|  |
| --- |
| **A**  **PROJECT REPORT ON** |
|  |
|  |
| NOLX\_Application |
|  |
|  |
| SUBMITTED IN  PARTIAL FULFILLMENT OF  **DIPLOMA IN Mobile COMPUTING (PG-DMC)** |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| **BY**  **86967- Aditya Sanjay Shukla**  **87057- Jayesh Nishikant Inamdar**  **87160- Chetan Sheshrao Sawarkar**  **87161- Harshal Rajendra Hatwar**  **87163- Ankit Kavadu Patil** |
|  |
|  |
| **UNDER THE GUIDENCE OF**  **Shreyas Jadhav** |
|  |
|  |
|  |
| **AT**  **SUNBEAM INSTITUTE OF INFORMATION TECHNOLOGY, PUNE** |

|  |  |
| --- | --- |
|  | |
| **SUNBEAM INSTITUTE OF INFORMATION TECHNOLOGY,**  **PUNE.** | |
|  | |
|  | |
|  | |
|  | |
| **CERTIFICATE** | |
|  | |
| This is to certify that the project | |
|  | |
| NOLX\_Application | |
|  | |
| Has been submitted by | |
|  | |
| **86967- Aditya Sanjay Shukla**  **87057- Jayesh Nishikant Inamdar**  **87160- Chetan Sheshrao Sawarkar**  **87161- Harshal Rajendra Hatwar**  **87163- Ankit Kavadu Patil** | |
|  | |
|  | |
| In partial fulfillment of the requirement for the Course of **PG Diploma in Mobile Computing (PG-DMC AUG2015)** as prescribed by The **CDAC** ACTS, PUNE. | |
|  | |
|  | |
| Place: Pune | Date: 30-JAN-2016 |
| |  |  | | --- | --- | | **Shreyas Jadhav** | **<Alumni Mentor Name>** | | **Project Guide** | **Alumni Mentor** |   **ACKNOWLEDGEMENT** | |

I would like to express my heartfelt gratitude to everyone who contributed to the successful completion of this N**OLX Project**. This project has been a significant learning experience, and it would not have been possible without the guidance, support, and encouragement of many individuals.

Firstly, I extend my sincere thanks to my **[Shreyas Jadhav]**, whose valuable insights and constant support played a crucial role in shaping this project. Their expertise and constructive feedback have helped me improve and refine the various aspects of the project.

I am also deeply grateful to my team members for their collaboration, discussions, and shared efforts in making this project a success. Their dedication and teamwork were instrumental in overcoming challenges and ensuring smooth development.

Additionally, I extend my appreciation to **[Sunbeam Infotech]** for providing the necessary resources, tools, and infrastructure that facilitated the development of this project.

Lastly, I would like to thank my friends for their unwavering support, encouragement, and patience throughout this journey. Their belief in my abilities has been a great motivation.

This project has been an enriching experience, allowing me to apply my technical knowledge in a real-world scenario. I hope this work will contribute positively to the field and serve as a foundation for future improvements

**ABSTRACT**

NOLX is an online marketplace designed to facilitate the buying and selling of second-hand products in a simple, efficient, and user-friendly manner. Taking inspiration from OLX, the platform allows users to list items for sale, browse available products, and communicate with potential buyers or sellers seamlessly. The application is built using React.js for the frontend, ensuring a dynamic and responsive user interface, while the backend is designed to manage authentication, product listings, and real-time interactions.

Users can create accounts, log in securely, and manage their listings effortlessly. The listing process enables sellers to upload product details, including images, descriptions, and pricing, making it easier for buyers to explore available options. A well-structured search and filtering mechanism ensures that users can quickly find the products they are looking for based on categories, price range, or location. To enhance user engagement, a real-time chat system is integrated, allowing direct communication between buyers and sellers without requiring third-party messaging apps.

The application also prioritizes security and reliability by implementing user verification, spam prevention measures, and a well-organized moderation system to prevent fraudulent listings. The design focuses on scalability, ensuring that the platform can accommodate a growing number of users and listings over time. By offering a smooth and intuitive experience, NOLX aims to become a trusted platform for second-hand transactions, providing convenience, efficiency, and transparency to buyers and sellers alike.

**INDEX**

|  |  |  |
| --- | --- | --- |
|  | **INTRODUCTION** | 1 |
|  | 1.1 Introduction | 2 |
|  | **Product Overview and Summary** |  |
|  | 2.1 Purpose |  |
|  | 2.2 Scope |  |
|  | 2.3 User Classes and Characteristics |  |
|  | 2.4 Design and Implementation Constraints |  |
|  | **REQUIREMENTS** |  |
|  | 3.1 Functional Requirements |  |
|  | 3.1.1 Use case for Administrator. |  |
|  | 3.1.2 Use case for Customer. |  |
|  | 3.2 Non - Functional Requirements |  |
|  | 3.2.1 Usability Requirement |  |
|  | 3.2.2 Performance Requirement |  |
|  | 3.2.3 Reliability Requirement |  |
|  | 3.2.4 Portability Requirement |  |
|  | 3.2.5 Security Techniques |  |
|  | **PROJECT DESIGN** |  |
|  | 4.1 Data Model |  |
|  | 4.1.1 Database Design |  |
|  | 4.2 Process Model |  |
|  | 4.2.1 Functional Decomposition Diagram |  |
|  | 4.2.2 Data Flow Diagram (DFD) |  |
|  | **TEST REPORT** |  |
|  | **PROJECT RELATED STATISTICS** |  |
|  | **CONCLUSION** |  |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Section** | **Table Title** | **Page** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Section** | **Figure Title** | **Page** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**1. INTRODUCTION**

The NOLX project is an online classified marketplace that enables users to buy and sell goods and services with ease. The platform serves as a bridge between buyers and sellers, allowing them to connect and conduct transactions without intermediaries. By leveraging digital technology, the project aims to create a seamless, efficient, and user-friendly experience for individuals looking to sell unused items or purchase pre-owned products at competitive prices.

The idea behind this project is to replicate the functionalities of NOLX, a widely used online marketplace, while incorporating new enhancements to improve the user experience. The system will allow users to list their items, browse products, search for specific goods, and complete transactions in a secure environment.

**Product Overview and Summary**

**2.1 Purpose**

The primary purpose of this NOLX-like platform is to provide an online marketplace where users can buy and sell new or used goods conveniently. It enables users to post advertisements, browse product listings, and communicate securely with potential buyers or sellers. The system aims to create a hassle-free experience for users while ensuring secure transactions, efficient communication, and easy navigation.

The platform is intended for individuals, businesses, and resellers looking to trade products without intermediaries. It minimizes operational costs and promotes a user-driven economy where people can make transactions directly.

**2.2 Scope**

The NOLX project will offer a comprehensive set of features that facilitate the smooth operation of an online classified marketplace. The key aspects of the system's scope include:

**Functional Scope**

1. **User Registration & Authentication:** Secure login and registration via email, or social media.
2. **Product Listing & Management:** Users can post, edit, and delete advertisements with images, descriptions, and pricing.
3. **Search & Filtering:** Advanced search functionality with category and price-based filters.
4. **Wishlist & Favorites:** Users can save items for future reference.
5. **Admin Dashboard:** User and listing moderation, fraud detection, and analytics.

**Technical Scope**

* **Frontend:** Developed using React.js for dynamic user interaction.
* **Backend:** Built with Node.js and Express.js for efficient request handling.
* **Database:** **MySQL** for structured data storage.

**2.3 User Classes and Characteristics**

This system is designed to cater to different types of users based on their needs and level of access.

**1. General Users (Buyers & Sellers)**

* **Characteristics:**
  + Can register and create a profile.
  + Can search, filter, and browse available products.
  + Can mark items as favorites or save them for later.

**2. Admins**

* **Characteristics:**
  + Have access to a dashboard for monitoring user activities.
  + Can remove fraudulent or inappropriate listings.
  + Can verify user accounts and take necessary actions against violations.
  + Can generate reports on platform activity.

**2.4 Design and Implementation Constraints**

The development of the NOLX project will be influenced by several constraints, which must be considered during implementation.

**1. Performance Constraints**

* The system must handle multiple concurrent users efficiently.
* Search and filtering operations should be optimized for fast response times.
* Image uploads should be compressed to reduce server load.

**2. Security Constraints**

* Secure authentication must be implemented using JWT
* User data and transactions must be encrypted to prevent unauthorized access.
* Admin privileges should be restricted to avoid unauthorized modifications.

**4. Technology Constraints**

* The application should use MySQL for data storage, and the backend must be built with Node.js and Express.js.
* Hosting services should support scalability and high availability ().
* Third-party APIs for my MYSQl server hosting (Clever cloud) should be compatible with the technology stack.

**3. REQUIREMENTS**

**3. Requirements**

**3.1 Functional Requirements**

Functional requirements define the core functionalities of the NOLX-like platform, specifying what the system should do to meet user needs. The primary users of this system include Administrators and Customers (Buyers & Sellers).

**3.1.1 Use Case for Administrator**

The Administrator is responsible for managing the platform, ensuring security, moderating content, and maintaining system performance.

**Administrator Functional Requirements**

1. **User Management**
   * View, verify, and manage user accounts.
   * Reset passwords for users upon request.
2. **Category & Listing Management**
   * Define product categories .
   * Edit, update, or delete categories when required.
3. **Reports & Analytics**
   * Generate reports on platform activity (number of listings, sales trends, most popular categories, etc.).
   * Monitor fraudulent activities or unusual transaction patterns.

**3.1.2 Use Case for Customer (Buyer & Seller)**

Customers include both buyers and sellers, who interact with the platform to post , search for products, and communicate with other users.

**Seller Functional Requirements**

1. **Account Registration & Authentication**
   * Sign up and log in using email, phone number.
2. **Post Advertisements**
   * Add product details (title, description, price, location).
   * Upload product images.
   * Select a category for the product.
3. **Manage Listings**
   * Edit, update, or delete product .
   * View responses or inquiries from potential buyers.
4. **Communication with Buyers**
   * Receive messages from interested buyers.

**Buyer Functional Requirements**

1. **Browse & Search Products**
   * Use search functionality with filters (category, price).
2. **View Product Details**
   * See seller information, product images, and descriptions.
3. **Contact Seller**
   * Send messages or inquiries to sellers through the chat system.
4. **Wishlist & Favorites**
   * Save products for future reference.
5. **Purchase & Payment (Optional)**
   * Complete purchases via online payment (if enabled).

**3.2 Non-Functional Requirements**

Non-functional requirements define the quality attributes, system constraints, and operational standards that the NOLX-like platform must meet. These requirements ensure that the system is usable, performant, reliable, portable, and secure.

**3.2.1 Usability Requirement**

Usability refers to how easily users can interact with the system. The platform should be intuitive, user-friendly, and accessible.

**Usability Requirements**

1. **Simple Navigation** – The interface should be designed with clear menus and intuitive navigation.
2. **Mobile-Friendly Design** – The platform should be responsive and work on various screen sizes.
3. **Consistent UI** – The design should follow Material Design or Bootstrap standards for uniformity.

**3.2.2 Performance Requirement**

Performance requirements ensure that the system runs smoothly under different conditions and loads.

**Performance Requirements**

1. **Response Time**
   * The system should load pages within 3 seconds for standard users.
   * The search feature should return results within 2 seconds.
2. **Data Processing**
   * The database should support thousands of product listings with efficient indexing.
3. **Scalability**
   * The system should be horizontally scalable using load balancers and microservices.
4. **Efficient Image Handling**
   * Images should be compressed and optimized to reduce page load times.

**3.2.3 Reliability Requirement**

Reliability refers to the system’s ability to function without failure over time.

**Reliability Requirements**

1. **Uptime Guarantee**
   * The system should be available 99.9% of the time (maximum downtime of 8.76 hours per year).
2. **Automatic Recovery**
   * In case of server failure, the system should recover.
3. **Error Handling & Logging**
   * All errors should be logged in real-time and stored in a monitoring system.
4. **Database Backup**
   * Data should be automatically backed up daily to prevent data loss.

**3.2.4 Portability Requirement**

Portability ensures that the system can run on different platforms and environments.

**Portability Requirements**

1. **Cross-Browser Compatibility**
   * The application should function correctly on Google Chrome, Firefox, Safari, Edge, and Opera.
2. **Device Compatibility**
   * The platform should be fully accessible on desktop, tablet, and mobile devices.
3. **Deployment**
   * The system should be clever cloudmajor modifications.
4. **Cross-Platform Mobile App** 
   * If a mobile app is developed, it should run on both Android using React Native.

**3.2.5 Security Techniques**

Security requirements protect user data, prevent unauthorized access, and ensure safe transactions.

**Security Techniques**

1. **User Authentication & Authorization**
   * Implement JWT (JSON Web Token) authentication for secure login.

**4. PROJECT DESIGN**

**4.1 Data Model**

Entities and Attributes

**1. address Table**

**Purpose:** Stores the user's address details.

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| address\_id | INT (Primary Key, Auto Increment) | Unique ID for the address. |
| user\_id | INT (Foreign Key) | References the users table. |
| street\_address | VARCHAR(255) | Street name and number. |
| city | VARCHAR(100) | City of residence. |
| pin | VARCHAR(10) | Postal code or PIN code. |
| country | VARCHAR(100) | Country of residence. |
| landmarks | VARCHAR(255) | Nearby landmarks for better identification. |
| extra\_directions | VARCHAR(255) | Additional instructions for finding the address. |

**2. category Table**

**Purpose:** Stores product categories.

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| category\_id | INT (Primary Key, Auto Increment) | Unique category ID. |
| category\_name | VARCHAR(100) | Name of the product category. |

**3. users Table**

**Purpose:** Stores user information.

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| user\_id | INT (Primary Key, Auto Increment) | Unique ID for each user. |
| name | VARCHAR(100) | Full name of the user. |
| password | VARCHAR(255) | Encrypted password. |
| email | VARCHAR(100) (Unique) | User's email address. |
| phone | VARCHAR(15) | Contact number (optional). |
| role | ENUM('Admin','User') | Defines the user's role (Admin or regular User). |
| location | VARCHAR(255) | User's general location. |
| history | TIMESTAMP (Default: Current Time) | Timestamp of user creation or last update. |

**4. wishlist Table**

**Purpose:** Stores users’ wishlists.

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| wishlist\_id | INT (Primary Key, Auto Increment) | Unique wishlist ID. |
| user\_id | INT (Foreign Key) | References users table. |
| product\_id | INT (Foreign Key) | References product table. |

**5. product Table**

**Purpose:** Stores products listed for sale.

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| product\_id | INT (Primary Key, Auto Increment) | Unique ID for each product. |
| product\_name | VARCHAR(255) | Name of the product. |
| seller\_id | INT (Foreign Key) | References the users table (seller’s ID). |
| category\_id | INT (Foreign Key) | References the category table. |
| description | VARCHAR(255) | Brief description of the product. |
| stock | INT | Available stock for the product. |
| price | DECIMAL(10,2) | Price of the product. |
| image\_url | TEXT | URL to the product image. |

**6. orders Table**

**Purpose:** Stores order details.

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| order\_id | INT (Primary Key, Auto Increment) | Unique order ID. |
| user\_id | INT (Foreign Key) | References the users table (buyer’s ID). |
| transaction\_id | INT (Foreign Key) | References the payment table. |
| order\_date | DATE (Default: Current Date) | Date when the order was placed. |
| shipping\_address | VARCHAR(255) | Delivery address. |
| total\_price | DECIMAL(10,2) | Total cost of the order. |
| order\_status | ENUM('Pending', 'Confirmed', 'Shipped', 'Delivered', 'Cancelled') | Status of the order. |

**7. order\_items Table**

**Purpose:** Stores the items in each order.

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| order\_item\_id | INT (Primary Key, Auto Increment) | Unique order item ID. |
| order\_id | INT (Foreign Key) | References orders table. |
| product\_id | INT (Foreign Key) | References product table. |
| quantity | INT (Default: 1) | Quantity of the product ordered. |
| price | DECIMAL(10,2) | Price per unit of the product. |

**8. payment Table**

**Purpose:** Stores payment details.

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| transaction\_id | INT (Primary Key, Auto Increment) | Unique transaction ID. |
| payment\_method | ENUM('UPI','Card','Cash') | Payment mode used. |
| payment\_status | ENUM('Pending','Success','Failed') | Status of the payment. |
| amount | DECIMAL(10,2) | Transaction amount. |
| upi\_id | VARCHAR(50) | UPI ID (if applicable). |
| card\_id | VARCHAR(50) | Card details (if applicable). |

**Key Relationships**

* **Users & Address**: One user can have multiple addresses (One-to-Many).
* **Users & Products**: A user can be a seller and list multiple products (One-to-Many).
* **Users & Wishlist**: Users can add multiple products to their wishlist (One-to-Many).
* **Products & Category**: Each product belongs to a specific category (Many-to-One).
* **Orders & Users**: A user can place multiple orders (One-to-Many).
* **Orders & Payment**: Each order has a corresponding payment transaction (One-to-One).
* **Order Items & Orders**: Each order contains multiple order items (One-to-Many).

**4.1.1 Database Design**

The following table structures depict the database design.

**Table 1 : Address**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Extra** |
| address\_id | int unsigned | NO | PRI | NULL | auto\_increment |
| user\_id | int unsigned | NO | MUL | NULL |  |
| street\_address | varchar(255) | NO |  | NULL |  |
| city | varchar(100) | NO |  | NULL |  |
| pin | varchar(10) | YES |  | NULL |  |
| country | varchar(100) | NO |  | NULL |  |
| landmarks | varchar(255) | YES |  | NULL |  |
| extra\_directions | varchar(255) | YES |  | NULL |  |

**Table 2 : Category**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Extra** |
| category\_id | int unsigned | NO | PRI | NULL | auto\_increment |
| category\_name | varchar(100) | NO |  | NULL |  |

**Table 3 : Order\_items**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Extra** |
| order\_item\_id | int | NO | PRI | NULL | auto\_increment |
| order\_id | int | NO | MUL | NULL |  |
| street\_address | int unsigned | NO | MUL | NULL |  |
| quantity | int | NO |  | 1 |  |
| price | decimal(10,2) | NO |  | NULL |  |

**Table 4 : Orders**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Extra** |
| order\_id | int | NO | PRI | NULL | auto\_increment |
| user\_id | int unsigned | NO | MUL | NULL |  |
| transaction\_id | int unsigned | YES | MUL | NULL |  |
| order\_date | date | NO |  | NULL |  |
| shipping\_address | varchar(255) | NO |  | NULL |  |
| total\_price | decimal(10,2) | NO |  | NULL |  |
| order\_status | enum('Pending',  'Confirmed',  'Shipped','Delivered',  'Cancelled') | YES |  | Pending |  |

**Table 5 : payment**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Extra** |
| transaction\_id | int unsigned | NO | PRI | NULL | auto\_increment |
| payment\_method | enum('UPI','Card','Cash') | NO |  | Cash |  |
| payment\_status | enum('Pending','Success',  'Failed') | YES |  | Pending |  |
| amount | decimal(10,2) | NO |  |  |  |
| upi\_id | varchar(50) | YES |  | NULL |  |
| card\_id | varchar(50) | YES |  | Pending |  |

**Table 6 : Product**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Extra** |
| product\_id | int unsigned | NO | PRI | NULL | auto\_increment |
| product\_name | varchar(255) | NO |  | NULL |  |
| seller\_id | int unsigned | NO |  | NULL |  |
| category\_id | int unsigned | NO | MUL | NULL |  |
| description | varchar(255) | YES |  | NULL |  |
| stock | int unsigned | NO |  | NULL |  |
| price | decimal(10,2) | YES |  | NULL |  |
| image\_url | text | YES |  | NULL |  |

**Table 7 : Users**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Extra** |
| user\_id | int unsigned | NO | PRI | NULL | auto\_increment |
| name | varchar(100) | NO |  | NULL |  |
| password | varchar(255) | NO |  | NULL |  |
| email | varchar(100) | NO | UNI | NULL |  |
| phone | varchar(10) | YES |  | NULL |  |
| role | enum('Admin','User') | YES |  | User |  |
| location | varchar(255) | YES |  | NULL |  |
| history | timestamp | YES |  | CURRENT\_TIMESTAMP | DEFAULT\_GENERATED |

**Table 8: Wishlist**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Extra** |
| wishlist\_id | int unsigned | NO | PRI | NULL | auto\_increment |
| user\_id | int unsigned | NO | MUL | NULL |  |
| product\_id | int unsigned | NO | MUL | NULL |  |

**4.2 Process Model**

The Process Model represents the process flow and functional decomposition of the NOLX project. It outlines the interactions between users, the system, and various components. The key components of the process model are:

**4.2.1 Functional Decomposition Diagram (FDD)**

The Functional Decomposition Diagram (FDD) breaks down the functionalities of the system into smaller, more manageable parts. It shows the major functions and sub-functions of the NOLX platform, which can be broken into levels of detail.

**Level 0: High-Level Overview of the System**

* **NOLX System**
  + User Management
  + Product Management
  + Order Management
  + Payment Management
  + Wishlist Management

**Level 1: Detailed Functional Decomposition**

* **User Management**
  + User Registration
  + User Login
  + Profile Management
* **Product Management**
  + Add Product
  + Update Product
  + Search Product
  + View Product Details
  + Delete Product
* **Order Management**
  + Create Order
  + View Order Details
* **Payment Management**
  + Initiate Payment
  + Confirm Payment
  + View Payment Status
* **Wishlist Management**
  + Add to Wishlist
  + View Wishlist
  + Remove from Wishlist

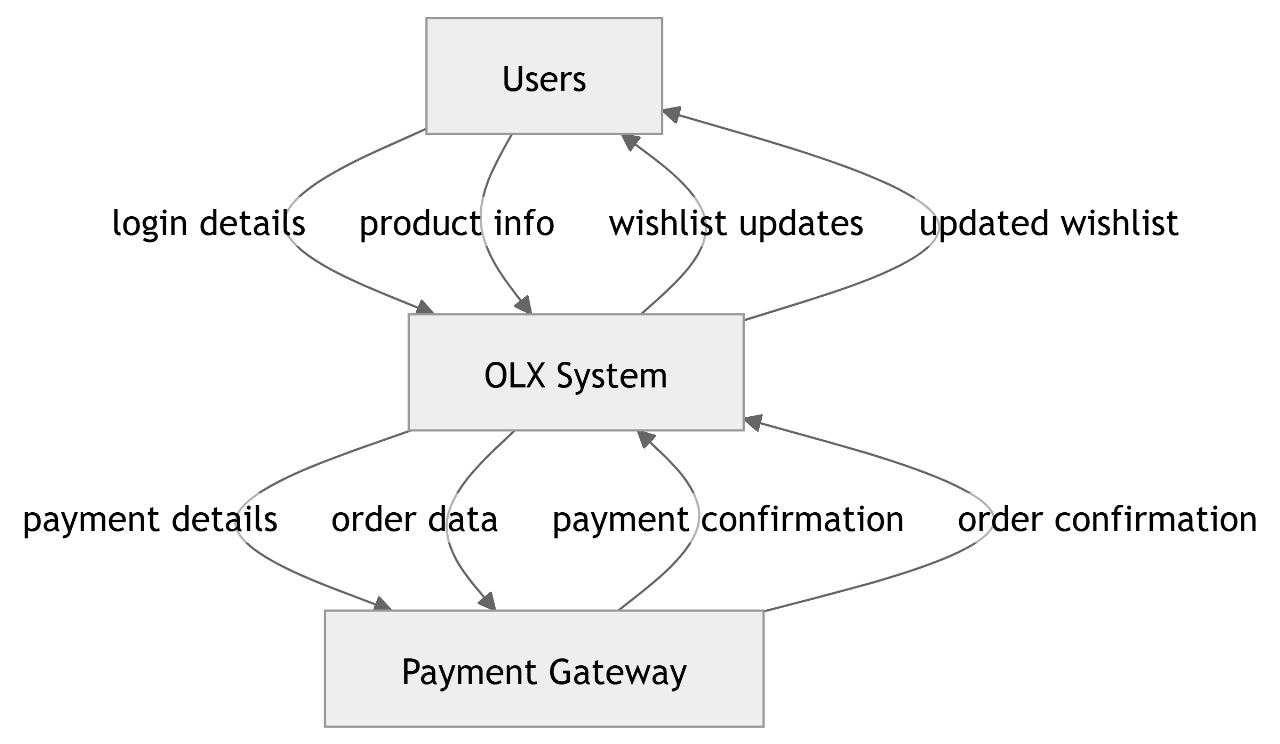
**4.2.2 Data Flow Diagram (DFD)**

A Data Flow Diagram (DFD) represents the flow of data within the system and how different processes interact with data stores. It is essential for understanding the system’s architecture and the flow of information between users, processes, and data stores.

**Level 1: Detailed DFD**

This level dives deeper into individual processes, describing the data flow between the system’s components.

* **Processes:**
  + User Registration & Login
    - Inputs: User Details
    - Outputs: User Account Information
  + **Product Management**
    - Inputs: Product Data
    - Outputs: Product Listings
  + **Order Management**
    - Inputs: Product Selection, User Address, Payment Information
    - Outputs: Order Confirmation, Order Details
  + **Payment Processing**
    - Inputs: Payment Details
    - Outputs: Payment Confirmation
  + **Wishlist Management**
    - Inputs: Product Additions/Removals
    - Outputs: Updated Wishlist



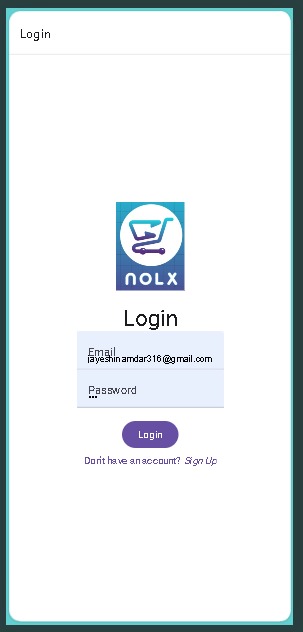
This diagram shows how the system receives data from users, processes it, and interacts with the payment gateway.

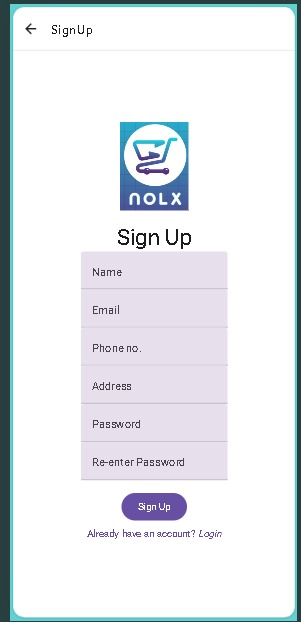
**7. CONCLUSION**

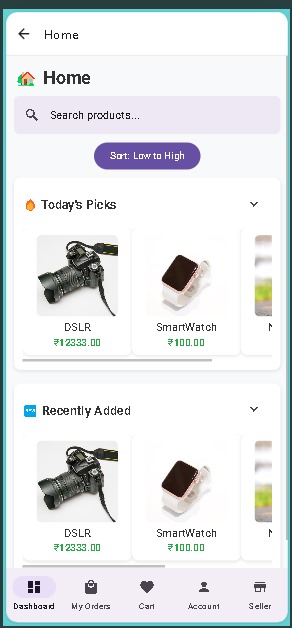
The NOLX project successfully implements an efficient and user-friendly online marketplace that allows users to buy and sell products seamlessly. The platform incorporates essential features such as user authentication, product listing, search functionality, wishlist management, order processing, and payment integration to ensure a smooth and secure transaction process.

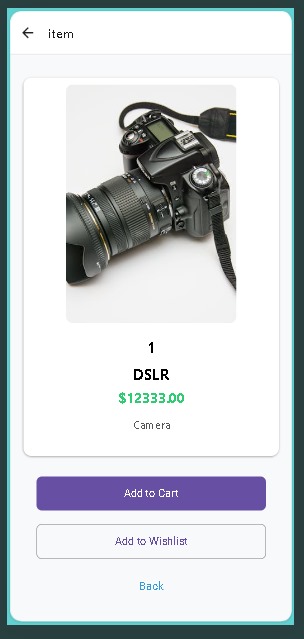


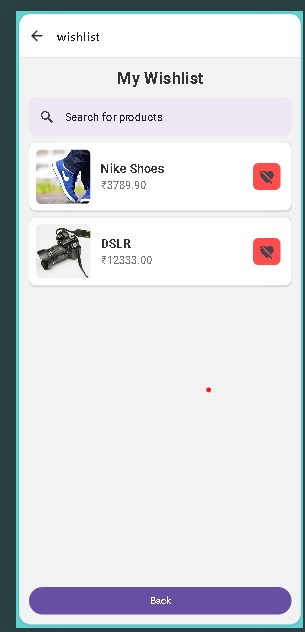
**logo**

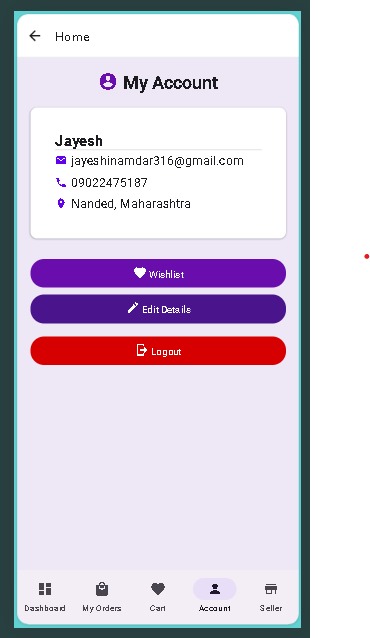


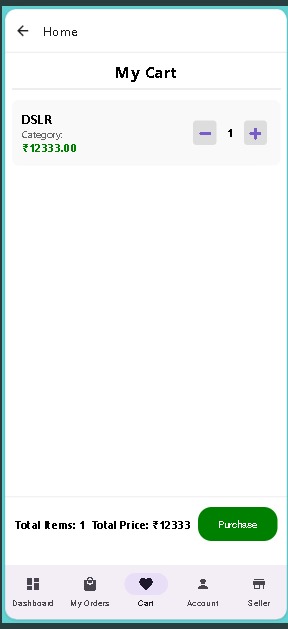


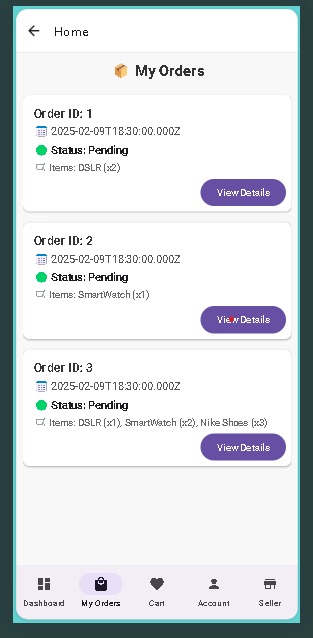


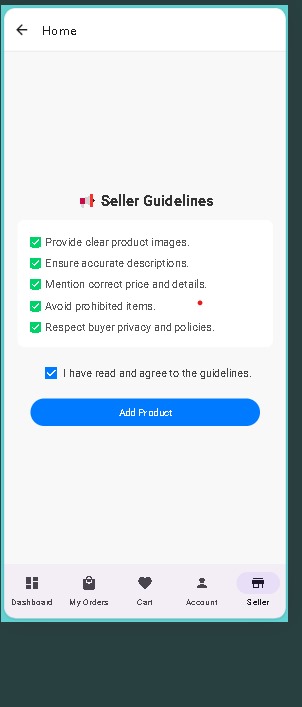


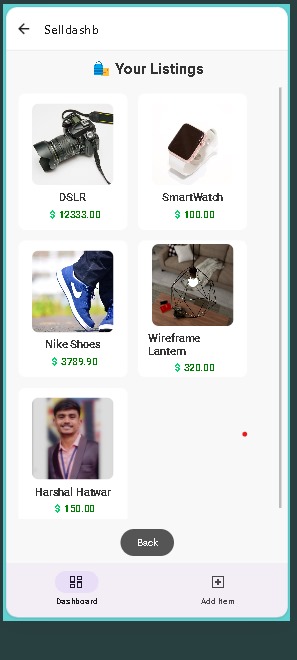












aa