# CHAPTER THREE - PROPOSED METHOD / SYSTEM / SOFTWARE

## 3.1 Description of the Proposed System

The Fashion Hub platform is a digital solution designed to empower boutique owners and fashion designers in Sekondi-Takoradi by providing an affordable, user-friendly, and localized e-commerce platform. The system integrates essential tools for managing fashion products, processing payments, and facilitating communication between vendors and customers.

**Key features of the Fashion Hub include:**

* **User Registration & Profiles:** Vendors can create and manage their business profiles, showcasing their products and services.
* **Product Management:** Fashion designers and boutique owners can easily upload, update, and manage their fashion items.
* **Payment Integration:** The platform supports Paystack, a local payment gateway, for easy and secure transactions.
* **Communication Tools:** The system integrates WhatsApp for direct customer communication, while the AI chatbot powered by BotPress and OpenAI assists with onboarding, product queries, and design assisting.
* **Mobile Optimization:** The platform is designed with mobile-first principles, ensuring accessibility for users with low bandwidth or minimal technical expertise.
* **Local Customization:** The platform accommodates local language preferences, currency, and shipping options specific to the Ghanaian market.

## 3.2 Systems Requirement Specifications

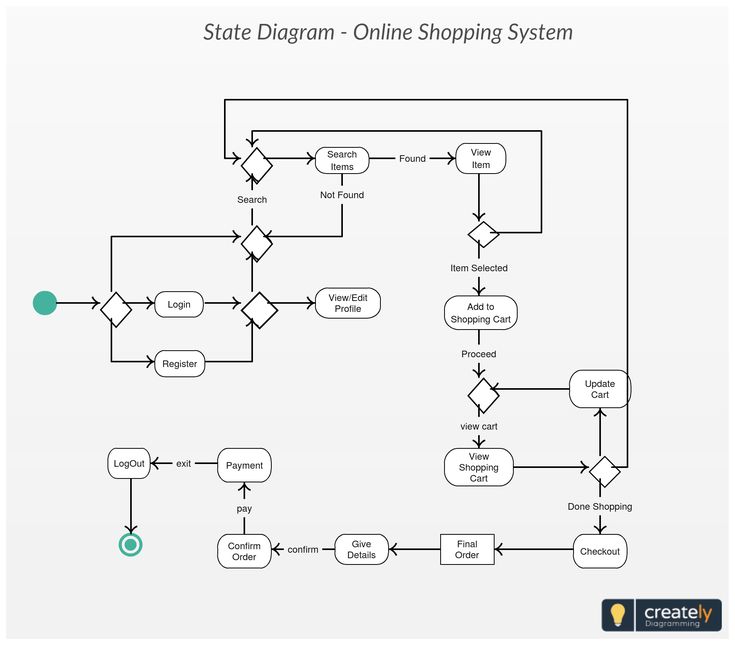
**Use Case Diagram:**



The use case diagram depicts the interactions between the users and the system. It covers the following key roles:

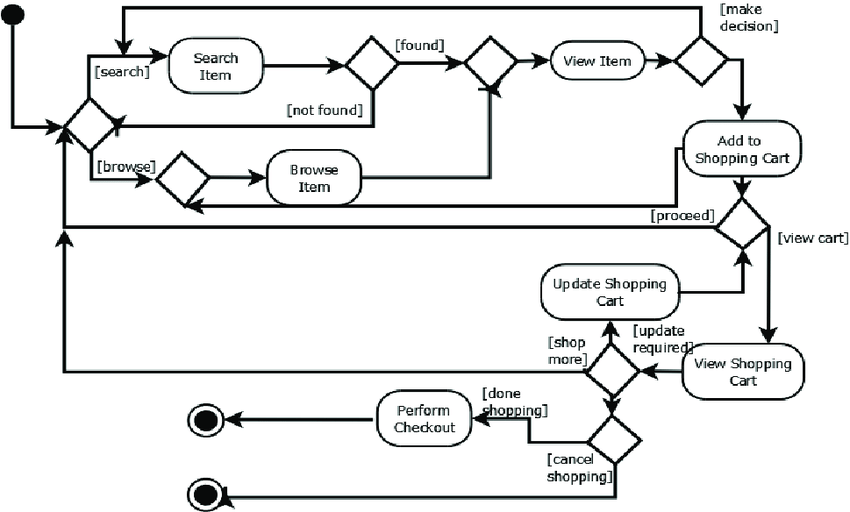
* **Admin:** Manages user accounts, product listings, and reviews. Admin has access to all functionalities on the platform.
* **Vendor (Boutique Owners / Designers):** Manages their business profile, uploads products, processes orders, and communicates with customers.
* **Customer:** Browses products, makes purchases, and communicates with vendors via the AI chatbot or WhatsApp.

**State Transition Diagram:**



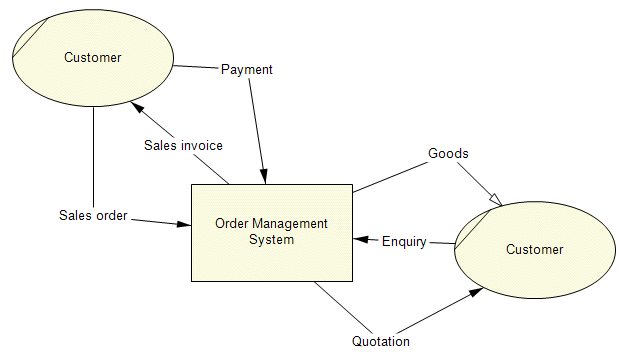
The state transition diagram illustrates the states of a user session on the platform and how transitions occur. This diagram shows the flow from registration, product browsing, checkout, to order confirmation.

**Activity Diagram:**



The activity diagram for product listing covers the steps a vendor follows to add a product to the platform. This includes filling in product details, uploading images, selecting categories, and setting the price.

**Data Flow Diagram (DFD):**



The DFD will depict the flow of information in the system, from customer requests to order processing. It includes major data stores such as user profiles, product catalog, orders, and payment details.

## **3.3 Architecture Design**

The system is designed using a **client-server architecture**, ensuring clear separation between the presentation layer (client-side) and the logic and data handling (server-side). Below is a detailed breakdown of the system components:

* **Client-Side (Frontend):**  
  The frontend is developed using **Laravel Blade templates**, which provide a clean and efficient server-side rendering engine integrated directly within the Laravel framework.
  + The interface is styled using **Bootstrap CSS**, ensuring responsive design.
  + **FontAwesome icons** are integrated for visually rich, scalable vector icons.
  + To improve user experience with animations and visual transitions, **AOS (Animate On Scroll)** library is employed.
* **Server-Side (Backend):**  
  The backend is built using the **Laravel framework (PHP)**, which serves as the core platform for handling logic, authentication, data validation, and API request routing.
  + Laravel’s built-in MVC structure provides clear organization of controllers, models, and views.
  + The application supports **RESTful API** endpoints for client-server communication and can be extended to support JSON-based API integrations.
  + Laravel’s queue system and scheduled tasks ensure **asynchronous job processing**, such as sending emails or interacting with AI services without blocking the main thread.
  + Security features like CSRF protection, rate limiting, and encrypted sessions are integrated natively.
* **AI Integration:**  
  The system includes intelligent chatbot functionality, powered by a combination of:
  + **BotPress**, an open-source conversational AI engine, which handles user intent recognition, dialogue flows, and context management.
  + **OpenAI APIs**, which enhance the chatbot’s ability to generate natural language responses, assist with complex queries, and provide contextually relevant suggestions.
  + The AI system enables **real-time user assistance**, **FAQ automation**, and **workflow support**, offering an intuitive and interactive experience for end users.
* **Hosting and Infrastructure:**
  + The **backend and database** are hosted on **Railway**, a modern cloud platform that provides continuous deployment, secure environment management, and built-in scalability for Laravel applications.
  + The **frontend** is hosted on **Vercel**, leveraging its serverless deployment model to ensure fast global content delivery, automatic version control, and robust performance optimization.
  + The infrastructure ensures **high availability**, **automatic scaling**, and **secure deployment pipelines**, with environment variables and secrets managed through Railway and Vercel dashboards.

## 3.4 Detailed Design

**Algorithm Description:**

The platform’s core algorithm involves the following steps:

1. **User Registration:** The user enters their details, which are stored in the database.
2. **Product Upload:** Vendors input product details and upload images.
3. **Order Processing:** The system confirms payment, updates the vendor's inventory.

**Flowchart:**

A flowchart for the checkout process can be created to visually represent the steps involved:

1. **Start** → Customer selects products.
2. **Proceed to Checkout** → User enters payment details.
3. **Payment Processing** → Payment gateway (Paystack) processes the transaction.
4. **Order Confirmation** → Confirmation sent to both vendor and customer.
5. **End.**

**Output Specifications:**

The expected outputs include:

* **Order Confirmation:** Sent to the customer and vendor.
* **Vendor Dashboard:** Displays sales reports, order status, and customer interactions.
* **AI Chatbot Responses:** Provides users with order updates, product inquiries, and basic troubleshooting.

## 3.5 Data Design

**Entity-Relationship Diagram (ERD):**

The ERD defines the relationships between key entities in the system, such as:

* **Users (Customers, Vendors, Admins)**: Stores user information such as name, contact, role, and payment history.
* **Products**: Includes product name, description, price, category, and vendor ID.
* **Orders**: Contains details like customer ID, product ID, order status, and payment status.
* **Payments**: Links to the order, tracking payment status and method used (Paystack).

**Database Design:**

The platform’s database consists of multiple tables:

* **Users:** id, name, email, role (vendor, admin, customer), password\_hash
* **Products:** id, vendor\_id, name, description, price, category, images
* **Orders:** id, user\_id, product\_id, quantity, status
* **Payments:** id, order\_id, payment\_method, status

## 3.6 Interface Design

**User Interface Screens:**

* **Home Page:** Showcases a product catalog with search and filter functionality.
* **Product Detail Page:** Displays product information, including images, price, and descriptions.
* **Vendor Dashboard:** Allows vendors to manage inventory, process orders, and view analytics.
* **Order Confirmation Page:** Provides confirmation details and order status.

**Decision Table for Checkout Process:**

|  |  |  |
| --- | --- | --- |
| **Condition** | **Action** | **Result** |
| Payment success | Confirm order and update inventory | Order confirmed |
| Payment failure | Notify user to retry | Order not processed |

## 3.7 Effectiveness and Efficiency of the Proposed System

The Fashion Hubis designed to be efficient and effective by:

* **Reducing Costs:** By providing an affordable e-commerce solution, it allows small businesses to access digital tools that were previously inaccessible.
* **Improving User Experience:** The user-friendly interface ensures that even individuals with minimal technical expertise can navigate the platform.
* **Enhancing Reach:** By leveraging AI-powered tools and WhatsApp, the platform enhances communication between vendors and customers, fostering better customer service and engagement.
* **Scalability:** The system is designed with scalability in mind, ensuring that it can handle a growing number of users and products over time.

## 3.8 Development Methodology

The **Agile development methodology** is adopted for this project. Agile is chosen because it promotes flexibility, iterative development, and continuous feedback, which are essential for the evolving nature of e-commerce and the fashion industry. This approach ensures that the platform can adapt quickly to changing market conditions, user needs, and emerging technologies. Regular sprints and feedback loops allow the team to make adjustments as needed, resulting in a final product that is closely aligned with the needs of users in Takoradi.

The Agile methodology is ideal for this project due to its focus on collaboration, adaptability, and rapid prototyping, all of which are critical in delivering a successful digital platform for local fashion entrepreneurs.