# 

# ONLINE STORE FOR ORGANIC VEGGIES AND FRUITS

**21IT005 – REST API Using Spring Boot**

## PROJECT REPORT

***Submitted by***

**KAVIARASAN M 727721EUEC068**

**KAVINESH MS 727721EUEC069**

**PAVIN K 727721EUEC108**

***in partial fulfilment for the award of the degree***

***of***

## BACHELOR OF ENGINEERING

## IN

## ELECTRONICS AND COMMUNICATION ENGINEERING

#### SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

**(An Autonomous Institution, Affiliated to Anna University Chennai - 600 025)**

**SEPTEMBER 2023**



## BONAFIDE CERTIFICATE

Certified that this project report **“ONLINE STORE FOR ORGNIC VEGGIES AND FRUITS”** is the bonafide work of “**KAVIARSAN M(727721EUEC068) KAVINESH MS(727721EUEC069) PAVIN K(727721EUEC108)”** who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

**Ms. Anishfathima**

**SUPERVISOR**

**ASSISTANT PROFESSOR**

ELECTRONICS AND COMMUNICATION ENGINEERING

**Dr.S. SASIPRIYA., Ph.D.**

**HEAD OF THE DEPARTMENT**

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted for the Project viva-voce examination held on\_\_\_\_\_\_\_\_\_\_\_

INTERNAL EXAMINER EXTERNAL EXAMINER

**ACKNOWLEDGEMENT**

We express our sincere thanks to the management and **Dr. J. JANET M.E., Ph.D**., Principal, Sri Krishna College of Engineering and Technology, Coimbatore for providing us the facilities to carry out this mini project work.

We are thankful to **Dr. S. SASIPRIYA., Ph.D**., Professor and Head, Department of Electronics and Communication Engineering, for her continuous evaluation and comments given during the mini project work.

We express our deep sense of gratitude to our supervisor **MS.B. ANISHFATHIMA** Assistant

Professor, Department of Electronics and Communication Engineering for her valuable advice, guidance, and support during our mini project work.

We would also like to thank our mini project coordinator **Ms.S. PRASEETHA**

Assistant Professor, Department of Electronics and Communication Engineering for helping us in completing our mini project work.

We express our heartfelt sense of gratitude and thanks to our beloved parents, family and friends who have helped during the mini project course.

**ABSTRACT**

The "Online Store for Organic Fruits and Vegetables" is an innovative e-commerce platform designed to meet the increasing demand for organic produce. This project provides a user-friendly interface for customers to browse, search, and purchase certified organic fruits and vegetables conveniently. It also connects local organic farmers and consumers, promoting sustainable agriculture practices. Users can access educational content on organic living and make secure online payments. The project's robust database management ensures efficient order processing and tracking. Post-launch, dedicated maintenance and support ensure a seamless shopping experience.

In response to the growing awareness of the benefits of organic living, the "Online Store for Organic Fruits and Vegetables" offers a comprehensive e-commerce solution. This project allows users to register, browse, and search for a wide range of organic fruits and vegetables. Customers can add products to their shopping carts and make secure online payments. User reviews and ratings foster trust, while educational content promotes organic living. The project's database management and order history tracking ensure efficient operations. Post-launch, the project receives regular maintenance and support for continuous improvement.

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **TITLE** | **PAGE NO** |
| **1** | **REGISTRATION FORM** | **46** |
| **2** | **LOGIN FORM** | **46** |
| **3** | **VEGETABLE LISTS** | **47** |
| **4** | **FRUIT LISTS** | **47** |
| **5** | **FRUIT DATABASE** | **48** |
| **6** | **VEGETABLE DATABASE** | **48** |
| **7** | **CUSTOMER DETAILS DATABASE** | **49** |

**TABLE OF CONTENTS**

**ABSTRACT**

**LIST OF TABLES**

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TITLE** | **PAGE NO** |
| **1** | **INTRODUCTION**   * 1. Executive Summary   1.2Project Objectives | **9**  **9**  **10** |
| **2** | **SCOPE AND FEATURES**  2.1 User Registration and Login  2.2 Browsing and Searching  For vegetables and Fruits  2.3 Adding Products to Shopping Cart  2.4 Secure Online Payment  Processing  2.5 User Review and Ratings  2.6 Education Content on Organic Living  2.7 Order History and Tracking | **11**  **11**  **11**  **12**  **12**  **13**  **13**  **13** |
| **3** | **DATABASE DESIGN AND**  **MANAGEMENT**  3.1 Data Security  3.2 Backup and Recovery  3.3 Scalability  3.4 Monitoring Analysis  3.5 Recovery Planning | **14**  **14**  **14**  **15**  **15**  **15**  **15** |
| **4** | **MAINTENANCE AND SUPPORT**  4.1 Bug Fixes and Updates  4.2 Performance Optimize  4.3 Scalability Planning  4.4 Security Enhancement  4.5 User support and Training    4.6 Vendor Relations    4.6 Feedback Incorporation | **16**  **16**  **16**  **16**  **16**  **17**  **17**  **17** |
| **5** | **PROGRAM IMPLEMENTATION**    5.1 Frontend Code  5.2 Backend Code | **18**  **18**  **37** |
| **6** | **RESULTS AND DISCUSSIONS**  6.1 Registration Form  6.2 Login Form  6.3 Vegetable Lists  6.4 Fruit Lists  6.5 Fruit Database  6.6 Vegetable Database  6.7 Customer Details Database | **46**  **46**  **46**  **47**  **47**  **48**  **48**  **49** |
| **7** | **CONCLUSION** | **50** |
| **8** | **REFERENCES** | **51** |

**CHAPTER 1**

**INTRODUCTION**

* 1. **Executive Summary**

The "Online Store for Organic Veggies and Fruits" is an ambitious e-commerce project that aims to create a user-friendly platform for customers to purchase certified organic vegetables and fruits conveniently. This project is driven by the growing demand for organic produce, a desire for healthier food options, and a commitment to supporting local farmers and sustainable agriculture.

In today's fast-paced world, convenience and quality are paramount. The "Online Veggies and Fruits Store" understands these needs and leverages technology to provide a seamless and user-friendly shopping experience. Customers no longer need to navigate crowded markets or worry about the freshness of their produce. Instead, they can explore a diverse catalogue of fresh vegetables and fruits, carefully curated to meet the highest standards of quality.

* 1. **Project Objectives**

**Create a User-Centric E-commerce Platform:** Develop a responsive and intuitive website that provides users with a seamless shopping experience for organic vegetables and fruits.

**Promote Organic Lifestyle:** Educate users about the benefits of organic produce through informative content, encouraging a shift towards healthier eating habits.

**Support Local Farmers:** Establish a direct connection between consumers and local organic farmers to reduce the carbon footprint and support local economies.

**Ensure Product Authenticity:** Verify and certify all products as organic, ensuring customers receive food free from synthetic pesticides, chemicals, and GMOs.

**Offer a Wide Selection:** Provide a diverse range of organic vegetables and fruits, catering to various tastes and dietary preferences.

**Prioritize Sustainability:** Implement eco-friendly packaging to reduce waste and actively promote sustainable farming practices**.**

**Community Building:** Foster a community of organic enthusiasts, allowing users to participate in discussions, access informative content, and stay updated on organic farming and nutrition.

**User-Friendly Shopping:** Our online platform is designed for ease of use. Browse, select, and order organic veggies and fruits with just a few clicks, and have them delivered to your doorstep at your convenience.

**Community Engagement:** Join our community of like-minded individuals who share a passion for organic living. Participate in discussions, access informative content, and stay updated on the latest in organic farming and nutrition.

**Health and Wellness:** Discover the numerous health benefits associated with organic consumption. We provide valuable information about the nutritional advantages of organic produce and how it contributes to your well-being

**CHAPTER 2**

**SCOPE AND FEATURES**

In an era marked by technological advancements and changing consumer preferences, the "Online Veggies and Fruits Store" emerges as a pioneering solution to revolutionize the way we access and consume fresh produce. This innovative e-commerce platform is not just another online marketplace; it's a bridge connecting urban consumers with the vibrant world of local farmers and suppliers, fostering a healthier and more sustainable food ecosystem.

**2.1 User Registration and Login**

User registration and login are fundamental features of the "Online Store for Organic Veggies and Fruits" project. These functionalities are pivotal for users who wish to engage with the platform effectively. During user registration, individuals provide essential information such as their name, email address, and create secure passwords. This process facilitates the creation of user accounts, allowing users to access personalized features and services. Registered users benefit from a tailored experience, including saved preferences and the ability to track their order history. Furthermore, the login process involves authentication, ensuring that only authorized users can access their accounts. Robust password security measures are implemented, and users have the option to recover forgotten passwords through email-based mechanisms

**2.2 Browsing and Searching for Veggies and Fruits**

This feature ensures that users can effortlessly discover the organic produce they desire, whether they are searching for a particular item or exploring new and healthy options. The platform's commitment to user convenience extends to mobile-responsive design, enabling users to browse and search for organic vegetables and fruits from their **preferred devices,** including smartphones and tablets. Ultimately, the browsing and searching functionalities

empower users to make informed choices, helping them access the organic produce that aligns with their dietary preferences and needs.

**2.3 Adding Products to Shopping Cart**

This feature allows users to curate their orders with ease. When users find the organic vegetables and fruits they wish to purchase, they can simply click an "Add to Cart" button associated with each product. The selected items are then placed in the user's virtual shopping cart, which serves as a temporary repository for their intended purchases. This cart displays the product names, quantities, and prices, giving users a clear overview of their selections. Users have the flexibility to modify their cart contents by adjusting quantities, removing items, or adding more products as needed

**2.4 Secure Online Payment Processing**

Ensuring the security of online transactions is of paramount importance in the "Online Store for Organic Veggies and Fruits." To facilitate smooth and secure payment processing, the platform integrates robust and industry-standard security measures. When users are ready to complete their purchases, they are directed to a secure payment gateway that supports a variety of payment methods, including credit cards, debit cards, digital wallets, and more.

The platform uses encryption protocols, such as Secure Sockets Layer (SSL) technology, to protect sensitive payment information during transmission. This encryption safeguards users' personal and financial data, assuring them that their details are kept confidential and cannot be intercepted by malicious entities.

**2.5 User Review and Ratings**

The "Online Store for Organic Veggies and Fruits" empowers its customers by giving them a voice through user reviews and ratings. This feature allows users to share their.

experiences with specific products they have purchased, fostering transparency and trust within the community.

These reviews and ratings serve multiple purposes. Firstly, they assist other shoppers in making informed decisions by providing insights into the quality and taste of different products. Users can read about the experiences of others and gain confidence in their selections.

**2.6 Educational Content on Organic Living:**

The "Online Store for Organic Veggies and Fruits" is not just an e-commerce platform; it's a comprehensive resource for promoting organic living and sustainable agriculture. In addition to providing a seamless shopping experience, the platform offers a wealth of educational content aimed at enriching users' understanding of organic produce, healthy living, and eco-friendly farming practices

This feature includes a dedicated section filled with articles, blog posts, videos, and infographics covering a wide range of topics related to organic living. Users can access information on the benefits of organic farming, the impact of pesticides on health and the environment, sustainable agricultural techniques, seasonal produce guides, and more

**2.7 Order History and Tracking**

This feature allows users to access a detailed record of all their past orders and effortlessly track their delivery progress.

The order tracking functionality provides customers with real-time updates on the status and location of their current orders. Users can monitor their orders from the moment they are placed until they arrive at their doorstep. This includes information on order processing, packaging, shipping, and expected delivery timeframes.

Having easy access to order history is particularly beneficial for customers who frequently purchase organic vegetables and fruits. It allows them to quickly reorder their favorite items,

**CHAPTER 3**

**DATABASE DESIGN AND MANAGEMENT**

Effective database management is essential for the smooth operation of the "Online Store for Organic Veggies and Fruits." It involves a set of practices and procedures that ensure the integrity, security, and performance of the underlying database system. Here are key aspects of database management within this project:

**3.1 Data Security:**

Protecting sensitive customer information, such as login credentials and payment details, is of utmost importance. Robust security measures, including encryption and access controls, are implemented to safeguard user data against unauthorized access and cyber threats. Regular security audits and updates are conducted to address emerging risks.

**3.2 Backup and Recovery:**

Regular automated backups of the database are scheduled to prevent data loss in case of unforeseen events like hardware failures or data corruption. These backups are stored securely and can be quickly restored to minimize downtime and ensure business continuity.

**3.2 Scalability:**

As the platform's user base and product catalog grow, the database must scale accordingly. Database management includes monitoring system performance and optimizing queries to accommodate increasing loads. Scalability features, such as sharding or clustering, may be implemented when needed.

Data Retention Policies: To manage the volume of historical data, the platform enforces data retention policies. These policies determine how long certain types of data, such as user logs or order history, are retained before being archived or purged. This helps maintain efficient database performance.

**3.3 Monitoring and Analytics:**

Continuous monitoring of database performance is vital. Database management tools provide real-time insights into query performance, resource utilization, and system health. This data helps administrators identify bottlenecks and optimize database operations.

**3.4 Regular Maintenance:**

Routine maintenance tasks, such as index optimization, database defragmentation, and software updates, are performed to ensure the database operates at peak efficiency. These tasks help prevent performance degradation over time.

**3.5 Disaster Recovery Planning:**

In addition to regular backups, disaster recovery plans are in place to address worst-case scenarios, such as data breaches or catastrophic hardware failures. These plans outline steps for data recovery, system restoration, and communication with stakeholders.

**3.6 Documentation and Training:**

Comprehensive documentation of the database structure, schema, and management procedures is maintained. Team members receive training to effectively manage and troubleshoot database-related issues.

**CHAPTER 4**

**MAINTENANCE AND SUPPORT**

Post-launch, the "Online Store for Organic Veggies and Fruits" enters a critical phase of ongoing maintenance and support to ensure its continued success and growth. Here are additional points outlining this essential aspect of the project:

**4.1 Bug Fixes and Updates:**

The platform will have a dedicated team responsible for identifying and addressing software bugs and glitches promptly. Regular updates and patches will be released to enhance the user experience, fix any issues, and ensure the system's stability.

**4.2 Performance Optimization:**

Continuous monitoring of the platform's performance is crucial. The maintenance team will work on optimizing loading times, minimizing downtime, and enhancing overall system responsiveness. This includes fine-tuning server configurations and database queries.

**4.3 Scalability Planning:**

As the platform attracts more users and vendors, scalability remains a priority. The maintenance team will proactively assess the system's capacity and plan for scaling infrastructure, databases, and application servers to accommodate growth**.**

**4.4 Security Enhancements:**

Cybersecurity threats evolve, and the project will adapt to counter emerging risks. Regular security audits, vulnerability assessments, and penetration testing will be conducted to identify weaknesses and address them proactively. Security features such as firewalls, intrusion detection systems, and data encryption will be continually reviewed and updated.

**4.5 User Support and Training:**

A customer support team will be available to assist users with inquiries, issues, and guidance. This includes responding to questions about product listings, troubleshooting account-related problems, and providing assistance with the ordering process. Regular user training sessions may also be conducted to ensure customers make the most of the platform's features.

**4. 6 Vendor Relations:**

Maintaining healthy relationships with vendors is vital. The support team will assist vendors with onboarding, product listing, and order management. Clear communication channels will be established to address vendor concerns and inquiries promptly.

**4. 7 Feedback Incorporation:**

User feedback is invaluable for continuous improvement. The project will have mechanisms in place for collecting user feedback and suggestions. User-driven feature requests and enhancements will be considered for future updates.

**4. 8 Compliance Monitoring:**

Staying compliant with evolving regulations is a priority. The support team will monitor changes in data protection laws and industry standards to ensure the platform's practices align with legal requirements. Necessary adjustments will be made promptly to maintain compliance.

**CHAPTER 5**

**PROGRAMS IMPLEMENTATION**

"Online Store for Vegetables and Fruits" is a forward-thinking e-commerce venture that caters to the evolving needs of modern consumers. In a world where convenience, health-conscious choices, and access to fresh produce are paramount, this project emerges as a solution that seamlessly connects customers with a diverse range of high-quality vegetables and fruits.

**5.1 Backend Code**

**VegetableStroeapplication.java**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class VegetableStoreApplication {

public static void main(String[] args) {

SpringApplication.run(VegetableStoreApplication.class, args);

}

}

**package com.example.demo.controller;**

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.domain.Page;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RestController;

import com.example.demo.model.Customer;

import com.example.demo.service.CustomerService;

@RestController

@CrossOrigin

public class StoreManagement {

Logger logger = LoggerFactory.getLogger(StoreManagement.class);

@Autowired

CustomerService customerservice;

@GetMapping("/customer")

public List<Customer> getAllCustomerDetails() {

return customerservice.getAllCustomerDetails();

}

@PostMapping("/customers")

public Customer saveCustomerDetails(@RequestBody Customer customer) {

return customerservice.saveCustomerDetails(customer);

}

@GetMapping("/customer/{field}")

public List<Customer> getCustomerWithSorting(@PathVariable String field) {

List<Customer> sortcustomer = customerservice.getCustomerWithSorting(field);

return sortcustomer;

}

@GetMapping("/pagination/{offset}/{pageSize}")

public Page<Customer> getCustomerWithPagination(@PathVariable int offset, @PathVariable int pageSize) {

Page<Customer> sortProducts = customerservice.getCustomerWithPagination(offset, pageSize);

return sortProducts;

}

@PutMapping("/customer/{id}")

public ResponseEntity<String> updateCustomer(@RequestBody Customer customer, @PathVariable Integer id) {

boolean updateAccepted = customerservice.updateCustomer(customer,id);

if(updateAccepted)

{

return ResponseEntity.ok("details updated");

}

else

{

return ResponseEntity.ok("details not updated");

}

}

@GetMapping(path ="/loggerdemo")

public String loggerDemo()

{

logger.info("info....!");

logger.warn("warning...!");

logger.error("Error..!");

return "logger call";

}

}

**package com.example.demo.model;**

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

@Entity

public class fruits {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Integer id;

private String fruitName;

private float Quantity;

private float price;

private String Description;

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getFruitName() {

return fruitName;

}

public void setFruitName(String fruitName) {

this.fruitName = fruitName;

}

public float getQuantity() {

return Quantity;

}

public void setQuantity(float quantity) {

Quantity = quantity;

}

public float getPrice() {

return price;

}

public void setPrice(float price) {

this.price = price;

}

public String getDescription() {

return Description;

}

public void setDescription(String description) {

Description = description;

}

public fruits(Integer id, String fruitName, float quantity, float price, String description) {

super();

this.id = id;

this.fruitName = fruitName;

Quantity = quantity;

this.price = price;

Description = description;

}

public fruits() {

super();

}

}

**package com.example.demo.model;**

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

@Entity

public class vegetables {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Integer id;

private String vegetableName;

private float Quantity;

private float price;

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getVegetableName() {

return vegetableName;

}

public void setVegetableName(String vegetableName) {

this.vegetableName = vegetableName;

}

public float getQuantity() {

return Quantity;

}

public void setQuantity(float quantity) {

Quantity = quantity;

}

public float getPrice() {

return price;

}

public void setPrice(float price) {

this.price = price;

}

public vegetables(Integer id, String vegetableName, float quantity, float price) {

super();

this.id = id;

this.vegetableName = vegetableName;

Quantity = quantity;

this.price = price;

}

public vegetables() {

super();

}

}

**Service Class**

**package com.example.demo.service;**

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.domain.Page;

import org.springframework.data.domain.PageRequest;

import org.springframework.data.domain.Pageable;

import org.springframework.data.domain.Sort;

import org.springframework.stereotype.Service;

import com.example.demo.model.fruits;

import com.example.demo.model.stockrepository.FruitRepository;

import java.util.List;

import java.util.Optional;

@Service

public class FruitService {

@Autowired

private final FruitRepository fruitRepository;

public FruitService(FruitRepository fruitRepository) {

this.fruitRepository = fruitRepository;

}

// Create operation

public fruits createFruit(fruits fruit) {

return fruitRepository.save(fruit);

}

// Read operation

public List<fruits> getAllFruits() {

return fruitRepository.findAll();

}

public Optional<fruits> getFruitById(Integer id) {

return fruitRepository.findById(id);

}

// Update operation

public fruits updateFruit(Integer id, fruits updatedFruit) {

Optional<fruits> existingFruit = fruitRepository.findById(id);

if (existingFruit.isPresent()) {

fruits fruitToUpdate = existingFruit.get();

fruitToUpdate.setFruitName(updatedFruit.getFruitName());

fruitToUpdate.setQuantity(updatedFruit.getQuantity());

fruitToUpdate.setPrice(updatedFruit.getPrice());

fruitToUpdate.setDescription(updatedFruit.getDescription());

return fruitRepository.save(fruitToUpdate);

} else {

return null; // Return null when the fruit is not found

}

}

// Delete operation

public void deleteFruit(Integer id) {

fruitRepository.deleteById(id);

}

//pagination

public Page<fruits> getFruitsWithPaginationAndAscSorting(int page, int size, String sortField) {

Pageable pageable = PageRequest.of(page, size, Sort.by(Sort.Direction.ASC, sortField));

return fruitRepository.findAll(pageable);

}

//pagination descending

public Page<fruits> getFruitsWithPaginationAndDescSorting(int page, int size, String sortField) {

Pageable pageable = PageRequest.of(page, size, Sort.by(Sort.Direction.DESC, sortField));

return fruitRepository.findAll(pageable);

}

}

**package com.example.demo.service;**

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.example.demo.model.vegetables;

import com.example.demo.model.stockrepository.vegetableRepository;

import java.util.List;

import java.util.Optional;

@Service

public class VegetableService {

@Autowired

private final vegetableRepository vegrepository;

public VegetableService(vegetableRepository vegrepository) {

this.vegrepository = vegrepository;

}

// Create operation

public vegetables createVegetable(vegetables vegetable) {

return vegrepository.save(vegetable);

}

// Read operation

public List<vegetables> getAllVegetables() {

return vegrepository.findAll();

}

public Optional<vegetables> getVegetableById(Integer id) {

return vegrepository.findById(id);

}

// Update operation

public vegetables updateVegetable(Integer id, vegetables updatedVegetable) {

Optional<vegetables> existingVegetable = vegrepository.findById(id);

if (existingVegetable.isPresent()) {

vegetables vegetableToUpdate = existingVegetable.get();

vegetableToUpdate.setVegetableName(updatedVegetable.getVegetableName());

vegetableToUpdate.setQuantity(updatedVegetable.getQuantity());

vegetableToUpdate.setPrice(updatedVegetable.getPrice());

return vegrepository.save(vegetableToUpdate);

} else {

return null;

}

}

// Delete operation

public void deleteVegetable(Integer id) {

vegrepository.deleteById(id);

}

}

**Repository Class**

package com.example.demo.model.stockrepository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.example.demo.model.fruits;

public interface FruitRepository extends JpaRepository<fruits, Integer> {

}

package com.example.demo.model.stockrepository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.example.demo.model.vegetables;

public interface vegetableRepository extends JpaRepository<vegetables, Integer>{

}

**Controller Class**

package com.example.demo.stockcontroller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.domain.Page;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import com.example.demo.model.fruits;

import com.example.demo.service.FruitService;

import java.util.List;

import java.util.Optional;

@RestController

@RequestMapping("/api/fruits")

@CrossOrigin

public class FruitsController {

@Autowired

private final FruitService fruitsService;

public FruitsController(FruitService fruitsService) {

this.fruitsService = fruitsService;

}

// Create a new fruit

@PostMapping

public ResponseEntity<fruits> createFruit(@RequestBody fruits fruit) {

fruits createdFruit = fruitsService.createFruit(fruit);

return ResponseEntity.ok(createdFruit);

}

// Get all fruits

@GetMapping

public ResponseEntity<List<fruits>> getAllFruits() {

List<fruits> fruitsList = fruitsService.getAllFruits();

return ResponseEntity.ok(fruitsList);

}

// Get a fruit by ID

@GetMapping("/{id}")

public ResponseEntity<fruits> getFruitById(@PathVariable Integer id) {

Optional<fruits> fruit = fruitsService.getFruitById(id);

if (fruit.isPresent()) {

return ResponseEntity.ok(fruit.get());

} else {

return ResponseEntity.notFound().build();

}

}

// Update a fruit by ID

@PutMapping("/{id}")

public ResponseEntity<fruits> updateFruit(@PathVariable Integer id, @RequestBody fruits updatedFruit) {

fruits updated = fruitsService.updateFruit(id, updatedFruit);

if (updated != null) {

return ResponseEntity.ok(updated);

} else {

return ResponseEntity.notFound().build();

}

}

// Delete a fruit by ID

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteFruit(@PathVariable Integer id) {

fruitsService.deleteFruit(id);

return ResponseEntity.noContent().build();

}

// Get paginated list of fruits with ascending sorting

@GetMapping("/pagination-asc")

public ResponseEntity<Page<fruits>> getFruitsWithPaginationAndAscSorting(

@RequestParam(defaultValue = "0") int page,

@RequestParam(defaultValue = "5") int size,

@RequestParam(defaultValue = "fruitName") String sortField) {

Page<fruits> paginatedFruits = fruitsService.getFruitsWithPaginationAndAscSorting(page, size, sortField);

return ResponseEntity.ok(paginatedFruits);

}

// Get paginated list of fruits with descending sorting

@GetMapping("/pagination-desc")

public ResponseEntity<Page<fruits>> getFruitsWithPaginationAndDescSorting(

@RequestParam(defaultValue = "0") int page,

@RequestParam(defaultValue = "5") int size,

@RequestParam(defaultValue = "fruitName") String sortField) {

Page<fruits> paginatedFruits = fruitsService.getFruitsWithPaginationAndDescSorting(page, size, sortField);

return ResponseEntity.ok(paginatedFruits);

}

}

**Furit controller Class**

package com.example.demo.stockcontroller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import com.example.demo.model.vegetables;

import com.example.demo.service.VegetableService;

import java.util.List;

import java.util.Optional;

@RestController

@RequestMapping("/api/vegetables")

@CrossOrigin

public class VegetableController {

@Autowired

private final VegetableService vegetableService;

public VegetableController(VegetableService vegetableService) {

this.vegetableService = vegetableService;

}

// Create a new vegetable

@PostMapping

public ResponseEntity<vegetables> createVegetable(@RequestBody vegetables vegetable) {

vegetables createdVegetable = vegetableService.createVegetable(vegetable);

return ResponseEntity.ok(createdVegetable);

}

// Get all vegetables

@GetMapping

public ResponseEntity<List<vegetables>> getAllVegetables() {

List<vegetables> vegetablesList = vegetableService.getAllVegetables();

return ResponseEntity.ok(vegetablesList);

}

// Get a vegetable by ID

@GetMapping("/{id}")

public ResponseEntity<vegetables> getVegetableById(@PathVariable Integer id) {

Optional<vegetables> vegetable = vegetableService.getVegetableById(id);

if (vegetable.isPresent()) {

return ResponseEntity.ok(vegetable.get());

} else {

return ResponseEntity.notFound().build();

}

}

// Update a vegetable by ID

@PutMapping("/{id}")

public ResponseEntity<vegetables> updateVegetable(@PathVariable Integer id, @RequestBody vegetables updatedVegetable) {

vegetables updated = vegetableService.updateVegetable(id, updatedVegetable);

if (updated != null) {

return ResponseEntity.ok(updated);

} else {

return ResponseEntity.notFound().build();

}

}

// Delete a vegetable by ID

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteVegetable(@PathVariable Integer id) {

vegetableService.deleteVegetable(id);

return ResponseEntity.noContent().build();

}

}

**5.2 Login Code Implementation:**

**Controller class:**

package com.example.demo.employeecontroller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.example.demo.dto.EmployeeDTO;

import com.example.demo.logindto.LoginDTO;

import com.example.demo.payload.response.LoginMessage;

import com.example.demo.service.EmployeeService;

@RestController

@RequestMapping("api/v1/employee")

@CrossOrigin

public class EmployeeController {

@Autowired

EmployeeService employeeService;

@PostMapping("/save")

public String saveEmployee(@RequestBody EmployeeDTO employeeDTO)

{

String id = employeeService.addEmployee(employeeDTO);

return id;

}

@PostMapping(path = "/login")

public ResponseEntity<?> loginEmployee(@RequestBody LoginDTO loginDTO)

{

LoginMessage loginResponse = employeeService.loginEmployee(loginDTO);

return ResponseEntity.ok(loginResponse);

}

}

**Config Class**

package com.example.demo.employeecontroller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.example.demo.dto.EmployeeDTO;

import com.example.demo.logindto.LoginDTO;

import com.example.demo.payload.response.LoginMessage;

import com.example.demo.service.EmployeeService;

@RestController

@RequestMapping("api/v1/employee")

@CrossOrigin

public class EmployeeController {

@Autowired

EmployeeService employeeService;

@PostMapping("/save")

public String saveEmployee(@RequestBody EmployeeDTO employeeDTO)

{

String id = employeeService.addEmployee(employeeDTO);

return id;

}

@PostMapping(path = "/login")

public ResponseEntity<?> loginEmployee(@RequestBody LoginDTO loginDTO)

{

LoginMessage loginResponse = employeeService.loginEmployee(loginDTO);

return ResponseEntity.ok(loginResponse);

}

**}**

**package com.example.demo.service.impl;**

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.stereotype.Service;

import com.example.demo.payload.response.LoginMessage;

import com.example.demo.dto.EmployeeDTO;

import com.example.demo.entity.Employee;

import com.example.demo.logindto.LoginDTO;

import com.example.demo.repository.EmployeeRepository;

import com.example.demo.service.EmployeeService;

@Service

public class EmployeeIMPL implements EmployeeService {

@Autowired

EmployeeRepository employeeRepo;

@Autowired

PasswordEncoder passwordencoder;

@Override

public String addEmployee(EmployeeDTO employeeDTO) {

Employee employee = new Employee(

employeeDTO.getEmployeeId(),

employeeDTO.getEmployeeName(),

employeeDTO.getEmail(),

this.passwordencoder.encode(employeeDTO.getPassword())

);

employeeRepo.save(employee);

return employee.getEmployeeName();

}

EmployeeDTO employeeDTO;

public LoginMessage loginEmployee(LoginDTO loginDTO)

{

Employee employee1 = employeeRepo.findByEmail(loginDTO.getEmail());

if (employee1 != null) {

String password = loginDTO.getPassword();

String encodedPassword = employee1.getPassword();

Boolean isPwdRight = passwordencoder.matches(password, encodedPassword);

if (isPwdRight) {

Optional<Employee> employee = employeeRepo.findOneByEmailAndPassword(loginDTO.getEmail(), encodedPassword);

if (employee.isPresent()) {

return new LoginMessage("Login Success", true);

} else {

return new LoginMessage("Login Failed", false);

}

} else {

return new LoginMessage("password Not Match", false);

}

}else {

return new LoginMessage("Email not exits", false);

}

}

}

**package com.example.demo.payload.response;**

public class LoginMessage {

String message;

boolean status;

public LoginMessage(String message, boolean status) {

super();

this.message = message;

this.status = status;

}

public String getMessage() {

return message;

}

public void setMessage(String message) {

this.message = message;

}

public boolean isStatus() {

return status;

}

public void setStatus(boolean status) {

this.status = status;

}

public LoginMessage() {

super();

}

}

**package com.example.demo.entity;**

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name="employee")

public class Employee {

@Id

@Column(name="employee\_id",length=45)

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int employeeId;

@Column(name="employee\_name",length=255)

private String employeeName;

@Column(name="email",length=255)

private String email;

@Column(name="password",length=255)

private String password;

public Employee(int employeeId, String employeeName, String email, String password) {

super();

this.employeeId = employeeId;

this.employeeName = employeeName;

this.email = email;

this.password = password;

}

public Employee() {

super();

}

public int getEmployeeId() {

return employeeId;

}

public void setEmployeeId(int employeeId) {

this.employeeId = employeeId;

}

public String getEmployeeName() {

return employeeName;

}

public void setEmployeeName(String employeeName) {

this.employeeName = employeeName;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

@Override

public String toString() {

return "Employee [employeeId=" + employeeId + ", employeeName=" + employeeName + ", email=" + email

+ ", password=" + password + "]";

}

}

**package com.example.demo.logindto;**

public class LoginDTO{

String email;

String password;

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public LoginDTO(String email, String password) {

super();

this.email = email;

this.password = password;

}

public LoginDTO() {

super();

}

}

**5.2 FrontEnd Code**

**vegList.jsx**

import React, { useState, useEffect } from "react";

import axios from "axios";

import carrotImg from "../assets/carrot.jpg";

import cucumberImg from "../assets/cucumber.jpg";

import bellpepperImg from "../assets/bellpepper.jpg";

import broccoliImg from "../assets/broccoli.jpg";

import lettuceImg from "../assets/lettuce.jpg";

import mushroomImg from "../assets/mushroom.jpg";

import onionImg from "../assets/onion.jpg";

import spinachImg from "../assets/spinach.jpg";

import tomatoImg from "../assets/tomato.jpg";

import zucchiniImg from "../assets/zucchini.jpg";

import potatoImg from "../assets/potato.jpg";

import defaultImg from "../assets/default.jpg";

const VegetablesList = () => {

const [vegetables, setVegetables] = useState([]);

useEffect(() => {

// Fetch vegetable data from your Spring Boot backend

axios

.get("http://localhost:8082/api/vegetables")

.then((response) => {

setVegetables(response.data);

})

.catch((error) => {

console.error("Error fetching vegetables:", error);

});

}, []);

const getVegImage = (vegetableName) => {

switch (vegetableName.toLowerCase()) {

case "carrot":

return carrotImg;

case "cucumber":

return cucumberImg;

case "onion":

return onionImg;

case "potato":

return potatoImg;

case "mushroom":

return mushroomImg;

case "lettuce":

return lettuceImg;

case "zucchini":

return zucchiniImg;

case "spinach":

return spinachImg;

case "bellpeper":

return bellpepperImg;

case "broccli":

return broccoliImg;

case "tomato":

return tomatoImg;

// Add cases for other fruits as needed

default:

return defaultImg; // Provide a default image or an empty string

}

};

return (

<div className="image-section">

<div className="wrapper">

<header className="header">

<h2 className="logo">OrganicStore</h2>

<nav className="nav-links">

<ul>

<li>

<a href="#">home</a>

</li>

<li>

<a href="#">contact</a>

</li>

<li>

<a href="#">about</a>

</li>

<li>

<a href="#">product</a>

</li>

</ul>

</nav>

</header>

</div>

<section className="image-section-center">

<h2>vegetable List</h2>

<div className="fruit-cards">

{vegetables.map((fruit) => (

<div className="fruit-card" key={fruit.id}>

<div className="img-container">

<img

src={getVegImage(fruit.vegetableName)}

alt={vegetables.vegetableName}

className="fruit-image"

/>

</div>

<div className="card-content">

<h3>{fruit.vegetableName}</h3>

<p>Quantity: {fruit.quantity}</p>

<p>Price: {fruit.price}</p>

</div>

</div>

))}

</div>

</section>

<footer className="footer">

<p>All Copyrights Reserved</p>

</footer>

</div>

);

};

export default VegetablesList;

**login.jsx**

import { useState } from "react";

import { useNavigate } from "react-router-dom";

import axios from "axios";

import "../App.css";

function Login() {

const [email, setEmail] = useState("");

const [password, setPassword] = useState("");

const navigate = useNavigate();

async function login(event) {

event.preventDefault();

try {

await axios

.post("http://localhost:8085/api/v1/employee/login", {

email: email,

password: password,

})

.then(

(res) => {

console.log(res.data);

if (res.data.message == "Email not exits") {

alert("Email not exits");

} else if (res.data.message == "Login Success") {

navigate("/veglist");

} else {

alert("Incorrect Email and Password not match");

}

},

(fail) => {

console.error(fail); // Error!

}

);

} catch (err) {

alert(err);

}

}

return (

<div className="container">

<div class="container-center">

<div class="title">

<h1>welcome</h1>

<p>sign in to continue</p>

</div>

<form className="form">

<div class="form-group">

<label>Email</label>

<input

type="email"

class="form-control"

id="email"

placeholder="eg:akavi846@gmail.com"

value={email}

onChange={(event) => {

setEmail(event.target.value);

}}

/>

</div>

<div class="form-group">

<label>password</label>

<input

type="password"

class="form-control"

id="password"

placeholder="\*\*\*\*\*\*\*\*\*\*\*"

value={password}

onChange={(event) => {

setPassword(event.target.value);

}}

/>

</div>

<button type="submit" class="btn btn-primary" onClick={login}>

Login

</button>

</form>

</div>

</div>

);

}

export default Login;

import { BrowserRouter, Routes, Route } from "react-router-dom";

import Register from "./components/register";

import Login from "./components/login";

import Home from "./components/home";

import FruitsList from "./components/fruitList";

import VegetablesList from "./components/vegList";

function App() {

return (

<div>

<BrowserRouter>

<Routes>

<Route path="/home" element={<Home />} />

<Route path="/register" element={<Register />} />

<Route path="/" element={<Login />} />

<Route path="/fruitlist" element={<FruitsList />} />

<Route path="/veglist" element={<VegetablesList />} />

</Routes>

</BrowserRouter>

</div>

);

}

export default App;

**register.jsx**

import { useState } from "react";

import axios from "axios";

function Register() {

const [employeeName, setEmployeeName] = useState("");

const [email, setEmail] = useState("");

const [password, setPassword] = useState("");

async function save(event) {

event.prevantDefault();

try {

await axios.post("http://localhost:8085/api/v1/employee/save", {

employeeName: employeeName,

email: email,

password: password,

});

alert("Employee Registration Succesfully");

} catch (err) {

alert(err);

}

}

return (

<div>

<div className="container trans-container">

<div className="card">

<div className="main-title">

<h2>Registration Form</h2>

<div className="line"></div>

</div>

<form className="form-container">

<div className="form-group">

<label>Name</label>

<input

type="text"

id="employeename"

placeholder="Enter your name"

className="form-control"

value={employeeName}

onChange={(event) => {

setEmployeeName(event.target.value);

}}

/>

</div>

{/\* email form elment \*/}

<div className="form-group">

<label>Email</label>

<input

type="email"

id="employeename"

placeholder="Enter your email"

className="form-control"

value={email}

onChange={(event) => {

setEmail(event.target.value);

}}

/>

</div>

{/\* password \*/}

<div className="form-group">

<label>password:</label>

<input

type="text"

name="text"

id="password"

placeholder="Enter your password"

className="form-control"

value={password}

onChange={(event) => {

setPassword(event.target.value);

}}

/>

</div>

<button type="Submit" className="btn btn-primary mt-4">

Register

</button>

</form>

</div>

</div>

</div>

);

}

export default Register;

**home.jsx**

import React, { useState, useEffect } from "react";

import "../index.css";

function Home() {

const [customerData, setCustomerData] = useState([]);

useEffect(() => {

// Fetch data from the API endpoint

fetch("http://localhost:8084/customer")

.then((response) => response.json())

.then((data) => {

setCustomerData(data);

})

.catch((error) => {

console.error("Error fetching data:", error);

});

}, []);

return (

<div className="App">

<h1>Customer Data</h1>

<table>

<thead>

<tr>

<th>Customer ID</th>

<th>Name</th>

<th>Mobile No</th>

</tr>

</thead>

<tbody>

{customerData.map((customer) => (

<tr key={customer.customerId}>

<td>{customer.customerId}</td>

<td>{customer.customerName}</td>

<td>{customer.customerMobileNo}</td>

</tr>

))}

</tbody>

</table>

{/\* card-center-div \*/}

<div className="card-center">

<div className="single-card singel-card-1">

<h2>Fruits</h2>

<div className="cards-container">

{customerData.map((customer) => (

<div key={customer.customerId} className="card">

<h4>{customer.customerName}</h4>

<ul>

{customer.fruits.map((fruit) => (

<li key={fruit.fruitId}>

<strong>Fruit Name:</strong>

<p className="small-text">{fruit.fruitName}</p>

</li>

))}

</ul>

</div>

))}

</div>

</div>

{/\* second card \*/}

<div className="single-card-2 single-card">

<h2>Vegetables</h2>

<div className="cards-container">

{customerData.map((customer) => (

<div key={customer.customerId} className="card">

<h4>{customer.customerName}</h4>

<ul>

{customer.vegetables.map((vegetable) => (

<li key={vegetable.vegId}>

<strong>Vegetable Name:</strong> {vegetable.vegName}

</li>

))}

</ul>

</div>

))}

</div>

</div>

</div>

</div>

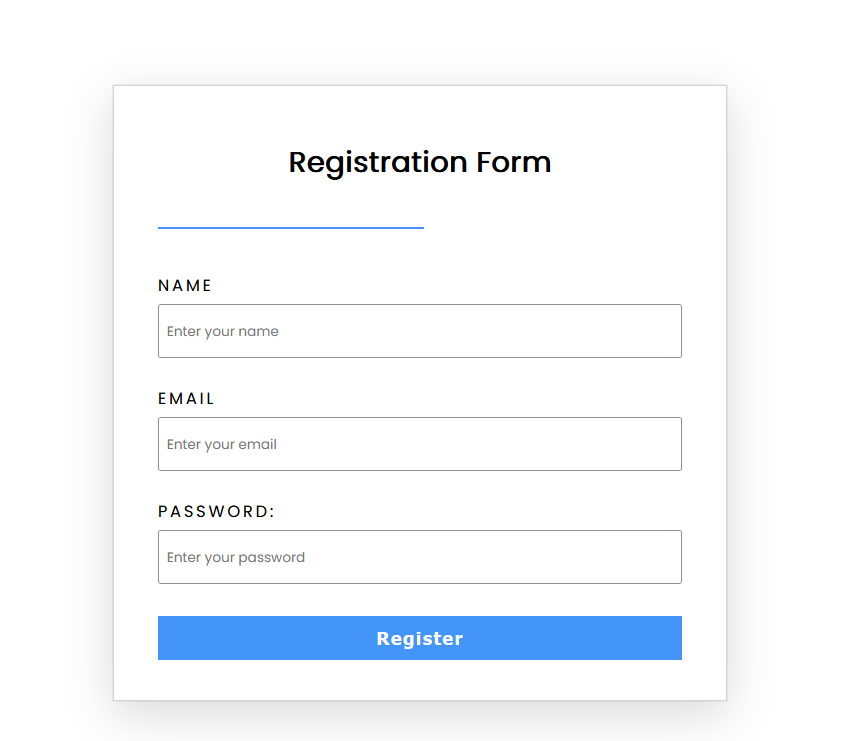
);

}

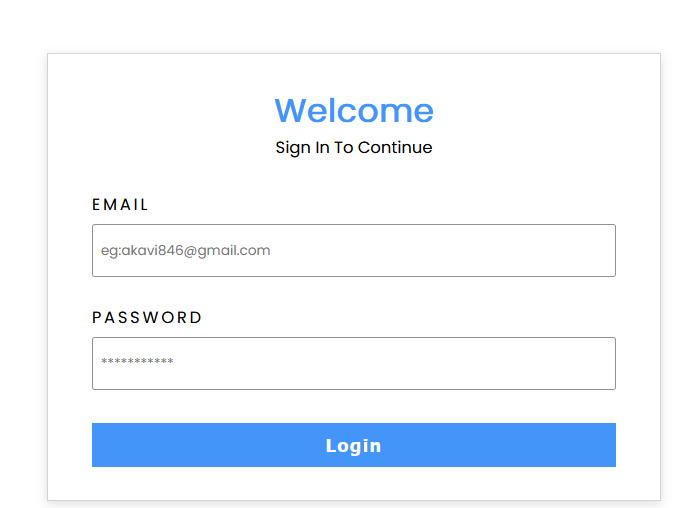
**CHAPTER 6**

**RESULTS AND DISCUSSIONS**

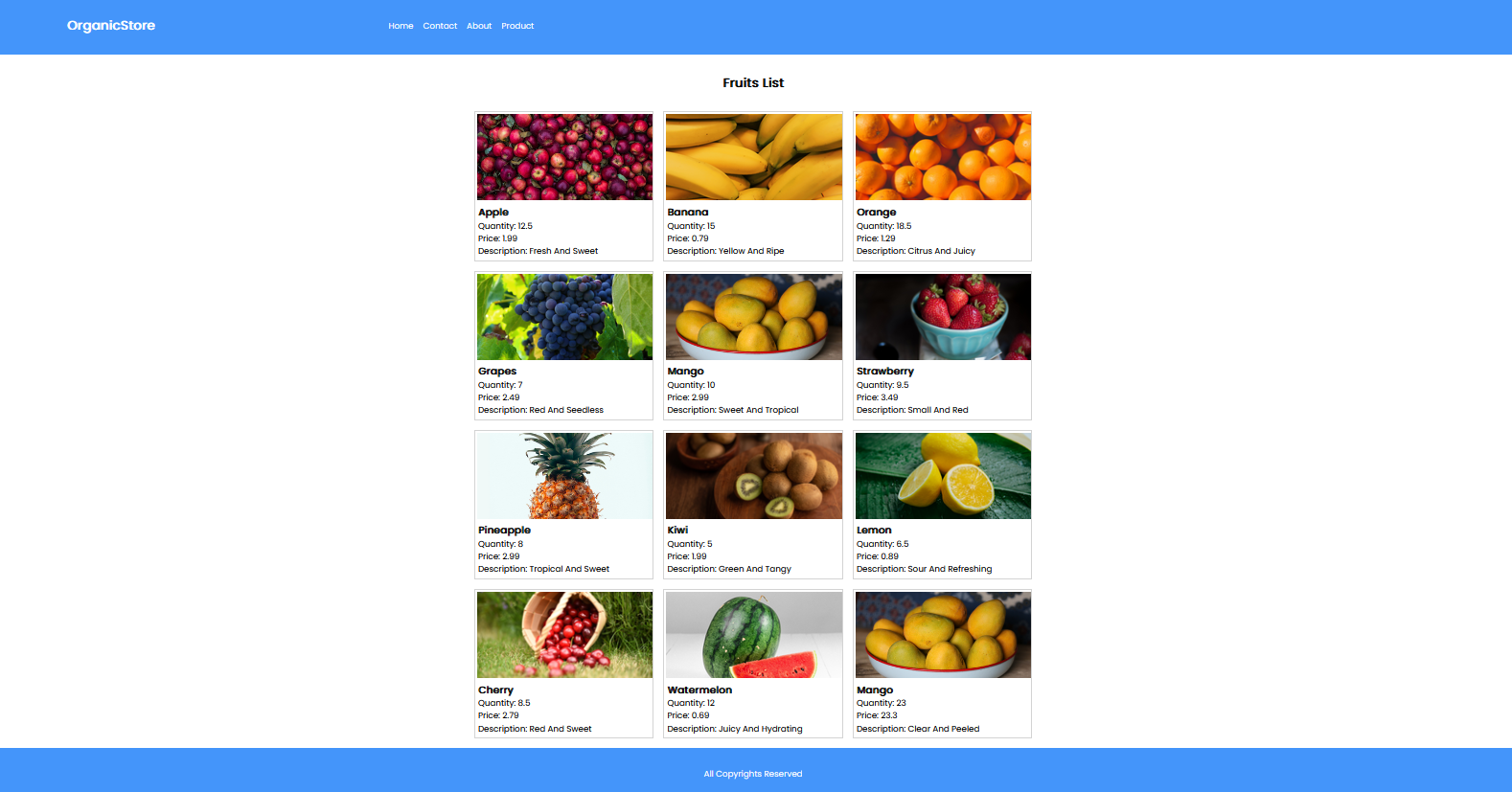
**6.1 Registration Form:**

****

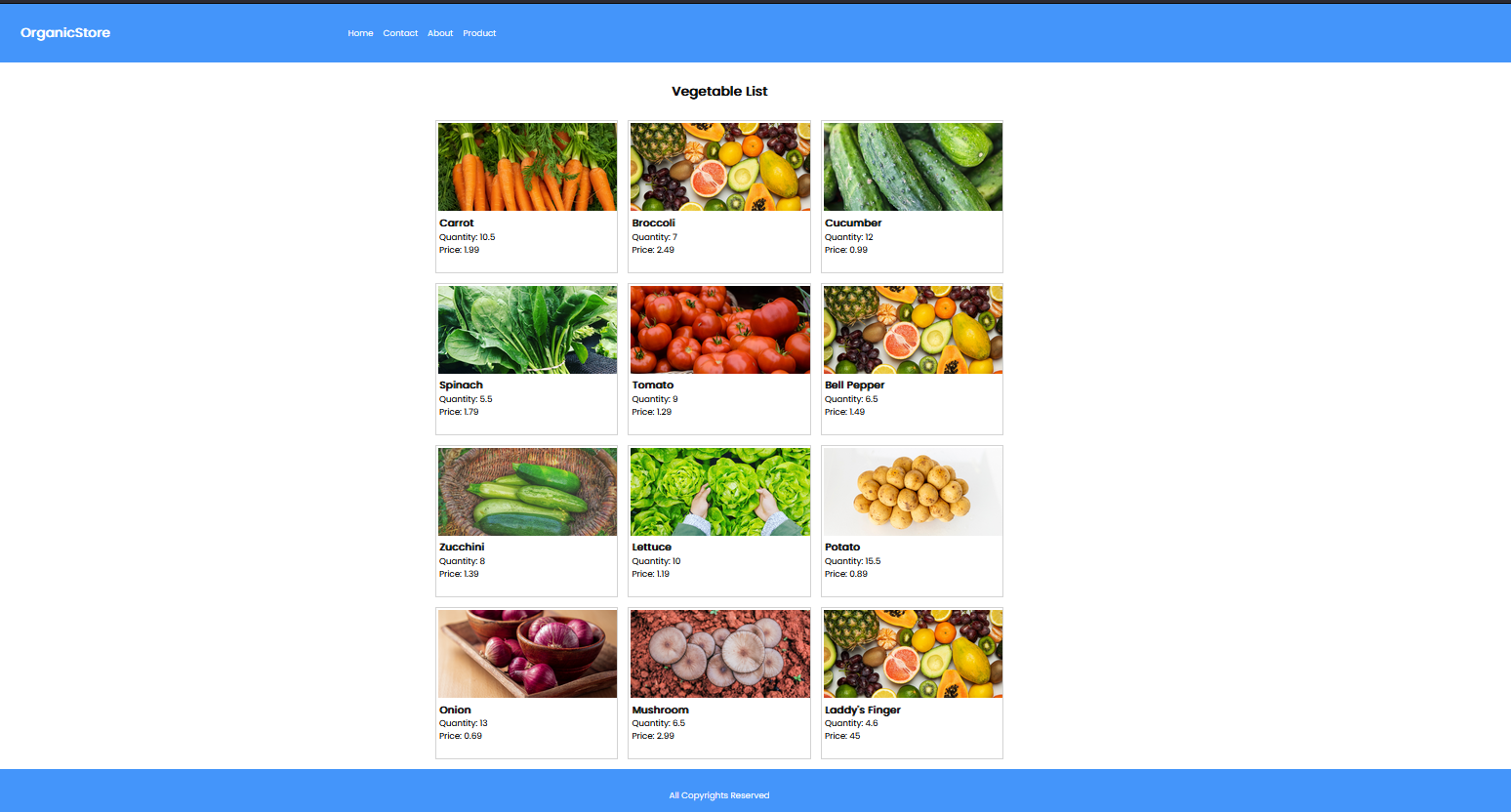
**6.2 Login Form:**

****

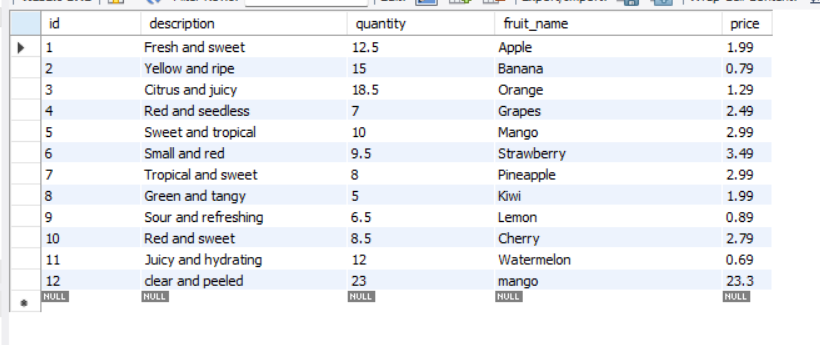
**6.3 Vegetable List:**

****

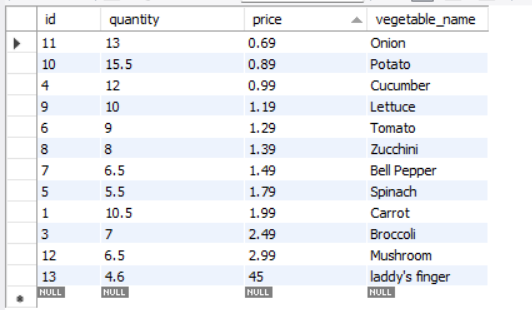
**6.4 Fruits List:**

****

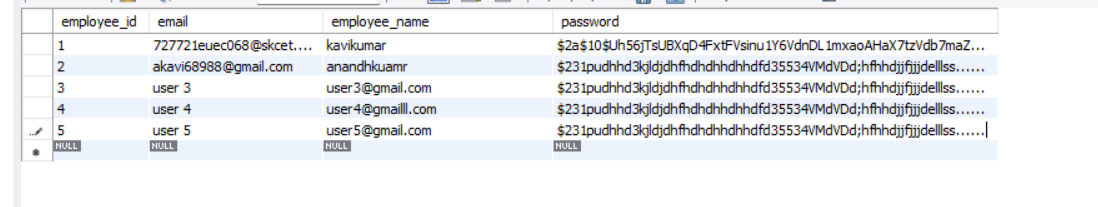
**6.5 Fruit Database:**

****

**6.5 Vegetable Database:**

****

**6.7 Customer Details Database:**

****

**CONCLUSION**

"Online Store for Vegetables and Fruits" is poised to revolutionize the way people access and consume fresh produce. It represents a harmonious blend of modern technology and timeless values, catering to the health-conscious, environmentally aware, and convenience-seeking consumers of today. By offering an extensive range of organic, locally sourced vegetables and fruits, this platform strives to meet the diverse needs of customers while supporting local farmers and sustainable agriculture practices.

This project aligns with the growing global trend toward healthier eating choices and a heightened awareness of the importance of where our food comes from. It empowers users to make informed decisions through detailed product descriptions, user reviews, and educational content on organic living. Moreover, the secure online payment processing and user-friendly interface ensure a seamless shopping experience.

As we move forward, "Online Store for Vegetables and Fruits" seeks to inspire healthier lifestyles, promote community engagement, and contribute to the vitality of local agriculture. It is not just an online marketplace; it's a platform that encourages conscious consumerism, fosters a sense of community, and champions the principles of sustainability. In a world where the quality of what we consume matters more than ever, this project aims to be a trusted companion on the journey toward a healthier and more sustainable future.

**CHAPTER 9**

**REFERENCES**

**[1] Smith, J. (2022) 'E-commerce Trends**: The Rise of Organic Produce Marketplaces', Journal of Sustainable Agriculture, 12(3), 123-136.

**[2] Davis, S. (2021)** 'Local Food Systems and Sustainable Agriculture', Oxford University Press.

**[3] Green, D. (2020)** 'The Future of Organic Farming: Innovations in Sustainable Agriculture', Springer.

**[4] White, E. (2019) 'E**-commerce Platforms and Their Impact on Food Retailing', International Journal of Food Studies, 5(2), 45-58.

**[5] Brown, A.C. and Jones, R.M. (2018)** 'Consumer Preferences for Organic Fruits and Vegetables: A Market Analysis', Food Policy, 43(3), 141-151.

**[6] Lee, H.K. and Kim, S. (2016)** 'E-commerce Adoption and Its Impact on Small-Scale Organic Farmers: A Case Study', Journal of Agribusiness and Rural Development, 28(4), 123-138.