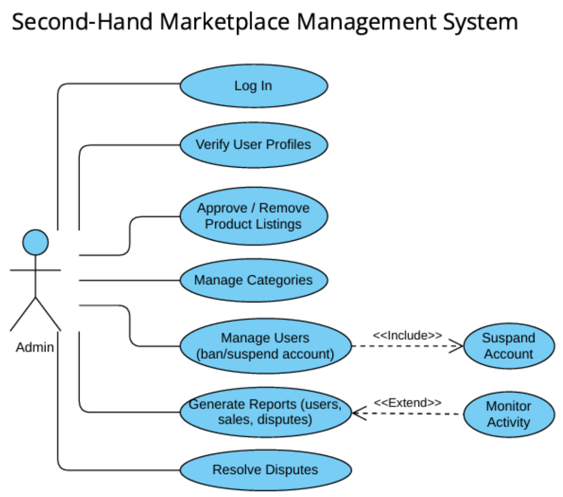
SOFTWARE DESIGN SYSTEM(SDS)

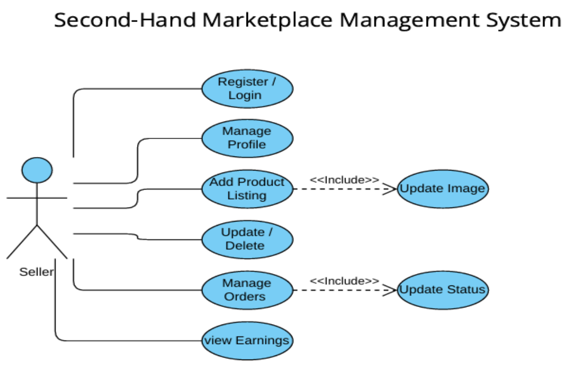
USE CASE DIAGRAM

- ADMIN:

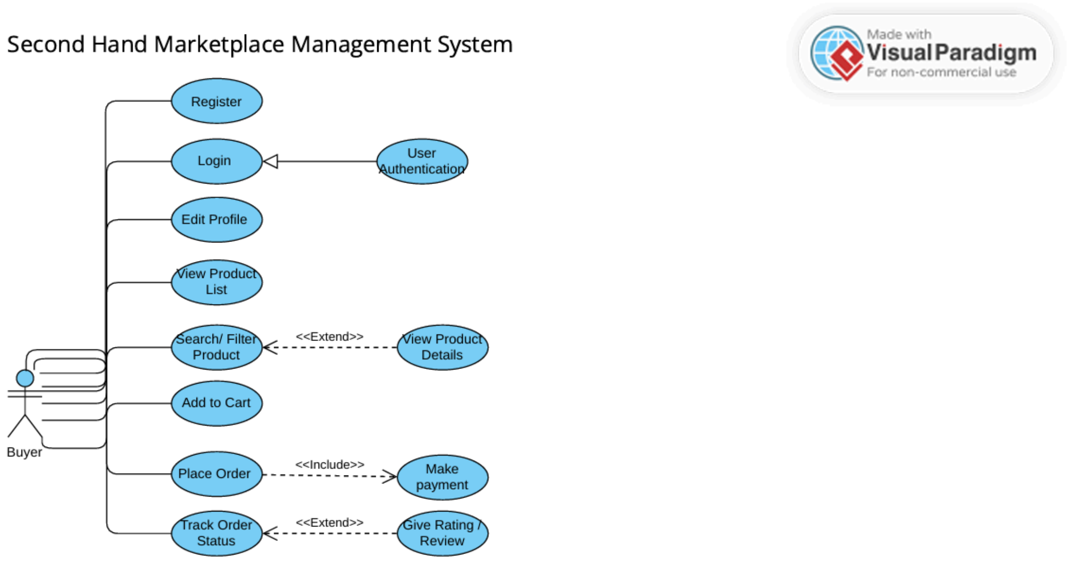


Second-Hand Marketplace Management System
Admin
Log In
Verify User Profiles
Approve / Remove Product Listings
Manage Categories
Manage Users (ban/suspend account)
<<Include>>
Suspand Account
Generate Reports (users, sales, disputes)
<<Extend>>
Monitor Activity
Resolve Disputes

- SELLER:



- BUYER:



EXTENDED USE CASE DIAGRAM

1. Use Case: UC-101 Register and Login (User Authentication)

Field	Description
Use Case ID	UC-101
Use Case Name	Register and Login (User Authentication)
Goal	To allow a user (Buyer, Seller, or Admin) to securely access the system or create a new account

Primary Actor	Buyer, Seller, Admin
Preconditions	The user is not currently logged into the system
Postconditions	If registration is successful, a new user account is created. If login is successful, the user is authenticated and redirected to their respective dashboard
Normal Flow(Basic Path)	1. The Actor accesses the Marketplace homepage. 2. The Actor selects the **"Register"** or **"Login"** option. 3. **(Registration):** The system displays the registration form (Username, Email, Password, User Type - Buyer/Seller). 4. The Actor enters the required details and submits the form. 5. The system validates the input, checks for unique username/email, and hashes the password. 6. The system creates the account record and redirects the Actor to the login page or dashboard. 7. **(Login):** The Actor enters their Email/Username and Password. 8. The system verifies the credentials against the database. 9. The system grants access and redirects the Actor to their designated dashboard

Alternative Flows	**A1: Forgot Password:** The Actor clicks "Forgot Password," enters their email, and the system sends a password reset link (or OTP) for the Actor to set a new password
Exception Flows	**E1: Invalid Credentials:** If the login credentials do not match, the system displays an "Invalid Credentials" error message and allows the Actor up to 3 retry attempts. **E2: Duplicate Registration:** If the provided Email or Username already exists, the system displays an error message "Account already exists" and prompts the user to log in.
Non-Functional Requirements	All passwords must be stored using strong **hashing** techniques

2. Use Case: UC-102 Verify User Profiles

Field	Description

Use Case ID	UC-102
Use Case Name	Verify User Profiles
Goal	To ensure platform trust and security by confirming the legitimacy of user profiles, particularly Sellers, who require verification to list products
Primary Actor	Admin
Preconditions	Admin is logged in. There are user profiles (primarily Sellers) with a 'Verification Pending' status.
Postconditions	The user's verification status is updated to 'Verified' or 'Rejected', and the user is notified

Normal Flow(Basic Path)	1. The Admin navigates to the **"User Verification"** section of the dashboard. 2. The system displays a queue of users awaiting verification, along with their submitted documents (e.g., ID proof). 3. The Admin selects a user and thoroughly reviews the submitted documentation against internal policy criteria. 4. **(Verify):** If documentation is satisfactory, the Admin clicks **"Verify Profile"**. The system updates the user's status to **'Verified'**. 5. **(Reject):** If documentation is insufficient or fraudulent, the Admin clicks **"Reject Verification"** and provides a mandatory reason. The system updates the status to **'Verification Rejected'**. 6. The system automatically sends an email notification to the user detailing the outcome.

Alternative Flows	**A1: Request Missing Documents:** Admin finds documentation incomplete and selects **"Request More Information"**. The system notifies the user of the missing items and returns the profile to the user for completion
Exception Flows	**E1: Document Access Error:** If the system fails to retrieve the user's submitted documents due to a storage error, an error message is displayed to the Admin, and the verification process is temporarily halted.
Non-Functional Requirements	All submitted verification documents must be stored securely using **data encryption at rest** and in compliance with privacy policies.

3. Use Case: UC-103 Approve/Remove Product Listings

Field	Description
Use Case ID	UC-103
Use Case Name	Approve/Remove Product Listings
Goal	To maintain product quality and platform integrity by moderating new product listings submitted by Sellers
Primary Actor	Admin

Preconditions	The Admin is logged in. There is at least one product with a 'Pending' status
Postconditions	The status of the selected Product is updated to 'Approved' (Available) or 'Removed' (Rejected), and the Seller is notified.

Normal Flow(Basic Path)	1. The Admin accesses the **"Admin Dashboard"** and navigates to the "Product Moderation" section. 2. The system displays a list of all products with the status **'Pending'**. 3. The Admin selects a Pending product to view its details (Description, Images, Seller Profile). 4. **(Approve):** If the listing meets all guidelines, the Admin clicks **"Approve"**. 5. The system updates the Product Status to **'Approved'** (or 'Available'). 6. **(Remove):** If the listing violates guidelines, the Admin clicks **"Remove"**, inputs a mandatory **Reason for Removal**, and clicks confirm. 7. The system updates the Product Status to **'Removed'** (or 'Rejected'). 8. The system sends an automated notification to the Seller regarding the status change.

Alternative Flows	**A1: Request Edit:** The Admin finds minor issues and selects **"Request Edit"**. The system sends a notification to the Seller detailing the required changes. The status remains 'Pending' (or changes to 'Editing Required').
Exception Flows	**E1: System Error:** If a database connection error occurs during the update process, the system logs the error and displays a message to the Admin, "Status update failed. Please try again."
Non-Functional Requirements	The Admin access must be secured with multi-factor authentication. All Admin moderation activities must be logged for auditing (**Monitor Activity**).

4. Use Case: UC-104 Manage Categories

Field	**Description**
Use Case ID	UC-104
Use Case Name	Manage Categories
Goal	To maintain and organize the product categorization structure (add, edit, or delete categories) of the marketplace
Primary Actor	Admin
Preconditions	Admin is logged in.

Postconditions	The list of available product categories (stored in the Categories table) is updated
Normal Flow(Basic Path)	1. The Admin navigates to the **"Category Management"** section. 2. The system displays the current hierarchical list of Categories. 3. **(Add Category):** The Admin inputs a new Category Name and an optional Parent Category, then clicks **"Add"**. 4. **(Edit Category):** The Admin selects an existing category, modifies its name or hierarchy, and clicks **"Save Changes"**. 5. **(Remove Category):** The Admin selects an obsolete category and confirms deletion. 6. The system validates the input and updates the Categories database table. 7. The system confirms the action to the Admin.

Alternative Flows	**A1: Merging Categories:** The Admin can select two similar categories and choose to merge them into a single category. The system automatically reassigns all associated products
Exception Flows	**E1: Deletion Conflict:** If the Admin attempts to delete a category that still contains active product listings, the system displays a warning and **prevents deletion** until all associated products are reassigned or archived.
Non-Functional Requirements	Category updates must be immediately reflected across the entire platform's **search and filtering interface**

5. **Use Case: UC-105 Manage Users (Suspend Account)**

Field	**Description**
Use Case ID	UC-105
Use Case Name	Manage Users (Ban/Suspend Account)
Goal	To maintain platform safety by restricting or terminating the access of users who violate policies or engage in fraud
Primary Actor	Admin
Preconditions	Admin is logged in. A user has been flagged for a policy violation, or a

	dispute has been resolved against them. (<<Include>> **Suspend Account**)
Postconditions	The user's account status is updated to **'Suspended'** or **'Banned'**, and their login access is immediately revoked.
Normal Flow(Basic Path)	1. The Admin navigates to the **"User Management"** section and finds the target user profile. 2. The Admin reviews the user's history (Disputes, Warnings, Listing Submissions) as evidence. 3. The Admin selects the user and clicks **"Suspend/Ban Account"**. 4. The system prompts the Admin to select the reason and duration (temporary Suspend or permanent Ban). 5. The Admin confirms the action. 6. The system updates the user's User Status to 'Suspended/Banned' and invalidates all current session tokens. 7. The system sends a formal notification to the user detailing the reason and duration of the action.

Alternative Flows	**A1: Issue Warning:** For minor offenses, the Admin can select **"Issue Warning"** instead. The system sends a formal warning notification but maintains the user's 'Active' status
Exception Flows	**E1: Admin Role Conflict:** The system will prevent the Admin from suspending or banning another Admin account, requiring a multi-admin consensus process
Non-Functional Requirements	This process requires **strong authentication** for the Admin (e.g., MFA) before confirming the action, due to the critical nature of revoking user access.

6. Included Use Case: UC-106 Suspend Account

Field	Description
Use Case ID	Uc-106
Use Case Name	Suspend Account
Goal	To immediately revoke login access and restrict platform activities for a user who has violated platform policies.
Primary Actor	Admin
Preconditions	The Admin is currently executing the Base Use Case **UC- Manage Users** (Ban/Suspend Account) or **UC-Resolve**

	Disputes. The target user's profile is identified.
Postconditions	The user's account status is updated to **'Suspended'** or **'Banned'**, their current session is terminated, and associated services (like active listings) are restricted.

Normal Flow(Basic Path)	1. The Base Use Case calls this Use Case (<<Include>> **Suspend Account**). 2. The system prompts the Admin to confirm the action and specify the **reason** and **duration** (e.g., 30 days suspension or permanent ban). 3. The Admin confirms the suspension/ban. 4. The system validates the Admin's credentials (may require a password/MFA for critical action). 5. The system updates the target user's User Status to 'Suspended/Banned' and records the action details in the activity log (<<Extend>> **Monitor Activity**). 6. The system immediately invalidates the user's current session tokens, logging them out. 7. The system sends a notification to the user detailing the action taken.

	8. Control returns to the Base Use Case.
Alternative Flows	**A1: Temporary Restriction:** If the duration is temporary, the system automatically schedules a task to reinstate the account status back to 'Active' upon expiry.
Exception Flows	**E1: Admin Role Conflict:** The system blocks the Admin from suspending another Admin account, requiring a different protocol. **E2: System Failure:** If the system fails to invalidate session tokens, it logs the error and retries the termination immediately.
Non-Functional Requirements	This action must be protected by a **secondary security check (MFA)** and performed with **minimal latency** to mitigate ongoing risk.

7. Use Case: UC-107 Generate Reports

Field	Description
Use Case ID	UC-107
Use Case Name	Generate Reports

Goal	To provide the Admin with detailed system analytics and operational data (users, sales, disputes) for strategic decision-making.
Primary Actor	Admin
Preconditions	Admin is logged in. The system contains transaction and user data
Postconditions	A detailed report file (e.g., PDF, CSV) is generated and made available to the Admin for download.

Normal Flow(Basic Path)	1. The Admin navigates to the **"Reporting/Analytics"** section. 2. The system displays a list of available report types (e.g., Sales Report, User Activity, Inventory Status, Dispute Summary). 3. The Admin selects a report type and specifies necessary parameters (e.g., date range, category, status). 4. The Admin clicks **"Generate Report"**. 5. The system queries the database and compiles the data (<<Extend>> **Monitor Activity** logs for security events). 6. The system generates the report file (e.g., PDF or CSV). 7. The system presents the report for immediate download or viewing.
Alternative Flows	**A1: Scheduled Reports:** The Admin can define and save a schedule (e.g., end of month) for the automatic generation and email delivery of specific reports.

Exception Flows	**E1: Timeout/Large Data Error:** If a requested report is too large to process immediately, the system notifies the Admin that the report generation is deferred to an offline process and will be emailed upon completion.
Non-Functional Requirements	Report generation must be optimized to ensure that **database load** from querying does not impact the live transaction performance of the marketplace.

8. Use Case: UC-108 Monitor Activity

Field	Description
Use Case ID	UC-108
Use Case Name	Monitor Activity
Goal	To log, track, and audit critical system events and user actions for security, compliance, and reporting purposes.
Primary Actor	Admin (The Admin reviews the logs, but the **System** is the initiator of the logging process).
Preconditions	A trigger event occurs (e.g., a report is generated, a user is suspended, or a suspicious login attempt happens).
Postconditions	A detailed timestamped record of the event is successfully written to the system's Activity Logs database table.

Normal Flow(Basic Path)	1. A Base Use Case (e.g., **UC-Generate Reports**) reaches an extension point (a critical action like data extraction or a security event). 2. The system initiates this Use Case (<<Extend>> **Monitor Activity**). 3. The system captures all relevant context: Event Type (e.g., 'Report Generation', 'Login Failure', 'User Suspension'), Timestamp, Actor ID, Target Entity ID, and outcome. 4. The system writes the log data to the **Audit Log** database asynchronously. 5. The Admin can later access these logs via the **UC- Generate Reports** function to review historical activity.
Alternative Flows	**A1: Alert Trigger:** If the monitored activity is a high-security event (e.g., 5 failed login attempts in 1 minute), the system triggers an immediate **real-time alert** to the security Admin team.
Exception Flows	**E1: Logging Failure:** If the system cannot connect to the Audit Log database, it fails gracefully, attempting to store the log entry locally or queue it for delayed submission, ensuring the main application functionality is unaffected.

Non-Functional Requirements	Logging must be performed **asynchronously** to avoid slowing down the performance of the Base Use Cases. Logs must be **immutable** and retained according to regulatory compliance (e.g., 7 years).

9. Use Case: UC-109 Resolve Disputes

Field	Description
Use Case ID	UC-109
Use Case Name	Resolve Disputes
Goal	To enable the Admin to fairly investigate and settle conflicts between Buyers and Sellers regarding transactions, quality, or communication
Primary Actor	Admin
Preconditions	Admin is logged in. A Buyer or Seller has previously initiated a dispute against a transaction.
Postconditions	The dispute status is updated to **'Resolved'**, and the necessary resolution (e.g., refund or dismissal) is executed.

Normal Flow(Basic Path)	1. The Admin accesses the **"Resolve Disputes"** section on the dashboard. 2. The system displays a list of active disputes (e.g., "Item Not as Described"). 3. The Admin selects a dispute and reviews all associated evidence (order details, chat logs, provided photos). 4. The Admin makes a final judgment and selects the resolution type (e.g., Full Refund, Partial Refund, Dispute Dismissed). 5. The Admin executes the resolution action (e.g., **triggers the refund process**). 6. The Admin updates the dispute status to **'Resolved'** and inputs the final decision details. 7. The system notifies both the Buyer and Seller of the final decision.
Alternative Flows	**A1: Dispute Escalation:** If the dispute reveals potential fraud, the Admin may escalate it, leading to the use of

	UC-405 (Manage Users) to suspend the involved user.
Exception Flows	**E1: Refund Processing Failure:** If the system fails to process a required refund (e.g., due to payment gateway error), the Admin is notified to process it manually, and the dispute status remains 'Resolution Pending' until completed.
Non-Functional Requirements	Dispute resolution documentation must be preserved for at least **7 years** for legal compliance and historical reference.

10. Generalization Include Use Case: UC-110 User Authentication

Field	**Description**
Use Case ID	UC-110
Use Case Name	User Authentication
Goal	To securely verify the identity and credentials of a user during registration or login.
Primary Actor	System (Initiated by Register/Login)
Preconditions	User has submitted credentials for verification.

Postconditions	Credentials are confirmed as valid or invalid.
Normal Flow(Basic Path)	1. The Base Use Case (UC-308 or UC-301) calls this Use Case. 2. The system retrieves the hashed password from the database using the provided email/username. 3. The system compares the submitted password (after hashing) with the stored hash value. 4. If the hashes match, authentication is successful. 5. If the hashes do not match, authentication fails. 6. Control returns to the Base Use Case with the authentication result.
Alternative Flows	**A1: Forgot Password:** Initiated during the Login process.
Exception Flows	**E1: Invalid Credentials:** Handled in the Login flow.

Non-Functional Requirements	Authentication must use secure **HTTPS** transmission and comply with data privacy standards.

11. Use Case: UC-111 Edit Profile

Field	**Description**
Use Case ID	UC-111
Use Case Name	Edit Profile
Goal	To allow the Buyer to manage and update their personal information (e.g., address, contact details).
Primary Actor	Buyer
Preconditions	The Buyer is logged in.
Postconditions	The Buyer's record in the Users table is updated with the new information.

Normal Flow(Basic Path)	1. The Buyer navigates to the **"Edit Profile"** or **"Account Settings"** section. 2. The system displays editable fields pre-filled with current data (e.g., Name, Address, Phone). 3. The Buyer modifies the desired information. 4. The Buyer clicks **"Save Changes"**. 5. The system validates the input and updates the user record. 6. The system displays a confirmation message.
Alternative Flows	**A1: Update Default Address:** The Buyer changes their primary delivery address, which is automatically saved for future checkouts.
Exception Flows	**E1: Invalid Data:** If the Buyer inputs an invalid address format, the system displays an error and prevents saving until corrected.
Non-Functional Requirements	The system must require the Buyer to re-authenticate (enter password) before saving changes to sensitive data (e.g., email).

12. Use Case: UC-112 View Product List

Field	Description
Use Case ID	UC-112
Use Case Name	View Product List
Goal	To display a general listing of available products on the marketplace.
Primary Actor	Buyer (Can also be Unregistered User)
Preconditions	The system contains active product listings (status = 'Approved').
Postconditions	The system displays the main product list page, usually sorted by date or popularity.
Normal Flow(Basic Path)	1. The Actor accesses the marketplace homepage or category page. 2. The system retrieves a curated list of active products from the Products table. 3. The system displays the list, showing key information for each item (Title, Price, Main Image).

	4. The Actor can scroll through the list or click on an item for details.
Alternative Flows	**A1: Sort List:** The Actor can select different sorting criteria (e.g., price ascending, date posted) to reorder the list display.
Exception Flows	**E1: Empty List:** If no products are available, the system displays a "No products currently available" message.
Non-Functional Requirements	The initial list load time should be optimized to be under **3 seconds**.

13. Use Case: UC-113 Search/Filter Product

Field	**Description**
Use Case ID	UC-113
Use Case Name	Search and Filter Products
Goal	To allow a Buyer to efficiently locate desired products by category, keywords, price, and other attributes.
Primary Actor	Buyer (Can also be Unregistered User)
Preconditions	The system contains active product listings (status = 'Approved')
Postconditions	The system displays a refined list of products matching the search and filter criteria.

Normal Flow(Basic Path)	1. The Actor accesses the product listing page. 2. The Actor inputs a search term (e.g., product name, brand) into the search bar. 3. **OR** The Actor utilizes filtering tools (e.g., Category, Price Range, Location, Product Condition). 4. The Actor clicks the "Search" or "Apply Filter" button. 5. The system executes a query on the Products table using the input criteria. 6. The system displays the resulting list of products, sorted by relevance or chosen preference. 7. The Actor can proceed to view product details from the list.
Alternative Flows	**A1: No Results:** If the search or filter returns no results, the system displays a "No matching products found" message and suggests relevant categories or popular items.

Exception Flows	**E1: Malformed Query:** If the search input is excessively long or contains unsupported characters, the system sanitizes the input before query execution to prevent errors
Non-Functional Requirements	Search and filtering results must be delivered with a high response speed (under 1 second) to ensure a fluid user experience

14. Extended Use Case: UC-111 View Product Details

Field	**Description**
Use Case ID	UC-114
Use Case Name	View Product Details
Goal	To allow a Buyer to examine all relevant information about a specific product before making a purchase decision
Primary Actor	Buyer (Can also be Unregistered User)
Preconditions	The Buyer has accessed a product list and selected an item. The product status is 'Approved'.
Postconditions	The system displays the comprehensive product page, including seller information and reviews.

Normal Flow(Basic Path)	1. The Buyer clicks on a product listing from the search results or any product list. 2. The system queries the Products table and the Users table (for Seller info) using the Product ID. 3. The system displays the detailed product page, including: Product Title, detailed Description, Condition, Price, Images, Seller Rating, and Location. 4. The system also displays past reviews and ratings for the seller and product (if available). 5. From this page, the Buyer can perform other actions like **Add to Cart** or **Initiate Negotiation**
Alternative Flows	**A1: Item Flagging:** If the Buyer suspects the listing is inappropriate or fraudulent, they can click a **"Report Listing"** button, initiating a notification to the Admin for review
Exception Flows	**E1: Listing Not Found:** If the product ID is invalid or the listing has been archived/removed, the system displays a 404 error page or a "Product Unavailable" message.

Non-Functional Requirements	Product images must load quickly and be scalable for different devices (mobile-friendly display).

15. Use Case: UC-305 Add to Cart

Field	**Description**
Use Case ID	UC-115
Use Case Name	Add to Cart
Goal	To allow the Buyer to temporarily save products for future purchase or multiple item checkout.
Primary Actor	Buyer (Logged In)
Preconditions	The Buyer is logged in. The selected product has an available quantity (stock > 0).
Postconditions	The selected product is added to the Buyer's Cart record in the database.

Normal Flow(Basic Path)	1. The Buyer is on the **View Product Details** page. 2. The Buyer confirms the desired quantity (default is 1) and clicks **"Add to Cart"**. 3. The system checks the current inventory of the product. 4. If stock is available, the system adds the item and quantity to the Buyer's Cart table. 5. The system displays a confirmation message (e.g., "Item added to cart") and updates the Cart icon count. 6. The Buyer can continue browsing or proceed to checkout.
Alternative Flows	**A1: Update Cart Quantity:** The Buyer can view the cart and change the quantity of an existing item. The system validates the new quantity against the available stock
Exception Flows	**E1: Out of Stock:** If the quantity requested is greater than the available stock, the system displays an error message "Insufficient stock available"

	and suggests buying the maximum available quantity
Non-Functional Requirements	Cart updates must use **AJAX** or similar technology to avoid page reloads, ensuring a smooth shopping experience

16. **Use Case: UC-116 Place Order**

Field	**Description**
Use Case ID	UC-116
Use Case Name	Place Order
Goal	To allow a Buyer to successfully purchase items from their shopping cart
Primary Actor	Buyer
Preconditions	The Buyer is logged in, and their Cart contains at least one available item
Postconditions	A new Order record is created, product inventory is reduced, and the Buyer's Cart is cleared

Normal Flow(Basic Path)	1. The Buyer views the products and clicks **"Add to Cart"**. 2. The Buyer navigates to the Cart and clicks **"Proceed to Checkout"**. 3. The system displays the Checkout page, showing the order summary and requiring Delivery Options (In-person/Shipping) and Address Confirmation. 4. The Buyer confirms the details and proceeds to the Payment step. 5. The system displays payment options. 6. The Buyer completes the payment process (<<Include>> **Make Payment**). 7. The system receives payment confirmation, creates the Order record, updates the Product quantity, and sets the Order Status to 'Processing'. 8. The system displays an "Order Confirmation" page and sends an email notification to the Buyer.

Alternative Flows	**A1: Price Negotiation:** Before proceeding to payment, the Buyer may initiate a **Negotiation/Offer** with the Seller, which must be accepted by the Seller before the Checkout process can begin
Exception Flows	**E1: Payment Failure:** If the payment transaction fails (e.g., card decline), the system displays a "Payment Failed" message and holds the order with a 'Payment Pending' status until a successful retry. **E2: Item Out of Stock:** If, during checkout, the quantity of an item becomes zero, the system removes the item from the Cart, notifies the Buyer, and asks them to proceed with the remaining items.
Non-Functional Requirements	All payment information must be securely handled using **encryption** and comply with relevant security standards.

17. Included Use Case: UC Make Payment

Field	**Description**
Use Case ID	UC-117
Use Case Name	Make Payment

Goal	To securely process the financial transaction for the order via integrated payment gateways.
Primary Actor	Buyer
Preconditions	The Base Use Case (**UC- Place Order**) is currently being executed. The order amount is finalized.
Postconditions	The payment gateway confirms the transaction, and the system records the payment as 'Successful' in the Transactions table.

Normal Flow(Basic Path)	1. The Base Use Case (UC-306) calls this Use Case (<<Include>> **Make Payment**). 2. The system redirects the Buyer to the payment gateway interface (or loads an embedded form). 3. The Buyer enters their payment method details (e.g., card number, expiry, CVV). 4. The payment gateway processes the transaction. 5. The payment gateway sends a **Success** confirmation back to the system. 6. The system updates the order and transaction tables. 7. Control returns to the Base Use Case.
Alternative Flows	**A1: Payment Via Wallet:** The Buyer selects an integrated digital wallet (e.g., PayPal, bKash) and completes the

	payment through that service's interface.
Exception Flows	**E1: Payment Failure:** If the transaction is declined by the bank, the gateway sends a **Failure** notification. The system displays an error and prompts the Buyer to try again or use a different method.
Non-Functional Requirements	Payment gateway integration must meet **PCI DSS compliance** standards for handling card data.

18. Use Case: UC-118 Track Order Status

Field	**Description**
Use Case ID	UC-118
Use Case Name	Track Order Status
Goal	To allow the Buyer to monitor the current status and fulfillment progress of their placed orders.
Primary Actor	Buyer
Preconditions	The Buyer is logged in. The Buyer has at least one order with status 'Processing', 'Shipped', or 'Ready for Meetup'.
Postconditions	The system displays the most current fulfillment status and location information for the selected order

Normal Flow(Basic Path)	1. The Buyer navigates to the **"Order History"** or **"My Orders"** section of their profile. 2. The system displays a list of all past and current orders. 3. The Buyer selects a specific order to view its details. 4. The system retrieves and displays the current **Order Status** (e.g., 'Processing', 'Shipped', 'Completed') and any associated **tracking number** or meetup schedule. 5. The Buyer receives real-time updates on status changes (<<Extend>> for all status updates).
Alternative Flows	**A1: Initiate Dispute:** If the order status is stuck or significantly delayed, the Buyer can click **"Raise a Dispute"**, leading to the use of UC-402 (Resolve Disputes) by the Admin.
Exception Flows	**E1: Tracking Data Error:** If the external shipping tracker is unavailable, the system displays a message indicating the temporary service outage and the last known status.

Non-Functional Requirements	The system must send **push or email notifications** to the Buyer when the order status is updated by the Seller

19. **Use Case: UC-119 Give Rating and Review**

Field	Description
Use Case ID	UC-119
Use Case Name	Give Rating and Review
Goal	To allow a Buyer to provide feedback on the Seller and the product after a successful transaction, promoting platform trust.
Primary Actor	Buyer
Preconditions	The Buyer is logged in. The transaction is marked as **'Delivered'** or **'Completed'**
Postconditions	A new Rating and Review record is created, and the Seller's average rating is updated

Normal Flow(Basic Path)	1. The Buyer navigates to the **"Order History"** section. 2. The Buyer identifies a completed order and clicks **"Rate Seller/Product"**. 3. The system displays a form requiring a star rating (e.g., 1-5) and a written review/comment. 4. The Buyer submits the rating and review. 5. The system validates the input, checks for inappropriate content, and saves the data to the Ratings and Reviews table. 6. The system recalculates and updates the Seller's average rating on their profile. 7. The system displays a confirmation message.
Alternative Flows	**A1: Edit Review:** The Buyer can edit or delete their submitted review within a specified timeframe (e.g., 7 days). The system updates the record and recalculates the rating.

Exception Flows	**E1: Content Flagged:** If the review is automatically flagged for profanity or malicious content, the system blocks the submission and sends it for manual review by the Admin.
Non-Functional Requirements	The rating system must be **statistically sound** and prevent users from submitting multiple ratings for the same transaction.

20. Use Case: UC-120 Update Profile Information (Manage Profile)

Field	Description
Use Case ID	UC-120
Use Case Name	Manage Profile
Goal	To allow a Seller to modify and maintain their personal, contact, and business-related profile details.
Primary Actor	Seller
Preconditions	The Seller is logged in.
Postconditions	The Seller's record in the Users table is updated with the new information.

Normal Flow(Basic Path)	1. The Seller navigates to the **"Manage Profile"** or **"Account Settings"** section. 2. The system displays editable fields (e.g., Name, Email, Address, Payout Bank Account details). 3. The Seller modifies the desired information. 4. The Seller enters their **current password** to confirm the changes. 5. The Seller clicks **"Save Changes"**. 6. The system validates the input and updates the user record. 7. The system displays a confirmation message.
Alternative Flows	**A1: Change Password:** The Seller can navigate to a separate **"Change Password"** function within the profile settings.
Exception Flows	**E1: Invalid Data:** If the Seller inputs an invalid format (e.g., incorrect phone number), the system displays an error and requires correction.

Non-Functional Requirements	Changes to sensitive data (e.g., Payout Details) must be protected by a **secondary security check** (e.g., OTP).

21. **Use Case: UC-121 Add Product Listing**

Field	Description
Use Case ID	UC-121
Use Case Name	Add Product Listing
Goal	To allow a Seller to list a second-hand item for sale on the marketplace
Primary Actor	Seller
Preconditions	The Seller is logged in and their profile has been **verified** by an Admin.
Postconditions	A new record is created in the Products database table, and its status is set to **'Pending'** for Admin review

Normal Flow(Basic Path)	1. The Seller navigates to the "Seller Dashboard" and clicks **"Add Product Listing"**. 2. The system displays the product submission form. 3. The Seller fills in all required fields: Product Title, Category, detailed Description, Price, Quantity, and Product Condition (Used/Like New). 4. The Seller uploads product images (<<Include>> **Update Image**). 5. The Seller clicks **"Submit Listing"**. 6. The system validates the input data and image formats. 7. The system saves the product data to the database with a 'Pending' status. 8. The system displays a confirmation message "Product Listing submitted for approval."

Alternative Flows	**A1: Save as Draft:** The Seller can click "Save as Draft" at any point, and the system saves the listing data without setting the status to 'Pending' or submitting it for review
Exception Flows	**E1: Missing Mandatory Fields:** If the Seller attempts to submit the listing without filling in a mandatory field (e.g., Price), the system highlights the missing fields and prevents submission. **E2: Image Upload Failure:** If the image upload fails due to file size or format constraints, the system displays an error and prompts the Seller to re-upload the images
Non-Functional Requirements	Product images must be compressed and optimized to ensure the page load time remains under the **3-second limit**. (Referenced from SRS)

22. **Included Use Case: UC-122 Update Image**

Field	**Description**
Use Case ID	UC-122
Use Case Name	Update Image

Goal	To securely upload, store, and associate an image (product photo or profile picture) with the appropriate entity (Product or User).
Primary Actor	Seller
Preconditions	The Seller is currently executing a Base Use Case that requires an image upload
Postconditions	The image file is successfully uploaded to the server, resized/optimized, and the database record (Product or User) is updated with the image URL/reference.

Normal Flow(Basic Path)	1. The Base Use Case (e.g., UC-201) calls this Use Case (<<Include>> **Update Image**). 2. The system displays the image upload interface. 3. The Seller selects one or more image files from their local device. 4. The Seller clicks **"Upload"**. 5. The system receives the file(s), validates the file type (e.g., JPEG, PNG) and size. 6. The system resizes and compresses the image(s) for optimization. 7. The system stores the optimized image(s) in the cloud/server storage. 8. The system updates the corresponding Product or User database record with the new image URL(s).

	9. Control returns to the Base Use Case.
Alternative Flows	**A1: Delete Existing Image:** The Seller can select and delete an existing associated image. The system removes the image from storage and deletes the corresponding database reference.
Exception Flows	**E1: Invalid File Type/Size:** If the file is not a supported type or exceeds the maximum size limit, the system displays an error message "Unsupported format or file too large" and rejects the upload. **E2: Storage Failure:** If the server fails to save the file, the system displays a "Storage Error" and logs the incident.
Non-Functional Requirements	Images must be checked for malicious content during upload. Image file names should be sanitized and stored securely.

23. Use Case: UC-123 Manage Product Listings (Update / Delete)

Field	**Description**
Use Case ID	UC-123
Use Case Name	Update/Delete

Goal	To allow a Seller to view, edit, or remove their existing product listings from the marketplace.
Primary Actor	Seller
Preconditions	The Seller is logged in and has existing product listings.
Postconditions	The selected product record in the Products table is updated, deleted, or its status is changed.
Normal Flow(Basic Path)	1. The Seller navigates to the **"My Listings"** section. 2. The system displays a list of the Seller's products with their current status. 3. **(Update Listing):** The Seller selects a listing, modifies its details, and clicks **"Update"**. (If images are updated, <<Include>> **Update Image**). 4. **(Delete Listing):** The Seller selects an item and clicks **"Remove/Archive"**. The product status is updated to 'Archived'. 5. The system validates and updates/archives the product record.

	6. The system displays a success message.
Alternative Flows	**A1: Relist Item:** The Seller can relist an 'Archived' item by changing its status to 'Pending'
Exception Flows	**E1: Deletion Conflict:** The system will prevent deletion of a listing if there is an **active, pending order** associated with it.
Non-Functional Requirements	Any update to an active listing should be immediately visible to Buyers.

24. Use Case: UC-124 Manage Orders

Field	**Description**
Use Case ID	UC-124
Use Case Name	Manage Orders
Goal	To allow a Seller to process, track, and update the status of orders placed for their products.
Primary Actor	Seller
Preconditions	The Seller is logged in. There is at least one order with the status **'Processing'** (successfully paid).
Postconditions	The order status is updated in the Orders table, and the Buyer is notified of the status change.

Normal Flow(Basic Path)	1. The Seller navigates to the **"Manage Orders"** section. 2. The system displays a list of incoming orders awaiting fulfillment. 3. The Seller selects an order to view details (Buyer info, delivery method). 4. The Seller updates the order status (<<Include>> **Update Status**) to reflect the current stage (e.g., 'Shipped', 'Ready for Meetup'). 5. The system saves the new status and sends an automatic notification to the Buyer. 6. Upon successful delivery/meetup confirmation, the Seller updates the status to **'Completed'**.
Alternative Flows	**A1: Cancel Order:** The Seller can select 'Cancel Order' with a reason, leading to a refund process and Buyer notification.
Exception Flows	**E1: Status Update Failure:** If the database fails to update the status, the system logs the error and displays a retry message to the Seller.

Non-Functional Requirements	The system must ensure **Buyer notification** is instantaneous upon Seller status change.

25. Included Use Case: UC-125 Update Status

Field	**Description**
Use Case ID	UC-125
Use Case Name	Update Status
Goal	To change the fulfillment status of a specific order, thereby communicating the progress to the Buyer.
Primary Actor	Seller
Preconditions	The Seller is currently executing the **Manage Orders** Base Use Case. The order must be in a state that allows status progression (e.g., 'Processing' can move to 'Shipped', but not vice-versa without Admin intervention).
Postconditions	The Orders database record is updated with the new status, a timestamp of the change is recorded, and the Buyer is notified.

Normal Flow(Basic Path)	1. The Base Use Case (**Manage Orders**) calls this Use Case (<<Include>> **Update Status**). 2. The Seller selects the desired new status from a dropdown menu (e.g., 'Processing' to **'Shipped'**, 'Shipped' to **'Completed'**). 3. The Seller enters any required associated information (e.g., tracking number for 'Shipped' status). 4. The Seller clicks **"Confirm Status Change"**. 5. The system validates the status change (ensures logical progression). 6. The system updates the Order Status field in the database and records the status change history. 7. The system automatically triggers a notification (email/push) to the Buyer. 8. Control returns to the Base Use Case.

Alternative Flows	**A1: Add Fulfillment Details:** If the status is 'Shipped', the Seller includes the courier name and tracking URL/number, which is saved alongside the status.
Exception Flows	**E1: Invalid Status Transition:** If the Seller attempts to move an order from 'Completed' back to 'Processing', the system displays an error "Invalid status transition" and requires Admin permission for reversal. **E2: Missing Required Field:** If the Seller confirms 'Shipped' status without entering a tracking number, the system blocks the update and prompts for the tracking information.
Non-Functional Requirements	Every status update must generate a **historical log record** for auditing and dispute resolution.

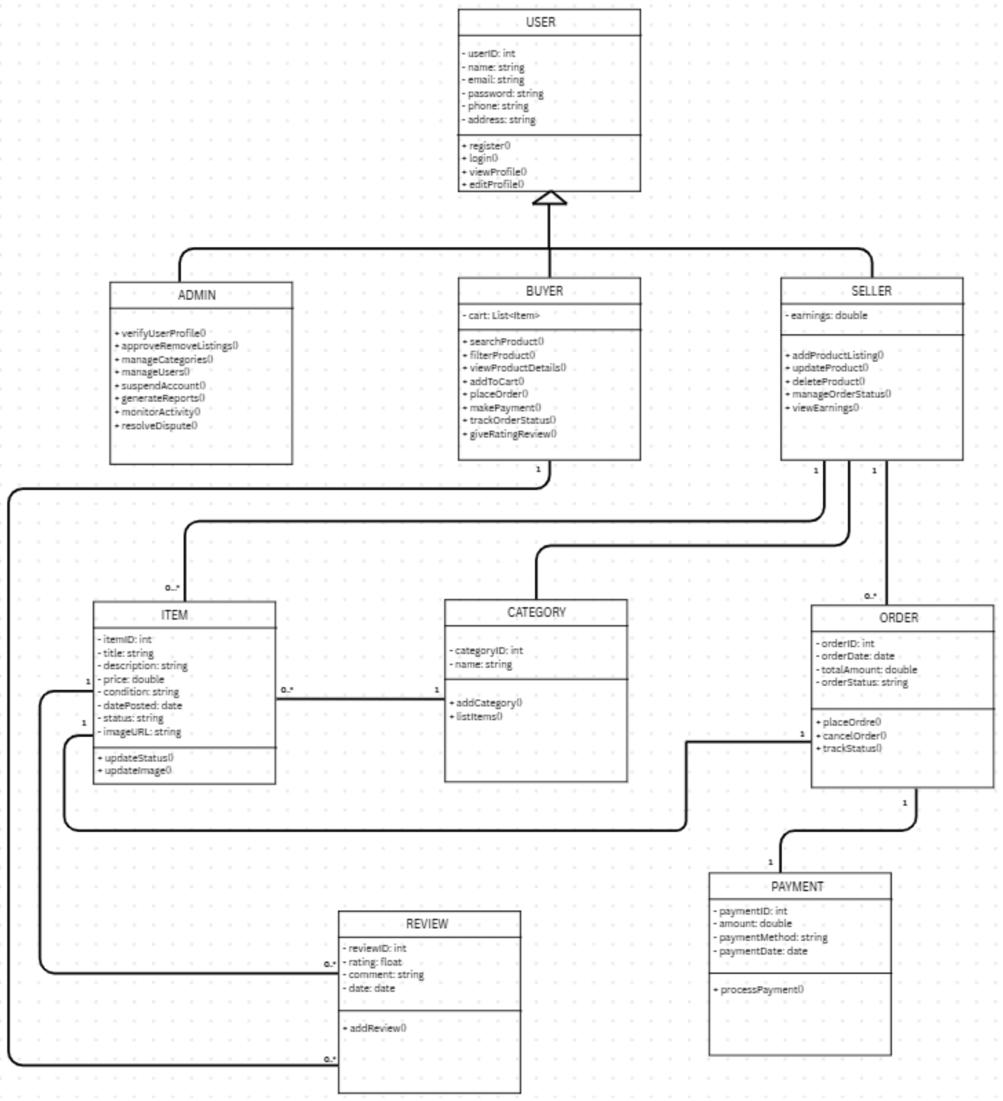
26. Use Case: UC-126 View Sales and Earnings

Field	Description
Use Case ID	UC-126
Use Case Name	View Earnings
Goal	To provide the Seller with a comprehensive overview of their transactional performance, revenue, and pending payouts.

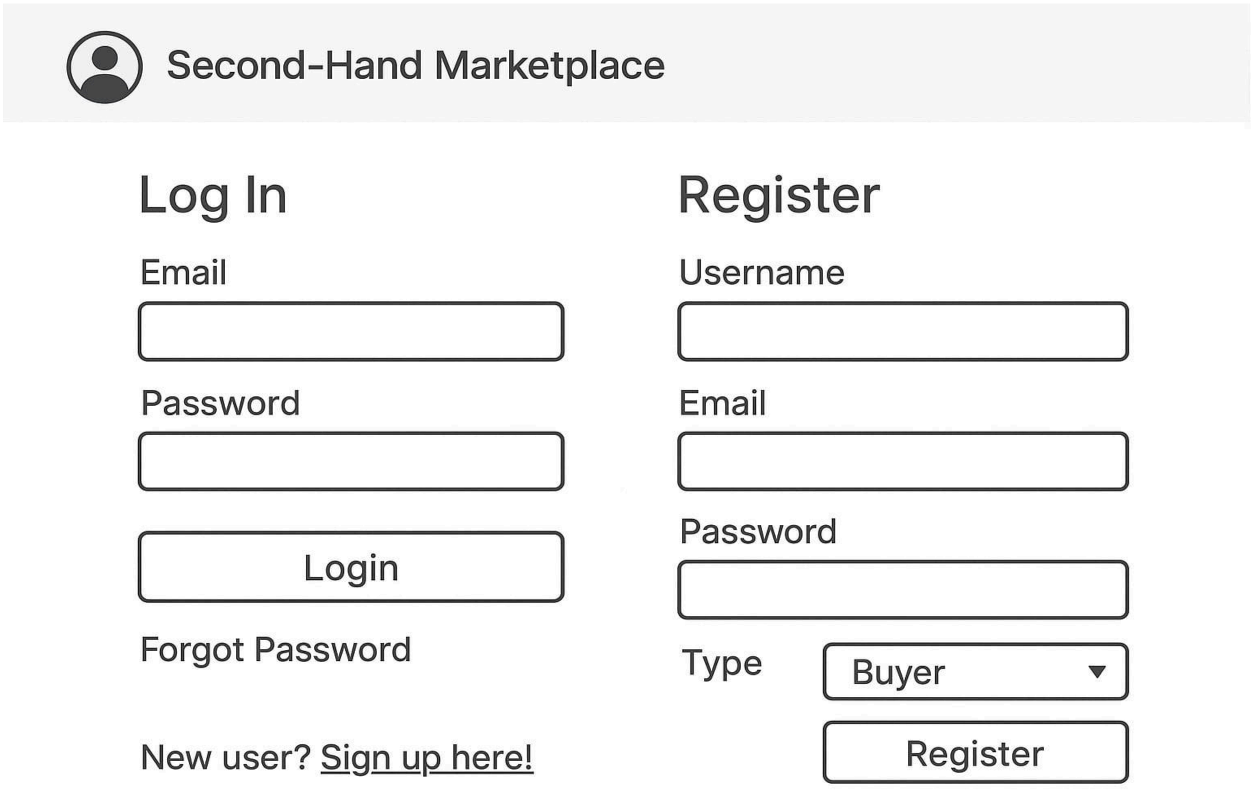
Primary Actor	Seller
Preconditions	The Seller is logged in. There are completed orders associated with the Seller.
Postconditions	The Seller is presented with real-time financial and sales data based on the selected period.
Normal Flow(Basic Path)	1. The Seller navigates to the **"view Earnings"** or **"Sales & Payouts"** section. 2. The system displays key metrics (e.g., Total Sales, Commission Deducted, Net Earnings, Available Balance). 3. The Seller can select a time period (e.g., last month, custom dates). 4. The system queries the Orders and Transactions tables to calculate the summary data. 5. The system presents the detailed transaction list and summary metrics in a structured format. 6. The Seller can proceed to **Request Payout** from this screen.

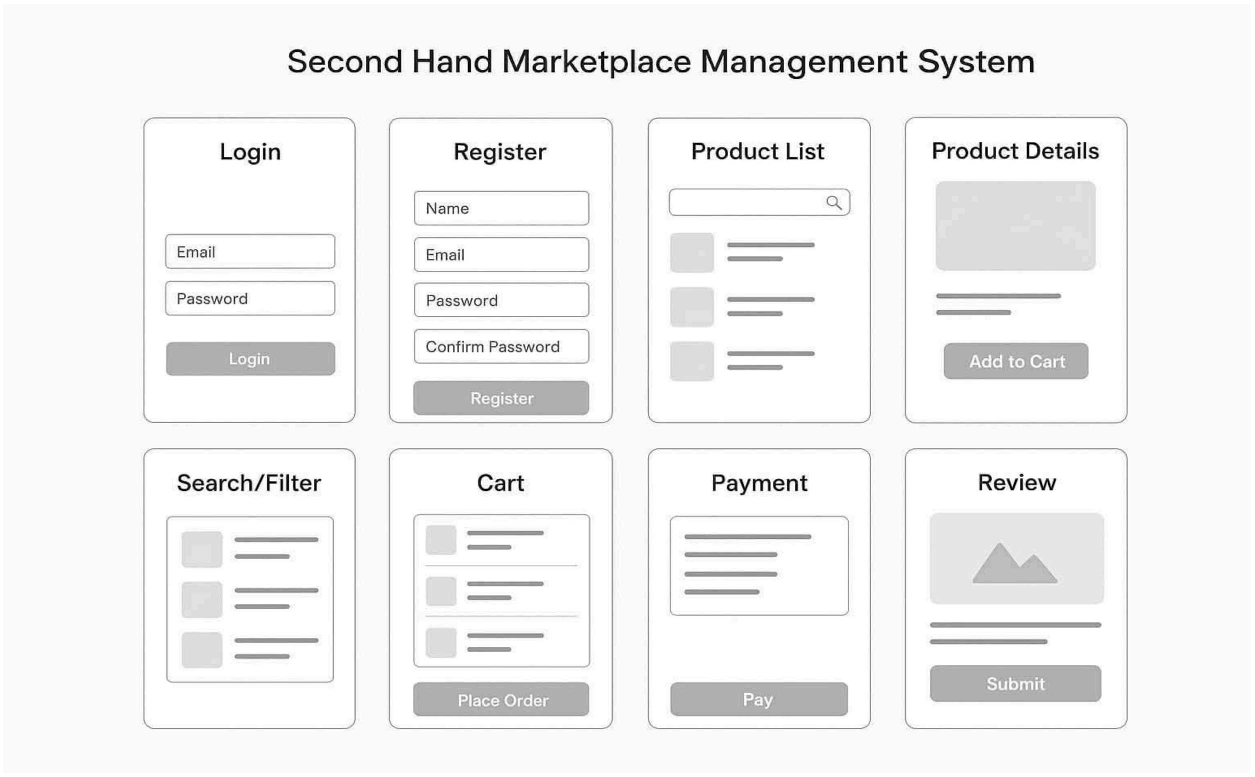
Alternative Flows	**A1: Export Report:** The Seller can generate a downloadable CSV/PDF report of all earnings within the selected period.
Exception Flows	**E1: Data Retrieval Failure:** If the system cannot connect to the financial database, it displays a maintenance notice and the last known earnings.
Non-Functional Requirements	Financial data must be **accurate and auditable**, reflecting transactions instantly.

CLASS DIAGRAM



USER INTERFACE(UI)





ARCHITECTURAL DESIGN PATTERN(MVC)

The Second-Hand Marketplace Management System (SHMMS) has been developed using the Model–View–Controller (MVC) architectural pattern. Employing MVC enables SHMMS to maintain a clear separation between user interface, business logic, and data management, supporting modular development and making system maintenance more manageable.

Overview

The MVC architecture divides the application into three interconnected layers: Model, View, and Controller. Each layer performs distinct roles, allowing developers to handle business logic, data processing, and user interface independently. This design ensures that updates to one layer (such as database changes or UI redesign) do not disrupt the others.

Components

Model:

The Model layer represents the data and the logic that governs access and modification of this data. It is directly connected to the database and performs operations such as creating, reading, updating, and deleting records.

Each entity—User, Product, Category, CartItem, Order, Payment, Conversation, and Chat—directly corresponds to its own database table in the relational schema. Each of these corresponds to database tables defined in the relational schema. The Model ensures data consistency and encapsulates the application's core business rules.

View:

The View layer handles presentation logic, displaying the data received from the Controller as structured and user-centric UI components.

In SHMMS, Views are constructed using PHP-based templating and static HTML, rendering product listings, user profile interfaces, order detail panels, and chat modules. This layer ensures intuitive user navigation and seamless platform interaction.

Controller:

The Controller acts as an intermediary between the Model and the View. It processes user requests, performs validation, invokes the necessary Model operations, and selects the appropriate View to render the response.

In SHMMS, controllers such as AuthController, ProductController, OrderController, and AdminController manage actions like registration, login, product management, order processing, and user verification.

Advantages of the MVC Architecture:

- **Separation of Concerns:** The Model, View, and Controller layers each have distinct responsibilities, which simplifies maintenance and enhances code organization.
- **Reusability:** Components such as authentication, navigation, and layout templates can be reused across modules.
- **Scalability:** The modular structure of MVC facilitates the integration of additional features, such as a bidding system or delivery tracking, allowing for easy expansion.
- **Parallel Development:** Multiple developers can work simultaneously on different layers (front-end, back-end, and database).
- **Security and Consistency:** Centralized control within controllers enforces uniform validation, access control, and error handling throughout the application.