Vaiyer Ponno

A Direct-to-Consumer (D2C) E-commerce Platform.

TABLE OF CONTENTS:

1. REQUIREMENT ANALYSIS

There are two requirements for the Vaiyer Ponno project: **functional** and **non-functional**. Additionally, requirements are categorized as **Normal** (1), **Expected** (2), and **Exciting** (3). This classification helps prioritize features and ensures that the platform delivers a comprehensive and satisfying user experience. Below are the detailed functional and non-functional requirements, marked with 1, 2, and 3.

1.1 Functional Requirements

These requirements define the essential features and capabilities that the Vaiyer Ponno platform must provide.

- User Authentication:
    - Sign In & Sign Up [1]
    - Password Recovery [2]
    - Multi-Factor Authentication [3]
- User Profiles:
    - Profile Creation and Management [1]
    - View Order History [1]
    - Customizable User Preferences [2]
- Product Management:
    - Create, Edit, and Delete Product Listings [1]
    - Product Categorization and Tagging [2]
    - Bulk Upload of Products [3]
- Shopping Cart:
    - Add/Remove Items to/from Cart [1]
    - View and Edit Cart [1]
    - Save Cart for Later [2]
- Checkout Process:
    - Order Placement [1]
    - Multiple Payment Options [2]
    - Guest Checkout [2]
- Order Management:
    - Order Tracking [1]
    - Order Cancellation and Returns [2]
    - Automatic Stock Updates [3]
- Communication:
    - Real-Time Messaging [1]
    - Email Notifications for Orders [1]
    - Push Notifications for Updates [2]
- Reviews and Ratings:
    - Product Reviews and Ratings [1]
    - Seller Ratings [2]

    - Review Filtering and Sorting [3]
- Payment Processing:
    - Secure Payment Gateway Integration [1]
    - Support for Multiple Currencies [2]
    - Automated Refund Processing [3]

1.2 Non-Functional Requirements

These requirements define the quality attributes and constraints that the platform must adhere to.

- Performance:
    - Scalable Architecture [1]
    - Optimized Page Load Times [2]
    - Efficient Database Queries [3]
- Security:
    - Data Encryption [1]
    - Regular Security Audits [2]
    - Advanced Threat Detection and Prevention [3]
- Usability:
    - Intuitive User Interface [1]
    - Accessible Design for All Users [2]
    - Customizable Themes and Layouts [3]
- Reliability:
    - 99.9% Uptime Guarantee [1]
    - Automated Backups [2]
    - Disaster Recovery Plan [3]
- Maintainability:
    - Modular Codebase [1]
    - Comprehensive Documentation [2]
    - Automated Testing and CI/CD Pipeline [3]
- Compliance:
    - Adherence to Local E-commerce Laws [1]
    - PCI-DSS Compliance for Payment Handling [2]
    - GDPR Compliance for User Data [3]

These are all the requirements.

2. Which model should I use?

Choosing the right Software Development Life Cycle (SDLC) model is crucial for the successful completion of the Vaiyer Ponno project. Different SDLC models are suited to different types of projects, depending on factors like complexity, timeline, and flexibility.

2.0 Introduction to SDLC Models

SDLC models provide a structured approach to software development. Each model defines a series of phases, from initial planning to deployment and maintenance, ensuring that all necessary steps are taken to deliver a successful software product. Let's explore some of the common models.

2.1 The Model I Chose

After evaluating various Software Development Life Cycle (SDLC) models, we decided to adopt the **Spiral Model** for the development of "Vaiyer Ponno." The decision was made based on several critical factors:

1. **Frequent Evolution of Software Functions:**
   The Spiral Model supports iterative development, allowing the software to evolve and improve with each iteration. This is crucial for Vaiyer Ponno as its features will need to adapt based on user feedback and market demands.
2. **Incorporation of Client Feedback:**
   Since Vaiyer Ponno is a business-oriented platform, client and user feedback are essential for refining the product. The Spiral Model facilitates regular reviews and adjustments, ensuring that the final product meets client expectations and market needs.
3. **Risk Analysis and Client Satisfaction:**
   The Spiral Model is designed to identify and mitigate risks early in the development process. This is particularly important for a business-critical project like "Vaiyer Ponno," where any potential issues must be addressed promptly to ensure client satisfaction and software reliability.
4. **Budget Constraints:**
   Given the limited budget for the project, the Spiral Model offers a controlled and phased approach to development. By focusing on risk management and iterative releases, we can manage costs effectively while delivering a high-quality product.
5. **Long-Term Commitment and Economic Considerations:**
   The project’s long-term nature means that economic priorities and requirements may shift over time. The Spiral Model’s flexibility allows us to adapt to these changes without disrupting the overall project timeline.
6. **Complex Requirements:**
   Vaiyer Ponno involves complex requirements that need careful evaluation and refinement. The Spiral Model’s iterative cycles provide the necessary framework to explore and clarify these requirements before moving forward with development.

7. **Phased Release for Customer Feedback:**
   As a new product line, Vaiyer Ponno will benefit from phased releases that allow for customer feedback at each stage. This iterative approach ensures that the platform can be adjusted and improved before its final release.

3. DOCUMENTATION

In the initial stages of a project, requirement documentation plays a crucial role in defining the scope and direction of the development process. There are four primary types of modeling used in requirement documentation, which serve as the foundation for understanding and designing the system. These models are:

1. **Scenario-Based Model**
2. **Flow-Oriented Model**
3. **Behavioral Model**
4. **Class-Based Model**

3.1 Scenario-Based Model

The Scenario-Based Model provides a narrative description of how the system will be used by different actors. It outlines the interactions between the system and its users, offering a clear understanding of how the software will function in real-world situations. Below are two scenarios that illustrate the system's operations:

3.1.1 SCENE 1: ACTOR – CLIENT

Use Case No	Details
1. Name	User Registration
Primary Actor	Customer
Secondary Actor	Database
Goal in context	To create an account on the website
Precondition	Internet, Browser, Email
Trigger	When someone wants to create an account
Events	- User enters required details - System verifies information - Account is created
Priority	High
When Available	First increment
Frequency of Use	Medium

Use Case No	Details
2. Name	User Login
Primary Actor	Customer
Secondary Actor	Database
Goal in context	To access personal account on the website
Precondition	Must have an account
Trigger	When someone wants to log in
Events	- User enters credentials - System authenticates details - User is granted access to their account
Priority	High
When Available	First increment
Frequency of Use	High
Channel to actor	Through browser
Channel to secondary actor	API / DB connection
Scenario	At the beginning
Exception	- Incorrect password - Account lockout after multiple failed attempts - Database connection issues

Use Case No	Details
3. Name	Browse Products
Primary Actor	Customer
Secondary Actor	Database
Goal in context	To view available products in the store
Precondition	Internet, Browser, Account (optional)
Trigger	When someone visits the product catalog
Events	- User navigates to product section - System displays available products - User can filter and sort products based on preferences
Priority	Medium
When Available	Second increment
Frequency of Use	High

Channel to actor	Through browser
Channel to secondary actor	API / DB connection
Scenario	Ongoing
Exception	- Products not loading - Slow filtering due to server load - Incorrect sorting of products

Use Case No	Details
4. Name	Add to Cart
Primary Actor	Customer
Secondary Actor	Database
Goal in context	To select products for purchase
Precondition	Must be logged in
Trigger	When someone selects a product
Events	- User clicks 'Add to Cart' - System verifies stock availability - Product is added to the user's cart
Priority	High
When Available	Second increment
Frequency of Use	High
Channel to actor	Through browser
Channel to secondary actor	API / DB connection
Scenario	Ongoing
Exception	- Insufficient stock - Cart not updating correctly - Database connection issues

Use Case No	Details
5. Name	Checkout Process
Primary Actor	Customer
Secondary Actor	Payment Gateway, Database

Goal in context	To purchase products in the cart
Precondition	Must have items in the cart
Trigger	When someone decides to complete a purchase
Events	- User initiates checkout - System calculates total cost - Payment details entered - Payment processed - Order confirmation sent to user
Priority	High
When Available	Third increment
Frequency of Use	Medium
Channel to actor	Through browser
Channel to secondary actor	API / Payment Gateway
Scenario	Ongoing
Exception	- Payment failure - Incorrect billing details - Cart abandonment

Use Case No	Details
6. Name	View Order History
Primary Actor	Customer
Secondary Actor	Database
Goal in context	To view past purchases and order statuses
Precondition	Must be logged in and have previous orders
Trigger	When someone wants to review past orders
Events	- User navigates to 'Order History' - System retrieves and displays past orders - User can view order details and status
Priority	Medium
When Available	Third increment
Frequency of Use	Medium
Channel to actor	Through browser
Channel to secondary actor	API / DB connection
Scenario	Ongoing

Exception	- Orders not displaying - Incorrect order details - Slow loading time

Use Case No	Details
7. Name	Manage User Profile
Primary Actor	Customer
Secondary Actor	Database
Goal in context	To update personal information and settings
Precondition	Must be logged in
Trigger	When someone wants to update their profile
Events	- User navigates to 'Profile Settings' - System displays current user information - User updates details - Changes are saved and confirmed
Priority	Medium
When Available	Second increment
Frequency of Use	Medium
Channel to actor	Through browser
Channel to secondary actor	API / DB connection
Scenario	Ongoing
Exception	- Error saving changes - Incorrect data validation - System not updating the profile correctly

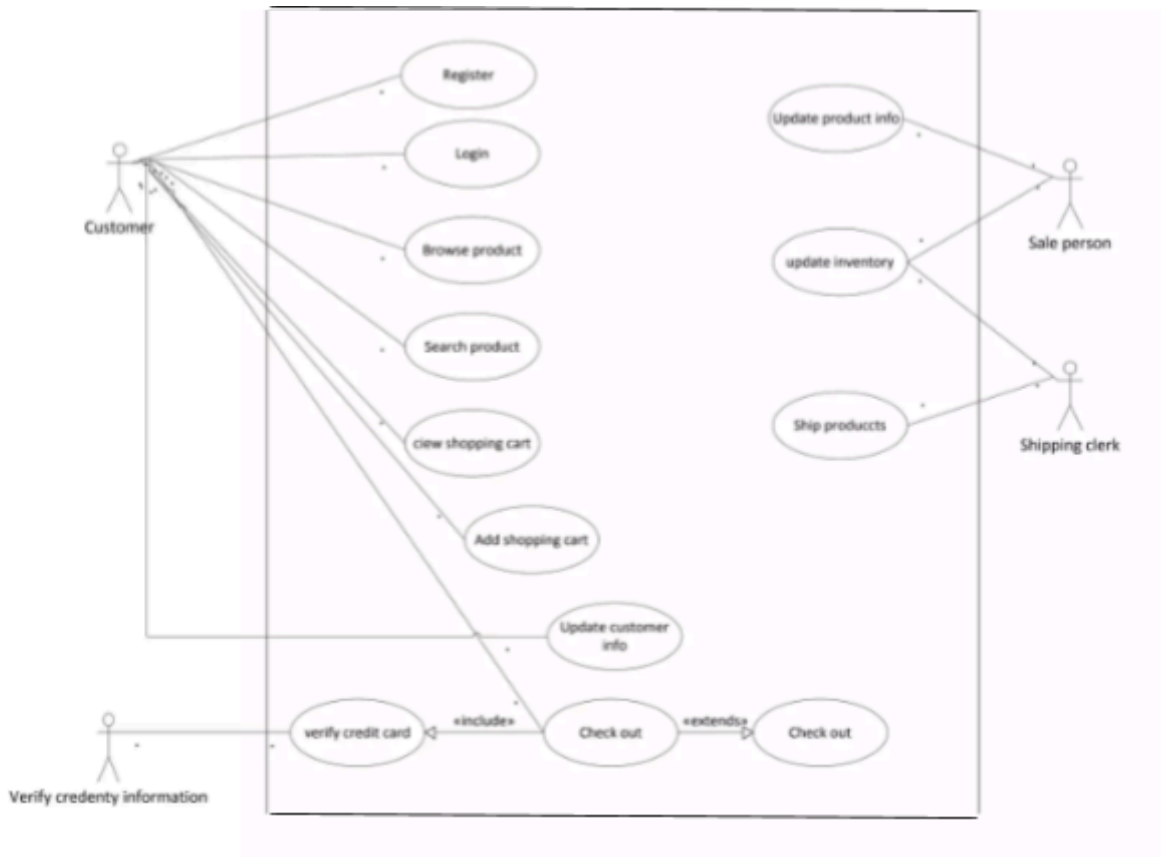
Use Case No	Details
8. Name	Product Review & Rating
Primary Actor	Customer
Secondary Actor	Database
Goal in context	To provide feedback on purchased products
Precondition	Must be logged in and have purchased the product
Trigger	When someone wants to leave a review or rating
Events	- User navigates to the purchased product - System displays 'Leave a Review' option - User submits review and rating - Review is saved and displayed on the product page

Priority	Medium
When Available	Third increment
Frequency of Use	Low
Channel to actor	Through browser
Channel to secondary actor	API / DB connection
Scenario	Ongoing
Exception	- Review not saving - Rating system malfunctioning - Delayed review approval

Use Case No	Details
9. Name	Contact Support
Primary Actor	Customer
Secondary Actor	Support Team, Database
Goal in context	To resolve issues or ask for assistance
Precondition	Must be logged in
Trigger	When someone needs help or has an issue
Events	- User navigates to 'Support' - User submits a support ticket - System sends the ticket to the support team - Support team responds to the ticket
Priority	High
When Available	Third increment
Frequency of Use	Low
Channel to actor	Through browser
Channel to secondary actor	API / Support Database
Scenario	Ongoing
Exception	- Support ticket not submitting - Delayed response from support - Issue not resolved

Use Case No	Details
10. Name	Admin Dashboard
Primary Actor	Admin
Secondary Actor	Database
Goal in context	To manage the platform, users, and products
Precondition	Admin privileges
Trigger	When an admin needs to monitor or manage the site
Events	- Admin logs into the dashboard - System displays site analytics, user activities, and product management options - Admin performs actions like banning users, updating products, etc.
Priority	High
When Available	Second increment
Frequency of Use	High
Channel to actor	Through browser
Channel to secondary actor	API / DB connection
Scenario	Ongoing
Exception	- Dashboard not loading - Admin actions not saving - System crashes during critical operations

3.1.2 USE CASE DIAGRAM



3.2 Class-Based Model for Vaiyer Ponno

3.2.1 Scenario Analysis

Scene - 1:

A customer visits the Vaiyer Ponno website and registers an account. After successfully authenticating, the customer browses various products across different categories. The customer adds a few selected products to the shopping cart and then proceeds to checkout. At checkout, the customer chooses a payment method, enters shipping details, and confirms the order.

Once the order is placed, the system processes the payment and generates an order confirmation, which is sent to the customer's email. The customer can then track the order status through their account dashboard.

In parallel, the order details are sent to the seller associated with the products. The seller receives a notification and prepares the order for shipment. The seller updates the order status as "shipped" in the system, and this status update is reflected in the customer's account.

The customer receives the product and confirms the delivery on the website. The system then prompts the customer to leave a review and rate the product. If any issues arise, the customer can contact the seller directly through the in-built messaging system or raise a support ticket, which the admin can monitor and address.

Finally, the customer logs out or can remain logged in to explore more products and services on the site.

Scene - 2:

The admin logs into the Vaiyer Ponno admin panel. Upon login, the admin is presented with a dashboard overview of the site's activities, including recent orders, payment statuses, and user activity.

The admin navigates to the "Payments" section, where they can view the payment status of all customers, ensuring that transactions are successfully completed. If there are any pending or failed payments, the admin can investigate and take necessary actions, such as contacting the customer or resolving payment gateway issues.

Next, the admin checks the "Support" section, where they can see a list of open support tickets submitted by customers and sellers. The admin can prioritize tickets, assign them to the relevant support staff, or respond directly to critical issues, ensuring that bugs are tracked and resolved promptly.

Additionally, the admin has the authority to manage user accounts. If any customer or seller violates the platform's terms of service, the admin can restrict or ban their account, either temporarily or permanently. The admin can also limit certain privileges, such as disabling a seller's ability to list new products or restricting a customer's access to specific features.

After completing these tasks, the admin logs out or continues to monitor the site for any other issues.

Scene - 3:

A seller registers on the platform, creates a seller profile, lists products, manages inventory, and views sales reports. The seller can also respond to customer inquiries and update product information.

Marked Nouns and Verbs:

- **Customer**: Browses, Views, Adds to Cart, Proceeds to Checkout, Selects Payment Method, Places Order, Tracks Order Status
- **Seller**: Registers, Creates Profile, Lists Products, Manages Inventory, Views Reports, Responds to Inquiries, Updates Product Information
- **Admin**: Manages Accounts, Monitors Transactions, Resolves Disputes, Updates Content, Manages Campaigns
- **Product**: Listed, Viewed, Added to Cart, Purchased, Updated
- **Order**: Placed, Tracked, Delivered
- **Payment Method**: Selected
- **Inventory**: Managed
- **Report**: Viewed
- **Inquiry**: Responded
- **Account**: Managed
- **Transaction**: Monitored
- **Dispute**: Resolved
- **Content**: Updated
- **Campaign**: Managed

Potential Classes and Methods:

Class Name	Methods (Functions)	Category
Customer	BrowseProducts(), ViewProduct(), AddToCart(),	External Entities
	Checkout(), SelectPaymentMethod(), PlaceOrder(),	
	TrackOrderStatus()	
Seller	Register(), CreateProfile(), ListProduct(),	External Entities
	ManageInventory(), ViewSalesReport(),	
	RespondToInquiry(), UpdateProductInfo()	
Admin	ManageAccounts(), MonitorTransactions(),	Organizational Units
	ResolveDisputes(), UpdateContent(), ManageCampaigns()	
Product	List(), View(), AddToCart(), UpdateInfo(), Purchase()	Things

Order	Place(), TrackStatus(), Deliver()	Things
PaymentMethod	Select()	Things
Inventory	Manage()	Things
Report	View()	Things
Inquiry	Respond()	Things
Account	Manage()	Things
Transaction	Monitor()	Things
Dispute	Resolve()	Things
Content	Update()	Things
Campaign	Manage()	Things

3.2.2 CLASS ANALYSIS

For the Vaiyer Ponno project, careful consideration, the following classes were accepted or rejected based on their relevance and systematic importance

No.	Name	Satisfies	Status
1	Client	1, 2, 4, 5, 7	Accepted
2	System	2, 4, 7	Accepted
3	Product	2, 5	Accepted
4	Order	2, 5	Accepted
5	Payment	2, 3, 5, 7	Accepted
6	Admin	1, 2, 4, 5, 7	Accepted
7	Dashboard	2, 5	Accepted
8	Status	2, 3	Rejected
9	Support Ticket	2, 5	Accepted
10	Transaction	2, 5	Rejected
11	Report	2, 3, 7	Accepted
12	Privileges	2, 3, 4, 5, 7	Accepted
13	Session	2, 7	Accepted
14	User	1, 2, 4, 5, 7	Accepted
15	Seller	1, 2, 4, 5, 7	Accepted
16	Bug	2, 3, 5, 7	Accepted
17	Email Notification	2, 5, 7	Accepted
18	SMTP	2, 3, 7	Rejected

19	Connection	2, 3, 7	Rejected
20	Product Category	2, 5	Accepted
21	Feedback	2, 3	Rejected
22	Shipping	2, 3, 5	Accepted
23	Inventory	2, 5,7	Accepted
24	Discount	2, 3, 5	Rejected
25	Rating	2, 5	Accepted
26	Services	3, 5,7	Accepted

Table: 3.2.2 – Class Analysis Table

Only 18 of the potential classes were accepted, while others were systematically integrated into the accepted classes.

3.2.3 CLASS DECLARATION TABLE

Below is the Class Declaration Table for the **Vaiyer Ponno** project:

1. System	
Attribute/Method	**Description**
Name	Name of the system
Counter	Number of active sessions
Status	Current status of the system (e.g., running, maintenance)
TimeStamp	Time of the last system update
Connection: TYPE Class Connection	Establishes a connection to the database
Client: TYPE Class Client	References the Client class
Admin: TYPE Class Admin	References the Admin class
Init() : self	Initializes the system
SwitchMode(type)	Switches between maintenance and running modes
ClearGC()	Clears the garbage collector

Table 3.2.3.1 – System Class

2. Client	
Attribute/Method	**Description**
ID	Unique identifier for the client
Name	Name of the client
Email	Client's email address
Phone	Client's phone number
Status	Current status of the client
Permissions	Permissions granted to the client
Connection: TYPE Class Connection	Establishes a connection to the system
System : TYPE Class System	References the System class
TimeStamp	Time of the last client interaction
Init() : self	Initializes the client
Registration()	Registers the client
Login()	Authenticates and logs in the client
Logout()	Logs the client out
SeeHistory()	Displays the client’s transaction history
Authenticate()	Verifies the client’s identity
Services(type)	Executes a service (e.g., order, payment)

Table 3.2.3.2 – Client Class

3. Admin	
Attribute/Method	**Description**
ID	Unique identifier for the admin
Name	Name of the admin
Email	Admin's email address
Phone	Admin's phone number
Role	Role of the admin (e.g., super admin, moderator)
Status	Current status of the admin
Permissions	Permissions granted to the admin
System : TYPE Class System	References the System class
TimeStamp	Time of the last admin interaction
Init() : self	Initializes the admin
Login()	Authenticates and logs in the admin

Logout()	Logs the admin out
ManageClients()	Manages client permissions and statuses
MonitorSystem()	Monitors system health and status
ResolveTickets()	Resolves support tickets

Table 3.2.3.3 – Admin Class

4. Product	
Attribute/Method	**Description**
ID	Unique identifier for the product
Name	Name of the product
Category	Category of the product
Price	Price of the product
Description	Description of the product
Stock	Stock availability of the product
Seller: TYPE Class Seller	References the Seller class
System : TYPE Class System	References the System class
AddProduct()	Adds a new product
UpdateProduct()	Updates product information
DeleteProduct()	Removes a product from the system
ViewProduct()	Displays product details

Table 3.2.3.4 – Product Class

5. Order	
Attribute/Method	**Description**
OrderID	Unique identifier for the order
ProductID	References the Product class
ClientID	References the Client class
Quantity	Quantity of the product ordered
TotalPrice	Total price of the order
OrderStatus	Current status of the order
TimeStamp	Time of the order placement
System : TYPE Class System	References the System class
PlaceOrder()	Places a new order
UpdateOrder()	Updates order details
CancelOrder()	Cancels the order

TrackOrder()	Tracks the status of the order

Table 3.2.3.5 – Order Class

6. Payment	
Attribute/Method	**Description**
PaymentID	Unique identifier for the payment
OrderID	References the Order class
ClientID	References the Client class
Amount	Amount paid
PaymentMethod	Payment method used (e.g., credit card, PayPal)
PaymentStatus	Status of the payment (e.g., pending, completed)
TimeStamp	Time of the payment transaction
System : TYPE Class System	References the System class
ProcessPayment()	Processes the payment
RefundPayment()	Refunds the payment
VerifyPayment()	Verifies payment status

Table 3.2.3.6 – Payment Class

7. Shipping	
Attribute/Method	**Description**
ShippingID	Unique identifier for the shipping
OrderID	References the Order class
ShippingAddress	Shipping address of the order
ShippingStatus	Status of the shipping (e.g., pending, shipped)
Carrier	Carrier service used for shipping
TimeStamp	Time of the shipping process
System : TYPE Class System	References the System class
InitiateShipping()	Initiates the shipping process
TrackShipping()	Tracks the shipping status
UpdateShipping()	Updates shipping details

Table 3.2.3.7 – Shipping Class

8. Seller	
Attribute/Method	**Description**
SellerID	Unique identifier for the seller

Name	Name of the seller
Email	Seller's email address
Phone	Seller's phone number
ProductList	List of products sold by the seller
System : TYPE Class System	References the System class
Init() : self	Initializes the seller
AddProduct()	Adds a product to the catalog
UpdateProduct()	Updates product details
RemoveProduct()	Removes a product from the catalog
ViewSales()	Displays sales statistics

Table 3.2.3.8 – Seller Class

9. Ticket	
Attribute/Method	**Description**
TicketID	Unique identifier for the support ticket
ClientID	References the Client class
AdminID	References the Admin class
IssueDescription	Description of the issue reported
TicketStatus	Current status of the ticket (e.g., open, resolved)
PriorityLevel	Priority level of the ticket (e.g., high, medium, low)
TimeStamp	Time when the ticket was created
System : TYPE Class System	References the System class
CreateTicket()	Creates a new support ticket
UpdateTicket()	Updates ticket details
ResolveTicket()	Resolves the support ticket
CloseTicket()	Closes the ticket after resolution

Table 3.2.3.9 – Ticket Class

10. Bug	
Attribute/Method	**Description**
BugID	Unique identifier for the bug
Description	Description of the bug

SeverityLevel	Severity level of the bug (e.g., critical, minor)
Status	Current status of the bug (e.g., open, fixed)
ReportedBy: TYPE Class Client	References the Client class who reported the bug
AssignedTo: TYPE Class Admin	References the Admin class responsible for fixing the bug
TimeStamp	Time when the bug was reported
System : TYPE Class System	References the System class
ReportBug()	Reports a new bug
UpdateBug()	Updates bug details
FixBug()	Marks the bug as fixed
CloseBug()	Closes the bug after it’s fixed

Table 3.2.3.10 – Bug Class

11. Privilege	
Attribute/Method	**Description**
PrivilegeID	Unique identifier for the privilege
Name	Name of the privilege
Description	Description of the privilege
RoleID	References the Role class
System : TYPE Class System	References the System class
AssignPrivilege()	Assigns privilege to a role or user
RevokePrivilege()	Revokes privilege from a role or user
UpdatePrivilege()	Updates privilege details

Table 3.2.3.11 – Privilege Class

12. Role	
Attribute/Method	**Description**
RoleID	Unique identifier for the role
RoleName	Name of the role (e.g., admin, client, seller)
Description	Description of the role’s purpose
Privileges: TYPE Class Privilege	List of privileges associated with the role
System : TYPE Class System	References the System class
AssignRole()	Assigns a role to a user

UpdateRole()	Updates role details
RemoveRole()	Removes a role from a user

Table 3.2.3.12 – Role Class

13. Report	
Attribute/Method	**Description**
ReportID	Unique identifier for the report
Title	Title of the report
Content	Content of the report
GeneratedBy: TYPE Class System	References the System class that generated the report
GeneratedFor: TYPE Class Admin/Client/Seller	References the user for whom the report was generated
TimeStamp	Time when the report was generated
System : TYPE Class System	References the System class
GenerateReport()	Generates a report based on specified criteria
ViewReport()	Allows viewing of the report
DownloadReport()	Allows downloading of the report

Table 3.2.3.13 – Report Class

14. Service	
Attribute/Method	**Description**
ServiceID	Unique identifier for the service
ServiceName	Name of the service provided
Description	Description of the service
Price	Price of the service
Duration	Duration of the service
ClientID: TYPE Class Client	References the Client class availing the service
System : TYPE Class System	References the System class
AddService()	Adds a new service
UpdateService()	Updates service details
DeleteService()	Deletes a service
ViewService()	Displays service details

Table 3.2.3.14 – Service Class

15. Session	
Attribute/Method	**Description**
SessionID	Unique identifier for the session
ClientID: TYPE Class Client	References the Client class associated with the session
AdminID: TYPE Class Admin	References the Admin class associated with the session
StartTime	Time when the session started
EndTime	Time when the session ended
Status	Current status of the session (e.g., active, inactive)
System : TYPE Class System	References the System class
StartSession()	Starts a new session
EndSession()	Ends the current session
ExtendSession()	Extends the session duration
TerminateSession()	Terminates the session

Table 3.2.3.15 – Session Class

16. Connection	
Attribute/Method	**Description**
ConnectionID	Unique identifier for the connection
ClientID: TYPE Class Client	References the Client class
AdminID: TYPE Class Admin	References the Admin class
ConnectionType	Type of connection (e.g., secure, standard)
Status	Current status of the connection
System : TYPE Class System	References the System class
InitiateConnection()	Initiates a new connection
TerminateConnection()	Terminates the connection
MonitorConnection()	Monitors the connection status

Table 3.2.3.16 – Connection Class

The End