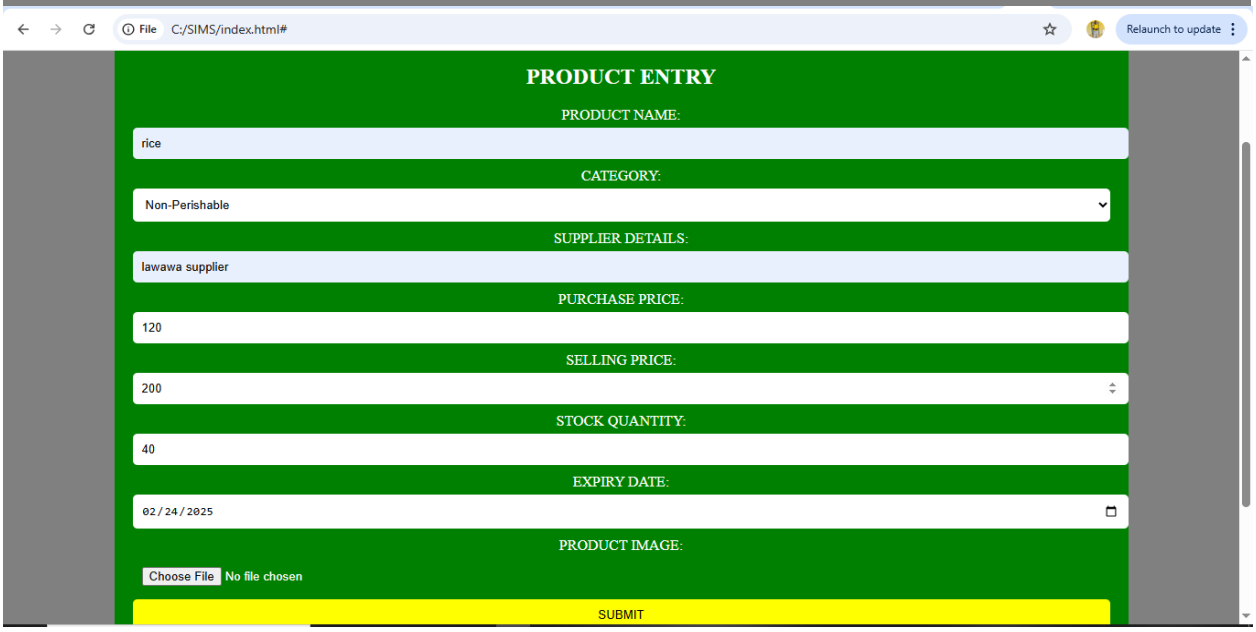
## Appendix Section

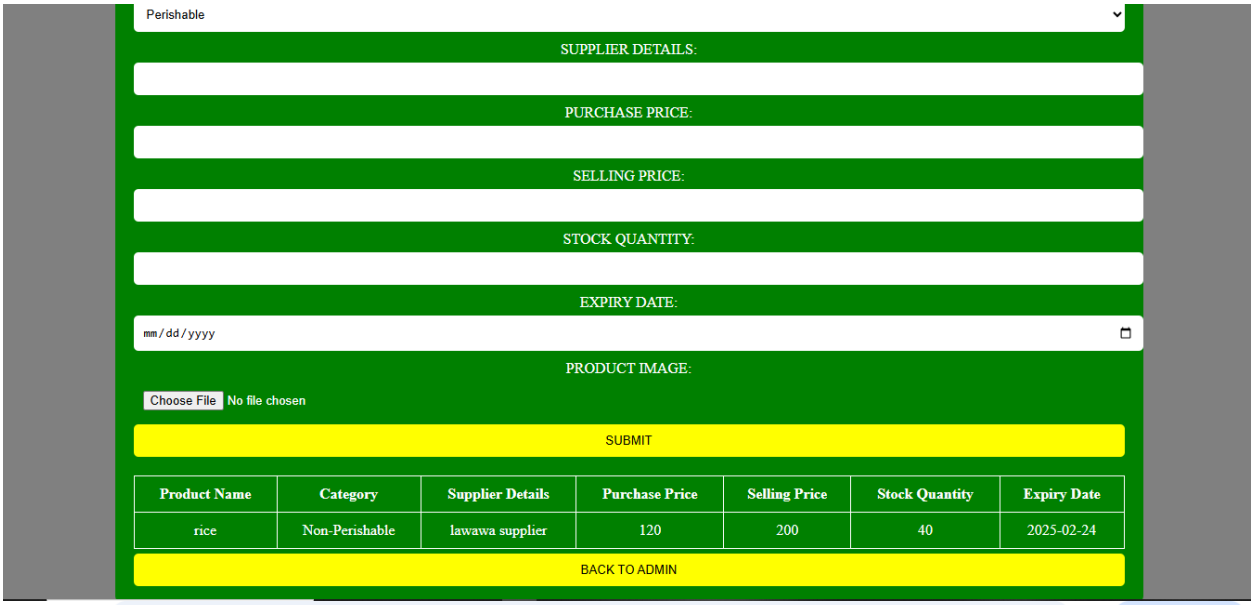
# Screenshot picture of the system testing

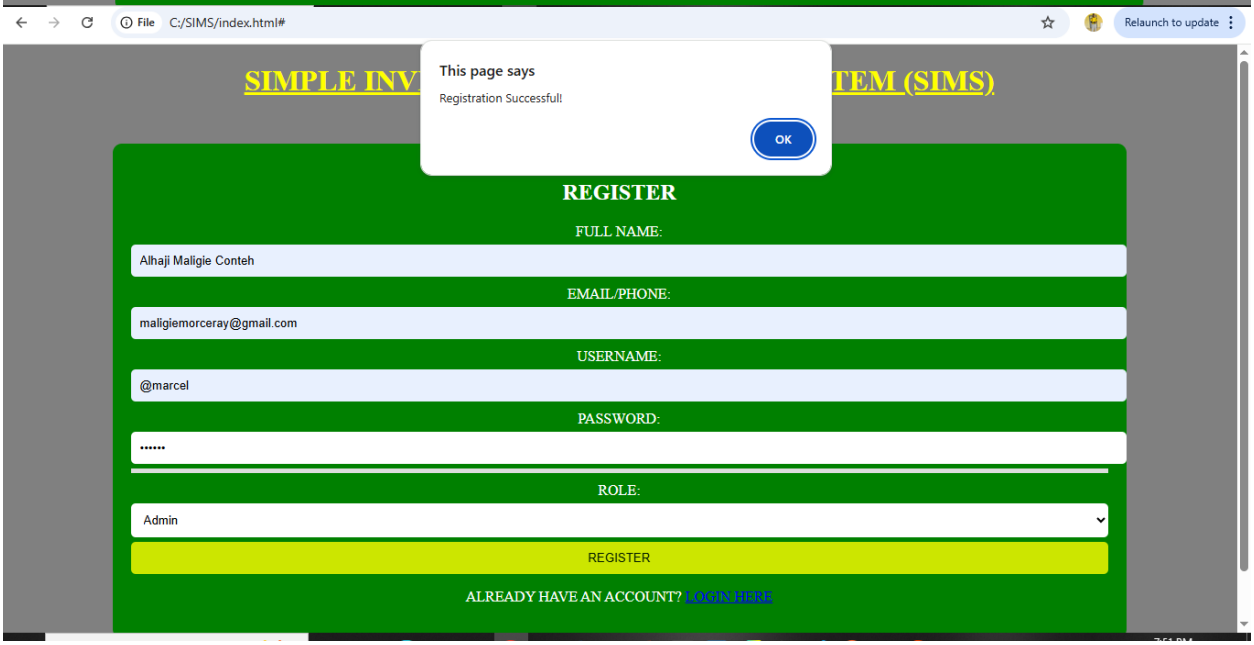


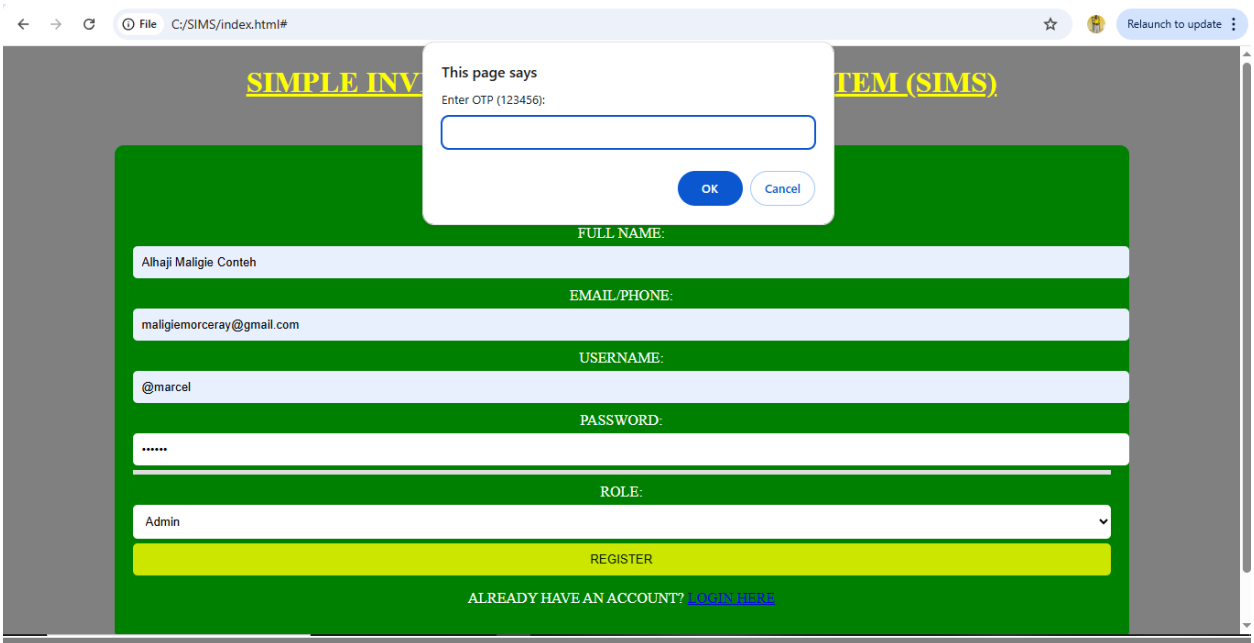




SIMPLE INV
TEM (SIMS)
This page says
Login Successful!
OK
LOGIN
USERNAME:
@marcel
EMAIL:
maligiemorceray@gmail.com
PASSWORD:
LOGIN
DON'T HAVE AN ACCOUNT? REGISTER HERE

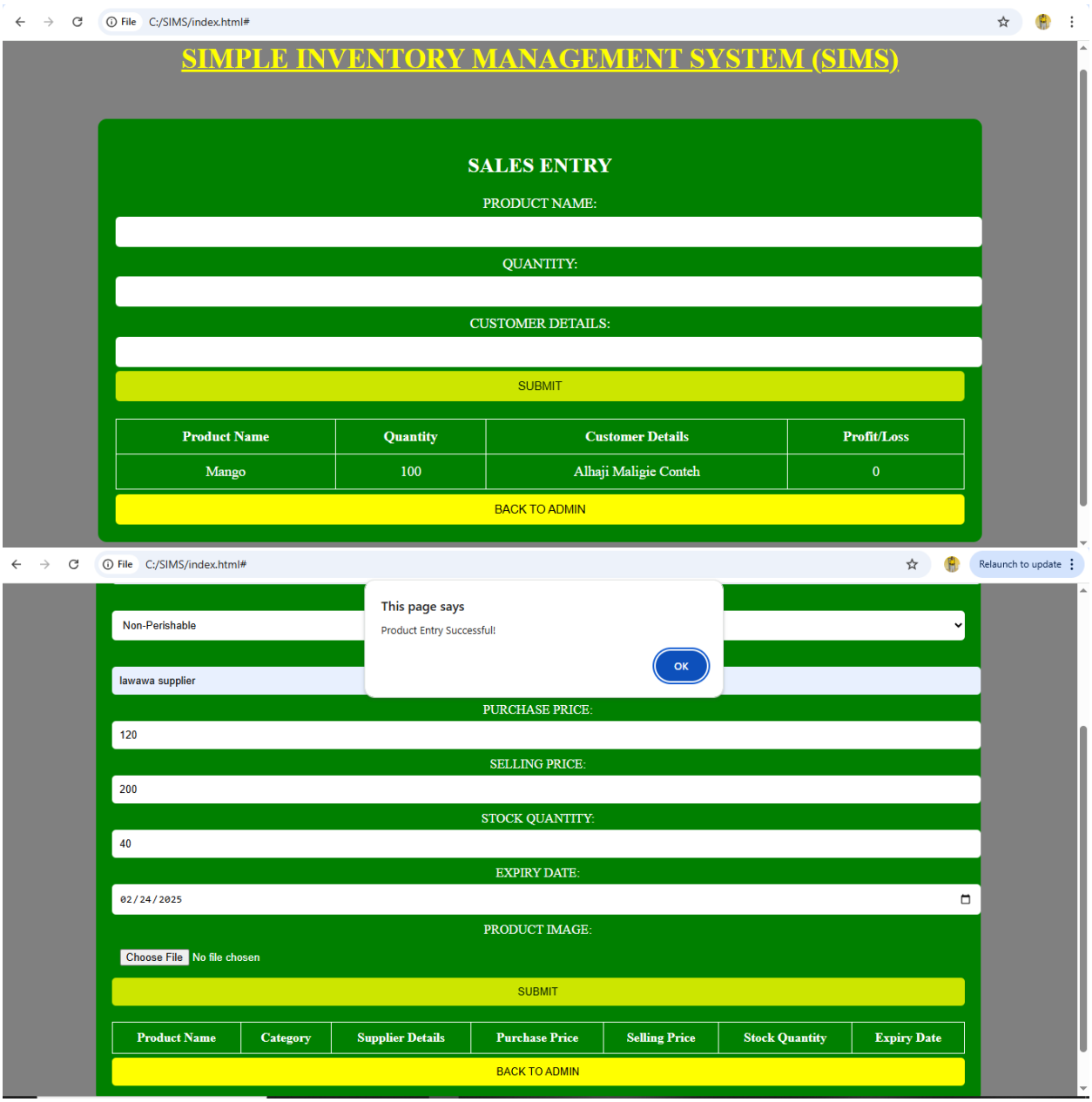

File C:/SIMS/index.html#
Relaunch to update
PRODUCT ENTRY
PRODUCT NAME:
rice
CATEGORY:
Non-Perishable
SUPPLIER DETAILS:
lawawa supplier
PURCHASE PRICE:
120
SELLING PRICE:
200
STOCK QUANTITY:
40
EXPIRY DATE:
02/24/2025
PRODUCT IMAGE:
Choose File
No file chosen
SUBMIT


Perishable
SUPPLIER DETAILS:
PURCHASE PRICE:
SELLING PRICE:
STOCK QUANTITY:
EXPIRY DATE:
mm / dd / yyyy
PRODUCT IMAGE:
Choose File
No file chosen
SUBMIT
Product Name
Category
Supplier Details
Purchase Price
Selling Price
Stock Quantity
Expiry Date
rice
Non-Perishable
lawawa supplier
120
200
40
2025-02-24
BACK TO ADMIN

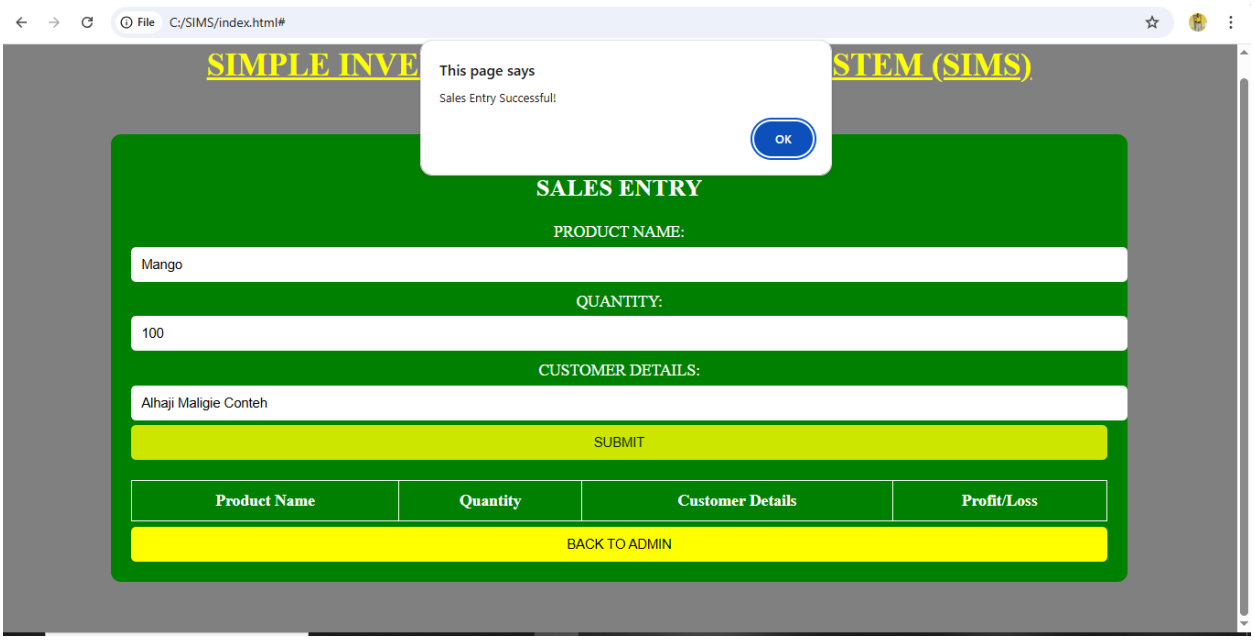

File
C:/SIMS/index.html#
Relaunch to update
SIMPLE INV
TEM (SIMS)
This page says
Registration Successful!
OK
REGISTER
FULL NAME:
Alhaji Maligie Conteh
EMAIL/PHONE:
maligiemorceray@gmail.com
USERNAME:
@marcel
PASSWORD:
ROLE:
Admin
REGISTER
ALREADY HAVE AN ACCOUNT? LOGIN HERE

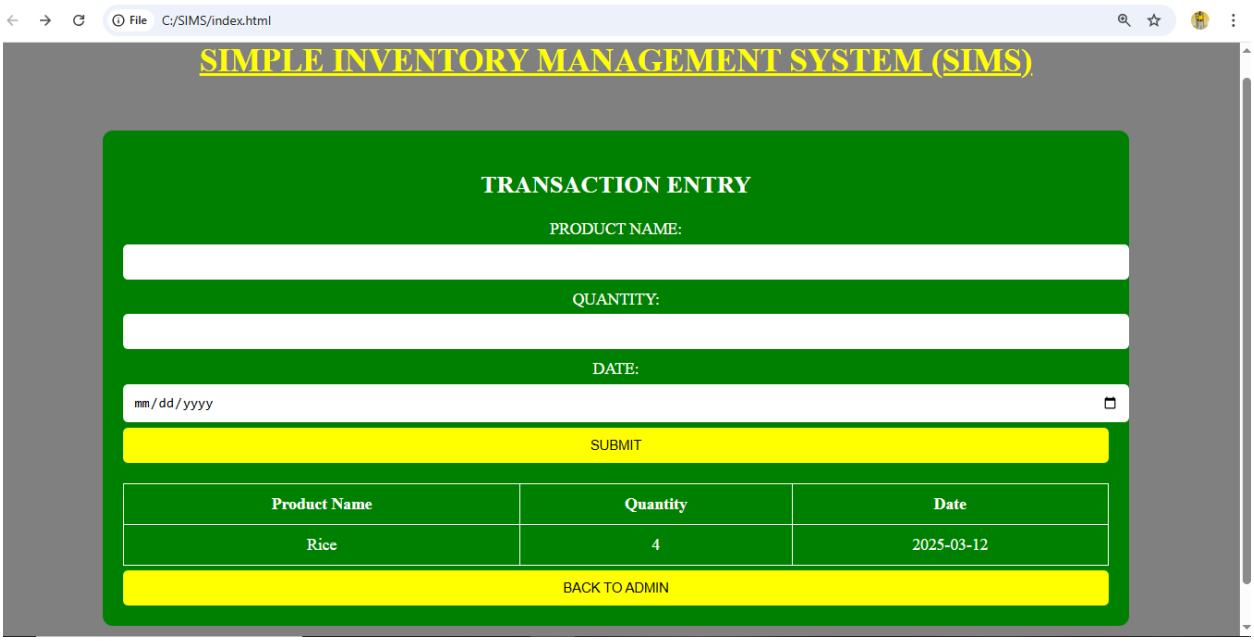

File C:/SIMS/index.html#
Relaunch to update
SIMPLE INV
TEM (SIMS)
This page says
Enter OTP (123456):
OK
Cancel
FULL NAME:
Alhaji Maligie Conteh
EMAIL/PHONE:
maligiemorceray@gmail.com
USERNAME:
@marcel
PASSWORD:
ROLE:
Admin
REGISTER
ALREADY HAVE AN ACCOUNT? LOGIN HERE


SIMPLE INVENTORY MANAGEMENT SYSTEM (SIMS)
SALES ENTRY
PRODUCT NAME:
QUANTITY:
CUSTOMER DETAILS:
SUBMIT
Product Name
Quantity
Customer Details
Profit/Loss
BACK TO ADMIN


File C:/SIMS/index.html#
SIMPLE INVENTORY MANAGEMENT SYSTEM (SIMS)
SALES ENTRY
PRODUCT NAME:
QUANTITY:
CUSTOMER DETAILS:
SUBMIT
Product Name
Quantity
Customer Details
Profit/Loss
Mango
100
Alhaji Maligie Conteh
0
BACK TO ADMIN


File C:/SIMS/index.html#
Relaunch to update
This page says
Product Entry Successful!
OK
Non-Perishable
lawawa supplier
PURCHASE PRICE:
120
SELLING PRICE:
200
STOCK QUANTITY:
40
EXPIRY DATE:
02/24/2025
PRODUCT IMAGE:
Choose File No file chosen
SUBMIT
Product Name
Category
Supplier Details
Purchase Price
Selling Price
Stock Quantity
Expiry Date
BACK TO ADMIN


File C:/SIMS/index.html#
SIMPLE INVE
STEM (SIMS)
This page says
Sales Entry Successful!
OK
SALES ENTRY
PRODUCT NAME:
Mango
QUANTITY:
100
CUSTOMER DETAILS:
Alhaji Maligie Conteh
SUBMIT
Product Name
Quantity
Customer Details
Profit/Loss
BACK TO ADMIN


File C:/SIMS/index.html
SIMPLE INVENTORY MANAGEMENT SYSTEM (SIMS)
TRANSACTION ENTRY
PRODUCT NAME:
QUANTITY:
DATE:
mm/dd/yyyy
SUBMIT
Product Name
Quantity
Date
Rice
4
2025-03-12
BACK TO ADMIN


```javascript
document.getElementById('login-form').addEventListener('submit', (e) => {
  e.preventDefault();
  const username = document.getElementById('username').value;
  const email = document.getElementById('email').value;
  const password = document.getElementById('password').value;

  // Simulate login validation
  if (username && email && password) {
    alert('Login Successful!');
    document.getElementById('login-dashboard').style.display = 'none';
    document.getElementById('admin-dashboard').style.display = 'block';
  } else {
    alert('Invalid credentials!');
  }
});
```

**GitHub URL** https://github.com/alkid12345/Future-Builders

**Application URL**

https://alkid12345.github.io/website.sims/