LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CC5002NI Software Engineering

35% Individual Coursework

Submission : Final Submission

Academic Semester: AY 2024-2025

Credit: 30

Student Name: Salina Thing

London met ID: 23047540

College ID: np01cp4a230017

Assignment Due Date: Monday, May 12, 2025

Assignment Submission Date: Monday, May 12, 2025

Submitted To: Rubin Thapa

23047540 SalinaThing SE_.docx

Islington College,Nepal

Document Details

Submission ID
trn:oid:::3618:95455567

Submission Date
May 12, 2025, 12:38 PM GMT+5:45

Download Date
May 12, 2025, 12:39 PM GMT+5:45

File Name
23047540 SalinaThing SE_.docx

File Size
33.3 KB

55 Pages

4,894 Words

28,044 Characters

16% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

- **60** Not Cited or Quoted 13%
  Matches with neither in-text citation nor quotation marks
- **8** Missing Quotations 3%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

7%	Internet sources
0%	Publications
15%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Table of Contents

Table of figures

Table of Tables

1. Introduction

This is an **individual coursework** of **35%** of **CS5002NI-Software Engineering** provided by our college, Islington College. In this project, we are required to carry out Object-Oriented Analysis and Design (OOAD) of the system for new inventory management system for a client organization, **Global Tech Corporation**.

1.1. Client background

Global Tech Corporation is a growing inventory management business which manages sales, controls stock, dispatches order, and generates reports. As the organization expands, the limitations of its existing manual or semi-automated inventory processes have become apparent. The company now seeks to improve operational efficiency, reduce human errors, and gain real-time visibility into stock levels with transactions by modernizing its operations.

Figure 1: Introduction of business

1.2. Aims

The **primary aim** of this coursework is to deal with real-world case scenarios and improve the inventory management system through systematic analysis. Design and agile development practices.

1.3. Objective

The objectives of my projects are mentioned below:

- To understand and apply Object-Oriented Analysis and Design principles to a real-world software system.
- To design the system using UML diagrams, including class, use case, sequence, activity, and communication diagrams.
- To manage the project using Agile methodology (Scrum).
- To model a fully functional Inventory Management System using UML diagrams.
- To simulate the development of the lifecycle by organizing the process using Agile methodologies (Scrum framework).

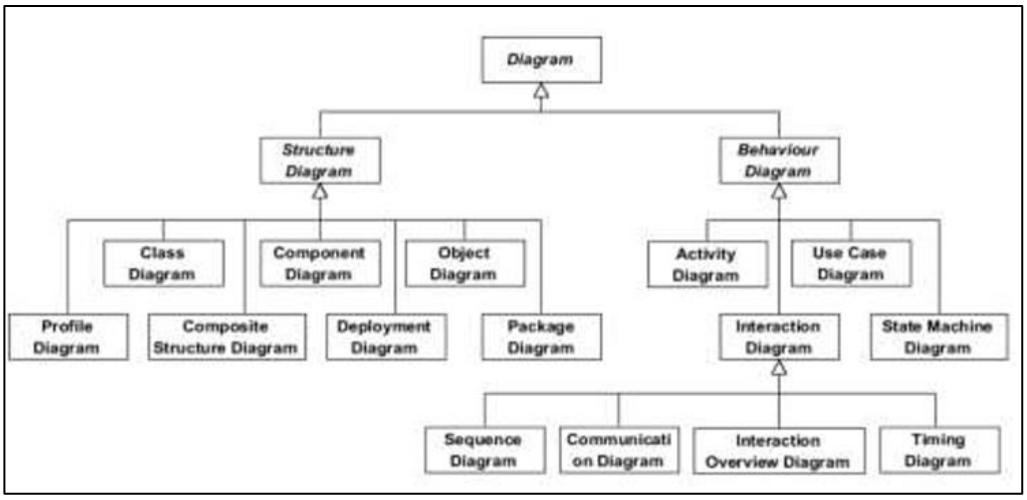


Figure 2: UML diagram

2. Work Breakdown Structure (WBS)

Work Breakdown structure is one of the most important tools used by many organizations at the beginning for proper project planning. It is the process of breaking down projects into smaller or different milestones to complete work in an effective or more manageable way. So that we can finish work with integrated scope, cost and schedule time in the targeted date (Duke, 2025).

WBS is a visual representation which is very helpful for project managers to allow work breakdown of all the tasks required to complete projects on time. In addition, WBS prevents from common issues like missed deadlines, cost overrun, etc are seen during project development (ProjectManager.com, 2025).

The Work Breakdown Structure of Global tech Corporation system are given below:

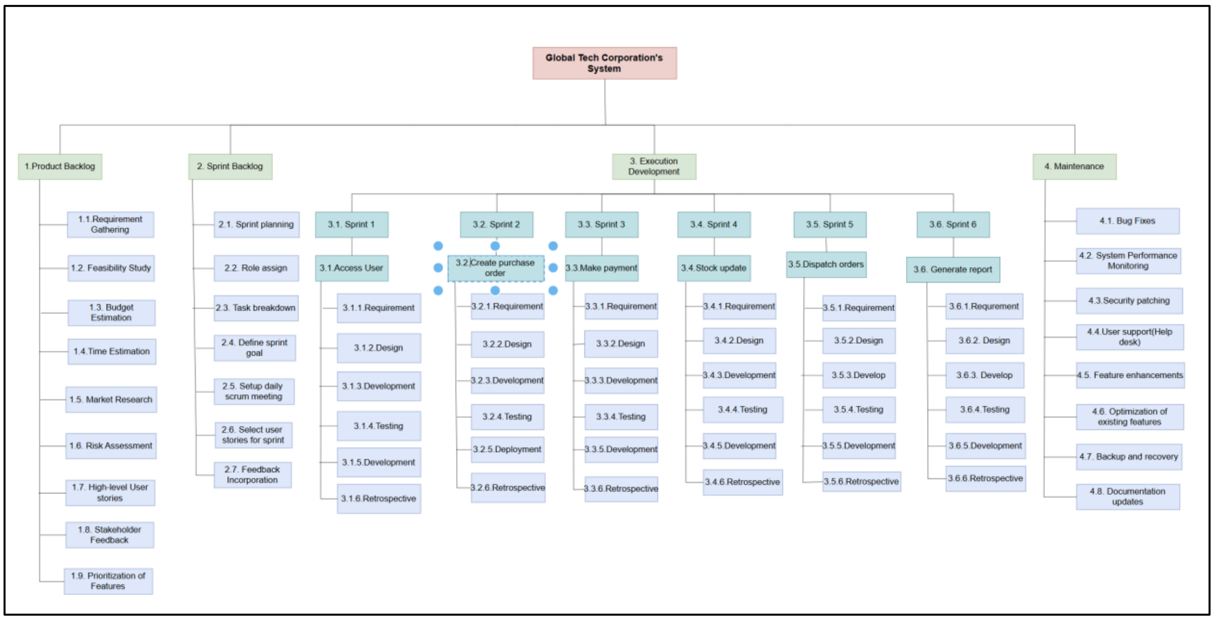


Figure 3: Work Breakdown Structure of my system

The provided WBS is a structured work breakdown of Global Tech Corporation's System based upon an Agile-based approach where the tasks are grouped logically with each steps planning. It provides guidelines to complete projects on the expected time frame in a manageable way. Like this, we can enhance project productivity, increase scalability, improve efficiency, improve project outcomes and lead to successful project of the system. The main four stages are:

a) **Product backlog**

   It includes the initial phase with requirement gathering, feasibility study, budgeting and prioritization features.

b) **Sprint backlog**

   Planning for sprints with tasks like sprint planning, role assignment, task breakdown, and setup of scrum meetings.

c) **Execution development**

   - Contains six sprints (Sprint 1 to Sprint 6) with each sprint focused on a specific feature (e.g., Access User, Create Purchase Order).
   - Each sprint follows the cycle: Requirement → Design → Development → Testing → Deployment → Retrospective.

d) **Maintenance**

   Post-development tasks like bug fixes, security patching, system monitoring, user support, feature enhancements, and documentation updates.

3. Gantt Chart

Gantt chart is a horizontal bar chart which is typically used to schedule, track and communicate deliverables, deadlines, dependencies, and resource assignments. It usually breaks down complex works into simpler form with different milestones and certain deadlines (teamgantt, 2025).

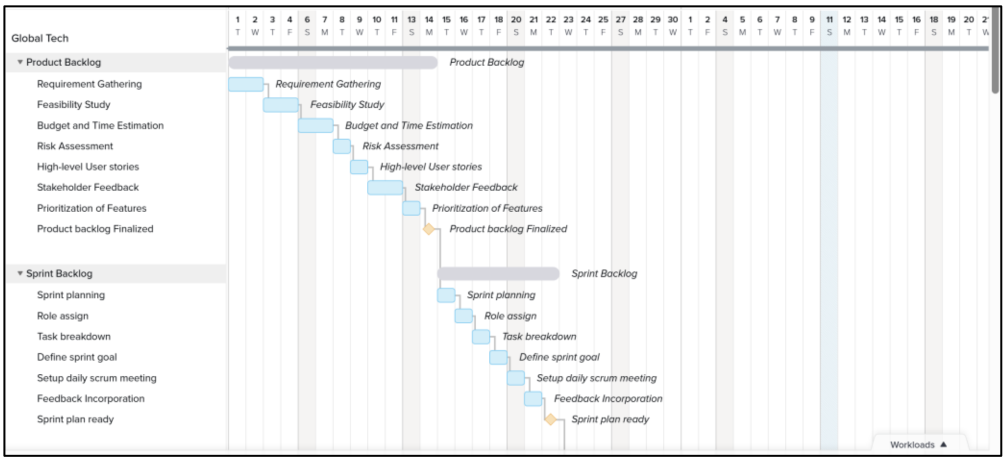


Figure 4: Gantt Chart(i)

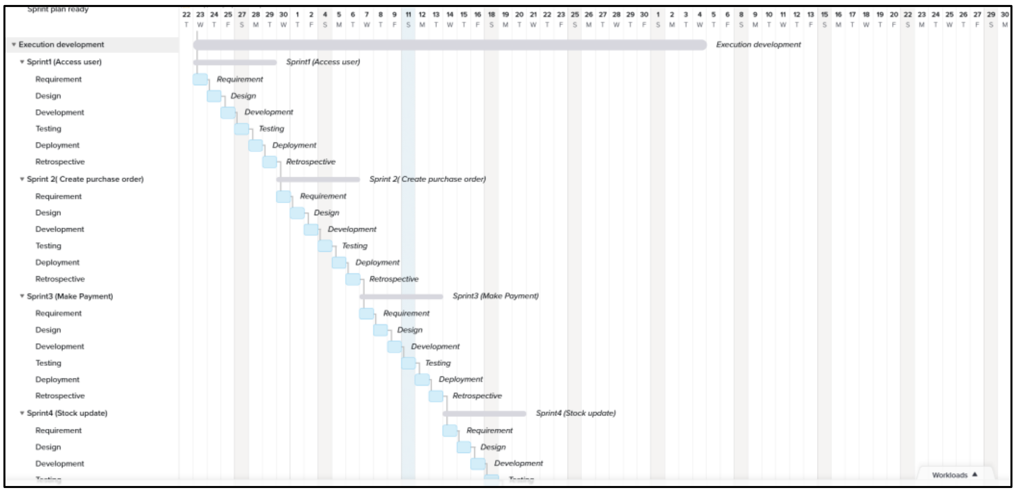


Figure 5: Gantt Chart(ii)

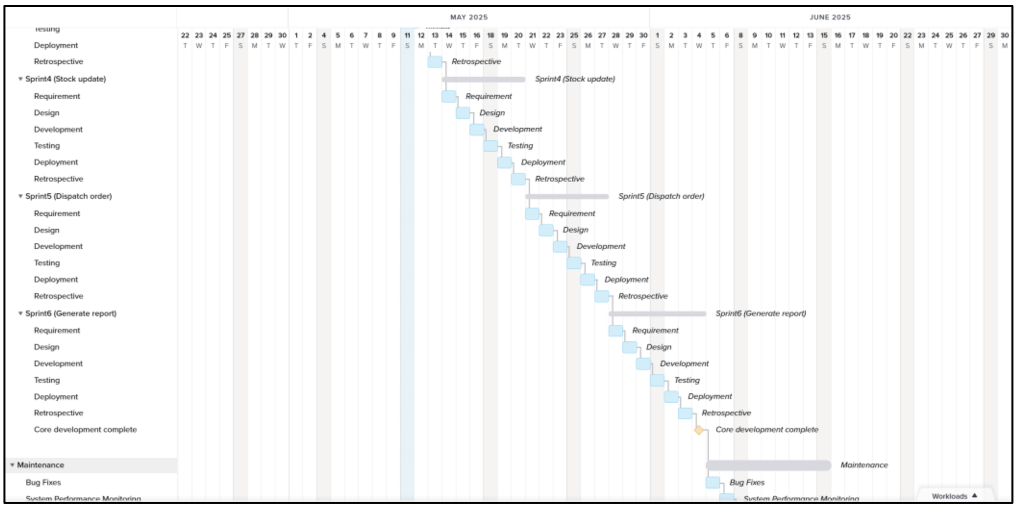


Figure 6: Gantt Chart(iii)

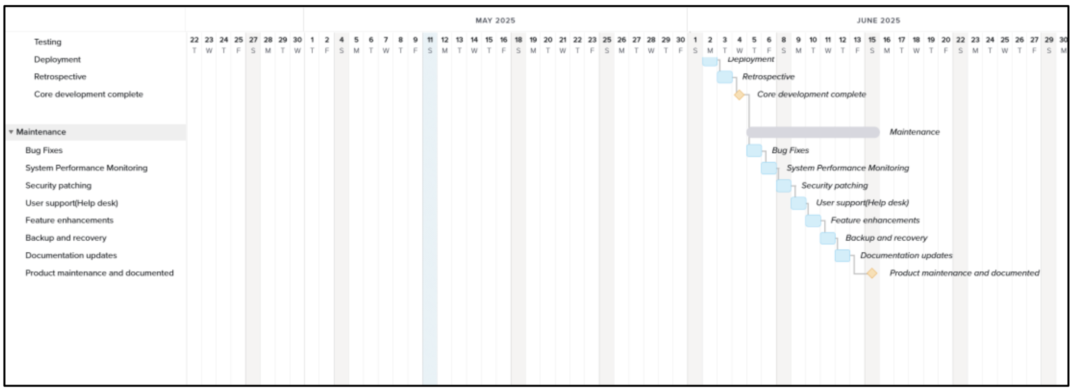


Figure 7: Gantt Chart(iv)

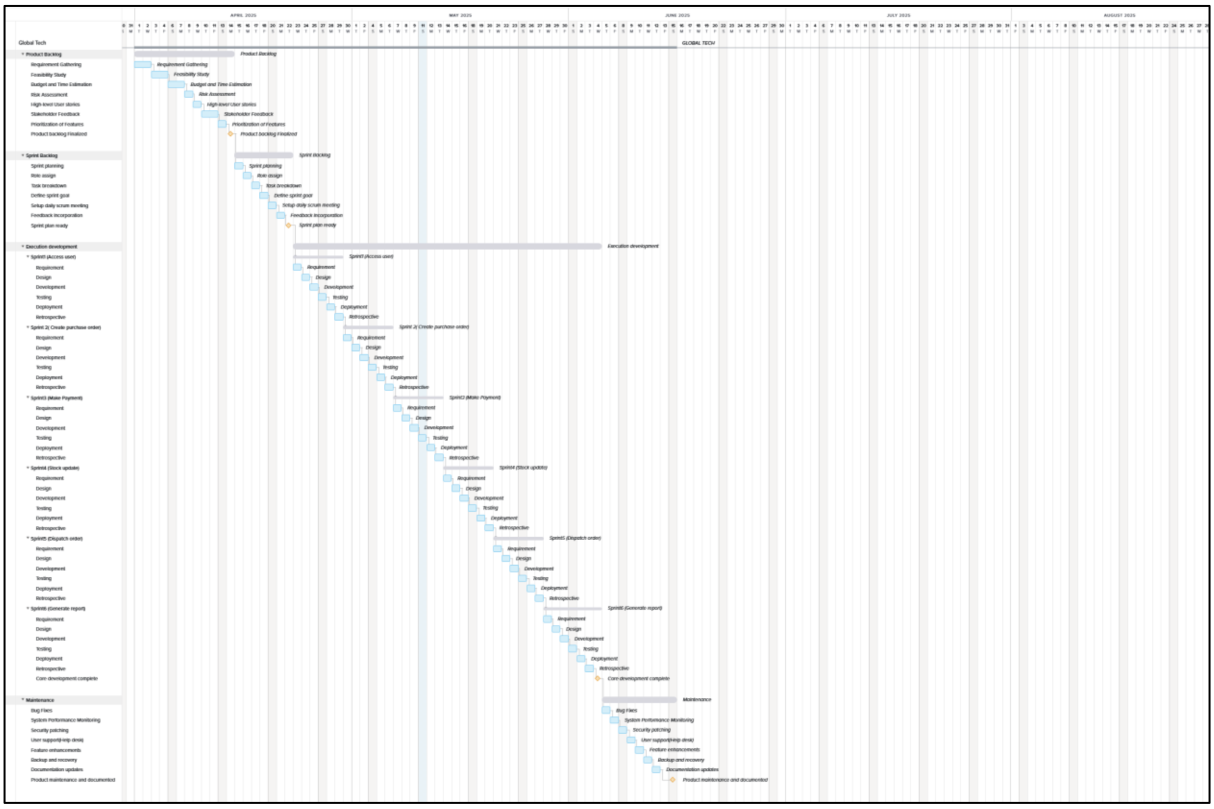
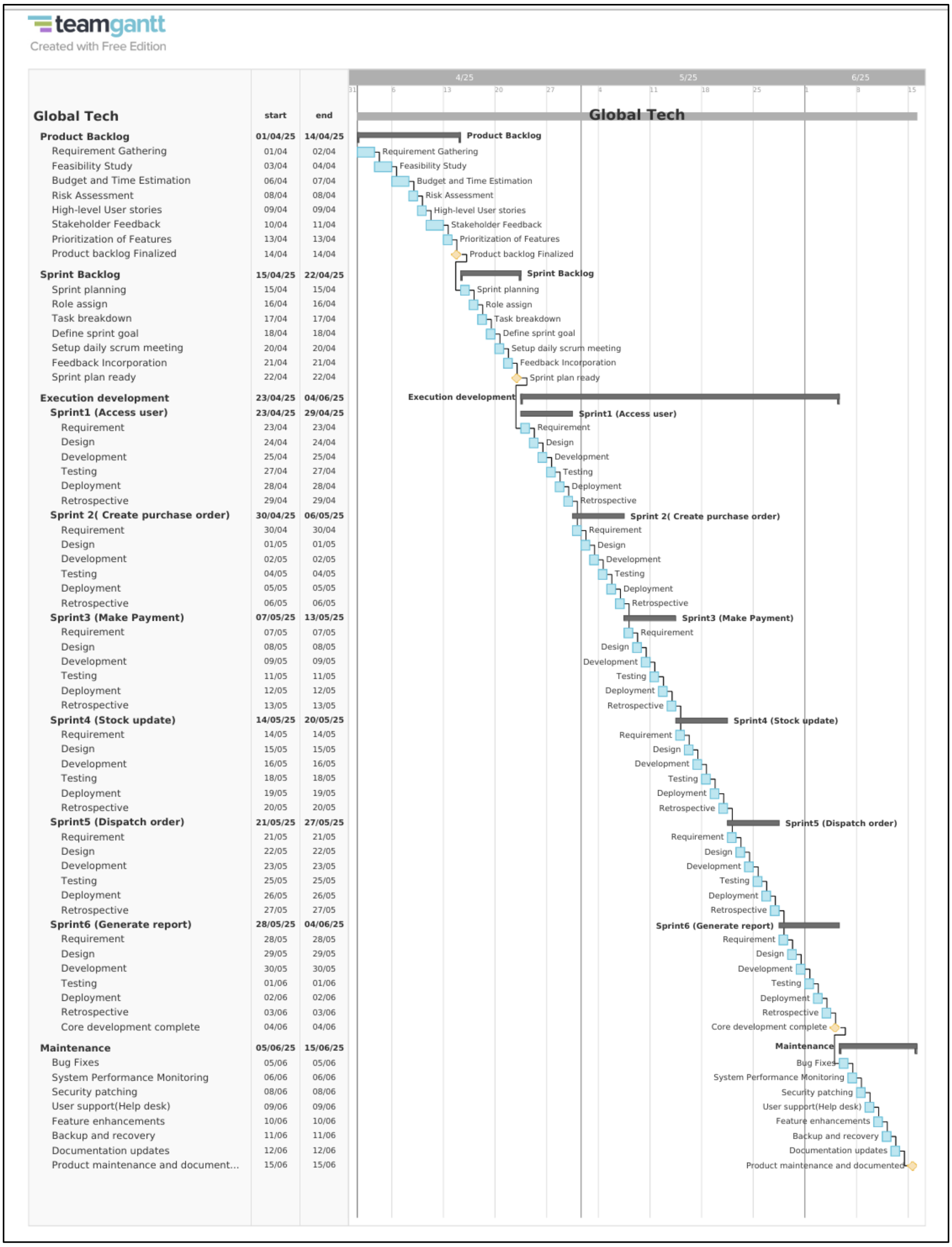


Figure 8: Overall Gantt Chart of Global Corporation



Global Tech	start	end
Product Backlog	**01/04/25**	**14/04/25**
Requirement Gathering	01/04	02/04
Feasibility Study	03/04	04/04
Budget and Time Estimation	06/04	07/04
Risk Assessment	08/04	08/04
High-level User stories	09/04	09/04
Stakeholder Feedback	10/04	11/04
Prioritization of Features	13/04	13/04
Product backlog Finalized	14/04	14/04
Sprint Backlog	**15/04/25**	**22/04/25**
Sprint planning	15/04	15/04
Role assign	16/04	16/04
Task breakdown	17/04	17/04
Define sprint goal	18/04	18/04
Setup daily scrum meeting	20/04	20/04
Feedback Incorporation	21/04	21/04
Sprint plan ready	22/04	22/04
Execution development	**23/04/25**	**04/06/25**
Sprint1 (Access user)	**23/04/25**	**29/04/25**
Requirement	23/04	23/04
Design	24/04	24/04
Development	25/04	25/04
Testing	27/04	27/04
Deployment	28/04	28/04
Retrospective	29/04	29/04
Sprint 2(Create purchase order)	**30/04/25**	**06/05/25**
Requirement	30/04	30/04
Design	01/05	01/05
Development	02/05	02/05
Testing	04/05	04/05
Deployment	05/05	05/05
Retrospective	06/05	06/05
Sprint3 (Make Payment)	**07/05/25**	**13/05/25**
Requirement	07/05	07/05
Design	08/05	08/05
Development	09/05	09/05
Testing	11/05	11/05
Deployment	12/05	12/05
Retrospective	13/05	13/05
Sprint4 (Stock update)	**14/05/25**	**20/05/25**
Requirement	14/05	14/05
Design	15/05	15/05
Development	16/05	16/05
Testing	18/05	18/05
Deployment	19/05	19/05
Retrospective	20/05	20/05
Sprint5 (Dispatch order)	**21/05/25**	**27/05/25**
Requirement	21/05	21/05
Design	22/05	22/05
Development	23/05	23/05
Testing	25/05	25/05
Deployment	26/05	26/05
Retrospective	27/05	27/05
Sprint6 (Generate report)	**28/05/25**	**04/06/25**
Requirement	28/05	28/05
Design	29/05	29/05
Development	30/05	30/05
Testing	01/06	01/06
Deployment	02/06	02/06
Retrospective	03/06	03/06
Core development complete	04/06	04/06
Maintenance	**05/06/25**	**15/06/25**
Bug Fixes	05/06	05/06
System Performance Monitoring	06/06	06/06
Security patching	08/06	08/06
User support(Help desk)	09/06	09/06
Feature enhancements	10/06	10/06
Backup and recovery	11/06	11/06
Documentation updates	12/06	12/06
Product maintenance and document...	15/06	15/06

Figure 9: Overall Gantt chart of Global Corporation

I have created Gantt chart from team gantt with **four major groups** and other sub-groups:

a) **Product backlog**
b) **Sprint backlog**
c) **Execution development**
d) **Maintenance.**

Overall, I have used hours in per sprint rest are per day. I have shown dependencies and milestone per chunks as per the requirements. I have used four milestone name product backlog finalized, sprint plan ready, core development complete and product maintenance and documented. My gantt chart has followed Agile methodology where feedback and update are done after completion of each sprint.

4. Use case Diagram

Use Case Diagram is one type of Unified Modeling Language (UML) diagram (behavior diagram) which shows the connection between actors and a system based on the requirement provided. It includes a high-level view of the system's functionality. The notations are:

a) Actors

Actors are the external entities which interact with the system. Like, external entities or entities in the DFD (Data Flow Diagram), actors connect with system usecase. It is represented by stick figures (geeksforgeeks, 2025).

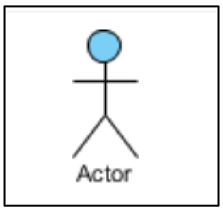


Figure 10: Notation of Actors

b) Use case

Use case are like scenes in the play. They are horizontally shaped ovals that represent the system function (Lucidchart, 2025). Every actor must linked to at least one use case in the system while some usecases may not connect to actors (visual-paradigm, 2025).

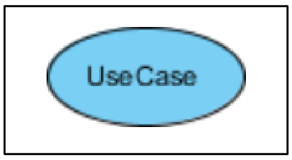


Figure 11: Notation of use case

c) Connection Link

The straight line that connects and interacts between actors and use case by solid link (visual-paradigm, 2025).

Figure 12: Notation of connection link

d) Boundary

The visual representation of system with certain scope or limits that separates use case from its external entities (geeksforgeeks, 2025).

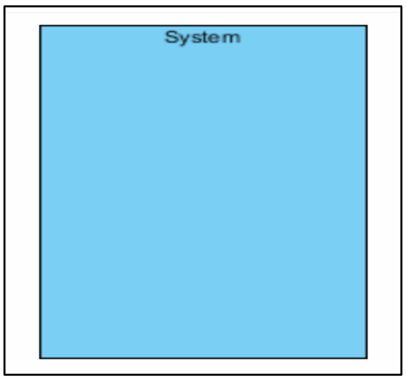


Figure 13: Notation of boundary

The use case diagram for Global Tech Management System's is given below:

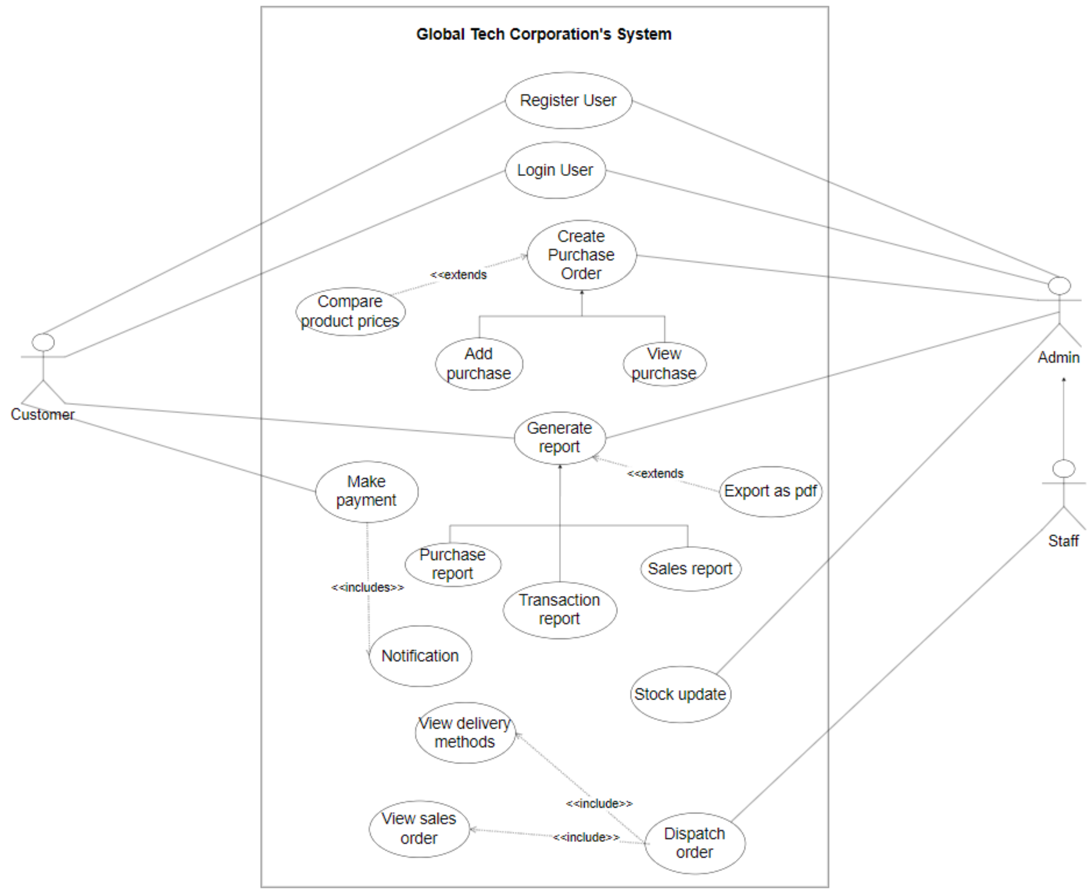


Figure 14:Use case diagram

The above use case diagram includes functionalities for the Global Tech Corporation's system. It is created to fulfill the needs of customers with the domain interest of buyers. The system provides various functionalities such as user authentication(register/login), purchasing product (add, edit, remove, view and update), make payment, dispatching order, generate report and so on.

4.1. High Level Use Case Description

a) Register User

Actors: Admin & Customer

Description: A new user (Admin/ Customer) can register by providing necessary details like name, address, role, email, etc. The system validates the information. The system stores information only if it is valid, and he/she receives confirmation of registration. Admin can also register staff accounts.

b) Login user

Actors: Admin & Customer

Description: This use case handles **the authentication.** The system verifies credentials, and allow to enter the system if valid or shows errors and put security measures like password reset and auto logout after multiple failed attempts.

c) Create Purchase Order

Actors: Admin

Description: After login, the Admin can create and manage purchase orders by adding products, setting quantities, and updating supplier details, while customers can optionally view product price comparisons. Once finalized, the order is saved with a status (pending, approved, received) and stock is updated accordingly.

d) Generate report

Actors: Admin & Customer

Description: This use case allows the user to generate different types of reports. Admin can generate reports like sales and transactions. Customers can generate reports on their purchase history, payments and delivery status. The system generate report and exported in the form of PDF format for easy sharing and record-keeping.

e) Make payment

Actors: Customer

Description: After confirming an order, the customer selects a payment method (card, mobile banking, etc.). The system securely processes the payment, updates the order status to "Paid", sends a receipt via email, and notifies the admin and staff. Failed transactions are logged, and customers are prompted to retry.

f) Stock update

Actors: Admin

Description: This use case allows the Admin to manage and update stock levels in the system. Admins can add new products, edit existing product details and remove items. The system update any items based on real time.

g) Dispatch order

Actors: Admin & Staff

Description: This use case handles view sales order and updates the delivery methods. Staff members handle the dispatching of goods. Admin can view and manage all recent orders details and update if necessary.

4.2. Expanded Use Case Description

a) Register User

Actor: Admin and Customer

Description: A new user (Admin/ Customer) can register by providing necessary details like name, address, role, email, etc. The system validates the information. The system stores information only if it is valid, and he/she receives confirmation of registration. Admin can also register staff accounts.

Typical course of Events:

Actor Action	System Response
1. The user clicks on the **"Register"** option.	2. The system shows a registration form requiring the details like Name, email, phone, admin role,etc.
3. The user fills in the required details and submits the form.	4. The system checks whether the email and username are unique or not.
	5. If the details are valid, the system saves the user information.
	6. The system confirms successful registration.
7. The user is now eligible to log in.	

Table 1: Expanded description of "Register user"

Alternative courses:

Line 4: If the email or username already exists, the system displays an error message: “Email is already registered, use another email or login in”.

The user can either enter different emails and proceed to login.

Line4: If any black field is left, the system highlights missing fields and prompts user to complete the field.

Line4: If password don’t meet security requirements then the system provide invalid password.

b) Make payment

Actor: Customer

Description: After confirming an order, the customer selects a payment method (card, mobile banking, etc.). The system securely processes the payment, updates the order status to "Paid", sends a receipt via email, and notifies the admin and staff. Failed transactions are logged, and customers are prompted to retry.

Typical course of Events:

Actor Action	System Response
1. The customer selects to the “payment section” option.	2. The system displays available payment methods (online payment, handincash,etc).
3. The customer chooses a payment method and enters payment details.	4.The system verifies the payment details.
	5.If the system valids, it further processes the payment by confirming product availability and calculation of costs.
	5. On successful confirmation, the system updated to "Paid" and sends notification.

Table 2:Expanded use case description of "Make Payment"

Alternative courses:

Line2: If payment details are incorrect, the system shows error message to the customer to re-enter them.

Line4: If payment is declined, the system provides an error message and asks for another method.

5. Collaboration diagram

A collaboration diagram, also known as a communication diagram, which illustrates the relationships and interactions among objects and the system in the Unified Modeling Language (UML). Sequence diagrams and collaboration diagrams deliver similar content but show it in different ways (visual-paradigm, 2024).

The notations of collaboration diagram are represented as following:

a) Object

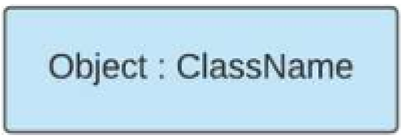


Figure 15: Notation of "Object"

b) Actors: It invokes interaction and must have unique name.

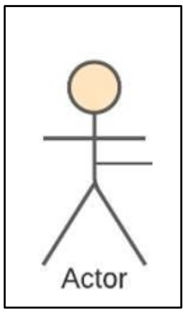


Figure 16: Notation of "Actor"

c) Link: It connects between actors and objects and single link is enough to support more than one messages.

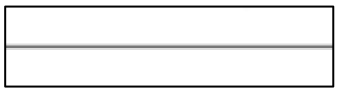


Figure 17: Notation of "Link"

d) Messages

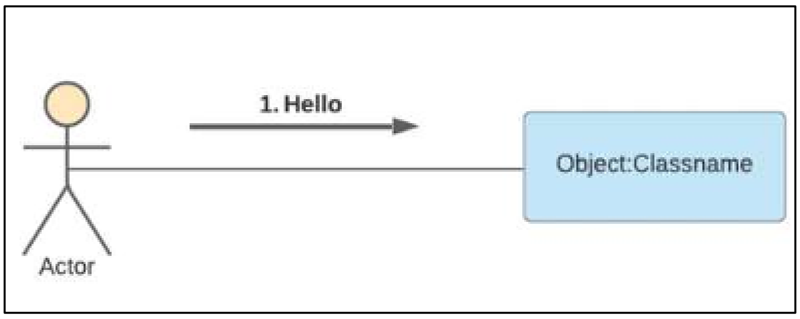


Figure 18: Notation of "messages"

(visual-paradigm, 2024)

I have studied one of the expanded descriptions usecase .i.e. “make payment” to understand the interaction between actors and the system in collaboration diagram. The possible domain classes from my use case are: **Order, Stock and Notification.**

Here, I have one **primary actor: Customer**, **boundary object: PaymentUI**, **control object: Payment** and **other object like Payment, Product, User and notification**. The actor connects with the boundary system. The control object will handle connection between actor and different object. If the payment is successful, the actor receives notification messages and if fails then also, it receives failure transaction.

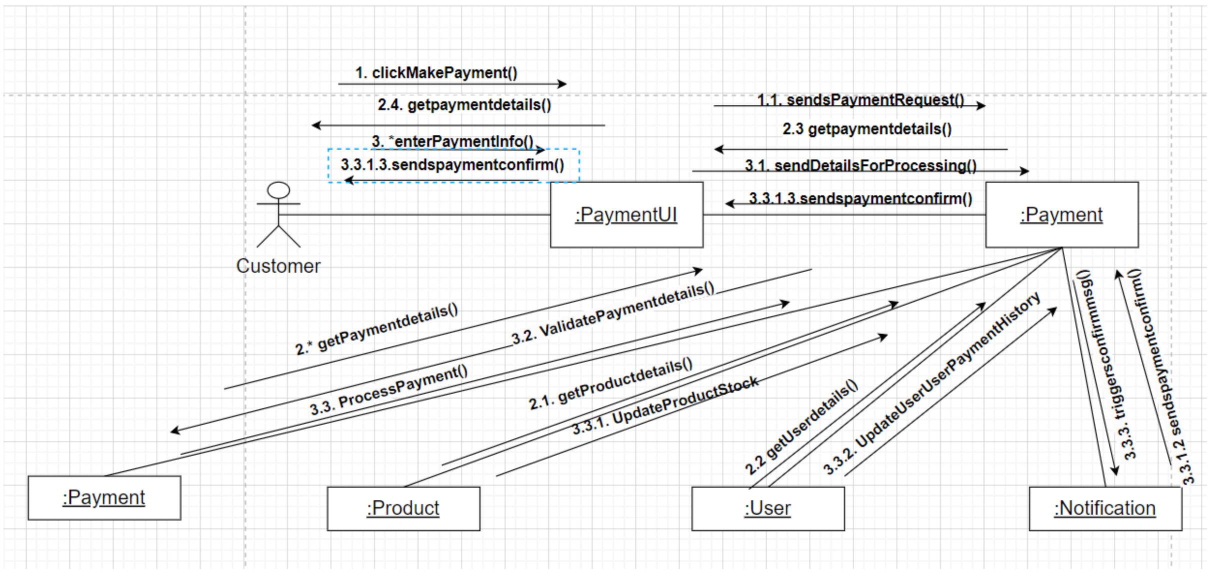

1. clickMakePayment()
2.4. getpaymentdetails()
3. *enterPaymentInfo()
3.3.1.3.sendspaymentconfirm()
1.1. sendsPaymentRequest()
2.3 getpaymentdetails()
3.1. sendDetailsForProcessing()
3.3.1.3.sendspaymentconfirm()
:PaymentUI
:Payment
Customer
2.* getPaymentdetails()
3.2. ValidatePaymentdetails()
3.3. ProcessPayment()
2.1. getProductdetails()
3.3.1. UpdateProductStock
2.2 getUserdetails()
3.3.2. UpdateUserUserPaymentHistory
3.3.3. triggersconfirmmsg()
3.3.1.2 sendspaymentconfirm()
:Payment
:Product
:User
:Notification

Figure 19: Collaboration diagram of "Register user"

6. Activity diagram

Activity diagram is one type of behavior diagram in Unified Modeling Language (UML) that represents the flow of works with different activities or processes within the system. As from the name, we came to know it represents how a system performs activity and moves from one state to another state. Also, it is useful in several ways like modeling workflows, simplifying complex logic, system design and analysis and describing different usecases (geeksforgeeks, 2025).

The notations of activity diagram are:

a) **Initial state:** This is the initial point of the diagram and represented by black filled circle.

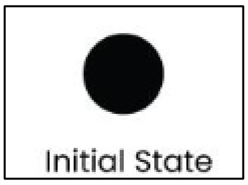


Figure 20: Notation of "Initial state"

b) **Action flow:** This is notation to determine the path from one activity state to another in the system. It is represented by arrow with dark head.

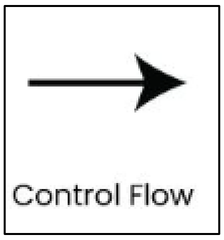


Figure 21: Notation of "Action flow"

c) **Activity:** Activity represents any action or event that takes place with in system and represented by rectangle with rounded corners.

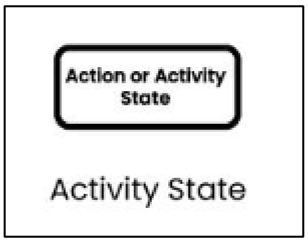


Figure 22: Notation of "Activity"

d) Decision node: This is actually a notation with two different inputs with conditions to make decisions as it is in diamond shape.

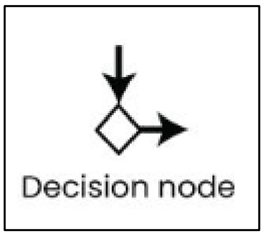


Figure 23: Notation of "Decision node"

e) **Fork:** It supports concurrent activities.i.e. no decision is made before splitting the activity into two parts.

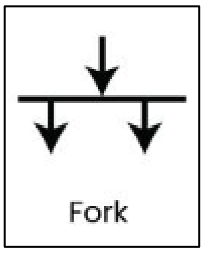


Figure 24: Notation of "Fork"

f) Join: It supports concurrent activities converging into one.

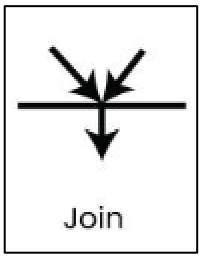


Figure 25: Notation of "Join"

g) **Merge:** We use merge node to combine different activities of decision node into one to proceed another activity in the system.

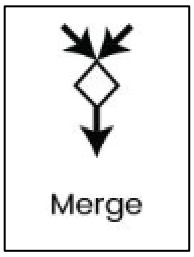


Figure 26: Notation of "Merge"

h) **End state:** The state where the activity ends. We use filled circle within circle notation.

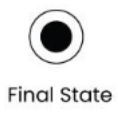


Figure 27: Notation of "End state "

i) **Swimelane:** It divides the activities to represent responsibilities over different actors.

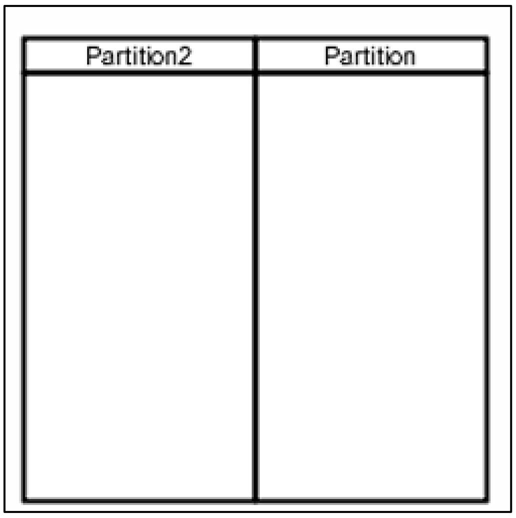


Figure 28: Notation of "Swimlane"

(geeksforgeeks, 2025)

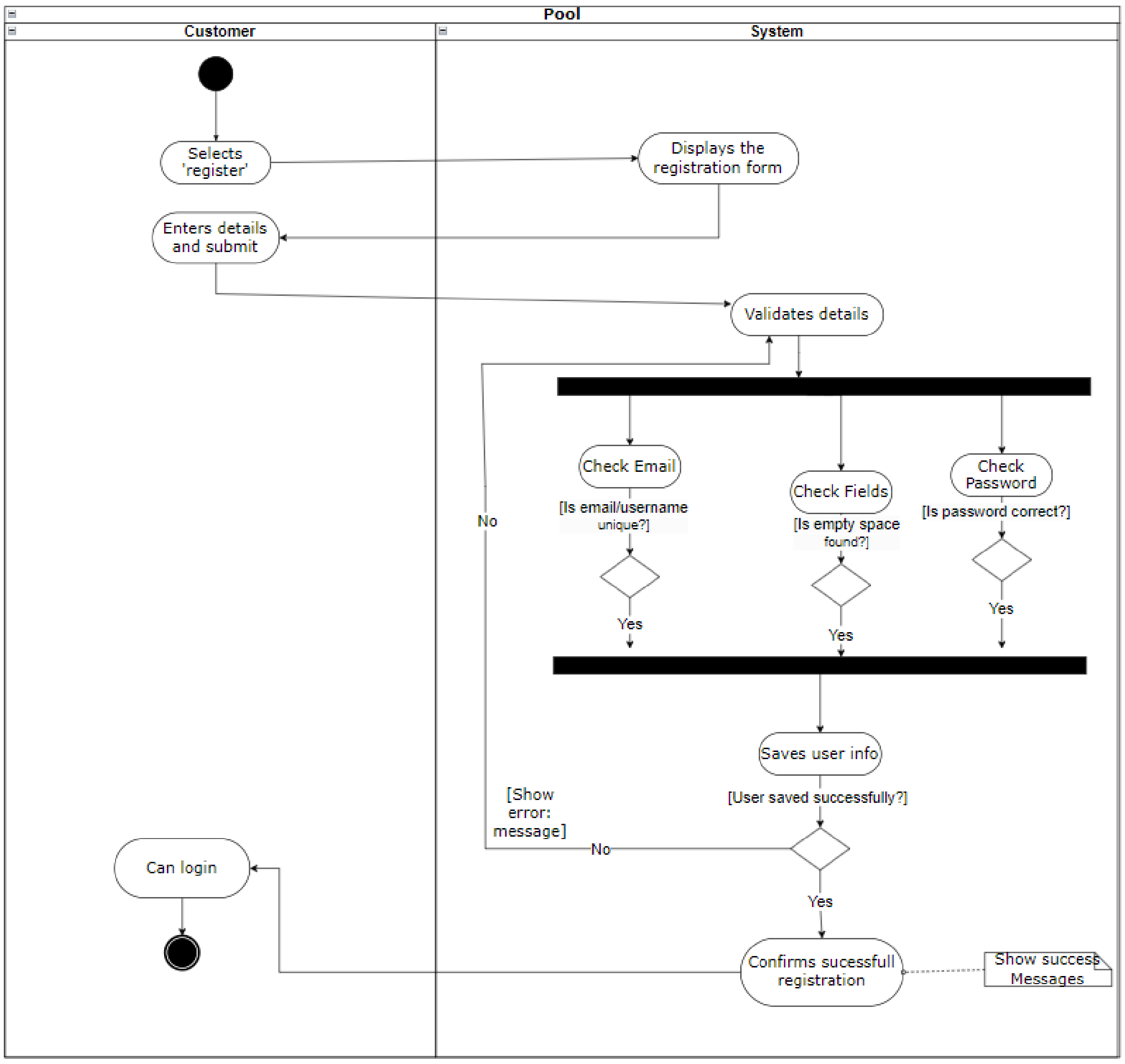


Figure 29: Activity Diagram "Register User"

7. Class diagram

Class diagram is one of the elements of UML (Unified Modeling Language) which shows the classes, attributes, methods and relationships among them of the systems. It makes it easier to interact with the system. It is used for constructing and visualizing object-oriented systems. It plays a crucial role in the software development lifecycle (geeksforgeeks, 2025).

7.1. Class diagram

The **class diagram notations** are:

a) **Class Name:** Class Name is typed at the top of box of the class, is centered and in bold.

b) **Attributes:** Attributes are class or behavior characteristics. It is written in the second compartment of box with visibility such as (.i.e. public, private) and each data type of attribute.

c) **Methods:** Methods are the operations or functions of the class. It is declared in the third compartment of the class with included visibility (private,public), return type and parameter of a method.

d) **Visibility Notations:** Visibility notations are the accessibility level of attributes and methods.

Public = "+" Accessible to all classes

Private= "-" Accessible only within the class

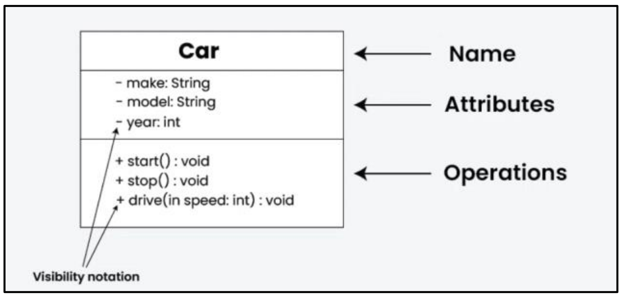


Figure 30: Class diagram notations

Domain class represents the concepts or main entity from the system. It directly corresponds to class in object-oriented programming and represents a real-world entity with attributes and behaviors (geeksforgeeks, 2024).

7.2. Relationships

a) Association

Any relationship between classes is called Association. It represents as solid line connecting two classes (geeksforgeeks, 2025).

Figure 31: Association relationship of class diagram

In my case, I have many classes which has connection to each other like Admin has connection with reports, customer with purchase order and staff with delivery details using multiplicity.

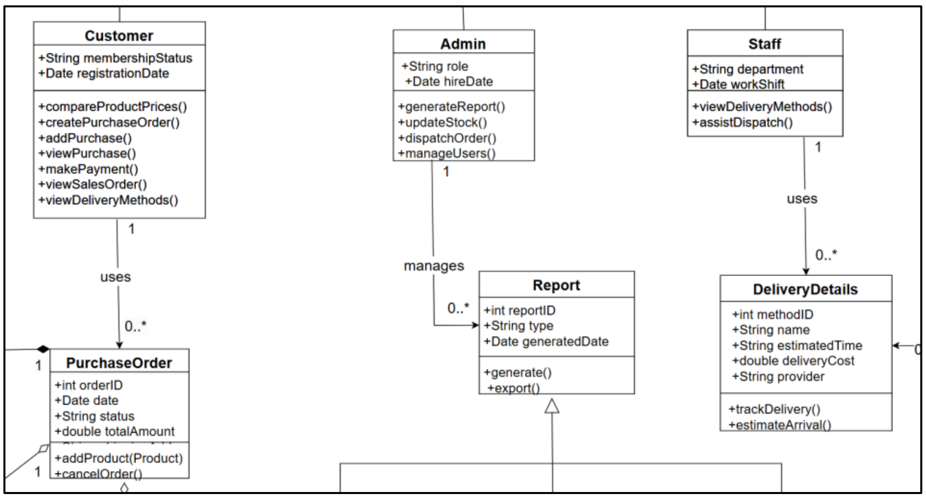


Figure 32: Example of Association relationship in my case

b) Inheritance

When child class inherits properties of parent class is called inheritance (geeksforgeeks, 2025).

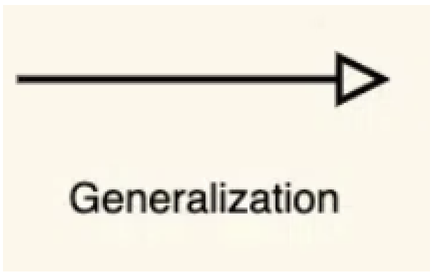


Figure 33: Inheritance relationship notation of class diagram

In my case, I have inheritance relationships in reports and user. The sub-class like customer, staff and admin inherit its parent property from user and similarly goes to report.

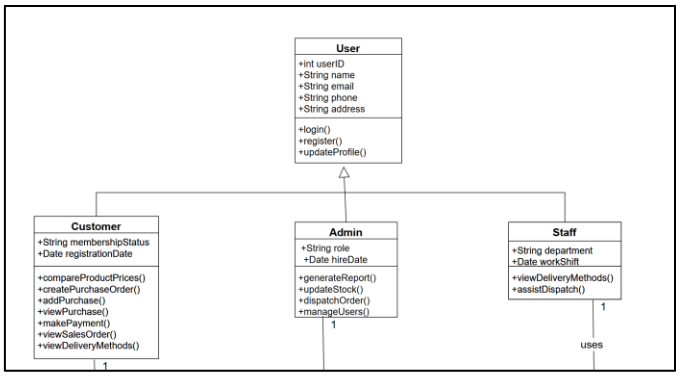


Figure 34: Example of Inheritance relationship in my case

c) Aggregation

Aggregation represents 'has a' relationship with solid line unfilled diamond. In this kind of relationship, the child class can exist independently of its parent class (geeksforgeeks, 2025).

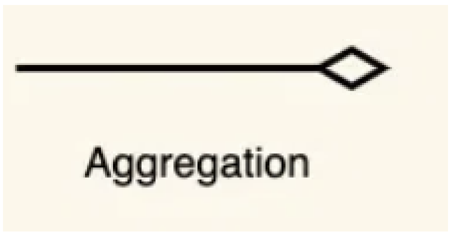


Figure 35: Aggregation relationship of class diagram

Like, Purchase order is considered as whole while payment and product are the parts of it. Product and payment can be multiple and both belongs to the purchase order. As, it means both are exist independently.

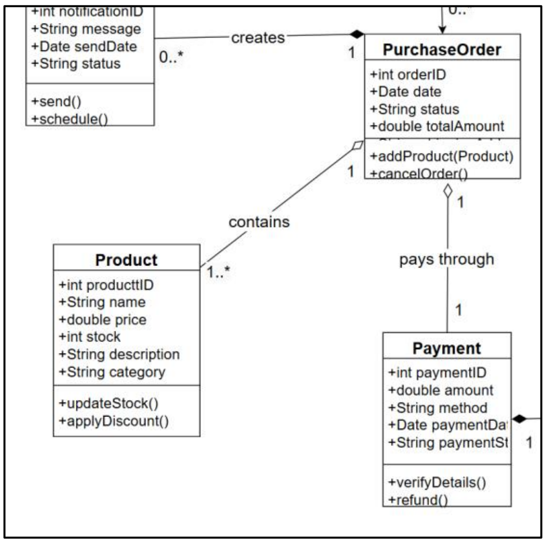


Figure 36: Example of Aggregation relationship in my case

d) Composition

Composition is the stronger form of aggregation. The part of class cannot exist independently and filled with diamond shape (geeksforgeeks, 2025).

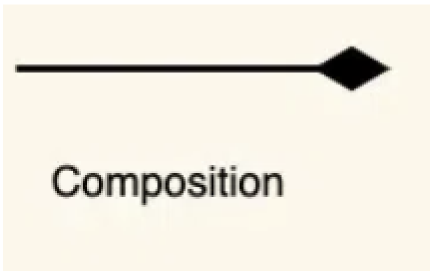


Figure 37: Composition relationship of class diagram

Here, In my case, I have payment details and notification as both are impossible without payment and purchase order. As, each payment details is possible only after payment and notification is possible only after purchase order. So, existence of one is important to other.

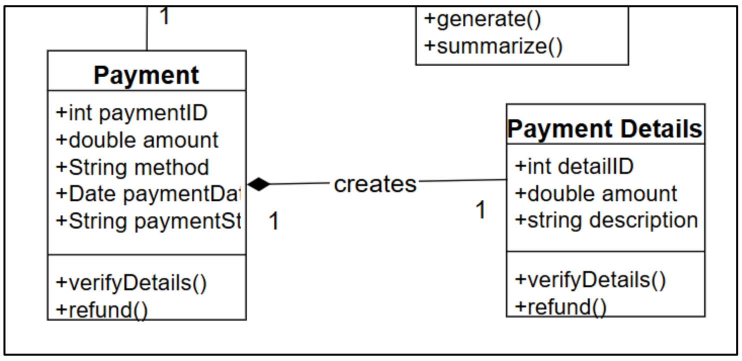


Figure 38: Example of Composition relationship in my case

e) Dependency

A dependency exists between two classes when one class relies on another, but the relationship is not as strong as association or inheritance (geeksforgeeks, 2025).

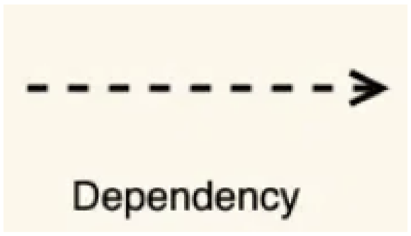


Figure 39: Dependency relationship notation

7.3. Multiplicity

Multiplicity describes how many **instances of one class can be connected to an instance of another class** through a given association.

Multiplicity can be expressed in the following ways:

- Exactly one - 1
- Zero or one - 0..1
- Many - 0..* or *
- One to many - 1..*
- exact number of instances - 1, 3, 6
- Default multiplicity is exact one

(UMLBoard, 2024)

The Class diagram with domain classes and usecases are mentioned below:

S.N.	Use case	Possible Domain Class
1.	Register user	User, RegistrationService, Profile
2	Login user	User, AuthenticationService
3	Create purchase order	PurchaseOrder, Product, PurchaseService, User
4	Compare purchase order	Product, PricingService
5	Add purchase	Purchase, Product, Cart, PurchaseService
6	View purchase	PurchaseOrder, Product
7	Generate report	Report, ReportService
8	Export as pdf	Report, PDFExportService
9	Make payment	Payment, PaymentService, PurchaseOrder, NotificationService
10	Notification	Notification, NotificationService
11	View delivery methods	DeliveryMethod, DeliveryService

12	View sales order	SalesOrder, Product, Customer
13	Dispatch order	Order, Inventory, DispatchService, DeliveryService
14	Stock update	Inventory, Product, StockService
15	Purchase report	Report, PurchaseOrder, User
16	Transaction report	Report, Payment, transactionService
17	Sales report	Report, SalesOrder

Table 3: Relationship between use case and possible domain class

S.N.	Domain Class	Use case
1.	User	Register user Login user
2.	Customer	Compare product prices Create purchase order Add purchase View purchase Make payment View sales order View delivery methods
3.	Admin	Generate report Update stock Dispatch order
4.	Staff	View delivery methods
5.	Product	Compare product prices Create purchase order Add purchase View purchase Update stock
6.	Purchase order	Create purchase order Add purchase View purchase Dispatch order Make payment
7.	Payment	Make payment Transaction report
8.	Notification	Make payment
9.	Dispatch order	View sales order View delivery methods

		Dispatching details
10.	Report	Generate report
11.	Purchase report	Generate report
12.	Sales report	Generate report
13.	Transaction report	Generate report Make payment
14.	Delivery methods	Dispatch order

Table 4: Class Diagram

7.4. Whole class diagram

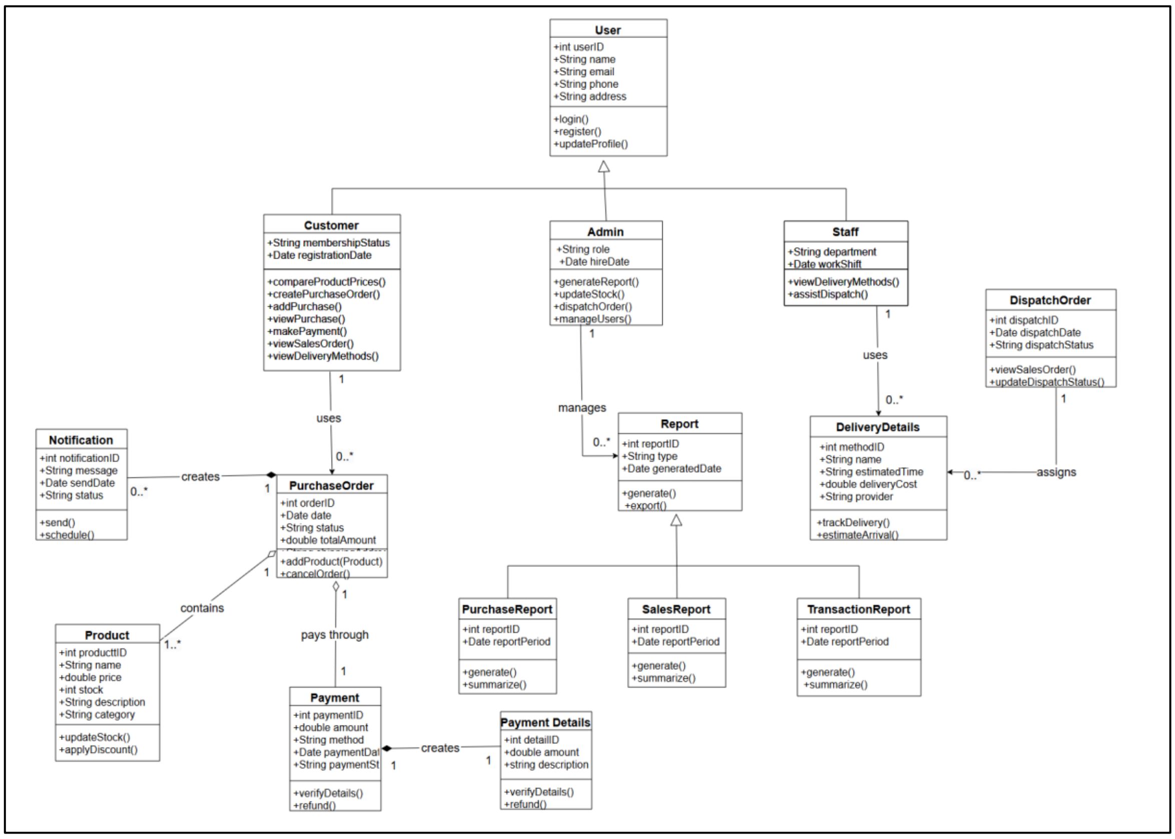


Figure 40: Class diagram in tabular format

In my project, I have 2 generalizations (User and Report), two aggregations (payment and product) and two compositions (notification and purchase details) with multiplicity.

8. Further Development

While developing my project, I have used object-oriented analysis and design diagrams. Now, it’s time to plan my next steps for further development so I need to choose the right patterns, technologies and tools to make my system more flexible, efficient and run more smoothly. This part is the most important phase to determine whether my system is alright with all the functions and matches the requirements of the users or not. So, Let’s go through many research and suggestion and apply in our system to proceed in the best possible way (asana, 2025).

I have chosen the **agile methodology** in my system as it works with scrum framework which includes sprints, provides the user feedback from users and stakeholders and keeps tracking the projects again and again. As, Agile breaks projects into several dynamic phases which are commonly known as sprints. Agile framework is an iterative methodology in which every sprint team reflects is most and improved in time and again (asana, 2025).

Agile create a flexible and responsive change to the project as it responds quickly without exhausted and complete project on time. It helps to improve collaboration between developers, stakeholders and users. Agile follows the **user centric approach** due to which we can track project on time using **tools** like Kanban boards, burndown charts, etc for visualizing tasks, progress and dependencies. It simply **improved risk management**, **iterative development for continuous improvement** and supporting **faster and more adaptive development**. This is how agile is most suitable methodology for my project to complete dynamic feature projects on time.

8.1. Architectural choices

The primary reason of choosing this architecture is due to its flexibility and scalability. Each microservice can be developed, tested, deployed and maintained independently which helps to detects bugs timely and updates faster. Since, each service operates independently, failure in one service don't crash entire system which can increase reliability. With microservices, development can happen simultaneously across different modules allowing the team members to work autonomously which can enhance team productivity and reduces interdependencies (geeksforgeeks, 2025).

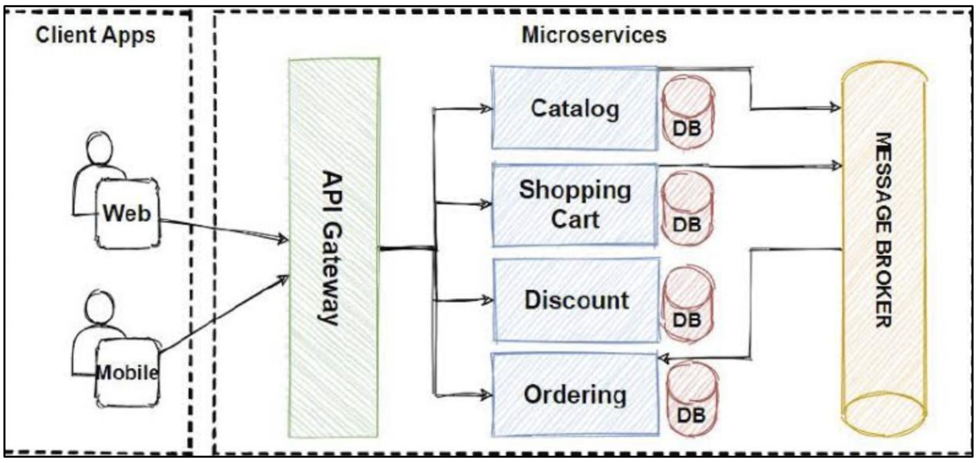


Figure 41: Microservice Architecture

In my project, Agile methodology offers an ideal architectural structure which are mentioned below:

- **User Authentication Service**
- **Product Management Service**
- **Payment Service**
- **Stock update service**
- **Order dispatch service**
- **Generate report service**
- **Review and Feedback Service**

With a clear scope and deliverables, each service is developed independently within Agile sprints. This matches the **iterative and incremental nature of Agile**, where each sprint delivers a functional module that can be tested and deployed.

By using **Microservices Architecture**, I ensure that my system is **scalable**, **easily deployable**, and aligned with required features which makes the overall development more structured, efficient, and long lasting.

8.2. Design Pattern

To enhance my project, I have chosen MVC (Model View Controller) design pattern. MVC is widely used architectural pattern in today's development practice to make system more scalable, efficient and flexible. The Model View Controller (MVC) design pattern specifies that an application consists of a data model, presentation information, and control information (geeksforgeeks, 2025). It divides application into three interconnected components as follows:

8.2.1. Model

Model is one of the components which represents the core **data**, **state**, and **business logic** of the application. It is responsible for:

- Communicating with the **database** or external data sources.
- Applying **validation** rules and business logic.
- Managing data updates and storing application state.

In my project, the model includes Java classes like user, product, order etc which directly map to database tables and handle core operations such as fetching, storing, or updating records. Model is applied in business logic without affecting data in the interface.

8.2.2. View

The View is responsible for presenting data to the user in a structured format.

It handles:

- The UI/UX components of the application.
- Displaying dynamic content passed from the controller.
- Managing user interactions visually.

In my project, JSP (JavaServer Pages) files are used to make login page, product catalog, shopping cart, and order confirmation. CSS and JavaScript are used to make page more responsive. We can make UI/Ux page with parallel development.

8.2.3. Controller

The Controller acts as a bridge between the View and the Model.

It:

- Receives and processes user inputs (e.g., form submissions).
- Invokes methods from the Model to update data.
- Chooses the appropriate View to display the result.

In my system, Java servlet classes serve as controllers. For example, the LoginServlet handles authentication by taking input from the login form, verifying it with the User model, and then redirecting to a dashboard or error page accordingly. Here, controller controls overall logic flows and manages different layers of application simultaneously.

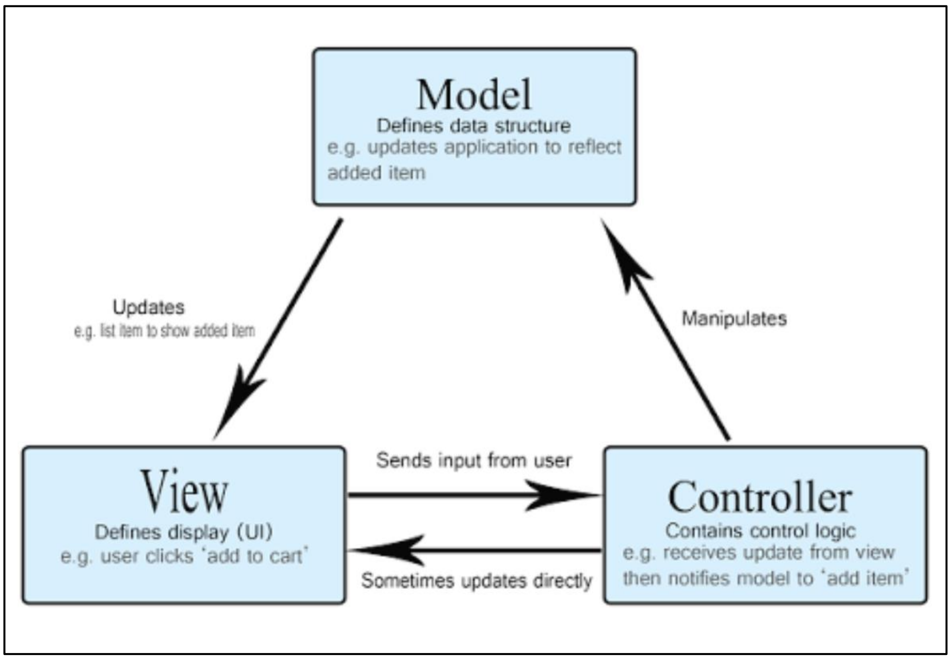

Model
Defines data structure
e.g. updates application to reflect added item
Updates
e.g. list item to show added item
Manipulates
View
Defines display (UI)
e.g. user clicks 'add to cart'
Sends input from user
Sometimes updates directly
Controller
Contains control logic
e.g. receives update from view then notifies model to 'add item'

Figure 42: MVC Architecture

MVC design helps in my projects:

- It **improves maintainability** by allowing individual components (UI, logic, data) to evolve independently.
- It improves **testability**, especially of the Model and Controller.
- It enables **role-based development**: front-end developers can focus on Views while back-end developers work on Models and Controllers.
- It **fits naturally with Agile development**, where features can be implemented and tested iteratively in components.

8.3. Development Plan

To make system more efficient and more systematic, I have a well-defined structure design and development plan. Using Agile methodology follows the portion of my plan which uses Scrum framework to breakdown the large project into smaller one. The development phases are tightly aligned with iterative cycles of **planning**, **design**, **development**, **testing**, **deployment**, and **retrospective review**. For completion of my project, I have use development phases, sprint-based timeline, milestone and dependencies.

In development phase, I have used tasks like: Requirement analysis, System design, Implementation, Testing, Deployment and Retrospective.

In sprint-based timeline:

Sprint	**Duration (Hours)**	**Key Deliverables**
1	20	User registration and login module (authentication, session)
2	18	Product catalog and category browsing
3	16	Cart and order placement modules
4	16	Payment and order tracking integration
5	15	Admin dashboard and product management
6	15	Final testing, bug fixes, deployment, and documentation

Total Sprint Hours: **100 hours**

Tools used:

a) **Draw.io**

Draw.io is an online platform or app which helps in making diagram and charts. The software allows to choose the layout or allow custom layout (Computer Hope, 2024).

Figure 43: Logo of Draw.io

b) **Figma**

It is diagrammatic tools like draw.io.

Figure 44: Logo of figma

Design: Figma (for UI wireframes) and Draw.io
Development: Java (JSP), CSS, MySQL
Deployment: Apache Tomcat, localhost
This plan ensures smooth progress of the system development while allowing flexibility and frequent evaluations for continuous improvement.

8.4. Testing Plan

Testing plan is the turning point for system security, functionality and reliability. It comprises the various structural phases to verify and authorize the system requirements under varying conditions as mentioned below:

a) White box testing

White box testing is a procedure in which the test expert has complete knowledge of the application source code along with the design documentation. Having this basic knowledge at this low level of depth allows them to do internal implementation work flow checking as well as trap faults which are not possible to find through black box or even gray box testing methodologies.

- **Unit Testing**

  Unit testing is done to check individual items or procedures of an application so that all of them run correctly. Unit testing allows the programmers to catch mistakes early and guarantees the application will still hold up to design standards throughout the creation process (CheckPoint, 2025).

- **Integration testing**

  Integration testing is that type of software testing in which the modules of the software are logically integrated and tested as a group. Integration testing basically determines if data is passing or being communicated among various modules and controls project level processes along with modules at some point of time (geeksforgeeks, 2025).

- **Sandwich testing**

  Sandwich Testing (hybrid integration testing) includes bottom-up approach testing and top-down approach testing, it also cannot be used in those systems that have a great interdependence among different modules and it accommodates parallel testing (geeksforgeeks, 2025).

- **Smoke testing**

  Smoke testing, or build verification or confidence testing, is a rapid test to determine whether a new software build is stable enough to proceed with further testing. It ensures that the main features are functioning correctly before proceeding to more comprehensive testing phase (Gillis, 2025).

b) Black box testing

Testing software which concentrates on the external behavior of a system without knowing how it is implemented inside is known as "black box" testing. Through this method, the implementors need not know the details of implementation or code organization but rather concentrate on analyzing the functionality of the software in terms of inputs and outputs. It is also known as specification-based testing and functional testing.

- **Regression Testing**

  Changes tend to cause issues in an application. While modifying a functionality or security of an application, regression testing confirms the code still passes the current test cases (CheckPoint, 2025).

- **Functional testing**

  Software testing that compares the system to functional requirements and specifications is known as functional testing. Functional testing verifies that the application satisfies the requirements or specifications. This evaluation is specifically interested in the processing result (geeksforgeeks, 2024).

- **Non-functional testing**

  Testing software that verifies the non-functional characteristics of a program is known as non-functional testing. It is intended to determine a system's preparedness on the basis of nonfunctional specifications that functional testing never considers (geeksforgeeks, 2024).

.

8.5. Maintenance plan

After completion of all the design and development, I have created a long-term plan for reliability, scalability and flexibility of my project. This plan outlines how I'll continue to manage and update the system after its initial deployment. The detailed maintenance plans are listed below:

- I'll set up a monitoring system to keep track of the system's performance, availability, and resource usage continuously to spot the error and resolve quickly.
- I'll follow a structured process to address and fix any bugs or issues reported by users in a timely manner.
- I'll make sure to keep up with the latest software updates, patches, and security fixes provided by third-party vendors for the system's underlying components and dependencies.
- As, when we used new technology, user needs to evolve, I'll be ready to make improvements and adapts the system per requirements to perform well in long terms.

9. Prototype

9.1. Register page

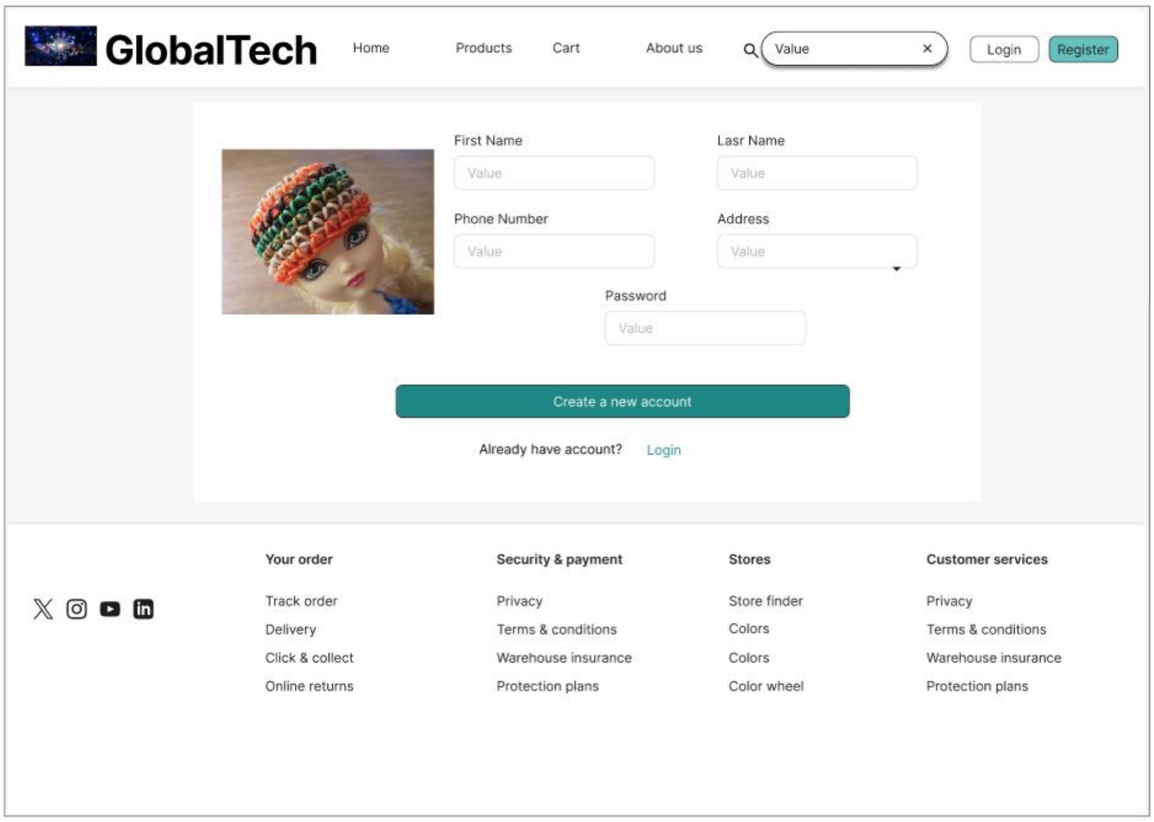


Figure 45: Prototype of register page

8.2. Login page

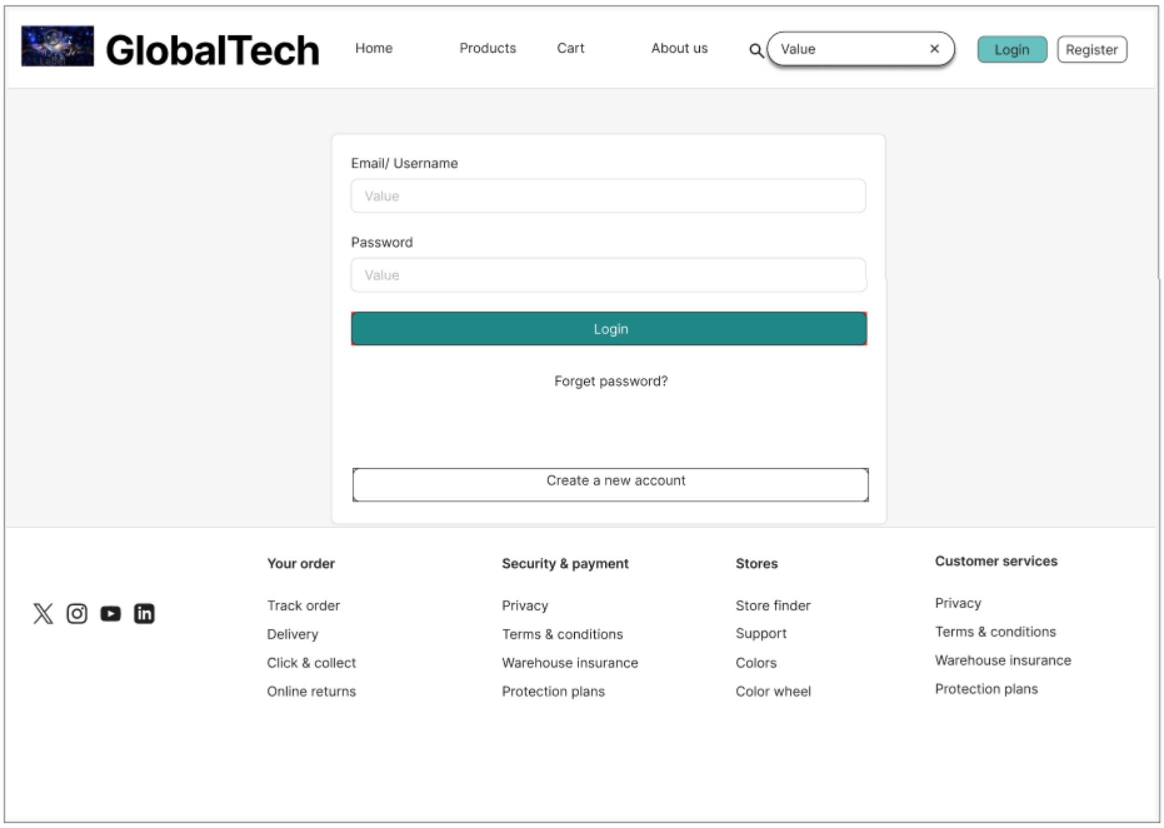


Figure 46: Prototype of Login page

8.3. Home page

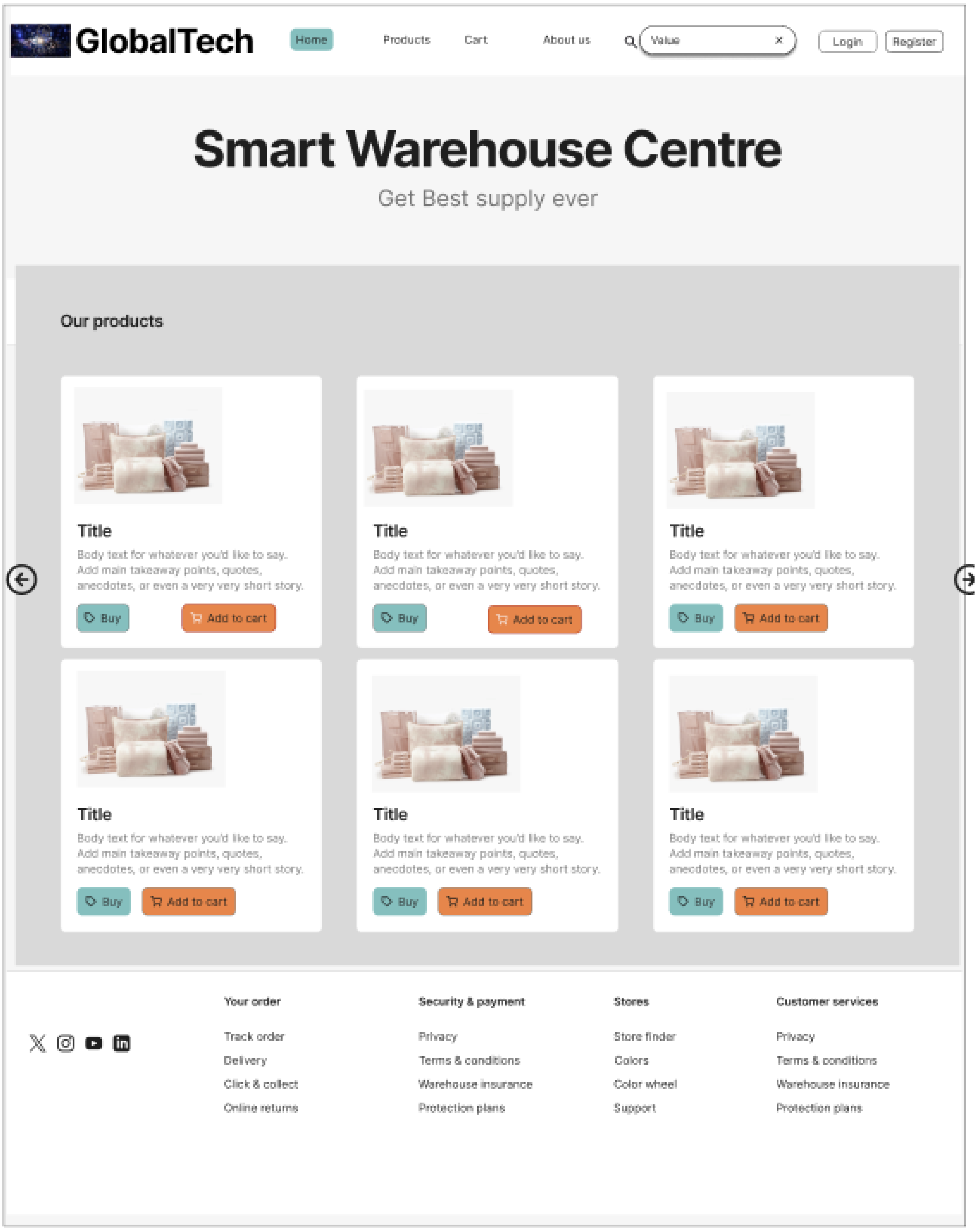


Figure 47: Home page

8.4. Customer Product page

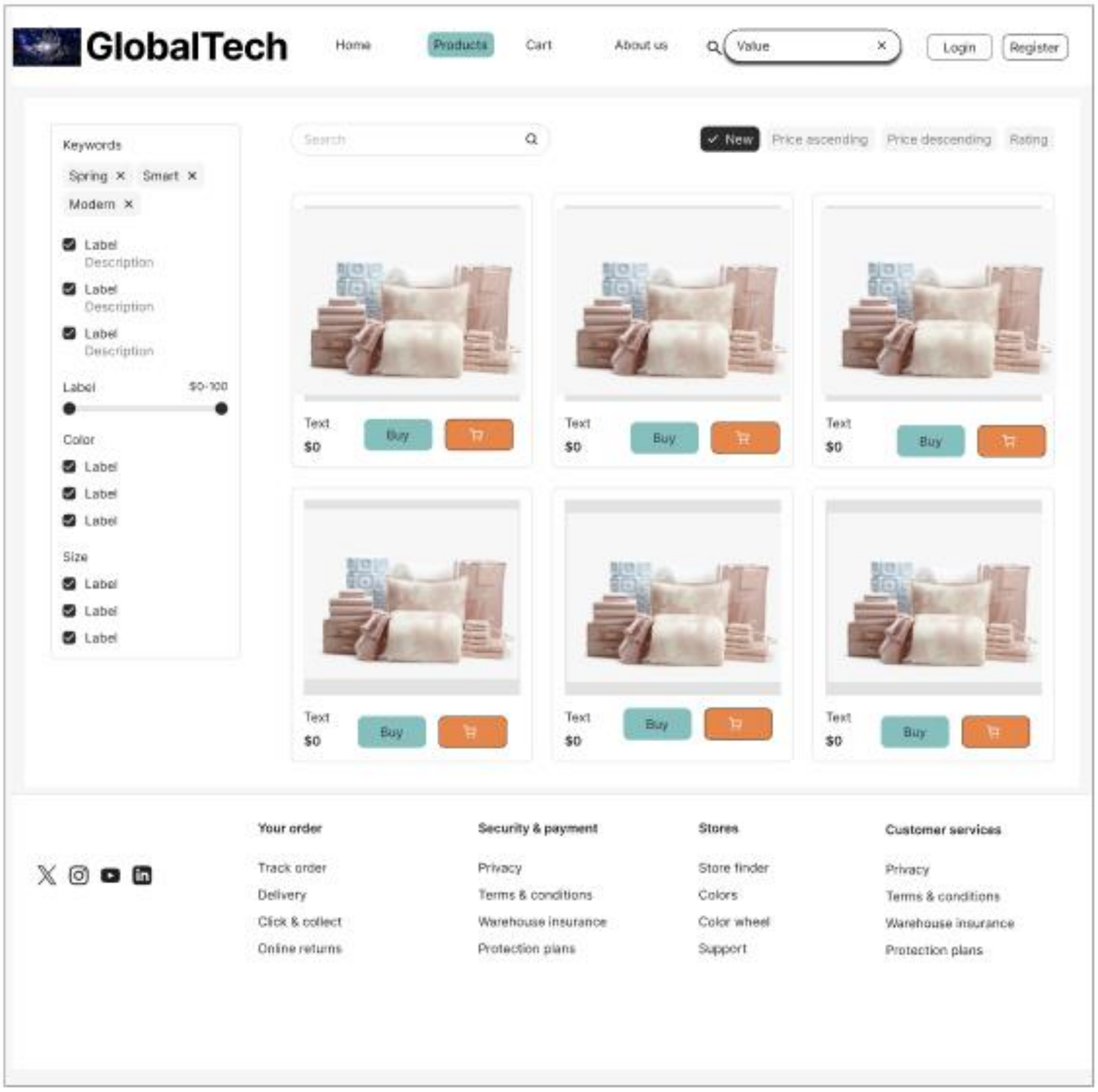


Figure 48: Customer product page

8.5. Customer Product details page

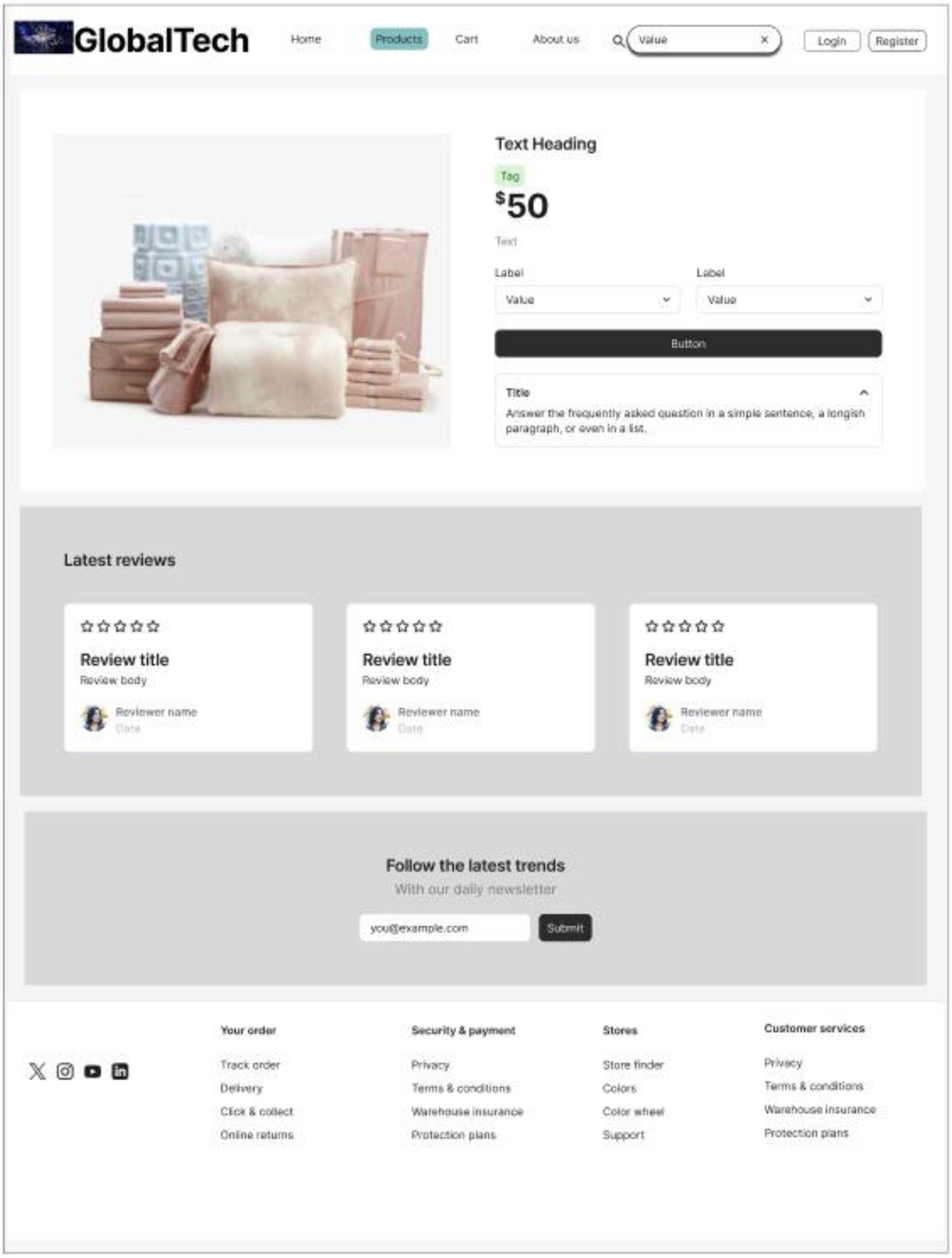
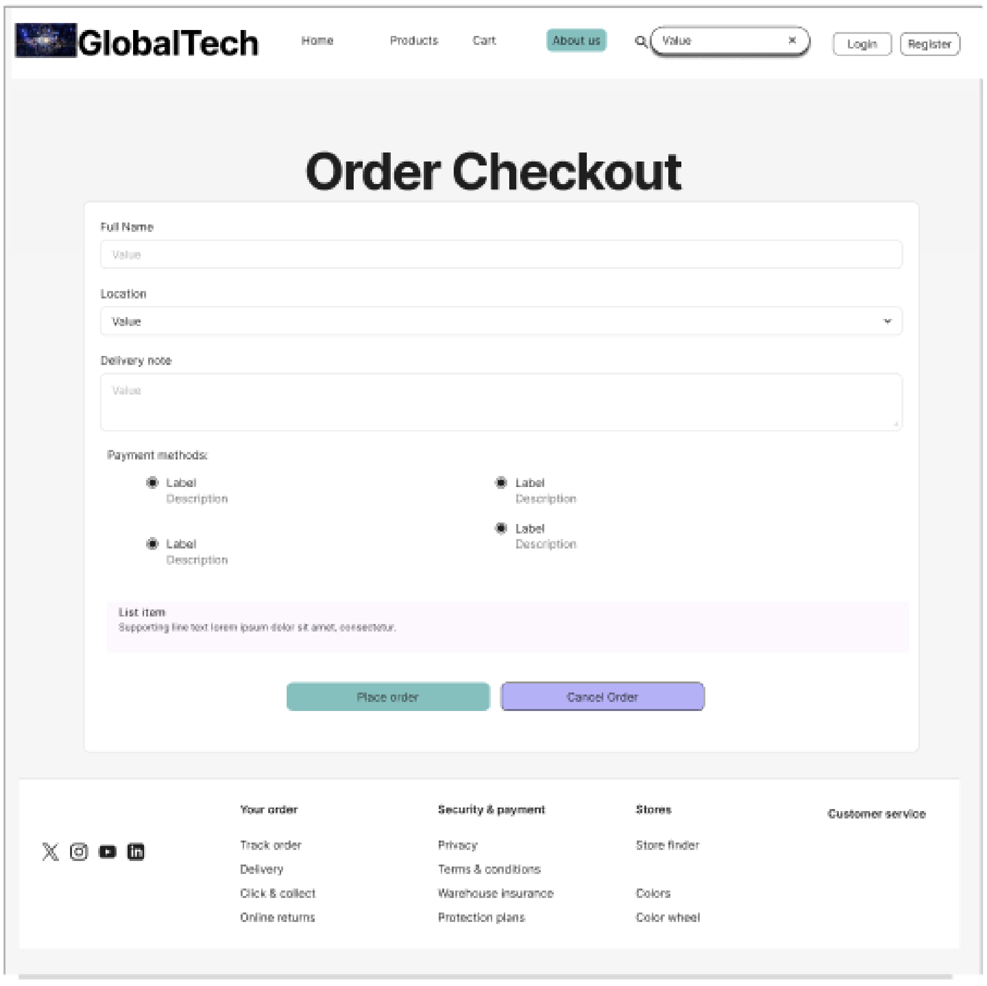


Figure 49: Customer Product detail page

8.6. Order page



Order Checkout

Full Name

Value

Location

Value

Delivery note

Value

Payment methods:

- Label Description
- Label Description
- Label Description
- Label Description

List item
Supporting line text lorem ipsum dolor sit amet, consectetur.

Place order | Cancel Order

Figure 50: Order checkout

8.7. Make payment page

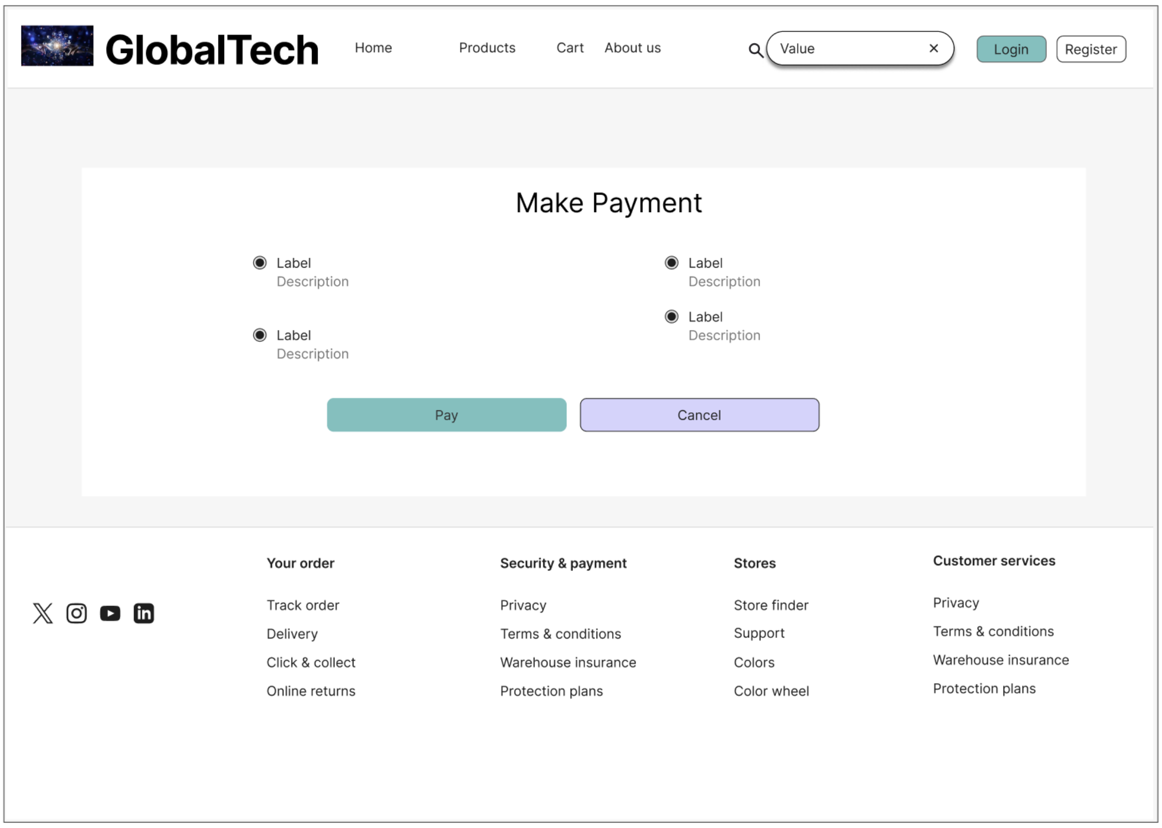


Figure 51: Make payment page

8.. Generate report page

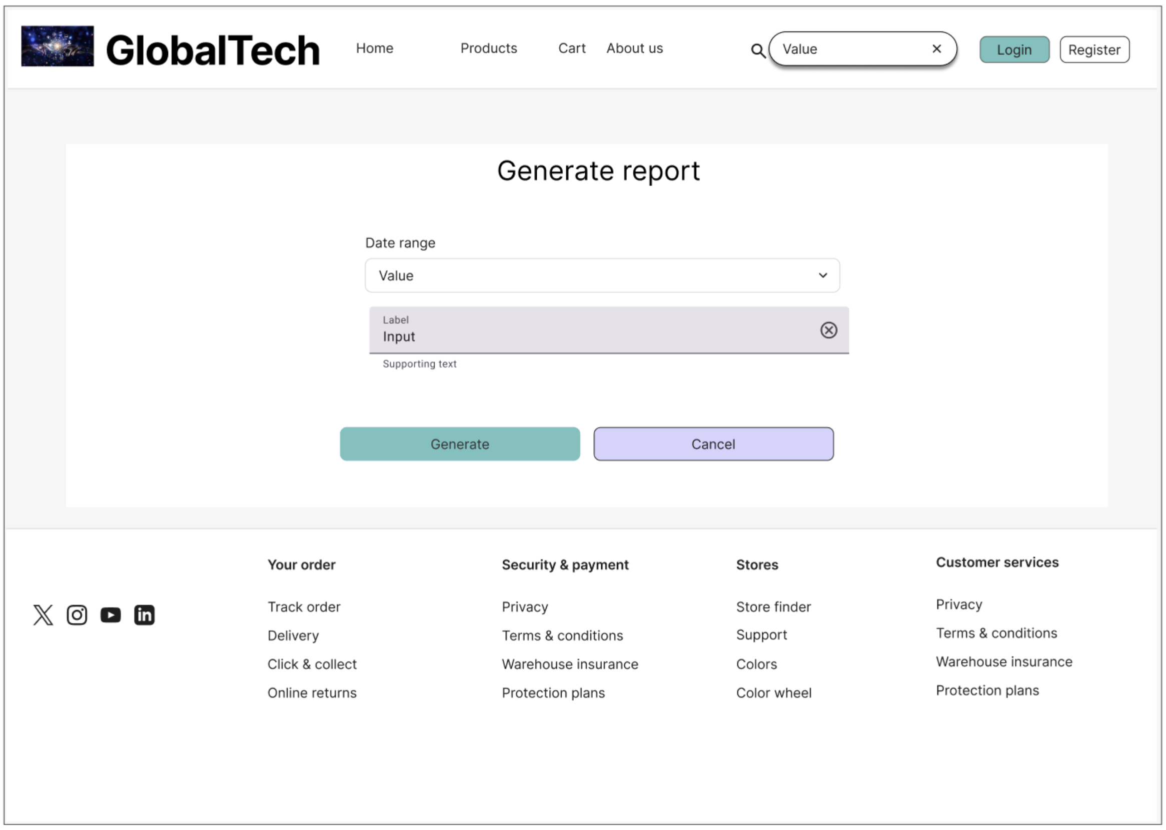


Figure 52: Generate report

8.9. Admin dashboard page

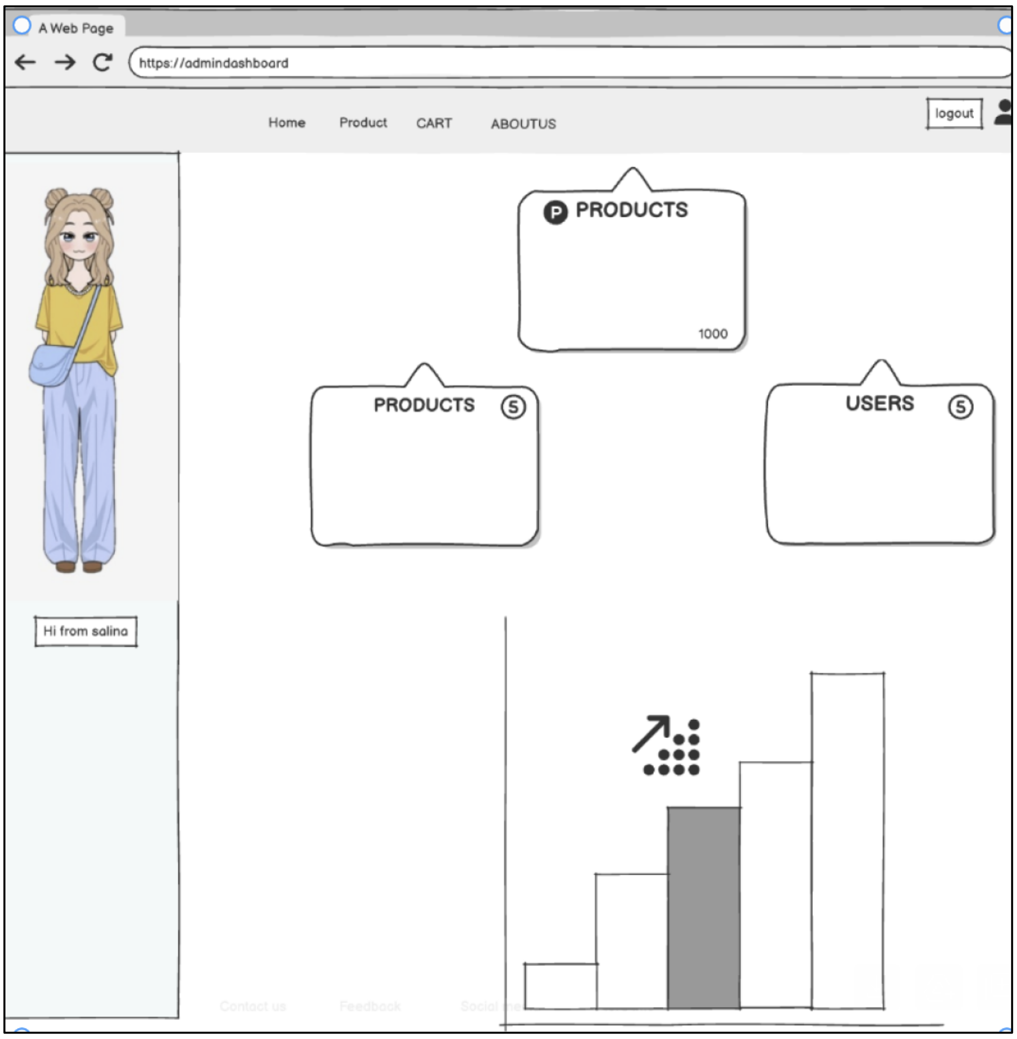


Figure 53: Admin dashboard

8.10. User page

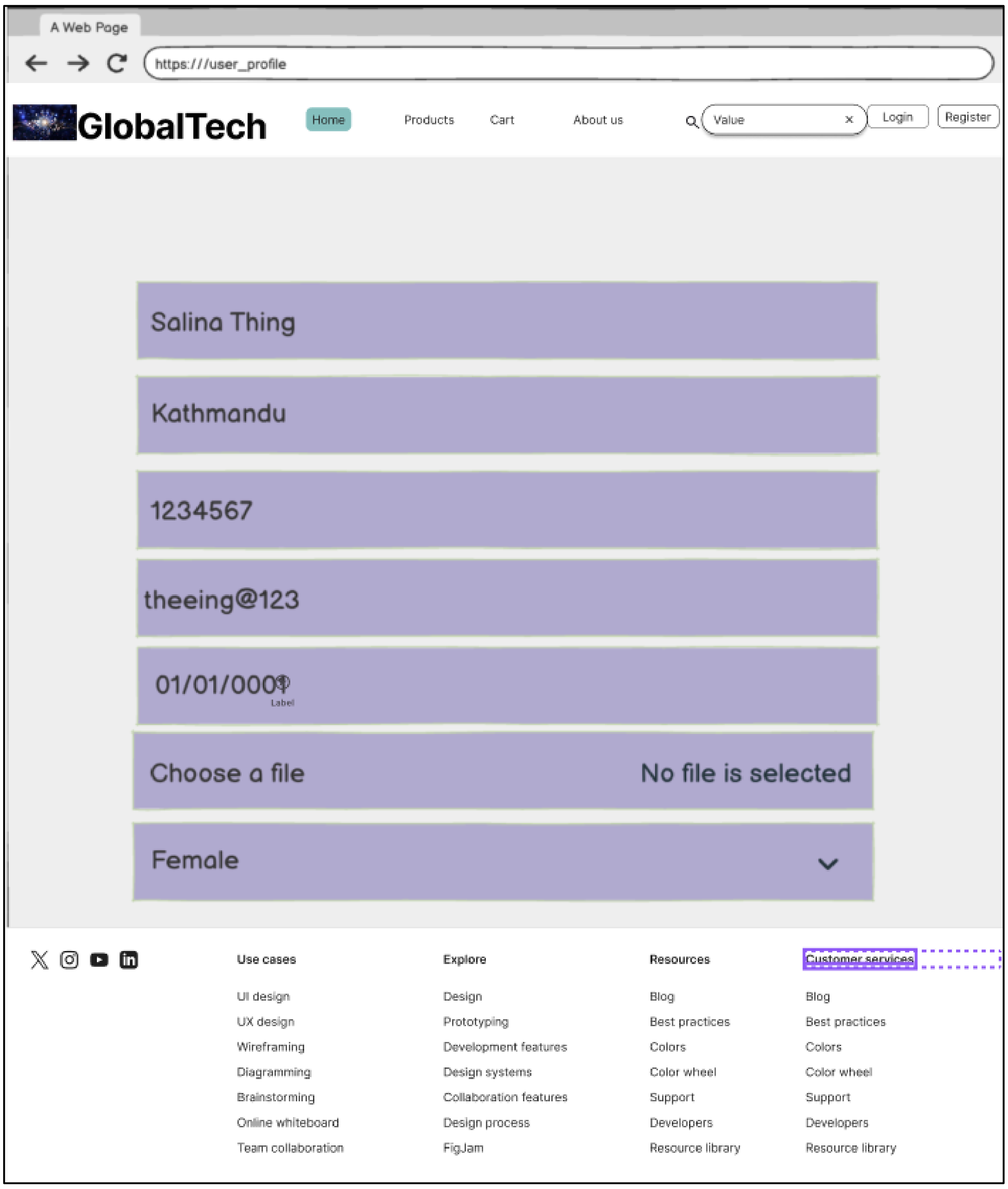
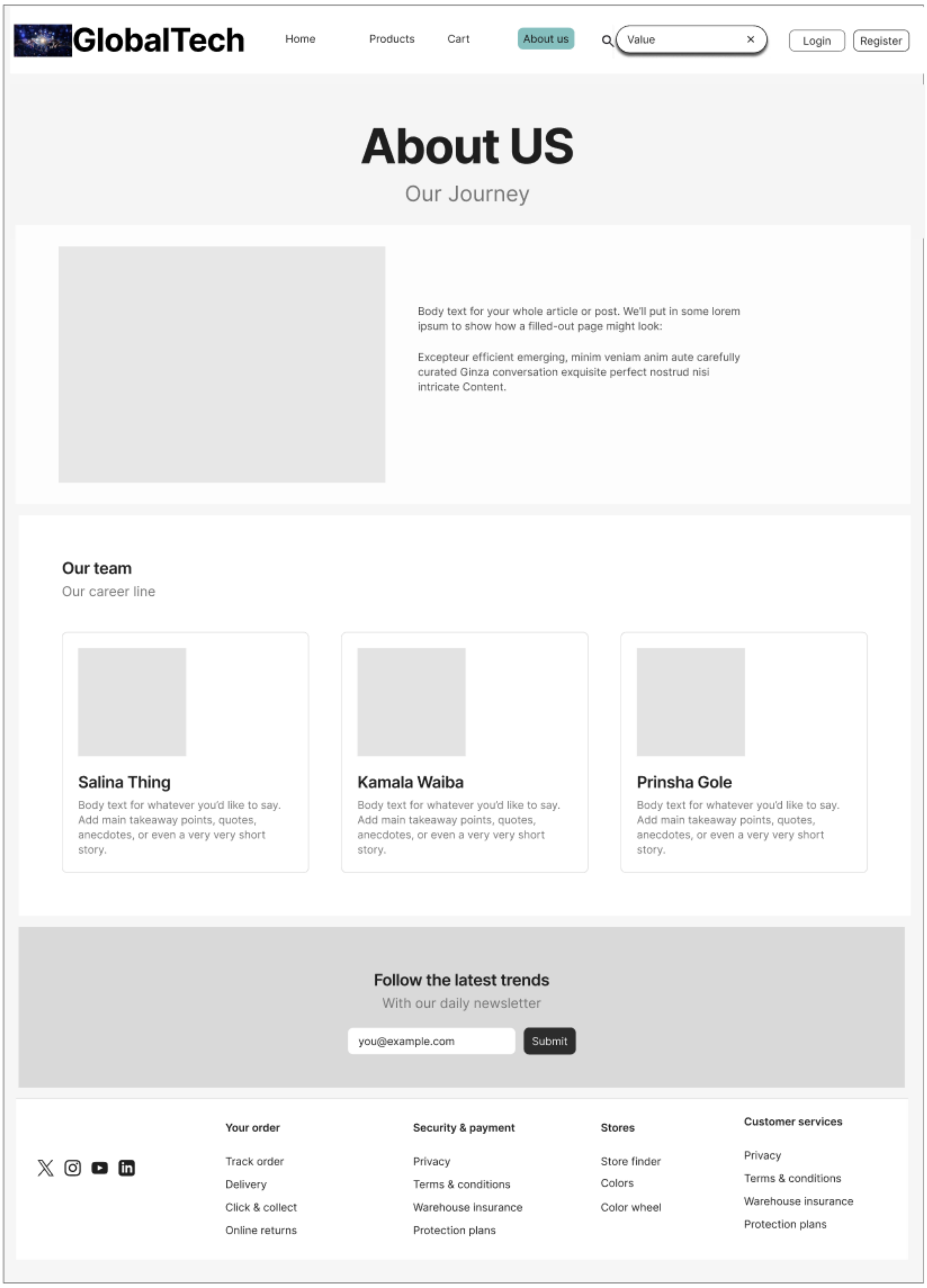


Figure 54: User page

8.11. About us page



GlobalTech

Home Products Cart About us Value Login Register

About US

Our Journey

Body text for your whole article or post. We'll put in some lorem ipsum to show how a filled-out page might look:

Excepteur efficient emerging, minim veniam anim aute carefully curated Ginza conversation exquisite perfect nostrud nisi intricate Content.

Our team

Our career line

Salina Thing

Body text for whatever you'd like to say. Add main takeaway points, quotes, anecdotes, or even a very very short story.

Kamala Waiba

Body text for whatever you'd like to say. Add main takeaway points, quotes, anecdotes, or even a very very short story.

Prinsha Gole

Body text for whatever you'd like to say. Add main takeaway points, quotes, anecdotes, or even a very very short story.

Follow the latest trends

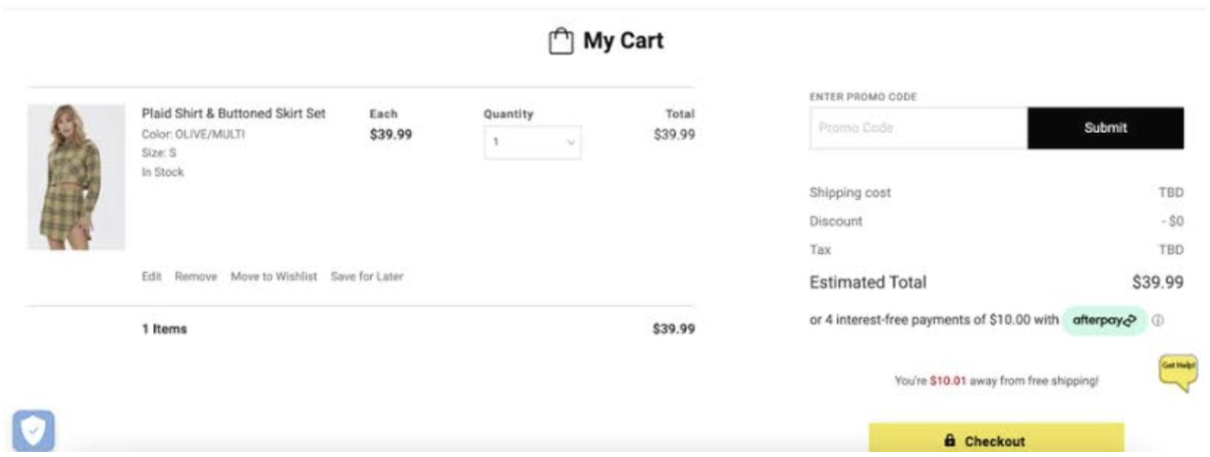
With our daily newsletter

you@example.com Submit

Your order	Security & payment	Stores	Customer services
Track order	Privacy	Store finder	Privacy
Delivery	Terms & conditions	Colors	Terms & conditions
Click & collect	Warehouse insurance	Color wheel	Warehouse insurance
Online returns	Protection plans		Protection plans

Figure 55: About us page

8.12. cart page



My Cart

	Each	Quantity	Total
Plaid Shirt & Buttoned Skirt Set Color: OLIVE/MULTI Size: S In Stock	$39.99	1	$39.99

Edit Remove Move to Wishlist Save for Later

1 Items $39.99

ENTER PROMO CODE

Promo Code | Submit

Shipping cost	TBD
Discount	- $0
Tax	TBD
Estimated Total	$39.99

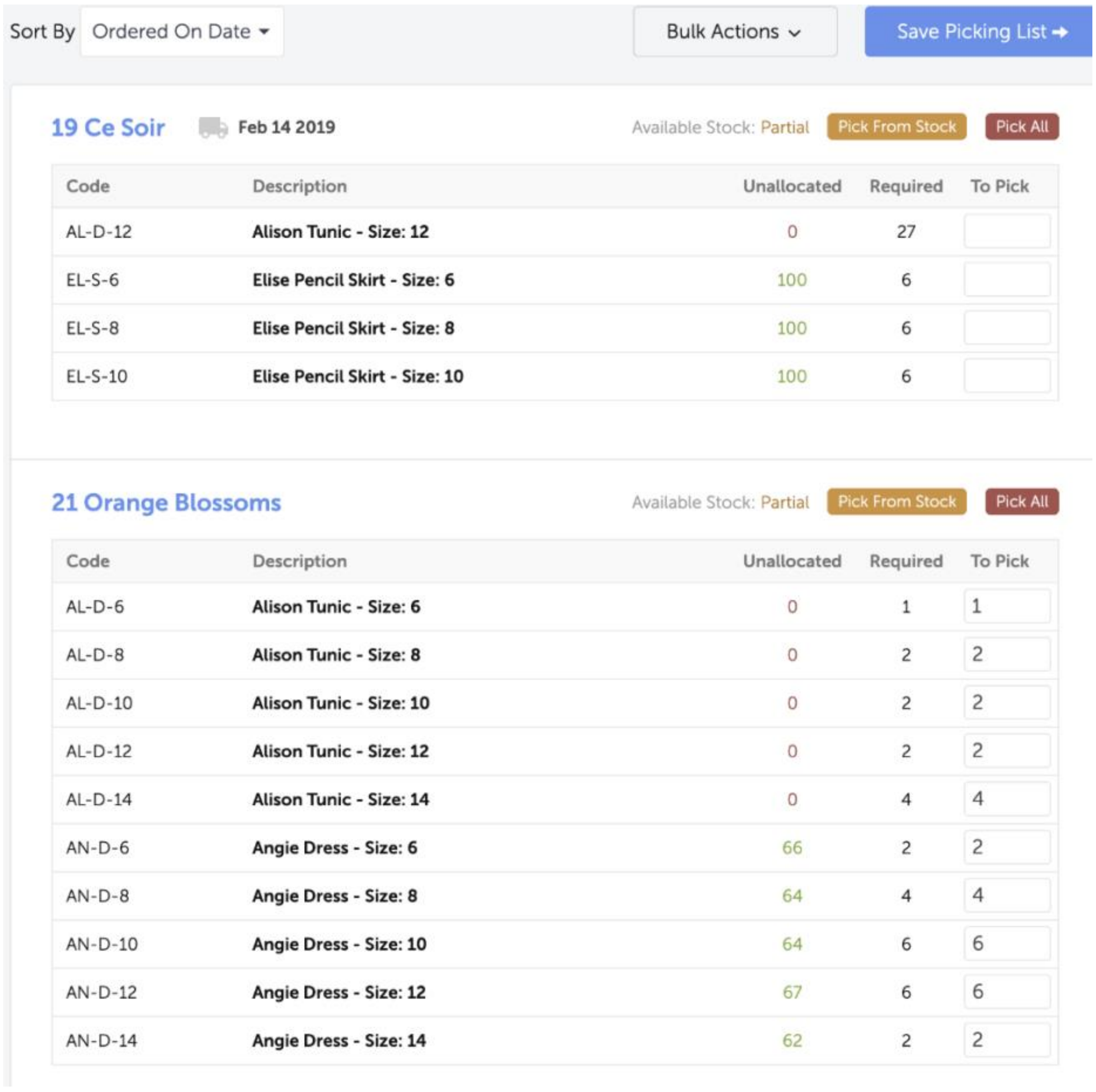
or 4 interest-free payments of $10.00 with afterpay

You're $10.01 away from free shipping!

Checkout

Figure 56: Cart page

8.13. Dispatch order page



19 Ce Soir Feb 14 2019 Available Stock: Partial Pick From Stock Pick All

Code	Description	Unallocated	Required	To Pick
AL-D-12	**Alison Tunic - Size: 12**	0	27	
EL-S-6	**Elise Pencil Skirt - Size: 6**	100	6	
EL-S-8	**Elise Pencil Skirt - Size: 8**	100	6	
EL-S-10	**Elise Pencil Skirt - Size: 10**	100	6	

21 Orange Blossoms Available Stock: Partial Pick From Stock Pick All

Code	Description	Unallocated	Required	To Pick
AL-D-6	**Alison Tunic - Size: 6**	0	1	1
AL-D-8	**Alison Tunic - Size: 8**	0	2	2
AL-D-10	**Alison Tunic - Size: 10**	0	2	2
AL-D-12	**Alison Tunic - Size: 12**	0	2	2
AL-D-14	**Alison Tunic - Size: 14**	0	4	4
AN-D-6	**Angie Dress - Size: 6**	66	2	2
AN-D-8	**Angie Dress - Size: 8**	64	4	4
AN-D-10	**Angie Dress - Size: 10**	64	6	6
AN-D-12	**Angie Dress - Size: 12**	67	6	6
AN-D-14	**Angie Dress - Size: 14**	62	2	2

Figure 57: Dispatch order

8.14. Compare price page

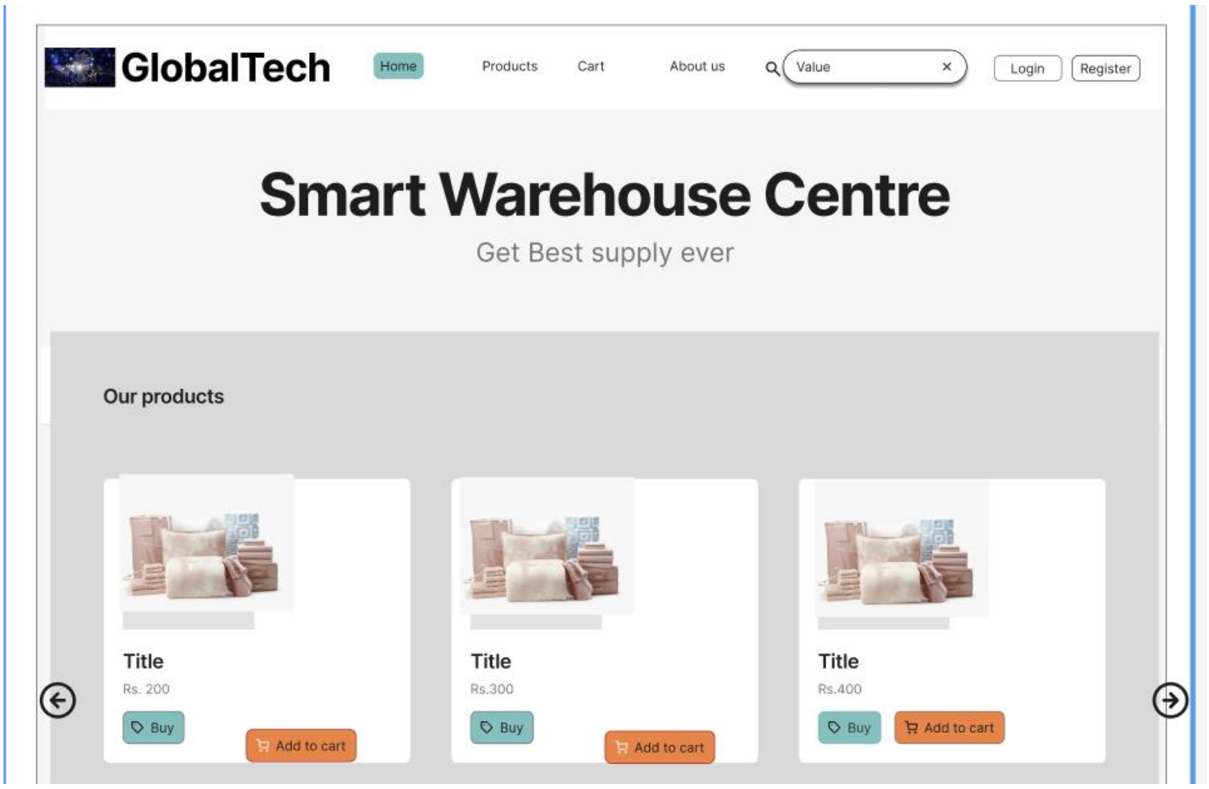
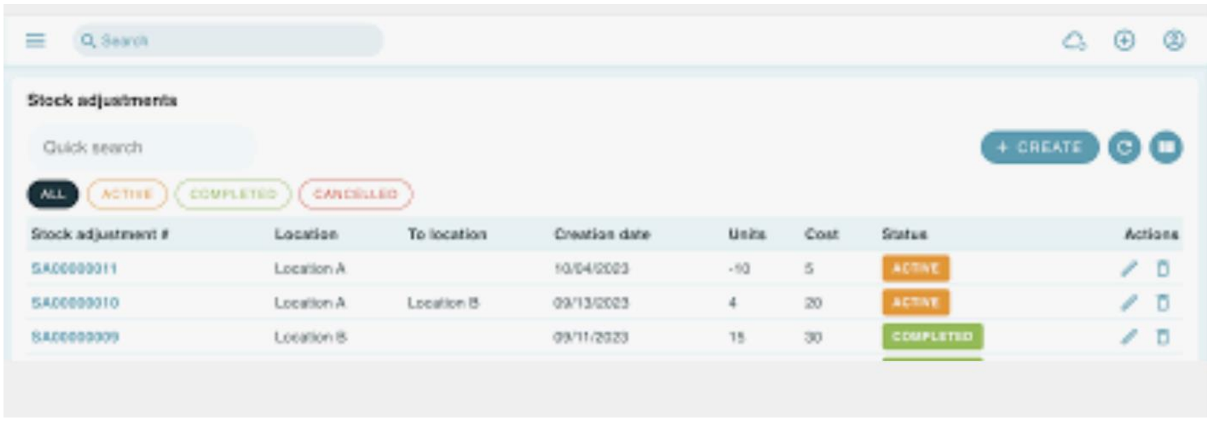


Figure 58: Compare product price

8.15. Add product page

8.16. Update stock page



Search

Stock adjustments

Quick search

+ CREATE

ALL ACTIVE COMPLETED CANCELLED

Stock adjustment #	Location	To location	Creation date	Units	Cost	Status	Actions
SA00000011	Location A		10/04/2023	-10	5	ACTIVE	
SA00000010	Location A	Location B	09/13/2023	4	20	ACTIVE	
SA00000009	Location B		09/11/2023	15	30	COMPLETED	

Figure 59: Update stock page

Conclusion

I have learnt to deal with real world based object oriented programs using Agile methodology. From creating a structured WBS and detailed UML diagrams to outlining communication flows and future development plans, each step was aligned with best practices in software engineering. The integration of design principles such as domain-driven class modeling and thoughtful use of relationships like inheritance and composition has added depth to the system architecture. Moving forward, this foundational work sets a clear path for iterative development, testing, and deployment, ensuring that the project stays aligned with both user needs and agile principles.

Bibliography

asana, 2025. *asana.* [Online]
Available at: https://asana.com/resources/agile-methodology
[Accessed 27 April 2025].

CheckPoint, 2025. *CheckPoint.* [Online]
Available at: https://www.checkpoint.com/cyber-hub/cyber-security/what-is-white-box-testing/#:~:text=White%20box%20testing%20is%20a,gray%20and%20black%20box%20testing.
[Accessed 27 April 2025].

Computer Hope, 2024. *Draw.io.* [Online]
Available at: https://www.computerhope.com/jargon/d/drawio.htm
[Accessed 11 May 2025].

Duke, R., 2025. *workbreakdownstructure.com.* [Online]
Available at: https://www.workbreakdownstructure.com/
[Accessed 4 March 2025].

geeksforgeeks, 2024. *Differences between Functional and Non-functional Testing.* [Online]
Available at: https://www.geeksforgeeks.org/differences-between-functional-and-non-functional-testing/
[Accessed 11 May 2025].

geeksforgeeks, 2024. *geeksforgeeks.* [Online]
Available at: https://www.geeksforgeeks.org/what-is-domain-class-in-uml/
[Accessed 27 April 2025].

geeksforgeeks, 2025. *Class Diagram Unified Modeling Language (UML).* [Online]
Available at: https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/
[Accessed 11 May 2025].

geeksforgeeks, 2025. *Difference between Integration Testing and Sandwich Testing.* [Online]
Available at: https://www.geeksforgeeks.org/difference-between-integration-testing-and-sandwich-testing/
[Accessed 11 May 2025].

geeksforgeeks, 2025. *geeksforgeeks.* [Online]
Available at: https://www.geeksforgeeks.org/use-case-diagram/
[Accessed 3 January 2025].

geeksforgeeks, 2025. *geeksforgeeks.* [Online]
Available at: https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/
[Accessed 30 March 2025].

geeksforgeeks, 2025. *geeksforgeeks.* [Online]
Available at: https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/
[Accessed 27 April 2025].

geeksforgeeks, 2025. *MVC Design Pattern.* [Online]
Available at: https://www.geeksforgeeks.org/mvc-design-pattern/
[Accessed 11 May 2025].

geeksforgeeks, 2025. *What are Microservices?.* [Online]
Available at: https://www.geeksforgeeks.org/microservices/
[Accessed 10 May 2025].

Gillis, A. S., 2025. *smoke testing.* [Online]
Available at: https://www.techtarget.com/searchsoftwarequality/definition/smoke-testing
[Accessed 11 May 2025].

Jason Smith, 2025. *dotcms.* [Online]
Available at: https://www.dotcms.com/blog/what-are-microservices-and-how-do-they-aid-agile-development-
[Accessed 27 April 2025].

Lucidchart, 2025. *Lucidchart.* [Online]
Available at: https://www.lucidchart.com/pages/uml-use-case-diagram
[Accessed 5 March 2025].

ProjectManager.com, 2025. *ProjectManager.com.* [Online]
Available at: https://www.projectmanager.com/guides/work-breakdown-structure
[Accessed 8 January 2022].

teamgantt, 2025. *teamgantt.* [Online]
Available at: https://www.teamgantt.com/what-is-a-gantt-chart#:~:text=A%20gantt%20chart%20is%20a,each%20task%20in%20the%20project.
[Accessed 30 December 2024].

UMLBoard, 2024. *Multiplicity.* [Online]
Available at: https://www.umlboard.com/docs/relations/multiplicity/
[Accessed 11 May 2025].

visual-paradigm, 2024. *visual-paradigm.* [Online]
Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml-collaboration-diagram/
[Accessed 30 March 2025].

visual-paradigm, 2025. *visual-paradigm.* [Online]
Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/
[Accessed 31 December 2024].