High-Level Solution Design Document

Project Bazaar for Client Thor

Version: 1.00

Updated: 28/02/2024

Author: Vaibhav Tyagi

Abstract

This project presents the development of Bazaar, an innovative e-commerce platform engineered to compete with industry frontrunners in the online retail sector. Bazaar is equipped with a comprehensive suite of features, including User Authentication, Product Discovery, Order Management, and Payment Handling, tailored to meet the diverse needs of modern consumers. Leveraging cutting-edge technology and user-centric design principles, Bazaar offers an intuitive browsing experience, empowering users to explore a vast array of products effortlessly. The platform prioritizes security and efficiency, ensuring seamless transactions and order fulfillment processes. By delivering a seamless and intuitive online shopping experience, Bazaar aspires to carve its niche as a leading player in the dynamic e-commerce landscape, poised to redefine the standards of digital retail.

Table of Contents

1. Introduction

Bazaar is an advanced e-commerce platform offering a seamless shopping experience across diverse product categories. With a user-friendly interface, it provides easy navigation, personalized recommendations, and secure payment options. Bazaar prioritizes security, partnering with trusted payment gateways for safe transactions. From fashion to electronics, users can explore a wide range of products from reputable brands. Bazaar aims to revolutionize online retail by delivering convenience, reliability, and quality service to modern consumers.

1.1. Purpose of this Document

This High-Level Design (HLD) document specifies the implementation, including inter-component dependencies, and provides sufficient design detail that any product based on this HLD will satisfy the product requirements.

1.2. Document Scope

Anyone interested in understanding the Bazaar Enhancements internal design should read this document.

2. Requirement Summary

The new requirements for the Bazaar platform encompass key aspects of network infrastructure, system functionalities, and user interactions. This breakdown provides a high-level overview to guide the development process effectively:

2.1 Network Infrastructure:

- *Scalability*: The platform should support an increasing number of users and products without compromising performance.
- *Security Measures*: Implement robust security protocols to safeguard user data, transactions, and ensure a secure browsing environment.

2.2 System Functionalities:

- *User Authentication Enhancement: Strengthen user authentication mechanisms to ensure secure access and protect user accounts.*
- *Order Processing Optimization: Streamline order processing for faster checkout, order confirmation, and tracking.*

2.3 User Interactions:

- *Intuitive User Interface: Enhance the user interface for intuitive navigation and an enjoyable shopping experience.*
- *Personalized Recommendations: Implement algorithms for personalized product recommendations based on user preferences and browsing history.*

2.4 Performance and Reliability:

- *Performance Optimization: Optimize system performance for efficient product browsing, order placement, and payment processing.*

- *System Reliability: Ensure high system availability and reliability to minimize downtime and enhance user trust.*

2.5 Accessibility:

- *Cross-Platform Compatibility: Ensure the platform is accessible and functions seamlessly across various devices and operating systems.*

- *Inclusive Design: Implement design features that enhance accessibility for users with different abilities.*

2.6 Collaboration with External Services:

- *Third-party Integration: Collaborate with external services, such as payment gateways, to provide diverse and secure payment options.*

- *API Development: Develop and maintain APIs for seamless integration with external systems.*

2.7 User Education and support:

- *User Guidance: Provide informative tooltips, guides, and support features to assist users in navigating and utilizing the platform effectively.*

- *Customer Support Integration: Implement efficient customer support mechanisms, including live chat and helpdesk features.*

3. Assumptions and Prerequisites

This document outlines key assumptions and foundational elements integral to the design process of Bazaar. The purpose is to ensure clarity and alignment among stakeholders regarding the fundamental principles guiding the platform's development.

i. **Internet Connectivity:** It is assumed that users will have reliable internet access to engage with Bazaar seamlessly across various devices.
ii. **Device Compatibility**: Bazaar is designed to be compatible with smartphones, tablets, and computers, ensuring accessibility for a wide range of users.
iii. **Product Availability:** The assumption is made that products showcased on Bazaar are readily available for purchase, maintaining user trust and satisfaction.
iv. **Security Measures:** Robust security measures will be implemented to safeguard user data and ensure secure transactions, maintaining the integrity of Bazaar's platform.
v. **Payment Gateway Integration:** Bazaar's functionality relies on seamless integration with trusted payment gateways for efficient transaction processing and user convenience.
vi. **Server Infrastructure:** Availability of stable server infrastructure is a prerequisite to support platform operations and ensure consistent performance.
vii. **Database Management:** Bazaar depends on a reliable database management system for efficient data storage, retrieval, and management, ensuring seamless user experiences.
viii. **Third-party Integrations:** Integration with third-party services such as shipping providers and analytics platforms enhances Bazaar's functionality and user experience.

These assumptions and foundational elements will be regularly reviewed and assessed throughout the design process to ensure their validity and alignment with Bazaar's objectives.

4. High-Level Design

The High-Level Design (HLD) for Bazaar delineates the overarching process flows and information pathways, incorporating visualizations to elucidate the seamless functionality of the platform. Addressing the integration nuances with a focus on five fundamental features, the HLD ensures a comprehensive understanding of Bazaar's design structure.

4.1. User Authentication

- **Components:**
  - Frontend (Mobile/Web Interface)
  - Backend Server
  - Database (User Information)
- **Flow:**
  - Users initiate the process by submitting login credentials through the frontend interface.
  - Backend server validates credentials by cross-referencing with the user information database.
  - Upon successful verification, the backend generates a token and transmits it to the frontend for efficient session management.

4.2. Product Browsing

- **Components:**
  - Frontend (Mobile/Web Interface)
  - Backend Server
  - Database (Product Catalog)
- **Flow:**
  - The frontend triggers a request for product data from the backend.
  - Backend server retrieves comprehensive product information from the database.
  - Product data is transmitted back to the frontend, facilitating an immersive display for users.

4.3. Shopping Cart Management

- **Components:**

    - Frontend (Mobile/Web Interface)
    - Backend Server
    - Database (User Cart)
- **Flow:**
    - Users seamlessly add products to their carts through the frontend.
    - Frontend communicates cart updates to the backend server.
    - Backend ensures real-time updates to the user's cart within the database.

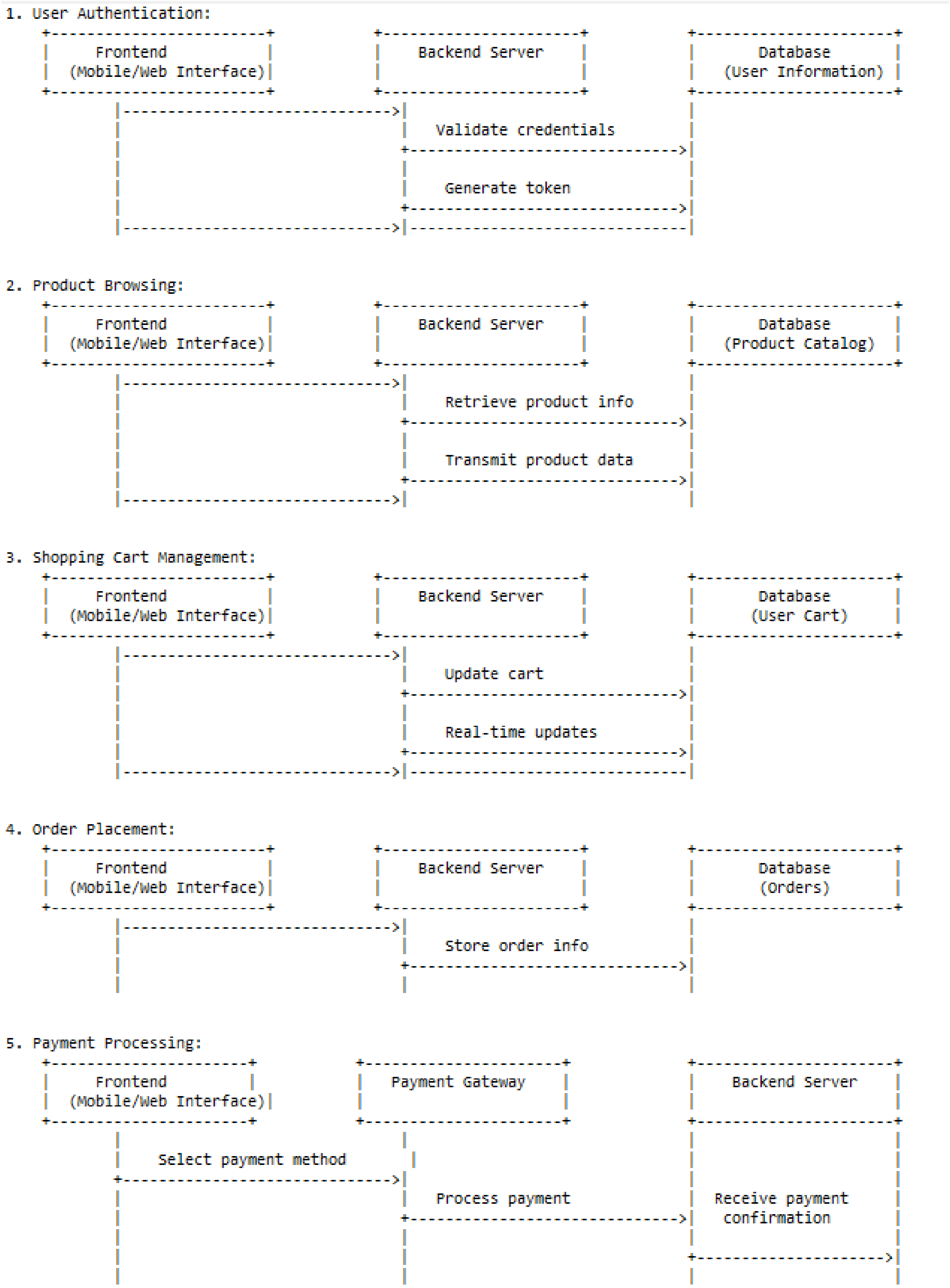
4.4. Order Placement

- **Components:**
    - Frontend (Mobile/Web Interface)
    - Backend Server
    - Database (Orders)
- **Flow:**
    - Users confirm their orders through the frontend interface.
    - Frontend transmits order details to the backend server.
    - Backend stores comprehensive order information securely in the database.

4.5. Payment Processing

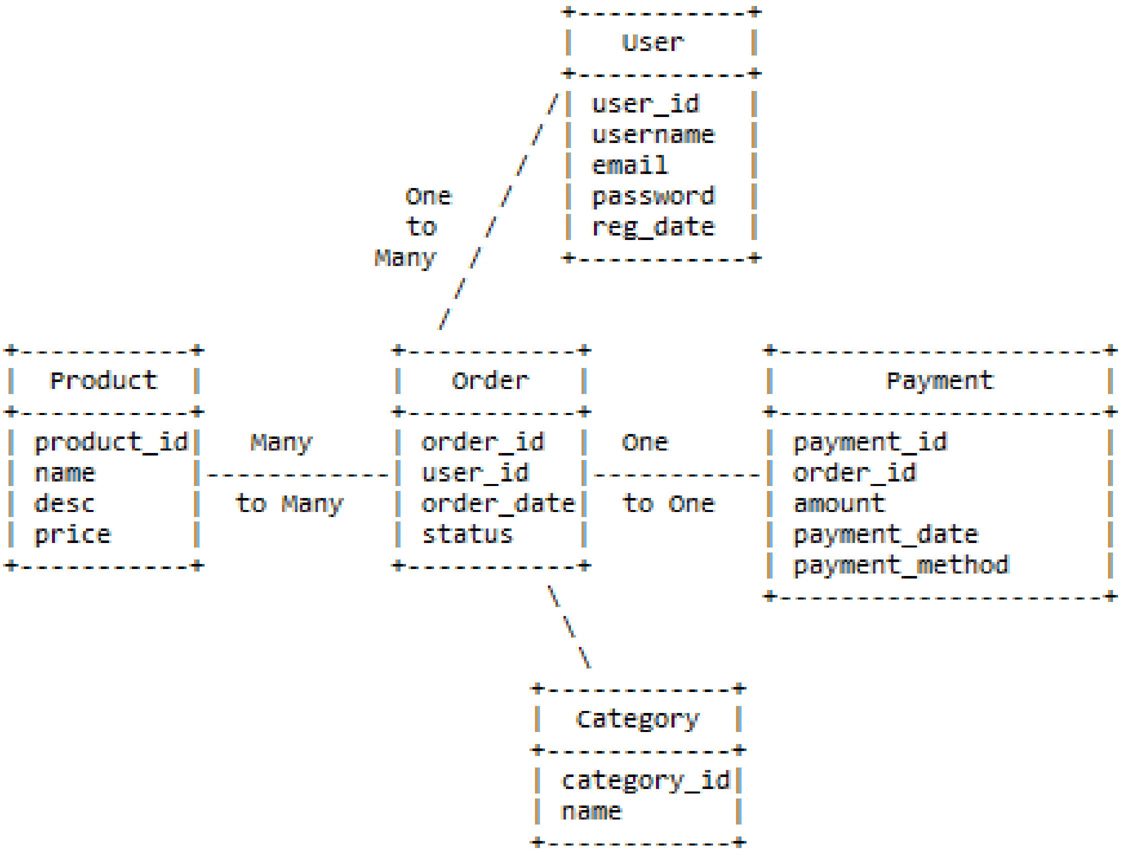
- **Components:**
    - Frontend (Mobile/Web Interface)
    - Payment Gateway
    - Backend Server
- **Flow:**
    - Users select their preferred payment method and initiate the payment process through the frontend.
    - Frontend communicates the payment request to the backend server.
    - Backend forwards the request seamlessly to the designated payment gateway.
    - The payment gateway processes the transaction securely.
    - Confirmation of the successful payment is relayed back to both the backend and frontend.

This High-Level Design provides a holistic view of Bazaar's foundational processes, aiding in the early assessment of assumptions, validations, and potential areas for further review. The integration aspects with other systems are seamlessly incorporated, ensuring a robust and efficient design structure for the Bazaar platform.



```
1. User Authentication:
    +------------------------+          +-----------------------+          +----------------------+
    |     Frontend           |          |    Backend Server     |          |       Database       |
    |  (Mobile/Web Interface)|          |                       |          |   (User Information) |
    +------------------------+          +-----------------------+          +----------------------+
            |------------------------------>|                              |
            |                               |  Validate credentials        |
            |                               +----------------------------->|
            |                               |                              |
            |                               |   Generate token             |
            |                               +----------------------------->|
            |------------------------------>|------------------------------|

2. Product Browsing:
    +------------------------+          +-----------------------+          +----------------------+
    |     Frontend           |          |    Backend Server     |          |       Database       |
    |  (Mobile/Web Interface)|          |                       |          |   (Product Catalog)  |
    +------------------------+          +-----------------------+          +----------------------+
            |------------------------------>|                              |
            |                               |   Retrieve product info      |
            |                               +----------------------------->|
            |                               |                              |
            |                               |   Transmit product data      |
            |                               +----------------------------->|
            |------------------------------>|                              |

3. Shopping Cart Management:
    +------------------------+          +-----------------------+          +----------------------+
    |     Frontend           |          |    Backend Server     |          |       Database       |
    |  (Mobile/Web Interface)|          |                       |          |      (User Cart)     |
    +------------------------+          +-----------------------+          +----------------------+
            |------------------------------>|                              |
            |                               |   Update cart                |
            |                               +----------------------------->|
            |                               |                              |
            |                               |   Real-time updates          |
            |                               +----------------------------->|
            |------------------------------>|------------------------------|

4. Order Placement:
    +------------------------+          +-----------------------+          +----------------------+
    |     Frontend           |          |    Backend Server     |          |       Database       |
    |  (Mobile/Web Interface)|          |                       |          |       (Orders)       |
    +------------------------+          +-----------------------+          +----------------------+
            |------------------------------>|                              |
            |                               |   Store order info           |
            |                               +----------------------------->|
            |                               |                              |

5. Payment Processing:
    +----------------------+           +----------------------+            +----------------------+
    |     Frontend         |           |  Payment Gateway     |            |    Backend Server    |
    |  (Mobile/Web Interface)|         |                      |            |                      |
    +----------------------+           +----------------------+            +----------------------+
            |                               |                              |                      |
            |    Select payment method       |                             |                      |
            +------------------------------>|                              |                      |
            |                               |  Process payment             |  Receive payment     |
            |                               +----------------------------->|   confirmation       |
            |                               |                              |                      |
            |                               |                              +--------------------->|
            |                               |                              |                      |
```

Figure 1 Flow between frontend, backend, and database for each module



```
                                     +-----------+
                                     |   User    |
                                     +-----------+
                                    /| user_id   |
                                   / | username  |
                                  /  | email     |
                          One    /   | password  |
                          to    /    | reg_date  |
                        Many   /     +-----------+
                              /
                             /
+-----------+            +-----------+            +----------------------+
|  Product  |            |   Order   |            |       Payment        |
+-----------+            +-----------+            +----------------------+
| product_id|   Many     | order_id  |  One       | payment_id           |
| name      |------------| user_id   |------------| order_id             |
| desc      |  to Many   | order_date|  to One    | amount               |
| price     |            | status    |            | payment_date         |
+-----------+            +-----------+            | payment_method       |
                                                  +----------------------+
                                    \
                                     \
                                      \
                                  +------------+
                                  |  Category  |
                                  +------------+
                                  | category_id|
                                  | name       |
                                  +------------+
```

FIGURE 2 ENTITY DIAGRAM FOR EACH ENTITY

ER diagram for the Bazaar application involves identifying and defining all relevant entities, attributes, and relationships based on the provided information. Let's start by identifying the entities and their attributes:

Entities:

1. User
2. Product
3. Order
4. Payment

Attributes:

1. User:
   - user_id (Primary Key)
   - username
   - email
   - password (hashed)

- registration_date

2. Product:
   - product_id (Primary Key)
   - name
   - description
   - price
   - category_id (Foreign Key)
3. Order:
   - order_id (Primary Key)
   - user_id (Foreign Key)
   - order_date
   - status
4. Payment:
   - payment_id (Primary Key)
   - order_id (Foreign Key)
   - amount
   - payment_date
   - payment_method

Based on these entities and attributes, we can now define the relationships between them:

Relationships:

1. User - Order: One-to-Many (One user can have multiple orders)
2. Product - Order: Many-to-Many (Many products can be part of many orders)
3. User - Payment: One-to-Many (One user can have multiple payments)
4. Order - Payment: One-to-One (Each order has one payment)

5. Plan

The plan for the Bazaar IT solution encompasses a detailed outline of all major components, their integration within the broader IT ecosystem (referred to as the context), and the connections facilitating seamless integration between these components.

5.1 User Authentication Component

- This component includes the frontend interface for user login and the backend server responsible for authentication.
- Within the context of the IT ecosystem, the user authentication component interfaces with the database containing user information.
- Integration is facilitated through secure API connections between the frontend, backend server, and database.

5.2 Product Browsing Component

- Comprising frontend and backend components, the product browsing component facilitates the retrieval and display of product information.
- It fits within the IT ecosystem by interacting with the product catalog database to retrieve product data.
- Integration involves API connections between the frontend, backend server, and product catalog database.

5.3 Shopping Cart Management Component

- The shopping cart management component consists of frontend and backend components responsible for managing user carts.
- Within the IT ecosystem, this component interacts with the user cart database to update cart contents.
- Integration is achieved through API connections between the frontend, backend server, and user cart database.

5.4 Order Placement Component

- This component encompasses frontend and backend functionalities for order confirmation and storage.
- It fits within the IT ecosystem by communicating with the orders database to store order information.
- Integration involves API connections between the frontend, backend server, and orders database.

5.5 Payment Processing Component

- Comprising frontend, backend, and payment gateway components, the payment processing component facilitates secure payment transactions.

- It fits within the IT ecosystem by connecting with external payment gateways for transaction processing.
- Integration involves API connections between the frontend, backend server, and payment gateway.

By delineating the plan for each major component of the Bazaar IT solution, detailing their integration within the broader IT ecosystem, and outlining the connections facilitating their integration, this plan ensures a comprehensive and cohesive approach to the development and implementation of the Bazaar platform.

5.1.Application Modules

The application modules for Bazaar are built using cutting-edge technologies and encompass various components essential for its functionality, including core components, user interfaces and layers, the database layer, business logic modules, licensed components, and APIs with external connections.

5.1.1 Core Components

- Bazaar's core components are developed using Java, leveraging the robustness and scalability of the language. These components are structured as microservices, allowing for independent development, deployment, and scalability. Each microservice is implemented using Spring Boot framework, facilitating rapid development and deployment of standalone services.

5.1.2 User Interfaces and Layers

- Bazaar offers intuitive and responsive user interfaces developed using HTML, CSS, and JavaScript for web interfaces, and JavaFX for desktop interfaces. These interfaces interact with the backend microservices through RESTful APIs, ensuring seamless communication between the frontend and backend layers.

5.1.3 Database Layer

- The database layer of Bazaar utilizes a relational database management system (RDBMS) such as MySQL or PostgreSQL for structured data storage. Additionally, a NoSQL database like MongoDB may be employed for handling unstructured or semi-structured data. These databases are accessed through Spring Data JPA or Hibernate for efficient data manipulation.

5.1.4 Business Logic Modules

- Bazaar's business logic modules are implemented within the backend microservices using Java. These modules handle tasks such as user authentication, product management, cart management, order processing, and payment validation. Business logic is encapsulated within service classes and interacts with the database layer using JPA repositories.

5.1.5 Licensed Components

- Bazaar may incorporate licensed components or third-party libraries to enhance its functionality. These components may include encryption libraries such as Bouncy Castle for secure data transmission, payment gateway APIs like PayPal or Stripe for handling payment transactions, and logging frameworks such as Log4j for effective logging and monitoring.

5.1.6 APIs and External Connections

- Bazaar integrates with external systems and services through APIs to extend its capabilities. Payment processing is facilitated through integration with payment gateway APIs such as PayPal API or Stripe API. Additionally, integration with shipping provider APIs like FedEx or UPS enables seamless order fulfilment. Analytics APIs such as Google Analytics may also be integrated for tracking user behaviour and performance metrics.

By incorporating these application modules and leveraging modern technologies such as Java, microservices architecture, Spring Boot, and relevant APIs, Bazaar ensures a robust, scalable, and feature-rich platform that delivers a seamless shopping experience to its users while maintaining efficiency and reliability in its operations.

5.2. Transactions and User Flows

The Transactions and User Flows section of Bazaar's design documentation details all necessary interactions between users and the system. These interactions encompass the various business processes implemented within the system to facilitate a seamless shopping experience. Below are the key transactions and user flows documented for Bazaar:

5.2.1 User Authentication Flow:

- User enters login credentials.
- System verifies credentials against the user database.
- Upon successful authentication, the user gains access to their account dashboard.

5.2.2 Product Browsing Flow:

- User navigates through product categories.
- User selects a specific product for detailed information.
- User adds the product to the shopping cart for potential purchase.

5.2.3 Shopping Cart Management Flow:

- User views and modifies items in the shopping cart.
- User proceeds to checkout after finalizing their selection.

5.2.4 Order Placement Flow:

- User confirms their order details.
- System processes the order and updates the order status.
- Confirmation email is sent to the user.

5.2.5 Payment Processing Flow:

- User selects a payment method during checkout.
- System communicates with the payment gateway API for payment processing.
- Payment confirmation is received, and the order status is updated accordingly.

These transactions and user flows represent the core interactions between users and the Bazaar platform, illustrating the seamless business processes implemented to ensure a user-friendly and efficient shopping experience.