

# **Online Grocery Portal**

## **CSE 6324-005: ADV TOPS IN SOFTWARE ENGINEERING**

**By Team-11**

Preethika Gaddamidi	1002077250
Chetana Kotha	1002067826
Rohan Vajanala	1002075325
Samyukth Challa	1002075109
Mallisha Sailesh Patkar	1002060037

## **Part 1: Project Initiation:**

### **Project Description:**

Grocery shopping plays a major and important role in humans daily life as they must shop to obtain the necessities for their living. “Online grocery portal” is an application that lets users order groceries and other home goods online and have them delivered right to their doors. Due to shifts in customer tastes and the rise of online shopping, online grocery portals have become increasingly common. They provide convenience and flexibility to customers who would rather buy from residence.

### **Project Objectives:**

1. **Enhanced Accessibility:** Create a user-friendly and intuitive online platform that is accessible on mobile and web platforms. This platform should be designed to accommodate people with a variety of needs, including those related to busy schedules, health concerns, limited mobility, and geographic restrictions.
2. **Convenience and Time saving:** Reduce the time and effort required for grocery shopping by enabling customers to explore, choose, and buy items from a variety of categories online instead of having to visit a store.
3. **Cost-effectiveness:** Provide tools that let consumers quickly evaluate costs, choose less expensive options, and take advantage of sales and discounts in order to help them save money on their grocery shopping.
4. **Product Variety and Quality Control:** Make certain that a wide selection of products, including fresh fruit, meat, dairy, pantry goods, and medical supplies, are available. These products should be purchased from reliable vendors and must fulfil strict quality standards in order to satisfy a wide range of consumer demands and tastes.
5. **Smooth buying Experience:** To increase customer happiness and loyalty, provide a safe and easy buying experience that includes features like customisable shopping carts, a variety of payment methods, and effective delivery logistics.
6. **Customisation and Suggestions:** Use algorithms to evaluate user preferences and purchasing patterns. Then, provide customised promos and product recommendations to improve the shopping experience and boost client loyalty.
7. **Customer Service and Feedback Mechanism:** Provide strong customer service channels and feedback systems to respond to user inquiries, quickly correct problems, and get insightful data for ongoing enhancements to the functionality and calibre of services provided by the online grocery portal.

## **Project Scope:**

The creation of an online grocery portal is included in the project scope in order to meet the changing demands and difficulties that contemporary customers have when they go food shopping. Users will be able to explore, choose, and buy food and household items from the comfort of their homes thanks to the portal's comprehensive solution, which offers cost-effectiveness, accessibility, and convenience. A wide spectrum of customers will be served by the platform, including those who are busy, have limited mobility or access to transportation, are immigrants, or have health concerns.

An easy-to-use interface that is accessible on both web and mobile devices, a large selection of product categories that include fresh produce, meat, dairy, pantry items, and medical supplies, reliable payment options, and effective delivery logistics are all important aspects of the online grocery portal. With the use of algorithms to deliver customised product suggestions and promotions based on user preferences and purchasing behaviour, personalisation will also be a key focus. Customer service channels and feedback systems will guarantee prompt resolution of issues and ongoing enhancements to the functionality and calibre of services provided by the platform. The creation of a comprehensive online grocery solution that improves the shopping experience and satisfies the many demands of contemporary consumers is the overall scope of the project.

## **Client Requirements:**

**User-Friendly Interface:** The client needs the online grocery portal to have an interface that is simple to use and intuitive for both web browsers and mobile devices.

**Comprehensive Product Range:** To satisfy a range of consumer demands, the portal should include a large selection of goods, such as fresh produce, meat, dairy, pantry items, and medical supplies, all supplied from reputable vendors.

**Accessibility Features:** The significance of accessibility features is underscored by the customer, who highlights their ability to cater to a wide range of user demands, including but not limited to hectic schedules, health concerns, mobility challenges, and remote locations from food shops.

**Tools for Cost Efficiency:** To help consumers save money on groceries, the portal should provide capabilities that let them compare costs, choose less expensive options, and take advantage of sales and discounts.

**Customisation and Suggestions:** By using algorithms to examine customer preferences and purchasing patterns, customised product suggestions and promotions are made in an effort to improve the shopping experience.

**Payment Methods:** In order to satisfy a variety of customer preferences, the client has to support credit/debit cards, digital wallets, and cash on delivery.

**Effective Delivery Logistics:** In order to fulfil orders on time and with reliability, the portal should guarantee effective delivery logistics. This includes optimising delivery routes and timetables to reduce client wait times.

**Channels for Customer Support:** putting in place reliable customer service channels, such as live chat, email support, and a dedicated hotline, to quickly respond to user questions and fix problems.

**Functional Requirements:**

**User Registration and Authentication:** Our website entails a registration procedure enabling users to establish accounts, accompanied by a login mechanism allowing registered users to access their accounts.

**Product Exploration:** Users have the capability to peruse products across diverse categories.

**Shopping Cart:**The platform permits users to add desired items to their shopping carts, adjust item quantities, and remove items from carts at their discretion.

**Secure Payment Processing:** We endeavour to create a protected payment gateway for users to conduct online transactions securely.

**Non-functional Requirements:**

**Data Security:** The website boasts a robust security infrastructure to safeguard users' personal and financial data against unauthorised access.

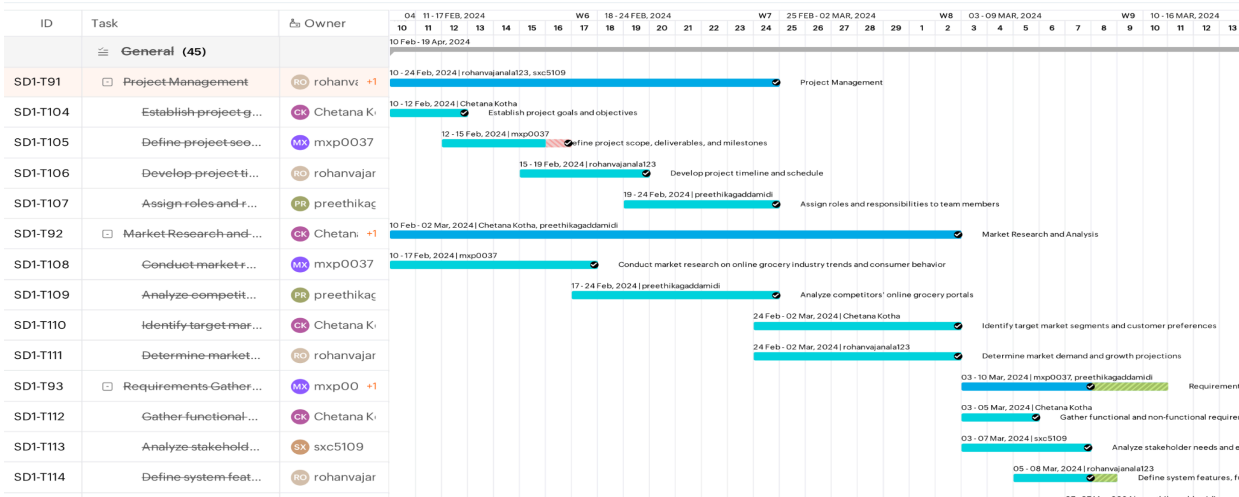
**Performance Optimization:** Our aim is to develop a rapid and dependable system capable of managing substantial traffic volumes (transactions) seamlessly, without encountering any glitches or downtime.

**User-Friendly Interface:** We aspire to craft an intuitive and user-centric interface facilitating effortless navigation and quick discovery of desired content.

**Universal Access:** The application ensures accessibility to all users, irrespective of their physical capabilities or impairments.

**Service Reliability:** The portal pledges consistent service delivery to users, free from interruptions or malfunctions.

**Project Timeline:**



Cost and time estimation: Calculate cost using COCOMO, LOC, FP using online software and tools.

COCOMO II - Constructive Cost Model

Software Size

Sizing MethodSource Lines of Code

SLOC

% Design Modified

% Code Modified

% Integration Required

Assessment and Assimilation (0% - 8%)

Software Understanding (0% - 50%)

Unfamiliarity (0-1)

New

20000

Reused

7500

0

0

20

4

Modified

9000

20

25

70

5

90

0

Software Scale Drivers

Precedentedness

Nominal

Architecture / Risk Resolution

Nominal

Process Maturity

Nominal

Development Flexibility

Nominal

Team Cohesion

Nominal

Software Cost Drivers

Product

Required Software Reliability

Nominal

Data Base Size

Nominal

Product Complexity

Nominal

Developed for Reusability

Nominal

Documentation Match to Lifecycle Needs

Nominal

Personnel

Analyst Capability

Nominal

Programmer Capability

Nominal

Personnel Continuity

Nominal

Application Experience

Nominal

Platform Experience

Nominal

Language and Toolset Experience

Nominal

Platform

Time Constraint

Nominal

Storage Constraint

Nominal

Platform Volatility

Nominal

Project

Use of Software Tools

Nominal

Multisite Development

Nominal

Required Development Schedule

Very Low

Maintenance

Off

Software Labor Rates

Cost per Person-Month (Dollars)

15

Calculate

Results

Software Development (Elaboration and Construction) Staffing Profile

Effort = 141.6 Person-months  
Schedule = 11.9 Months  
Cost = \$2123

Total Equivalent Size = 24485 SLOC  
Effort Adjustment Factor (EAF) = 1.43

Acquisition Phase Distribution

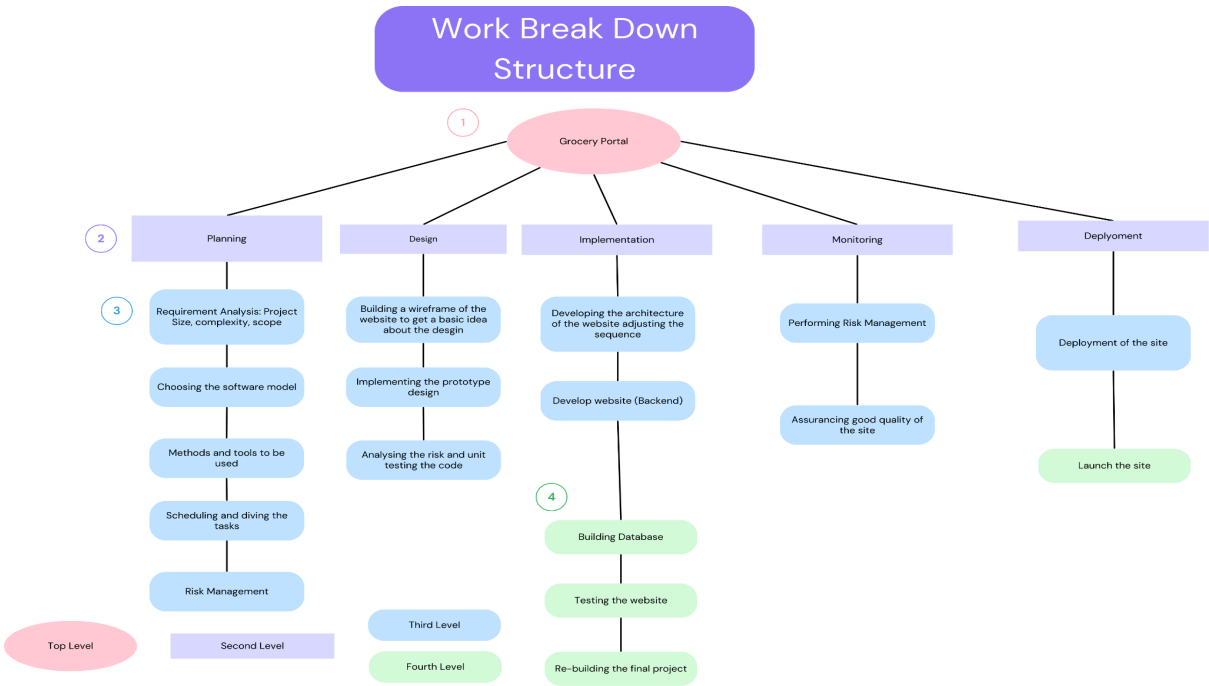
Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	8.5	1.5	5.7	\$127
Elaboration	34.0	4.4	7.6	\$510
Construction	107.6	7.4	14.5	\$1614
Transition	17.0	1.5	11.5	\$255

Software Effort Distribution for RUP/MBASE (Person-Months)

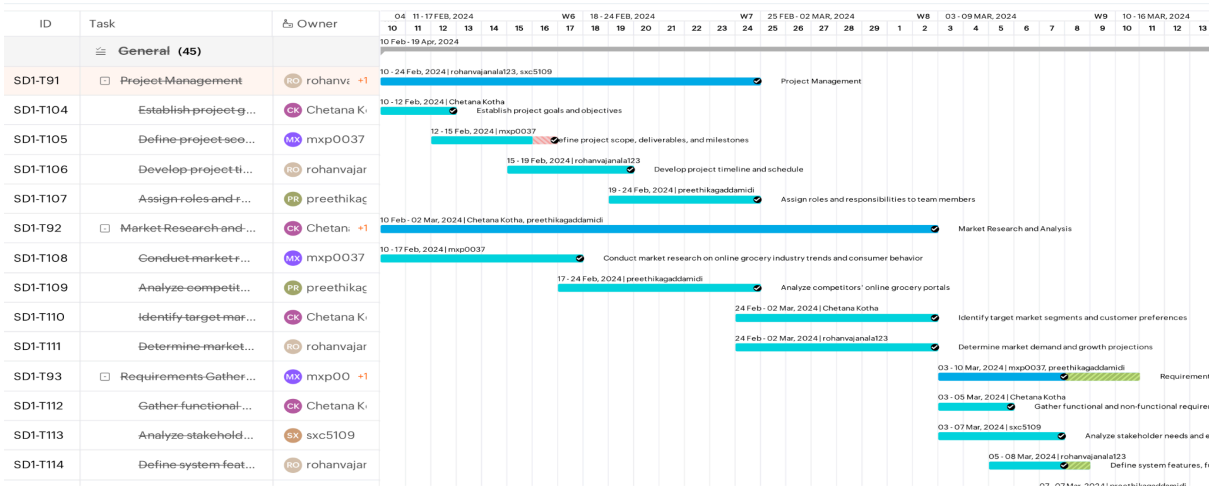
Phase/Activity	Inception	Elaboration	Construction	Transition
Management	1.2	4.1	10.8	2.4
Environment/CM	0.8	2.7	5.4	0.8
Requirements	3.2	6.1	8.6	0.7
Design	1.6	12.2	17.2	0.7
Implementation	0.7	4.4	36.6	3.2
Assessment	0.7	3.4	25.8	4.1
Deployment	0.3	1.0	3.2	5.1

Part 2: Project Planning

Create a WBS for the project. Include at least three levels of decomposition:



Project Schedule: Develop a project schedule using a Gantt chart or another appropriate tool. Identify critical path activities and milestones.



## **Part 3: Risk Management**

**Risk Identification:** Identify at least five potential risks associated with the project, and categorise them as technical, organisational, or external risks.

### **Organisational risk:**

**Inventory control:** Losses may result from either an excess or a shortage of inventory. Goods that are perishable are especially susceptible to spoiling.

**Employee Safety and Retention:** Mishaps, injuries, or employee turnover can cause operational disruptions and negatively affect customer service.

### **Technical risk:**

**Cybersecurity risks:** Sensitive consumer data can be compromised by malware, hacking, or data breaches, which can harm a store's reputation.

### **External risk:**

**Regulatory Compliance:** Violations of labour laws, health codes, or food safety rules may lead to fines, legal action, or even closure.

**Reputation management:** A store's reputation may be damaged and consumers may stop coming in as a result of bad press brought on by events like food contamination or customer complaints.

**Economic Factors:** Shifts in consumer purchasing patterns, inflation, and recession are examples of economic fluctuations that can impact sales and profitability.

**Market Competition:** Gaining a larger market share and making more money can be at risk while competing with other supermarkets, internet retailers, and other grocery stores.

**Disruptions in the supply chain :** It is caused by pandemics, natural disasters, or political upheaval that raise prices and cause shortages of certain products.

Natural disasters can physically harm the store and cause operational disruptions. Examples of these events include floods, storms, and earthquakes

**Risk Assessment: Assess the identified risks in terms of their likelihood and impact. Create a risk matrix.**

To assess the risks in terms of their likelihood and impact, we can create a risk matrix. In the risk matrix, the likelihood and impact of each risk are typically categorised into different levels, such as low, medium, and high.

Here's a risk matrix for the identified risks:

Risk	Likelihood	Impact	Risk Level
Market Competition	High	High	High
Disruptions in Supply Chain	Medium	High	High
Inventory Control	Medium	High	High
Employee Safety and Retention	Medium	High	High
Cybersecurity Risks	Medium	High	High
Regulatory Compliance	Medium	High	High
Reputation Management	Medium	High	High
Economic Factors	High	High	High
Natural Disasters	Low	High	Medium

In this matrix, the likelihood is assessed as low, medium, or high, while the impact is also categorised as low, medium, or high. The risk level is determined based on the combination of likelihood and impact, with high-risk levels indicating areas that require immediate attention and mitigation strategies.

**Risk Mitigation Plan:** Develop a risk mitigation plan for the high-priority risks. Describe specific actions to minimise or eliminate these risks.

**Risk Mitigation Strategies:** Create and put into action plans to lessen the probability or effect of hazards that have been recognised. Purchasing backup generators, for instance, can help reduce power interruptions.

**Contingency Planning:** Create backup plans so you can react quickly to unforeseen circumstances. These could include crisis communication plans, backup supplies, or emergency response procedures.

**Insurance coverage:** Obtain the correct insurance coverage to guard against monetary losses brought on by a variety of dangers, such as liability claims or property damage.



**Education and Training:** To reduce the risks which are associated with human error, train staff on cyber security best practices, compliance standards, and safety protocols.

**Monitoring and Review:** Keep a close eye on how well risk management techniques are working and make required adjustments to plans in response to evolving situations or emerging risks.

**Communication:** To keep stakeholders, including staff members, vendors, and clients, aware about possible risks and mitigation strategies, keep lines of communication open.

#### **Part 4: Quality Assurance:**

Project success depends on maintaining high-quality deliverables throughout the entire project lifecycle. Here's a thorough key points we implemented for accomplishing this:

1. **Defining Specific Requirements:** Work with stakeholders to identify specific, well-defined requirements from the outset. These specifications must be time-bound, relevant, quantifiable, attainable, and specific (SMART).
2. **Creating a Plan for Quality Management:** Make a strategy for quality control that describes the procedures, resources, and roles involved in maintaining the project's quality. The strategy ought to encompass the methods for measuring, overseeing, and regulating quality.
3. **Using Agile Methodologies:** Implement agile techniques like Scrum or Kanban, where appropriate. Iterative development, regular testing, and ongoing input are all encouraged by agile, and these are critical for preserving quality throughout the project.
4. **QA Procedures into Practice:** Create quality assurance procedures to confirm that deliverables adhere to standards. Code reviews, design reviews, testing, and peer inspections are a few examples of this.
5. **Conduct Testing:** To find and fix errors, thoroughly test the deliverables. User Acceptance Testing (UAT), System Testing, Integration Testing, and Unit Testing are all included in this. Tools for automated testing can make this procedure more efficient.
6. **Applying Measures for Quality Control (QC):** Put QC procedures in place to keep an eye on and manage the deliverables' quality. To find and address deviations from quality standards, this may entail doing audits, performance reviews, and inspections.
7. **Engaging the Parties:** Engage stakeholders at every stage of the project's lifetime to get their input and make sure expectations are met. Collaboration and regular communication among stakeholder's aid in the early detection and proactive resolution of any problems.
8. **Continuous Improvement:** Based on measurements, lessons learned, and feedback, continuously assess and enhance quality procedures. Review performance often in

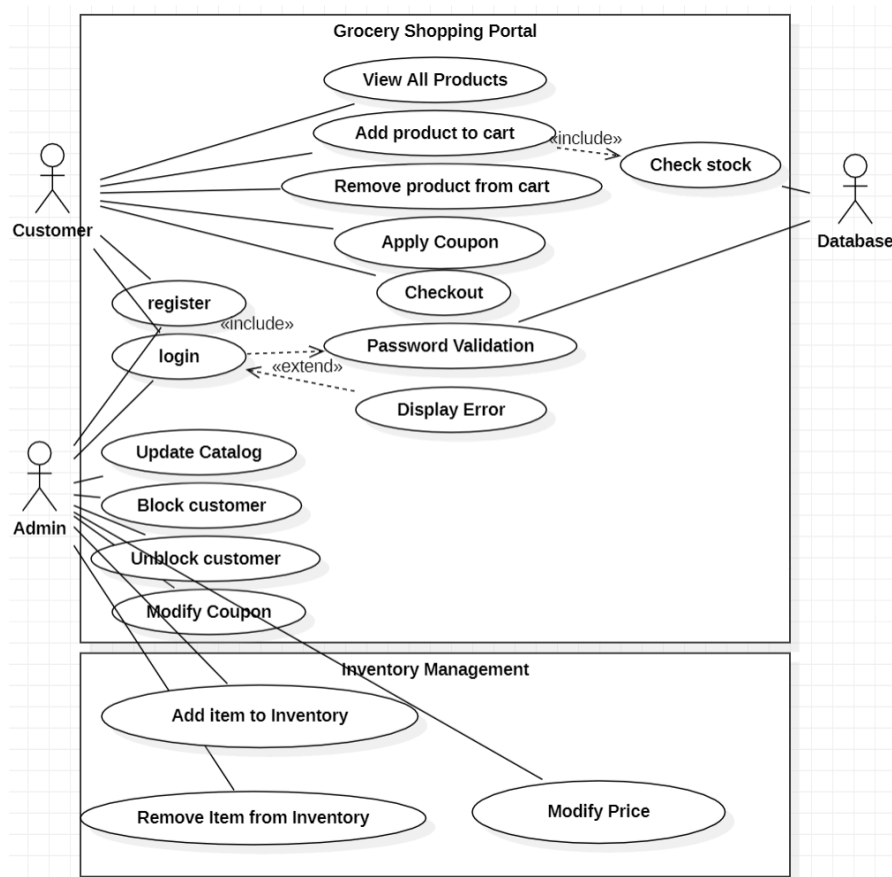
relation to quality goals and adapt as necessary to improve the calibre of outputs.

## **Part 5: Design : Include ER, UML, ...diagram.**

### **Use case Diagram:**

The use cases are:

- Register
- Login
- View All Products
- Add/Remove Products in cart.
- Apply Coupon & Modify Coupon
- Update Catalog
- Block/Unblock Customer
- Checkout

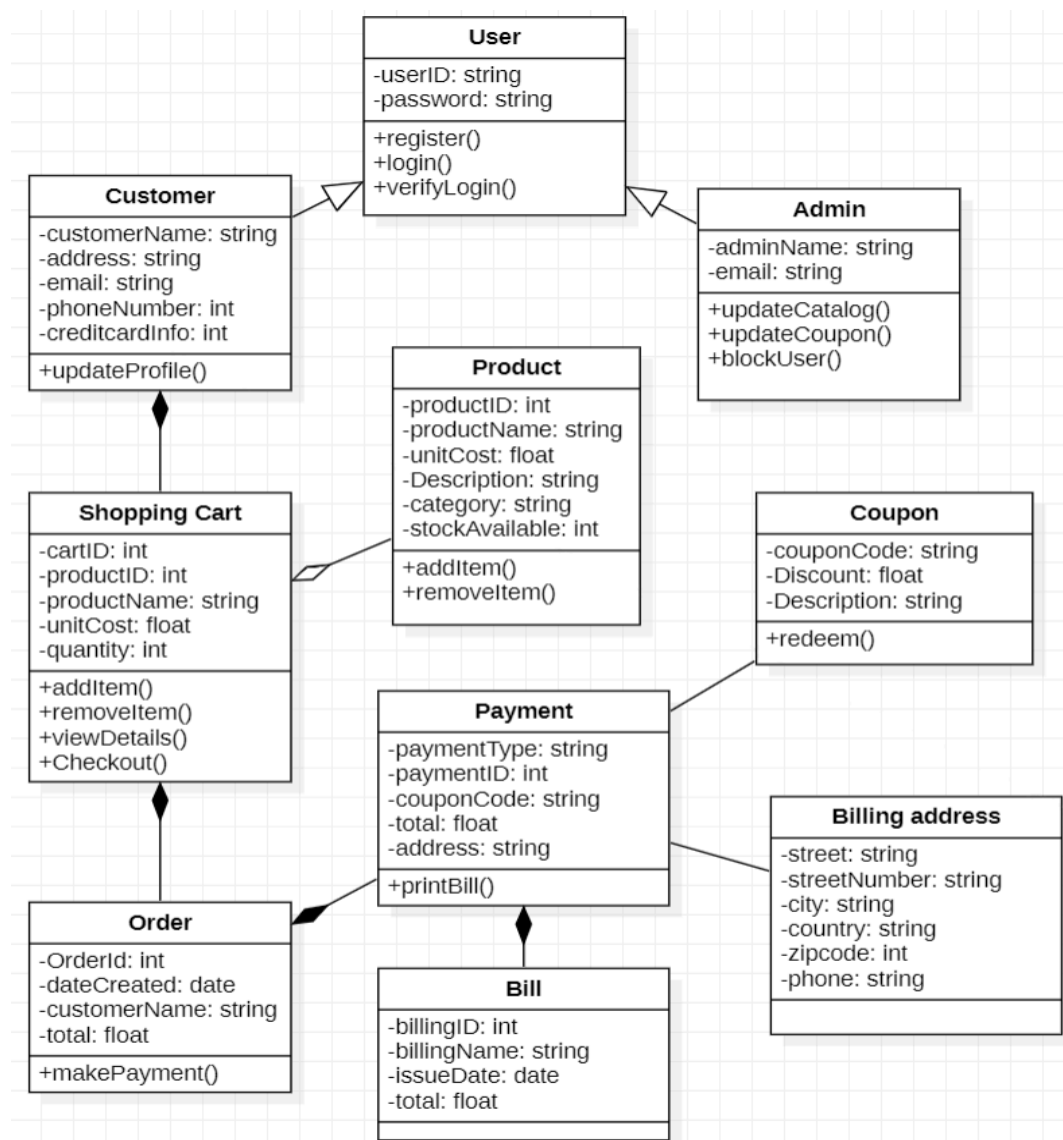


**USE CASE DIAGRAM**

## Class diagram:

The objects in this class are:

- User
- Customer
- Admin
- Shopping Cart
- Product
- Order
- Payment
- Coupon
- Billing Address & Bill.

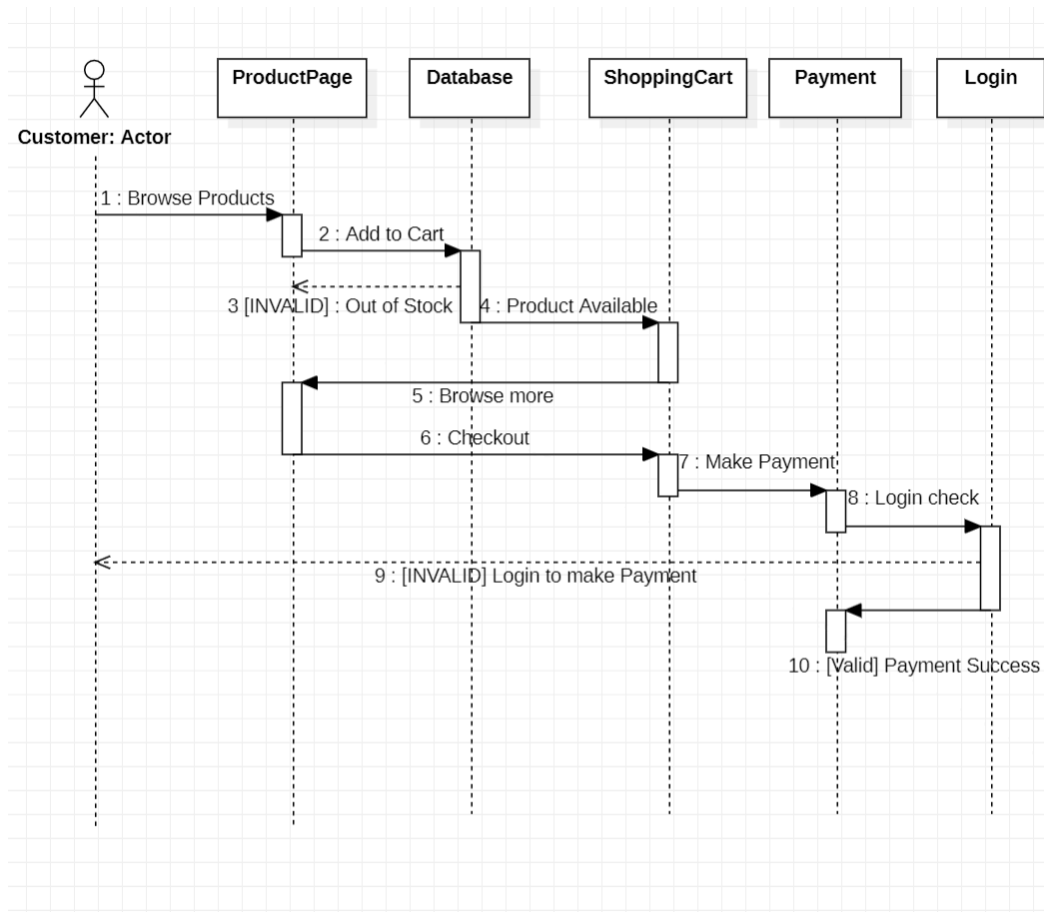


**CLASS DIAGRAM**

## Sequence diagram:

A sequence diagram shows how objects interact with one another sequentially, or the manner in which these interactions occur. This makes it very simple to follow the interactions of the system without even having to study the instructions. If any unanticipated results are obtained, it also makes it easier to follow the errors.

The customer is the Actor in our system. The objects of the system are Login, Product Page, Database, Shopping Cart, and Payment.



Sequence diagram

## Part 6: Implementation

Describe any kind of tools, framework, libraries, .. to be used in your project.

Frontend	HTML5, CSS, JavaScript
DataBase	SQLITE3
Backend	Python
IDE	Visual Studio Code
Python Framework	Django
Testing	Selenium

### Implementation:

The implementation of the project is done in four major steps.

#### 1) Registration and Login:

This step allows new users to register the website. For returning users, they can use the login functionality to enter their credentials, and the system will validate provided information with the stored user data in the database for authentication.

This process is necessary if the user wishes to make payments for products in their shopping cart during checkout. Viewing the product page does not require the user to log in or register.

```
for a {{ site_name }} account and sign in below:{% endblocktrans %}</p>

<div class="socialaccount_ballot">

  <ul class="socialaccount_providers">
    {% include "socialaccount/snippets/provider_list.html" with process="login" %}
  </ul>

  <div class="login-or">{% trans 'or' %}</div>

</div>

{% include "socialaccount/snippets/login_extra.html" %}

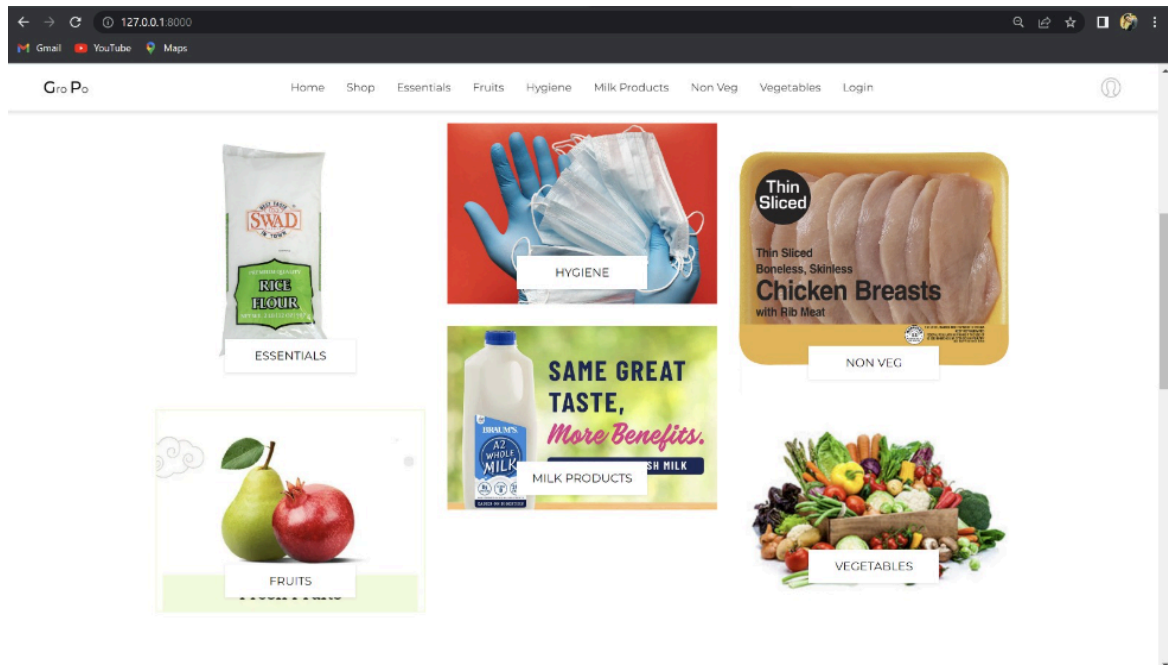
{% else %}
<p>{% blocktrans %}If you have not created an account yet, then please
<a href="{{ signup_url }}" class="text-primary">sign up</a> first.{% endblocktrans %}</p>
{% endif %}

<form class="login" method="POST" action="{{ url 'account_login' }}">
  {% csrf_token %}
  {{ form|crispy }}
  {% if redirect_field_value %}
    <input type="hidden" name="{{ redirect_field_name }}" value="{{ redirect_field_value }}" />
  {% endif %}
  <a class="btn btn-outline-dark" href="{{ url 'account_reset_password' }}">{% trans "Forgot Password?" %}</a>
  <button class="btn btn-primary" type="submit">{% trans "Sign In" %}</button>
</form>

</div>
</div>
</div>
```

## 2) Browsing the products:

We can browse the products using the user-friendly interface of the website. User's can navigate through the website, and select a category from the navigation bar, and then browse through the products to choose the ones they wish to purchase.

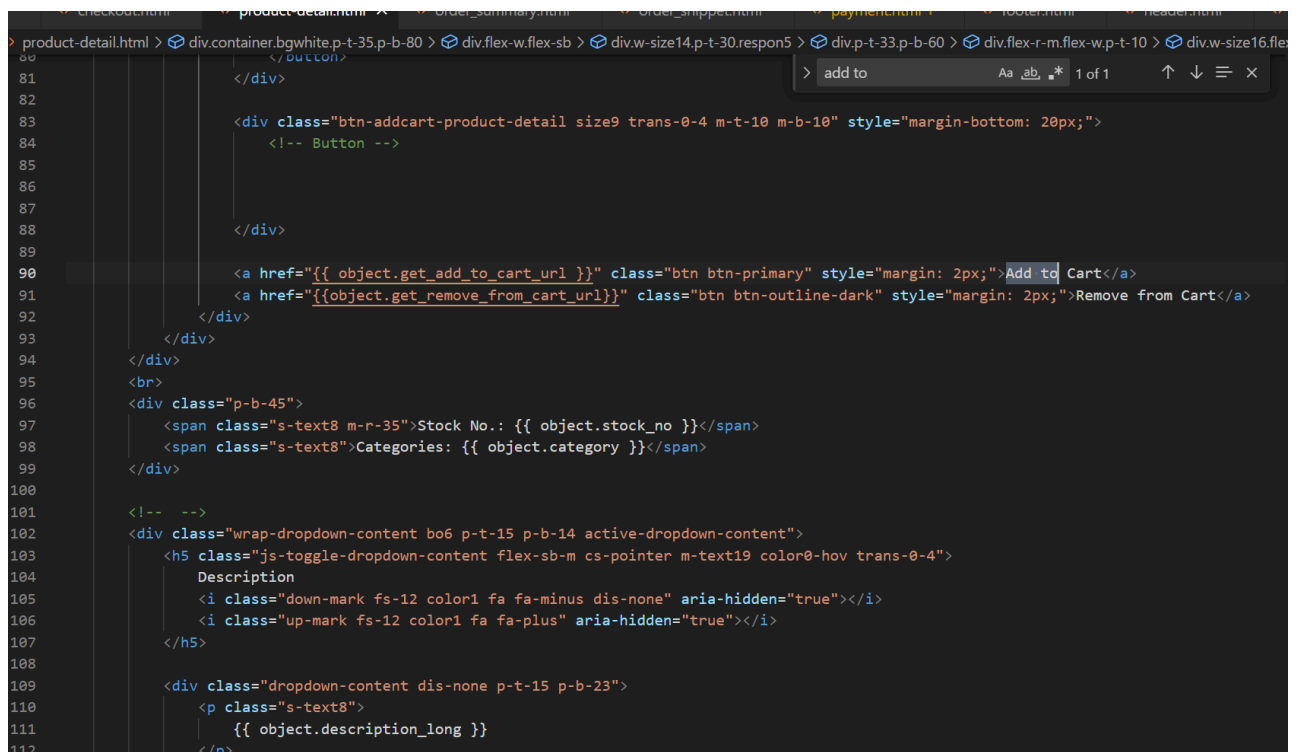
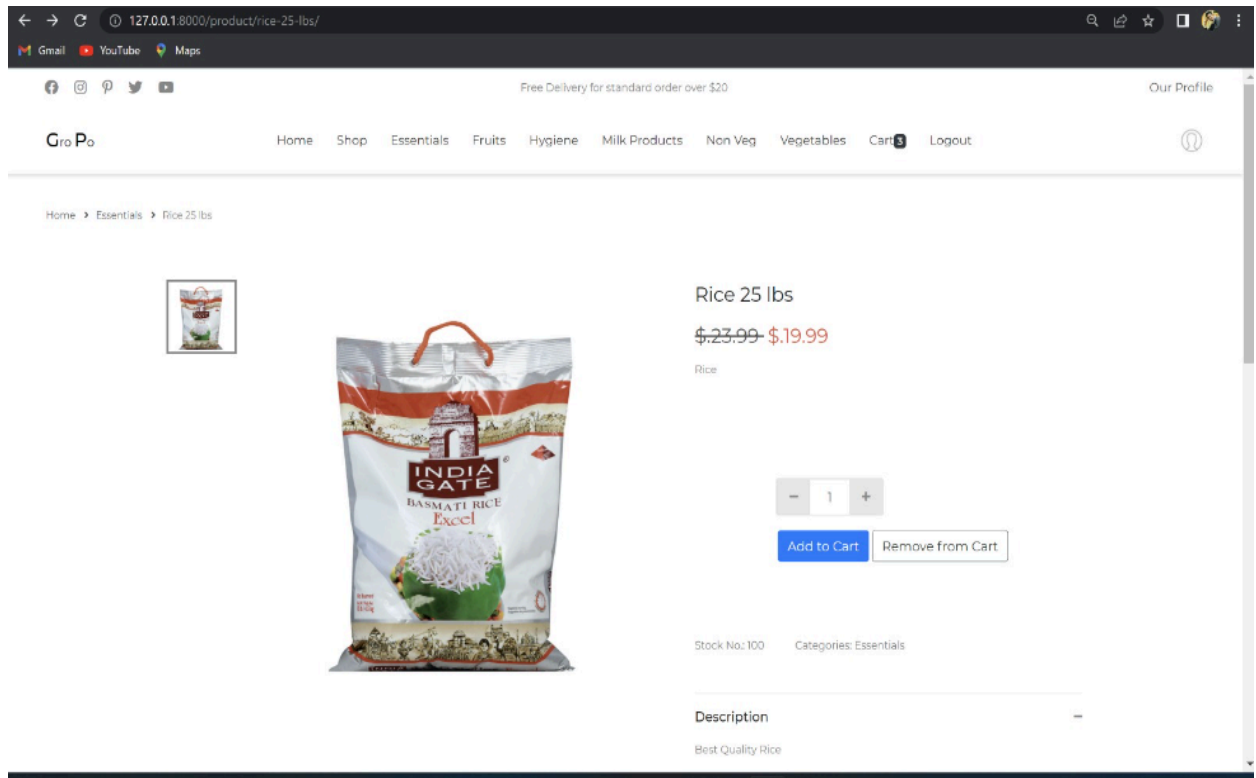


```
product-detail.html > footer.bg6.p-t-45.p-b-43.p-l-45.p-r-45 > div.flex-w.p-b-90 > div.w-size7.p-t-30.p-l-15.p-r-15.respon4 > ul > li.p-b-9
168     <a href="#" class="fs-18 color1 p-r-20 fa fa-youtube-play"></a>
169   </div>
170 </div>
171 </div>
172
173 <div class="w-size7 p-t-30 p-l-15 p-r-15 respon4">
174   <h4 class="s-text12 p-b-30">
175     Categories
176   </h4>
177
178   <ul>
179     <li class="p-b-9">
180       <a href="/category/Essentials/" class="s-text7">
181         Essentials
182       </a>
183     </li>
184
185     <li class="p-b-9">
186       <a href="/category/Vegetables/" class="s-text7">
187         Vegetables
188       </a>
189     </li>
190
191     <li class="p-b-9">
192       <a href="/category/Fruits/" class="s-text7">
193         Fruits
194       </a>
195     </li>
196
197     <li class="p-b-9">
198       <a href="/category/NonVeg/" class="s-text7">
199         Non veg
200       </a>
201     </li>
```

### 3) Adding products to the Cart:

The website has a cart feature that enables users to add products to the shopping cart, and also update the quantity of items and remove them if necessary.

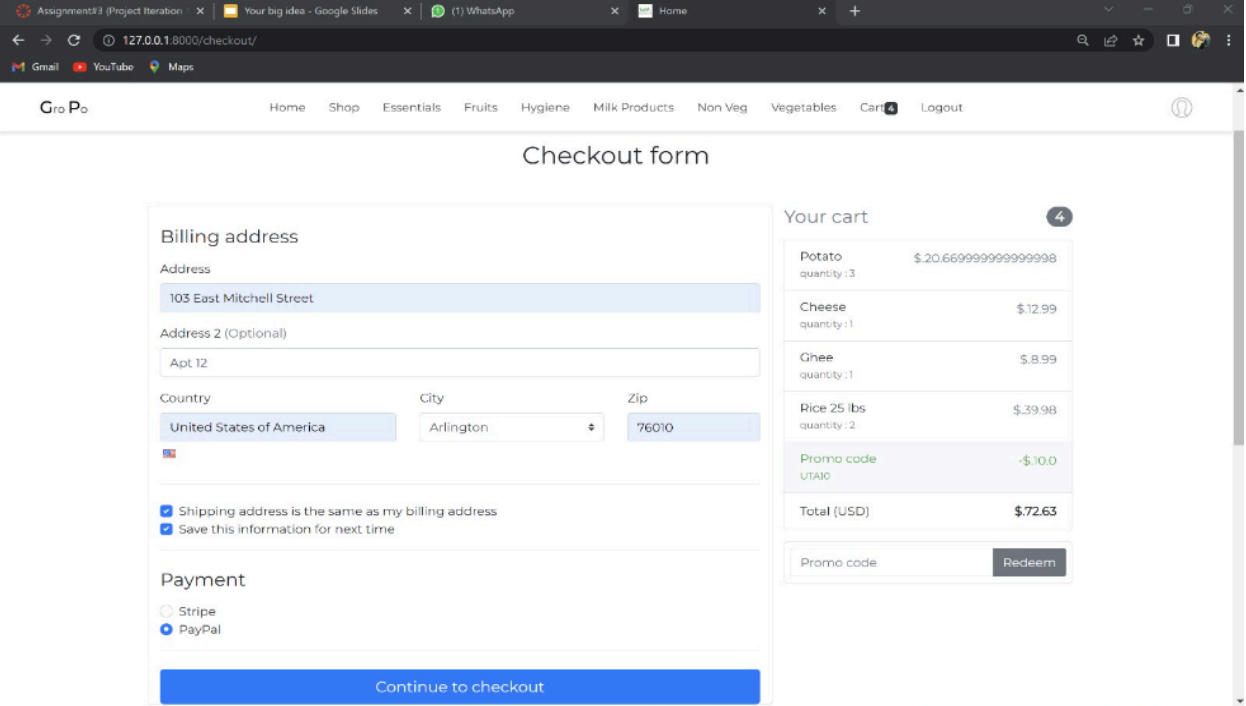
This information is stored in the database, ensuring that it remains available across all the sessions.



#### 4) Applying coupons and Checkout:

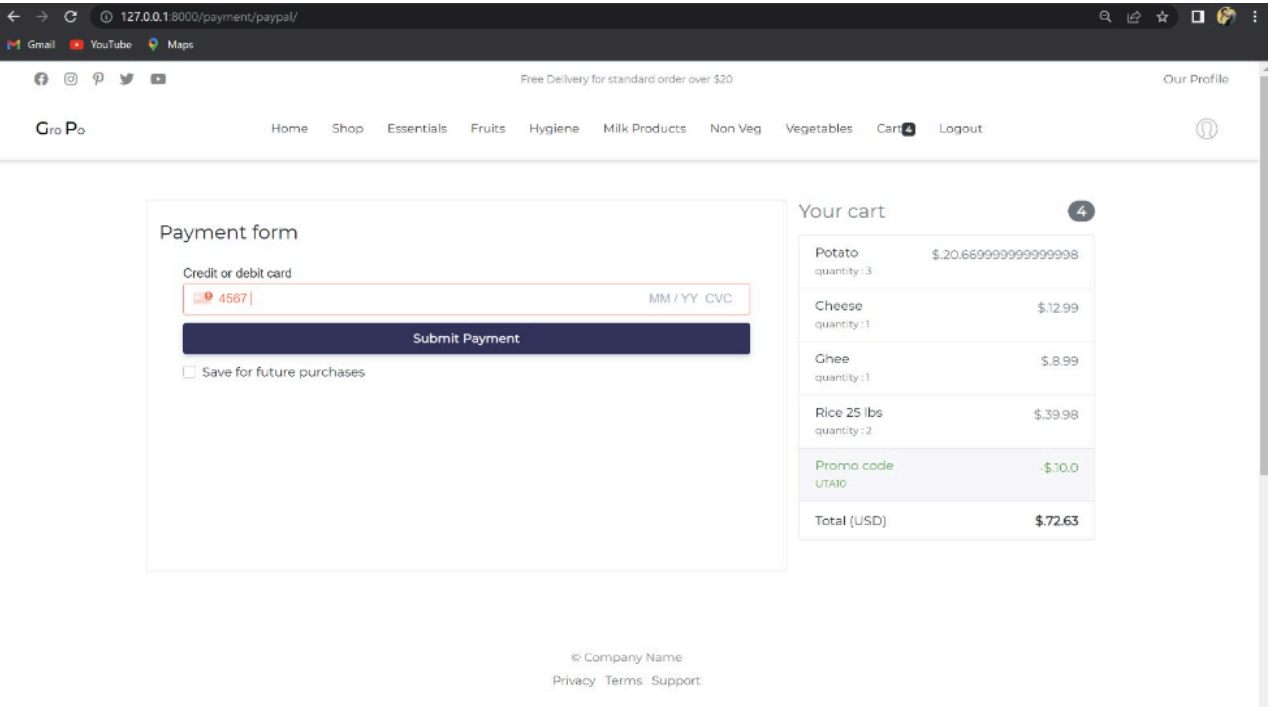
To avail a discount on products, the users can apply a unique coupon or any discount code, which is validated against the ones stored in the database.

The payment process is always carried out through a secure payment gateway to avoid cybersecurity risks.



The screenshot shows the checkout form for GroPo. The form is divided into two main sections: Billing address and Payment. The Billing address section includes fields for Address (103 East Mitchell Street), Address 2 (Optional) (Apt 12), Country (United States of America), City (Arlington), and Zip (76010). There are checkboxes for "Shipping address is the same as my billing address" and "Save this information for next time". The Payment section has radio buttons for Stripe and PayPal (selected). A blue "Continue to checkout" button is at the bottom. On the right, the "Your cart" section shows a list of items: Potato (quantity: 3, \$20.66), Cheese (quantity: 1, \$12.99), Ghee (quantity: 1, \$8.99), and Rice 25 lbs (quantity: 2, \$39.98). A promo code UTA10 is applied, reducing the total by \$10.00. The total (USD) is \$72.63. A "Redeem" button is next to the promo code field.

Item	Quantity	Price
Potato	3	\$20.66
Cheese	1	\$12.99
Ghee	1	\$8.99
Rice 25 lbs	2	\$39.98
Promo code UTA10		-\$10.00
<b>Total (USD)</b>		<b>\$72.63</b>



The screenshot shows the payment form for GroPo. The form is divided into two main sections: Payment form and Your cart. The Payment form section includes a "Credit or debit card" field with a card number (4567) and a "Submit Payment" button. There is a checkbox for "Save for future purchases". The "Your cart" section shows the same list of items as the checkout form: Potato (quantity: 3, \$20.66), Cheese (quantity: 1, \$12.99), Ghee (quantity: 1, \$8.99), and Rice 25 lbs (quantity: 2, \$39.98). A promo code UTA10 is applied, reducing the total by \$10.00. The total (USD) is \$72.63.

Item	Quantity	Price
Potato	3	\$20.66
Cheese	1	\$12.99
Ghee	1	\$8.99
Rice 25 lbs	2	\$39.98
Promo code UTA10		-\$10.00
<b>Total (USD)</b>		<b>\$72.63</b>

© Company Name  
Privacy Terms Support



```
135 <script src="https://js.stripe.com/v3/"></script>
136 {% if card %}
137     <div style="padding: 5px 30px;">
138         <div class="custom-control custom-checkbox">
139             <input type="checkbox" class="custom-control-input" name="use_default_card" id="use_default_card">
140             <label class="custom-control-label" for="use_default_card">Use default card:
141                 **** *  {{ card.last4 }}
142             <span>Exp: {{ card.exp_month }}/{{ card.exp_year }}</span></label>
143         </div>
144     </div>
145 {% endif %}
146
147 <div class="current-card-f Corresponds to the HTTP POST method; form data are included in the body of the form and sent to the server.
148     <form action="." method="post" class="stripe-form">
149         {% csrf_token %}
150         <input type="hidden" name="use_default" value="true">
151         <div class="stripe-form-row">
152             <button id="stripeBtn">Submit Payment</button>
153         </div>
154         <div id="card-errors" role="alert"></div>
155     </form>
156 </div>
157 <div class="new-card-form">
158     <form action="." method="post" class="stripe-form" id="stripe-form">
159         {% csrf_token %}
160         <div class="stripe-form-row" id="creditCard">
161             <label for="card-element" id="stripeBtnLabel">
162                 Credit or debit card
163             </label>
164             <div id="card-element" class="StripeElement StripeElement--empty"><div class="__PrivateStripeElement" style=
165         </div>
166         <div class="stripe-form-row">
167             <button id="stripeBtn">Submit Payment</button>
168         </div>
169         <div class="stripe-form-row">
170             <div class="custom-control custom-checkbox">
171                 <input type="checkbox" class="custom-control-input" name="save" id="save_card_info">
172                 <label class="custom-control-label" for="save_card_info">Save for future purchases</label>
173             </div>
174         </div>
175         <div id="card-errors" role="alert"></div>
```

## Part 7: Testing

Explain and discuss both manual testing and automated testing. Result from unit test, integration test, system testing and user acceptance testing along with the automation result.

Test case	Test case Name	Description	Actual Result	Expected Result	Status
<i><b>TID01</b></i>	Registration Page	<ul style="list-style-type: none"><li>• This involves confirming whether users can create an account that meets the standard criteria with all mandatory fields correctly filled, and ensuring that the system responds appropriately to any errors caused by missing or invalid data.</li><li>• The objective is to check whether a new user can register successfully.</li></ul>	Successfully Registered as "USERID".	Successfully Registered as "USERID".	<b>PASS</b>
<i><b>TID02</b></i>	Login Page	<ul style="list-style-type: none"><li>• The task is to authenticate the login ID and password by checking their correctness.</li><li>• Additionally, it involves testing if the system correctly handles instances where the user enters incorrect login credentials, resulting in login failure.</li></ul>	Login successful.	Login successful.	<b>PASS</b>
<i><b>TID03</b></i>	Product Page	<ul style="list-style-type: none"><li>• The goal is to confirm whether the product details page shows precise and correct information related to the product such as the name, short description, product image, and price.</li></ul>	Display Product Page	Display Product Page	<b>PASS</b>
<i><b>TID04</b></i>	Cart Page	<ul style="list-style-type: none"><li>• This involves testing whether users can add new products to their shopping cart, and ensuring that the correct item is added to the cart</li></ul>	Display Cart	Display Cart	<b>PASS</b>

<b>TID05</b>	Updated Cart	<ul style="list-style-type: none"> <li>Verifies whether the users can update the items in the cart(+/- to update quantity).</li> </ul>	Update the Cart and Discount Field.	Update the Cart and Discount Field.	<b>PASS</b>
<b>TID06</b>	Payment Page	<ul style="list-style-type: none"> <li>Verifies whether the users can proceed to checkout by providing their billing and shipping information.</li> </ul>	Payment Page	Payment page	<b>PASS</b>
<b>TID07</b>	Sign out page	<ul style="list-style-type: none"> <li>This involves testing whether the user can successfully sign out or log out of the system.</li> </ul>	Sign out Successful	Sign out Successful	<b>PASS</b>
<b>TID08</b>	Block/ unblock of user	<ul style="list-style-type: none"> <li>This involves testing if the admin can block/unblock a user</li> </ul>	User has been blocked/unblocked	User has been blocked/unblocked	<b>PASS</b>
<b>TID09</b>	Coupon Application	<ul style="list-style-type: none"> <li>This involves testing if the admin can modify a coupon</li> </ul>	Coupon modified	Coupon modified	<b>PASS</b>

We have used selenium for the testing phase. As, testing an online grocery portal using Selenium can have several benefits such as:

- Selenium allows for the automation of repetitive tasks involved in testing, such as form filling, buttons, and verifying elements of the page. This saves time and effort compared to manual testing.
- It also allows us to create test scripts that can be rerun whenever changes are made to the application. This will make sure that new features or bug fixes don't break existing functionalities.
- It supports different web browsers, such as Firefox, Chrome, Safari, and Microsoft Edge. It will make sure that the online grocery portal works consistently across various browsers, providing a better user experience for customers.
- And, Selenium allows for parallel execution of tests across multiple environments and browsers simultaneously. It speeds up the testing process and improves efficiency.

```
def loginTest(driver, username = 'malli', password = '123456'):
    login_text = driver.find_element(By.XPATH, '//a[text()="Login"]')
    hover = ActionChains(driver).move_to_element(login_text)
    hover.perform()
    loginBtn = WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.XPATH, '//ul[@class="']
    loginBtn.click()
    username_input = driver.find_element(By.XPATH, "//input[@name='login']")
    username_input.send_keys(username)
    password_input = driver.find_element(By.XPATH, "//input[@name='password']")
    password_input.send_keys(password)
    signin_button = driver.find_element(By.XPATH, "//button[text()='Sign In']")
    signin_button.click()
    try:
        # Wait for an element on the home page to be visible (replace 'element_id' with the actual ID of a
        WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.XPATH, "//div[contains(te
        print("Successfully logged in and on the home page.")
    except:
        print("Failed to reach the home page after logging in.")
```

```
def logoutTest(driver):
    loginTest(driver)
    logout_test = driver.find_element(By.XPATH, '//a[text()="Logout"]')
    logout_test.click()
    signout_button = driver.find_element(By.XPATH, "//button[text()='Sign Out']")
    signout_button.click()
    try:
        # Wait for an element on the home page to be visible (replace 'element_id' with the actual ID of a
        WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.XPATH, " //div[contains(t
        print("Successfully logged out from the home page.")
    except:
        print("Failed to log out....")
```

DevTools listening on ws://127.0.0.1:55458/devtools/browser/2a1bf9bf-b3a2-45d4-98b4-cc5fa64331fe  
 Successfully logged in and on the home page.  
 Successfully logged out from the home page.

DevTools listening on ws://127.0.0.1:55373/devtools/browser/1eb7a687-651f-459d-b8b6-78dc97d849c7  
 Successfully logged in and on the home page.  
 Product added to the cart.

```

def signUpTest(driver):
    userName = 'ab' + str(random.randint(30,100000))
    email = userName + '@gmail.com'
    password = str(random.randint(10000,99999))
    login_text = driver.find_element(By.XPATH,'//a[text()="Login"]')
    hover = ActionChains(driver).move_to_element(login_text)
    hover.perform()
    signUpText = WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.XPATH, '//ul
    signUpText.click()
    #input fields
    username_input = driver.find_element(By.XPATH,"//input[@name='username']")
    username_input.send_keys(userName)
    email_input = driver.find_element(By.XPATH,"//input[@name='email']")
    email_input.send_keys(email)
    password_input = driver.find_element(By.XPATH,"//input[@name='password1']")
    password_input.send_keys(password)
    pass_again_input = driver.find_element(By.XPATH,"//input[@name='password2']")
    pass_again_input.send_keys(password)
    signUpBtn = driver.find_element(By.XPATH,"//button[contains(text(),'Sign Up')]")
    signUpBtn.click()
    loginTest(driver,username= userName, password=password)

```

```

✓ def cartTest(driver):
    loginTest(driver)
    shopText = driver.find_element(By.XPATH,"//a[text() = 'Shop']")
    shopText.click()
    pannerText = driver.find_element(By.XPATH,"//a[contains(text(),'Paneer')]")
    pannerText.click()
    addToCartBtn = driver.find_element(By.XPATH,"//a[contains(text(),'Add to Cart')]")
    addToCartBtn.click()
    try:
        # Wait for an element on the home page to be visible (replace 'element_id' with the actual ID
        WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.XPATH, " //div[contai
        print("Product added to the cart.")
    except:
        print("Failed to add the product")

```

## **Part 8: Maintenance:**

### **Describe list of improvements that you can add to your project in future:**

For an online grocery portal, different types of maintenance are important to ensure it's smooth operation, scalability and security. Below are some key types of maintenance:

1. **Corrective Maintenance:** Fixing issues, errors or bugs that arise in the system to keep the grocery portal secure, stable, and performing optimally. Reactive measures to address issues as they arise, such as bug fixes, root cause analysis, and change management.
2. **Preventive Maintenance:** This type of maintenance aims to prevent potential issues from occurring in the first place. Such as, regular monitoring, updating software components, and also conducting security audits to identify vulnerabilities to keep the online grocery portal secure, stable, and performing optimally. This includes regular updates, security audits, backups, performance monitoring, user training, and scalability planning.
3. **Adaptive Maintenance:** As the online grocery market evolves, the portal needs to adapt to changes in technology, user preferences, and market trends. It involves modifying the system to accommodate these market changes.
4. **Perfective Maintenance:** This focuses on enhancing the system's functionality and performance based on user feedback and business requirements. It involves adding new features, improving user experience, or optimising performance.

**In considering future improvements for our online grocery portal project, several enhancements can be envisioned to ensure its sustained efficiency, scalability, and security:**

**Refined User Experience:** Continuously refine the interface and experience based on user feedback and evolving design trends. This could involve simplifying the checkout process, incorporating personalised recommendations, and refining search functionalities.

**Integration of Innovative Technologies:** Explore incorporating emerging technologies like artificial intelligence (AI) and machine learning (ML) to automate processes such as inventory management and predictive analytics. This can enhance operational efficiency and user satisfaction.

**Scalability and Performance Optimization:** Proactively optimize the backend infrastructure and codebase to handle increasing user traffic and data volume. Consider cloud-based solutions, caching mechanisms, and database query optimization to improve response times.

**Strengthened Security Measures:** Enhance security protocols to protect user data and prevent cyber threats. This might involve regular security assessments, implementing multi-factor authentication, and encrypting sensitive data.

**Mobile Optimization:** Ensure the platform is optimized for mobile devices to cater to the rising trend of mobile shopping. This could entail developing a dedicated mobile app or ensuring responsive design for seamless browsing and purchasing on smartphones and tablets.

**Analytics and Business Intelligence:** Implement robust analytics and reporting capabilities to gain insights into user behaviour, sales trends, and inventory management. This data-driven approach can inform strategic decisions and facilitate targeted marketing efforts.

**Community Engagement and Feedback Mechanisms:** Foster a sense of community by implementing feedback mechanisms like user forums, surveys, and customer support channels. Actively engage with users to understand their needs and preferences, driving continuous improvement and customer satisfaction.

### **Part 9: Roles and Responsibilities:**

<b>Name</b>	<b>Roles and Responsibilities</b>
Preethika Gaddamidi	<ul style="list-style-type: none"><li>● Database Integration</li><li>● Backend Python for Product Page</li><li>● Documentation</li><li>● Selenium Testing</li></ul>
Chetana Kotha	<ul style="list-style-type: none"><li>● Back End Python for Product Page</li><li>● Back End for Cart</li><li>● Documentation</li><li>● Selenium Testing</li></ul>
Rohan Vajanala	<ul style="list-style-type: none"><li>● Front End Python for Login Page</li><li>● Front End for Cart</li><li>● Documentation</li><li>● Selenium Testing</li></ul>
Samyukth Challa	<ul style="list-style-type: none"><li>● Front End UI for Product Page</li><li>● Payment Page</li><li>● Documentation</li><li>● Selenium Testing</li></ul>
Mallisha Patkar	<ul style="list-style-type: none"><li>● Back End Python for Login Page</li><li>● Back End for Cart</li><li>● Documentation</li><li>● Selenium Testing</li></ul>

## **Part 10: Future Scope:**

**Technological Advancements:** AI integration for personalised recommendations and streamlined operations, optimising inventory management and real-time customer insights.

**Sustainable Practices:** Eco-friendly packaging and delivery methods adoption, including renewable materials and efficient transportation to reduce environmental impact.

**Expansion:** Diversification of product offerings and entry into new markets through partnerships with local suppliers and introduction of niche products.

**Community Engagement:** Integration of social media and user-generated content for community-driven feedback, fostering customer loyalty and interaction.

## **Part 11: References:**

### **Database and Framework:**

1. **SQLite3 Documentation:** <https://www.sqlite.org/docs.html>
  - For understanding and implementing database solutions.
2. **Django Documentation:** <https://docs.djangoproject.com/en/5.0/>
  - Framework (Django) for the web application development
3. **Django Implementation:**  
<https://www.youtube.com/watch?v=SIyxjRJ8VNY&list=PLsyebzWxl7r2ukVgTqIQcl-1T0C2mzau>
  - Series of videos for implementation of Website

### **Software and Tools:**

1. **Python:** <https://www.python.org>
  - Used for backend development.
2. **HTML,CSS,JS:** <https://www.w3schools.com>
  - Used for Front-end development.For designing modern and responsive user interfaces.