Hamdard University
Department of Computing
Final Year Project

Online Auction App
FYP-026/FL-24

Software Design Specifications

Submitted by
M.Umair (491-2021)
Bismah Imran (1797-2021)
Maham Mirza (1922-2021)

Supervisor
Osama Ahmed Khan

Spring 2021
Document Sign off Sheet

1.1.1 Document Information

Project Title	Online Auction App
Project Code	FYP-026/FL-24
Document Name	Software Requirements Specifications
Document Version	1.0
Document Identifier	FYP-026/FL-24 SRS
Document Status	Draft
Author(s)	Bismah Imran , M,Umair
Approver(s)	Osama Ahmed Khan
Issue Date	17-01-2025

Name	Role	Signature	Date
M.Umair	Team Lead		
Bismah Imran	Team Member 2		
MahamMirza	Team Member 3		
Osama Ahmed Khan	Supervisor		
	Co-Supervisor		
	Project Coordinator		

Revision History

Date	Version	Description	Author
9-12-24	1.0	Data input	Bismah Imran
13-12-24	1.0	completion	Bismah Imran
2-01-25	1.0	Diagrams	M.Umair
14-01-25	1.0	Rechecking	Bismah, Umair

Definition of Terms, Acronyms, and Abbreviations

Term	Description
Online Auction	A digital platform where users can bid on items, with the highest bidder winning the item at the close.
Auction Engine	The core system managing auction functionalities such as item listings, bidding processes, and timers.
User Roles Bidder	Individuals participating in auctions by placing bids.
Seller/Auctioneer	Users listing items for auction.
Admin	System managers handling user management, disputes, and auction oversight.
Timed Auction	An auction type where bids are accepted for a fixed duration.
Reserve Auction	An auction type where the item is sold only if bidding meets a preset reserve price.
"Buy Now" Option	A feature allowing buyers to purchase items at a fixed price without bidding.
Real-Time Bidding	A system enabling instantaneous bid placement and updates, ensuring seamless interactions.
Chat Module	An integrated feature allowing direct communication between buyers and sellers for negotiations.
User Authentication	A secure login mechanism, including encrypted passwords and potentially two-factor authentication.
Notification System	Alerts users about bid status, auction closures, and system updates in real-time.
Data Encryption	A method of safeguarding user data and transactions against unauthorized access.
Caching Mechanisms	Technology like Redis used to speed up data retrieval for active auctions and frequent queries.
Fraud Detection	Systems and algorithms to identify and mitigate fraudulent activities, such as fake listings or shill bidding.

Term	Description
COD (Cash on Delivery)	A payment option where buyers pay upon receiving the product, often integrated with logistics systems.
Payment Gateway	Third-party services facilitating secure online transactions, including COD and escrow payments.
Accessibility Standards	Compliance with guidelines like WCAG to ensure the app is usable by individuals with disabilities.
Regulatory Compliance	Adherence to laws and regulations governing online auctions, including data protection and e-commerce.
Scalability	The app's ability to handle increased traffic and user growth without performance issues.
Dispute Resolution	Mechanisms to address and resolve conflicts between buyers and sellers effectively.

2 Table of Contents

1 Introduction

The Online Auction System is built with the primary goal of easing the process of conducting auctions for users, which raises the need for constructive non-standard approaches. This application has also enhanced the bidding mechanism, so instead of offering items at fixed prices, bidders can participate in an item's bidding process and buy it for a more competitive price. Another enhancement is the implementation of a user profile management system that allows users to register and manage their profiles securely, an auction module building which encompasses the description of items and the conditions of the bids, and a time-dependent bidding module where users can track the history of the bids and update their status. The ability to communicate directly for price negotiations is facilitated via a chatting system, while the admin side allows for a streamlined management of the auctions, user accounts, and user disputes. Users are also instantly notified about the changes in the auction status and bids, in addition to the fact that the application is optimized for any mobile device which means that it is easy for users to use the application when they are busy doing other tasks.

1.1 Purpose of Document

The purpose of this document is to provide comprehensive Software Design Specifications (SDS) for the Online Auction App. Outlining the design methodology, architecture and detailed design considerations. This document is intended to guide the development team and serve as a reference for stakeholders.

1.2 Intended Audience

This document is intended for

- Development Team members
- Project managers
- System architectures
- QA testers
- Stakeholders and supervisors

1.3 Project Overview

The Online Auction App is a digital platform designed to make buying and selling items easier and more dynamic. Unlike regular e-commerce apps with fixed prices, this app lets users bid on products, giving buyers a chance to get better deals and sellers an opportunity to earn more through competitive bidding.

It includes features like real-time updates on bids, user profiles to manage activities, and a chat system for direct communication between buyers and sellers. The app is secure, ensuring user data and transactions are safe, and works smoothly on mobile devices so users can participate in auctions anytime, anywhere. Administrators help manage the system, handle disputes, and ensure everything runs smoothly.

Overall, this app bridges the gap between traditional e-commerce and live auction systems, providing a modern and efficient solution for online trading. Whether you're a buyer looking for bargains or a seller wanting to reach a wider audience, this app offers everything you need for a seamless auction experience

1.4 Scope

- The Online Auction App is designed to create a modern, interactive platform for buying and selling items through real-time bidding. It allows users to register, create profiles, and post items for auction while providing a secure and user-friendly environment. Buyers can browse auctions, place bids, and receive instant updates about their bid status. Sellers benefit from reaching a wider audience and maximizing profits through competitive bidding.

- The app includes built-in chat features, enabling buyers and sellers to communicate directly, making negotiations and transactions smoother. It is optimized for mobile devices, ensuring that users can participate in auctions anytime and anywhere. Additionally, the app incorporates strong security measures to protect user data and transactions, such as encrypted communication and secure login processes. Powered by Firebase, the app ensures real-time updates and scalability, allowing it to handle a growing number of users and auctions seamlessly.

- The scope of this project focuses on delivering a reliable, accessible, and engaging online auction experience, bridging the gaps in traditional e-commerce platforms and empowering both buyers and sellers with innovative tools and features.

.

2 Design Considerations

2.1 Problem Identification and Preliminary Considerations

This section outlines the critical issues and requirements that need to be addressed before creating a comprehensive design solution for the online auction app. These considerations serve as the foundation for a robust, user-centric system design.

2.1.1 User Accessibility and Experience

- Ensure the app is intuitive and user-friendly for both buyers and sellers.
- Provide seamless navigation, from account creation to bidding and selling processes.
- Optimize the app for different devices (mobile, tablet, and desktop) and platforms (iOS, Android, and web browsers).

2.1.2 Security and Trust

- Implement secure authentication (e.g., two-factor authentication).
- Safeguard user data and financial transactions through encryption and compliance with data protection laws.
- Include features like verified user badges, feedback systems, and dispute resolution mechanisms to build trust.

2.1.3 Bidding and Auction Processes

- Support multiple auction types, such as timed auctions, reserve auctions, and "Buy Now" options.
- Real-time bidding updates with notifications for outbids and auction endings.
- Prevent fraudulent practices, such as shill bidding or fake listings, with automated monitoring.

2.1.4 Payment and Transaction Management

- Provide an option for sellers to enable or disable COD for their listings based on their preferences.
- Clearly indicate to buyers during checkout whether COD is available for their selected items.
- Specify delivery regions or conditions where COD is applicable.

2.1.5 Scalability and Performance

- Design the system to handle a growing user base and large volumes of concurrent bids without performance degradation.
- Employ efficient database management for real-time data updates.
- Optimize load balancing and caching mechanisms to reduce latency.

2.1.6 Regulatory and Compliance Requirements

- Adhere to regional and international e-commerce and auction laws.
- Ensure taxation rules are applied correctly based on user locations.
- Address legal requirements for selling restricted or regulated items.

2.1.7 Customer Support and Feedback

- Provide multiple channels for customer support (chat, email, FAQs, etc.).
- Enable users to report issues directly through the app.
- Regularly collect user feedback to drive continuous improvement.

2.1.8 Marketing and Growth

- Include features for promotional offers, such as discounted listing fees or special bidding events.
- Integrate social media sharing to increase user engagement and app visibility.
- Use analytics to track user behavior and improve marketing strategies.

2.2 Assumptions and Dependencies

2.2.1 Design-Specific Assumptions and Dependencies

2.2.2 Dynamic Real-Time Updates

- **Assumption:** The app must support real-time updates for auction events (e.g., bidding activity, auction expiration).
- **Dependency:** The design relies on WebSocket technology or similar real-time communication protocols to ensure low-latency updates. The system must scale to handle concurrent users during high-traffic auctions.

2.2.3 Scalable Modular Architecture

- **Assumption:** The system will need to support modular features like different auction types (e.g., timed, reserve) and payment methods, including COD and escrow services.
- **Dependency:** The design assumes a microservices architecture, where components such as the bidding engine, payment processing, and notifications are independently scalable.

2.2.4 Customizable User Interfaces

- **Assumption:** The app will offer different user roles (buyers, sellers, and admins), each with tailored dashboards and workflows.

- **Dependency:** The design depends on frameworks (e.g., React, Angular) that allow responsive, role-specific, and customizable UI elements.

2.2.5 Geographical and Regional Constraints

- **Assumption:** The system must accommodate region-specific features, such as language, currency, and tax rules.
- **Dependency:** The design incorporates APIs for localization, exchange rates, and tax calculations. Additionally, the system must dynamically adapt based on the user's location.

2.2.6 Secure Payment and Delivery Integration

- **Assumption:** For payments, COD will require integration with logistics providers to track deliveries and confirm transactions securely.
- **Dependency:** The design assumes APIs for delivery tracking, payment reconciliation, and secure data transfer between the app and third-party services.

2.2.7 Fraud and Dispute Management

- **Assumption:** Users will expect mechanisms to detect fraudulent activities and resolve disputes efficiently.
- **Dependency:** The design assumes the integration of AI-driven fraud detection tools and a scalable dispute resolution module for automated and manual review.

2.2.8 Flexibility in Feature Addition

- **Assumption:** New features (e.g., social sharing, in-app chat, or AI recommendations) may need to be added post-launch without disrupting existing workflows.
- **Dependency:** The system must employ a flexible plugin or API-based structure to allow feature expansion with minimal downtime.

2.2.9 Third-Party Services

- **Assumption:** Critical features like notifications, analytics, and performance monitoring will rely on third-party services (e.g., Firebase, Mixpanel, Twilio).
- **Dependency:** The design assumes stable APIs and continued support from these services, with fallback mechanisms in place for potential service interruptions.

2.2.10 Compliance with Accessibility Standards

- **Assumption:** The app must comply with accessibility guidelines to cater to users with disabilities.
- **Dependency:** The design depends on accessible design principles (e.g., WCAG) and tools for testing accessibility compliance during the development process.

2.2.11 Support for High Availability and Disaster Recovery

- **Assumption:** The app should remain operational during high traffic and recover quickly from unexpected downtimes.
- **Dependency:** The design assumes the use of cloud services with built-in redundancy, auto-scaling, and disaster recovery mechanisms.

2.3 Risks and Volatile Areas

2.3.1 Emerging User Requirements

- **Source of Change:** Users may demand additional features like advanced filtering, new auction types, or expanded payment options (e.g., cryptocurrency, BNPL).
- **Impact:** Requires iterative development and system modularity to integrate new features without disrupting core functionalities.
- **Mitigation:**
  - Design the system with a modular architecture to allow independent addition of new features.
  - Use Agile development methodologies to respond to changing user requirements in short cycles.

2.3.2 Technological Advancements

- **Source of Change:** Adoption of new technologies (e.g., AI for fraud detection, AR for virtual item previews, blockchain for transparent bidding).
- **Impact:** Upgrades or overhauls in existing systems may be necessary to stay competitive.
- **Mitigation:**
  - Leverage APIs and cloud-based services that allow easy integration with cutting-edge technologies.
  - Regularly review technological trends to proactively plan for updates.

2.3.3 Regulatory and Compliance Changes

- **Source of Change:** Updates to e-commerce, auction, or data protection laws across regions.
- **Impact:** Non-compliance could lead to legal penalties or service restrictions in specific regions.
- **Mitigation:**
  - Maintain a legal and compliance team to monitor regulatory updates.
  - Design the system to support region-specific configurations for tax laws, restricted items, and data handling.

2.3.4 Scalability Demands

- **Source of Change:** A sudden increase in users or high-profile auctions leading to system overload.
- **Impact:** Performance degradation, slow bidding updates, or system crashes could harm user trust.
- **Mitigation:**
    - Utilize cloud infrastructure with auto-scaling capabilities to handle peak loads.
    - Employ load testing during development to ensure system stability under high traffic.

2.3.5 Fraud and Security Threats

- **Source of Change:** Increasing sophistication in fraud techniques (e.g., fake accounts, shill bidding, item misrepresentation).
- **Impact:** Loss of user trust, financial losses, and reputational damage.
- **Mitigation:**
    - Integrate AI-driven fraud detection and monitoring tools.
    - Regularly update security protocols and conduct penetration testing.

2.3.6 Competition and Market Dynamics

- **Source of Change:** Competitors introducing innovative features or offering lower transaction fees.
- **Impact:** Risk of user attrition and reduced market share.
- **Mitigation:**
    - Monitor competitors and user feedback to stay ahead in innovation.
    - Implement loyalty programs and promotional campaigns to retain users.

2.3.7 Logistics and Payment Ecosystem Changes

- **Source of Change:** Dependency on third-party logistics providers and payment gateways that may revise APIs or terms.
- **Impact:** Disruptions in payment processing or COD fulfillment.
- **Mitigation:**
    - Partner with multiple logistics providers and payment gateways to reduce single points of failure.
    - Periodically update integrations to align with third-party changes.

2.3.8 Contingency Paths for Changes

1. Maintain a flexible microservices architecture that allows isolated updates without affecting the entire system.
2. Use feature toggles to enable or disable new features in response to unexpected challenges.

3 Develop a rollback plan for major updates to revert to stable versions if unforeseen issues arise.

3.1. System Architecture

This section provides a high-level overview of the online auction app, describing how its functionality and responsibilities are divided among various components and subsystems. The architecture ensures scalability, maintainability, and efficient collaboration between components to deliver a seamless user experience.

3.1.1 High-Level Overview

The system follows a modular and microservices-based architecture, ensuring that each component operates independently while interacting with others through defined interfaces. It is divided into the following layers:

3.1.1.1 Presentation Layer

- **Purpose**: User interaction via mobile and web applications.
- **Components**:
  - Mobile App (iOS/Android)
  - Web Interface (React/Angular-based)
- **Responsibilities**:
  - Provide an intuitive UI for buyers, sellers, and admins.
  - Handle real-time updates for bids, notifications, and auction statuses.
  - Ensure responsiveness and accessibility across devices.

3.1.1.2 Business Logic Layer

- **Purpose**: Manage core functionalities and workflows of the auction system.
- **Components**:
  - Auction Engine: Manages auction creation, bidding processes, and timer functionalities.
  - User Management Service: Handles registration, login, roles, and profile management.
  - Payment Module: Manages payment methods, including COD, escrow, and online payments.
  - Notification Service: Sends real-time alerts (e.g., outbids, auction end) via email, SMS, and push notifications.
- **Responsibilities**:
  - Ensure real-time bid updates.
  - Validate bids, manage auction rules, and resolve conflicts.
  - Integrate fraud detection algorithms.

3.1.1.3 Data Layer

- **Purpose**: Manage storage and retrieval of system data.
- **Components**:
  - Relational Database (e.g., PostgreSQL): For structured data like user profiles, auction details, and transaction records.
  - NoSQL Database (e.g., MongoDB): For unstructured data like logs, notifications, and analytics.
  - Cache Layer (e.g., Redis): For high-speed access to frequently queried data, like active auctions and bids.
- **Responsibilities**:
  - Maintain data consistency and integrity.
  - Provide APIs for seamless data retrieval and updates.

3.1.1.4 Integration Layer

- **Purpose**: Facilitate communication with third-party services.
- **Components**:
  - Payment Gateways: Support secure transactions for COD and online payments.
  - Logistics APIs: Handle shipping, tracking, and COD verification.
  - Analytics and Monitoring Tools: Collect and analyze user activity and system performance.
- **Responsibilities**:
  - Manage external dependencies like payment and delivery services.
  - Ensure API reliability and fallback mechanisms for third-party failures.

3.1.1.5 Infrastructure Layer

- **Purpose**: Provide a robust environment for system deployment and scalability.
- **Components**:
  - Cloud Hosting (e.g., AWS, Azure, GCP): For deployment, scalability, and disaster recovery.
  - Load Balancers: Distribute traffic across servers to avoid overload.
  - Monitoring Tools (e.g., Prometheus, Grafana): Track system performance and uptime.
- **Responsibilities**:
  - Enable auto-scaling to handle peak traffic during high-profile auctions.
  - Provide disaster recovery and backups to ensure data safety.

3.1.2 Interaction Between Components

3.1.2.1 User Interaction Flow:

- Users interact with the app via the Presentation Layer.

- User requests (e.g., placing a bid, viewing auctions) are processed by the Business Logic Layer.
- Data retrieval and updates occur in the Data Layer, ensuring minimal latency through caching.
- Notifications are sent via the Notification Service for key events (e.g., auction updates, bid success).

3.1.2.2 Auction Process Flow:

- Sellers create auctions via the UI, processed by the Auction Engine.
- Buyers place bids in real-time, validated and updated in the system through the Business Logic Layer.
- Auction results are finalized, and payment/fulfillment processes are initiated using Payment and Logistics APIs.

3.1.2.3 COD Workflow:

- Buyers select COD at checkout, triggering the Logistics API for order confirmation.
- Delivery updates are tracked, and cash is reconciled via logistics partners before releasing funds to the seller.

3.1.3 Scalability and Extensibility

- The architecture supports future feature additions (e.g., new auction types, advanced analytics) via modular components.
- High scalability is achieved through containerized microservices (e.g., using Docker and Kubernetes).
- APIs ensure seamless integration with new third-party services as the system evolves.

3.2 Software Architecture

The software architecture for the Online Auction System is designed with a layered approach for simplicity and modularity.

3.2.1. Layers of Architecture

1. **User Interface Layer**:
    - Manages interactions with the users (bidders, auctioneers, admins).
    - Example technologies: React.js, Angular, or mobile frameworks like Flutter.
2. **Middle Tier (Business Logic)**:
    - Handles the main business logic.
    - Manages bid validation, auction rules, and real-time updates.
    - Technologies: Node.js, Java Spring Boot, or .NET Core.
3. **Data Access Layer**:

   - Interfaces between the application and the database.
   - Implements ORM frameworks like Hibernate or Sequelize.
4. **Database Layer**:
   - Stores all application data persistently.
   - Technologies: Firebase, PostgreSQL, or MySQL.

4. Design Strategy

The design strategy for the online auction app is centered around scalability, modularity, and user-centric functionality. This section outlines the key abstractions, mechanisms, and decision-making processes that shape the system's high-level organization while addressing considerations such as extensibility, reuse, interface paradigms, data management, and concurrency.

4.1 Future System Extension or Enhancement

- **Strategy:**
  - Use a modular architecture where each component (e.g., bidding engine, payment gateway, notification system) operates independently.
  - Adopt microservices for core functionalities to enable easy addition or replacement of features.
  - Ensure API-driven development, allowing integration with new third-party services without disrupting the system.
- **Reasoning:**
  - User needs and market trends evolve, necessitating a flexible system that supports new auction types (e.g., reverse auctions) or payment methods (e.g., cryptocurrencies).
  - The microservices approach ensures isolated updates and prevents downtime.
- **Trade-offs:**
  - Increased complexity in managing inter-service communication and dependencies.
  - Potential latency in distributed systems compared to monolithic architectures.

4.2 System Reuse

- **Strategy:**
  - Design reusable components for common functionalities like user authentication, payment processing, and notifications.
  - Employ shared libraries and frameworks for tasks such as logging, error handling, and security.
- **Reasoning:**
  - Reuse reduces development effort for future projects or additional platforms (e.g., mobile and web).
  - Standardized components improve system consistency and reduce maintenance overhead.

- **Trade-offs:**
  - Additional effort is required upfront to ensure components are generic and well-documented.
  - Customization for specific use cases may require extending the reusable components.

4.3 User Interface Paradigms

- **Strategy:**
  - Use a responsive design paradigm to ensure a seamless experience across devices.
  - Implement role-based interfaces tailored to buyers, sellers, and administrators.
  - Focus on real-time interactivity with Web Socket or Server-Sent Events (SSE) for updates during auctions.
- **Reasoning:**
  - The app serves diverse user roles and device types, making adaptable UIs essential.
  - Real-time feedback during bidding increases user engagement and ensures competitive fairness.
- **Trade-offs:**
  - Real-time UI updates require robust backend systems and increased network demands.
  - Implementing role-specific interfaces can increase design complexity.

4.4 Data Management

- **Strategy:**
  - Use a combination of relational databases (e.g., PostgreSQL) for structured data (e.g., user profiles, auction details) and NoSQL databases (e.g., MongoDB) for unstructured data (e.g., logs, analytics).
  - Employ caching mechanisms (e.g., Redis) for frequently accessed data like active auctions and bid histories.
  - Design the system to support data partitioning and replication for high availability.
- **Reasoning:**
  - A hybrid database approach optimizes for performance and scalability.
  - Caching reduces latency and ensures real-time data availability during critical auction moments.
- **Trade-offs:**
  - Managing hybrid databases introduces additional complexity.
  - Consistency issues may arise between cached data and the primary database, requiring robust synchronization mechanisms.

4.5 Concurrency and Synchronization

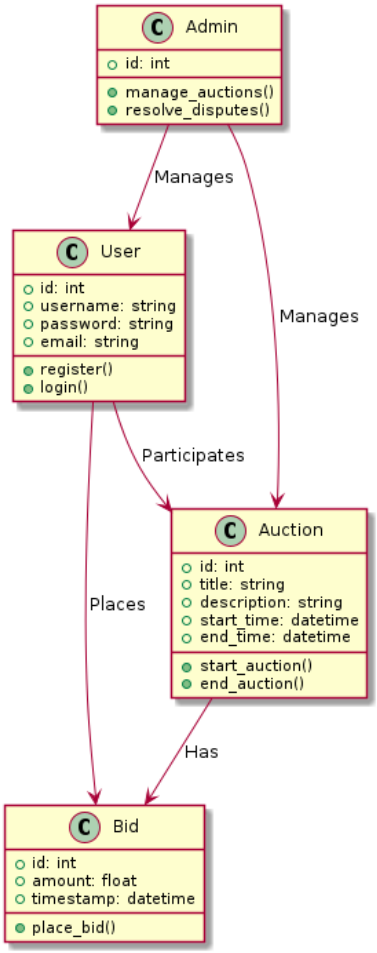
- **Strategy:**

- Use optimistic locking or versioning for handling concurrent updates in critical operations like placing bids.
- Employ event-driven architecture with message queues (e.g., RabbitMQ, Kafka) to manage high-concurrency scenarios.
- Design backend services to be stateless, enabling horizontal scaling for increased concurrent user loads.

- **Reasoning:**
  - Auctions involve real-time interactions, making concurrency management crucial to avoid race conditions or data corruption.
  - Event-driven systems decouple components, improving scalability and reliability under heavy loads.
- **Trade-offs:**
  - Increased complexity in managing distributed transactions.
  - Real-time bid synchronization requires significant infrastructure investment.

5. Detailed System Design

5.1 Design Class Diagram



Certainly! Below is the revised version within the range of 5.2:

5.2 Database Design

5.2.1 Data 1: User

- **Name:** User
- **Alias:** Bidder, Auctioneer
- **Content Description:** ID + Username + Password + Email + Role + Created_At

Table: User

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
ID	Unique identifier for user	Integer	-	No	Auto-increment	PK
Username	Name of the user	String	100	No	NULL	
Password	Encrypted password	String	255	No	NULL	
Email	User’s email address	String	150	No	NULL	
Role	Role of the user (bidder/seller)	String	50	No	'Bidder'	
Created_At	Account creation timestamp	DateTime	-	No	CURRENT_TIMESTAMP	

5.2.2 Data 2: Admin

- **Name:** Admin
- **Alias:** System Manager
- **Where-used/how-used:** Used for managing users, auctions, and system disputes.
- **Content Description:** ID + Username + Password + Email + Created_At

Table: Admin

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
ID	Unique identifier for admin	Integer	-	No	Auto-increment	PK
Username	Name of the admin	String	100	No	NULL	
Password	Encrypted password	String	255	No	NULL	
Email	Admin's email address	String	150	No	NULL	
Created_At	Account creation timestamp	DateTime	-	No	CURRENT_TIMESTAMP	

5.2.3 Data 3: Auction

- **Name:** Auction
- **Alias:** Item, Listing
- **Where-used/how-used:** Used for creating auctions, placing bids, and notifications.
- **Content Description:** ID + Title + Description + Start_Time + End_Time + Starting_Bid + Current_High_Bid + User_ID + Created_At

Table: Auction

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
ID	Unique identifier for auction	Integer	-	No	Auto-increment	PK

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
Title	Title of the auction item	String	200	No	NULL	
Description	Description of the auction item	Text	-	Yes	NULL	
Start_Time	Auction start time	DateTime	-	No	NULL	
End_Time	Auction end time	DateTime	-	No	NULL	
Starting_Bid	Minimum starting bid	Float	-	No	NULL	
Current_High_Bid	Current highest bid	Float	-	Yes	NULL	
User_ID	ID of user creating the auction	Integer	-	No	NULL	FK (User.ID)
Created_At	Auction creation timestamp	DateTime	-	No	CURRENT_TIMESTAMP	

5.2.4 Data 4: Bid

- **Name:** Bid
- **Alias:** Offer
- **Where-used/how-used:** Used for placing and tracking bids in auctions.
- **Content Description:** ID + Amount + Timestamp + User_ID + Auction_ID

Table: Bid

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
ID	Unique identifier for the bid	Integer	-	No	Auto-increment	PK

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
Amount	Bid amount placed by the user	Float	-	No	NULL	
Timestamp	Time the bid was placed	DateTime	-	No	CURRENT_TIMESTAMP	
User_ID	ID of user placing the bid	Integer	-	No	NULL	FK (User.ID)
Auction_ID	ID of auction the bid belongs to	Integer	-	No	NULL	FK (Auction.ID)

5.2.5 Data 5: Notification

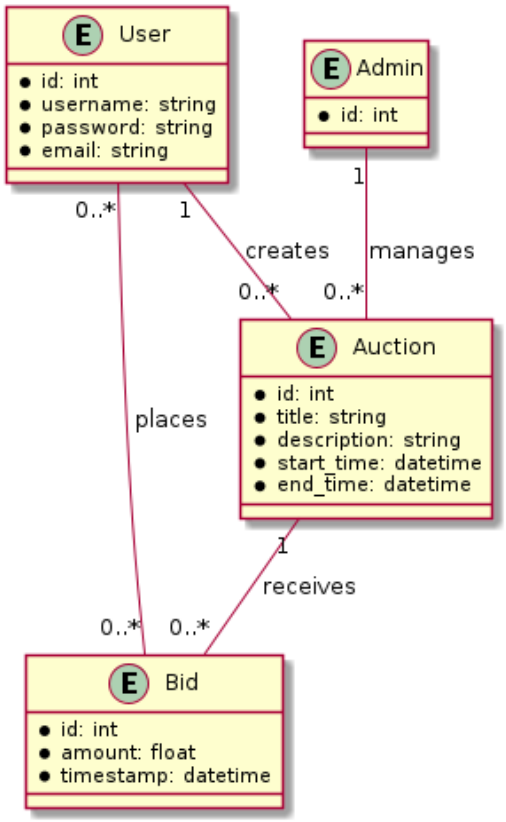
- **Name:** Notification
- **Alias:** Alert
- **Where-used/how-used:** Used to notify users about auction and bid updates.
- **Content Description:** ID + Message + User_ID + Timestamp

Table: Notification

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
ID	Unique identifier for notification	Integer	-	No	Auto-increment	PK
Message	Notification content	Text	-	No	NULL	
User_ID	ID of user receiving the notification	Integer	-	No	NULL	FK (User.ID)
Timestamp	Time the notification was sent	DateTime	-	No	CURRENT_TIMESTAMP	

This structure follows the provided format for the database design within the 5.2 section range. Let me know if you need any more adjustments!

5.3 Entity-Relationship (ER) Diagram and Description



5.3.1 ER Data Model

1. **User Table**

Attribute Name	Data Type	Constraints
ID	Integer	Primary Key, Not null, Unique
Username	String	Not null
Password	String	Not null
Email	String	Not null, Unique

---

2. **Admin Table**

Attribute Name	Data Type	Constraints
ID	Integer	Primary Key, Not null, Unique

---

3. **Auction Table**

Attribute Name	Data Type	Constraints
ID	Integer	Primary Key, Not null, Unique
Title	String	Not null
Description	String	Optional
Start_Time	DateTime	Not null
End_Time	DateTime	Not null

---

4. **Bid Table**

Attribute Name	Data Type	Constraints
ID	Integer	Primary Key, Not null, Unique
Amount	Float	Not null
Timestamp	DateTime	Not null
User_ID	Integer	Foreign Key (references User.ID)
Auction_ID	Integer	Foreign Key (references Auction.ID)

5.3.2 Relationships

Relationship	Source Table	Source Attribute	Target Table	Target Attribute	Cardinality
User places Bid	User	ID	Bid	User_ID	1 to Many
Bid is placed on Auction	Bid	Auction_ID	Auction	ID	Many to 1
Admin creates Auction	Admin	ID	Auction	Admin_ID	1 to Many
Admin manages Auction	Admin	ID	Auction	Admin_ID	Many to Many*

5.3.3 E/R Model Description

1. **Entities:**
    - **User**: Represents the users participating in the auction system.
    - **Admin**: Represents the administrators managing the auctions.
    - **Auction**: Represents the auctioned items.

- **Bid**: Represents the bids placed by users on auctions.

2. **Relationships:**
   - **User places Bid**: A user can place multiple bids, but each bid is associated with one user.
   - **Bid is placed on Auction**: A bid is linked to one auction, and an auction can have multiple bids.
   - **Admin creates Auction**: Admins are responsible for creating auctions.
   - **Admin manages Auction**: Admins also manage the auctions after they are created.
3. **Cardinalities:**
   - A **User** can place 0 or more **Bids**, but a **Bid** is placed by 1 **User**.
   - A **Bid** belongs to 1 **Auction**, but an **Auction** can receive 0 or more **Bids**.
   - An **Admin** can create or manage 0 or more **Auctions**.

5.3.4 Data Dictionary

5.3.4.1 Data 1: User

Attribute Name	Description	Data Type	Constraints
User ID	Unique identifier for each user	Integer	Primary Key, Not null, Unique
User Name	Name of the user	String	Not null
Email	Email address of the user	String	Not null, Unique
Password	Encrypted password for the user	String	Not null
Contact Number	Phone number of the user	String	Optional

5.3.4.2 Data 2: Item

Attribute Name	Description	Data Type	Constraints
Item ID	Unique identifier for each auction item	Integer	Primary Key, Not null, Unique
Item Name	Name of the auction item	String	Not null
Item Description	Detailed description of the auction item	Text	Optional
Starting Bid	Minimum bid amount for the item	Decimal	Not null
Auction End Time	End time of the auction for the item	DateTime	Not null

5.3.4.3 Data 3: Bid

Attribute Name	Description	Data Type	Constraints
Bid ID	Unique identifier for each bid	Integer	Primary Key, Not null, Unique
Bid Amount	Amount placed by the bidder	Decimal	Not null
Bid Time	Time when the bid was placed	DateTime	Not null
User ID	Identifier of the user who placed the bid	Integer	Foreign Key (references User ID)
Item ID	Identifier of the item being bid on	Integer	Foreign Key (references Item ID)

5.3.4.4 Data 4: Chat

Attribute Name	Description	Data Type	Constraints
Chat ID	Unique identifier for each chat instance	Integer	Primary Key, Not null, Unique
Sender ID	Identifier of the user sending the message	Integer	Foreign Key (references User ID)
Receiver ID	Identifier of the user receiving the message	Integer	Foreign Key (references User ID)
Message Content	Content of the chat message	Text	Not null
Timestamp	Time when the message was sent	DateTime	Not null

5.3.4.5 Data 5: Notification

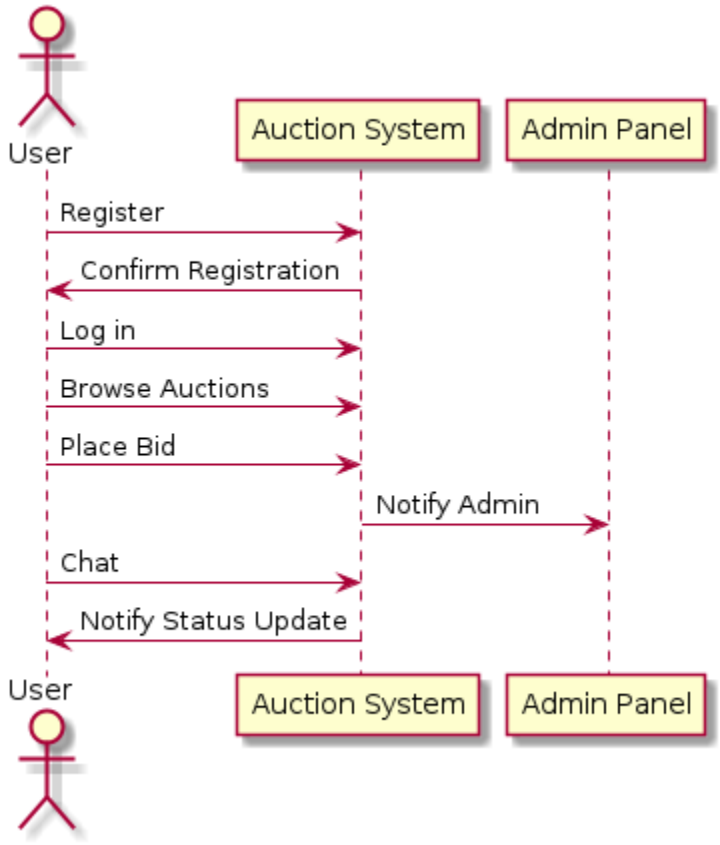
Attribute Name	Description	Data Type	Constraints
Notification ID	Unique identifier for each notification	Integer	Primary Key, Not null, Unique
Notification Type	Type of notification (e.g., Bid Update, Auction Status)	String	Not null
User ID	Identifier of the user receiving the notification	Integer	Foreign Key (references User ID)
Message Content	Details of the notification	Text	Optional

Attribute Name	Description	Data Type	Constraints
Timestamp	Time when the notification was sent	DateTime	Not null

5.4 Application Design

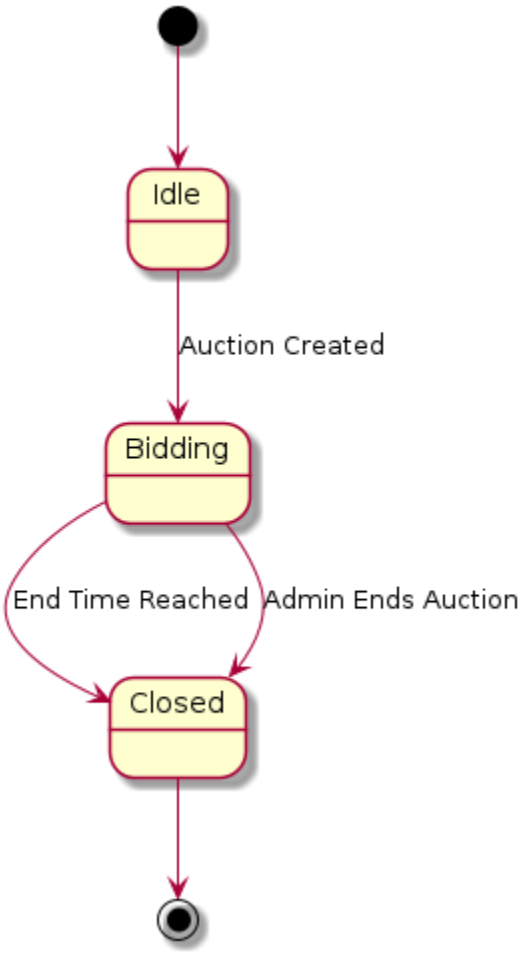
5.4.1 Sequence Diagram

5.4.1.1Sequence Diagram 1



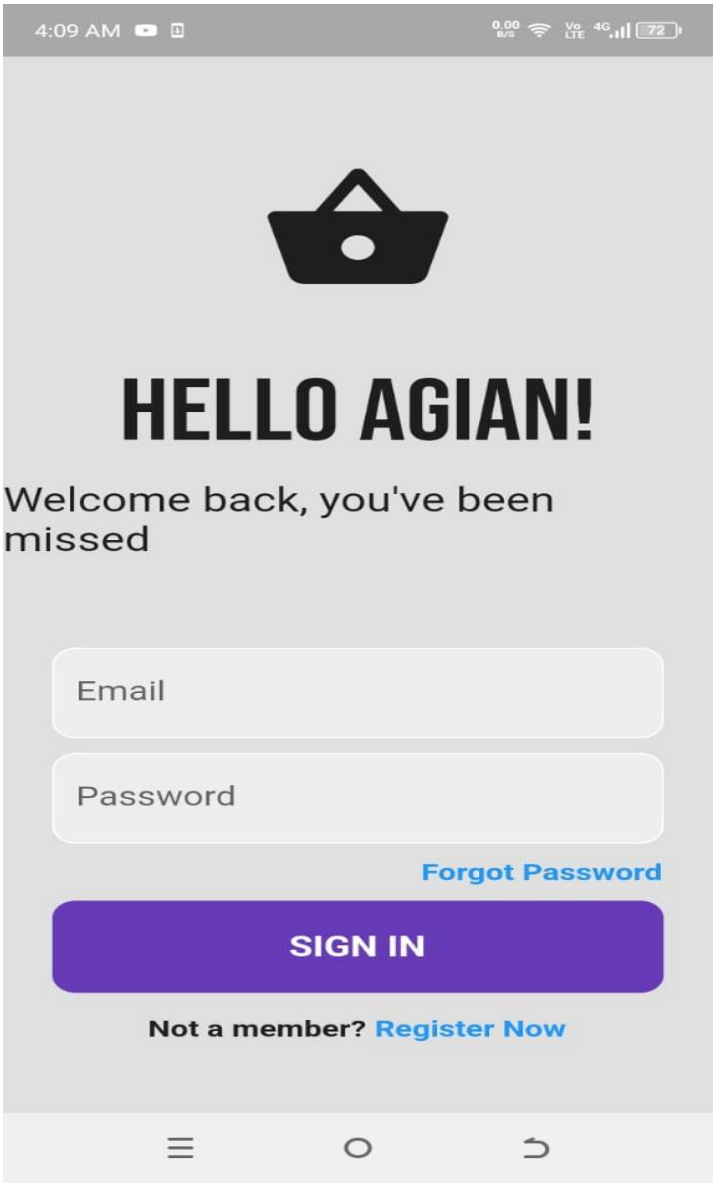
5.4.2 State Diagram

5.4.2.1 State Diagram 1

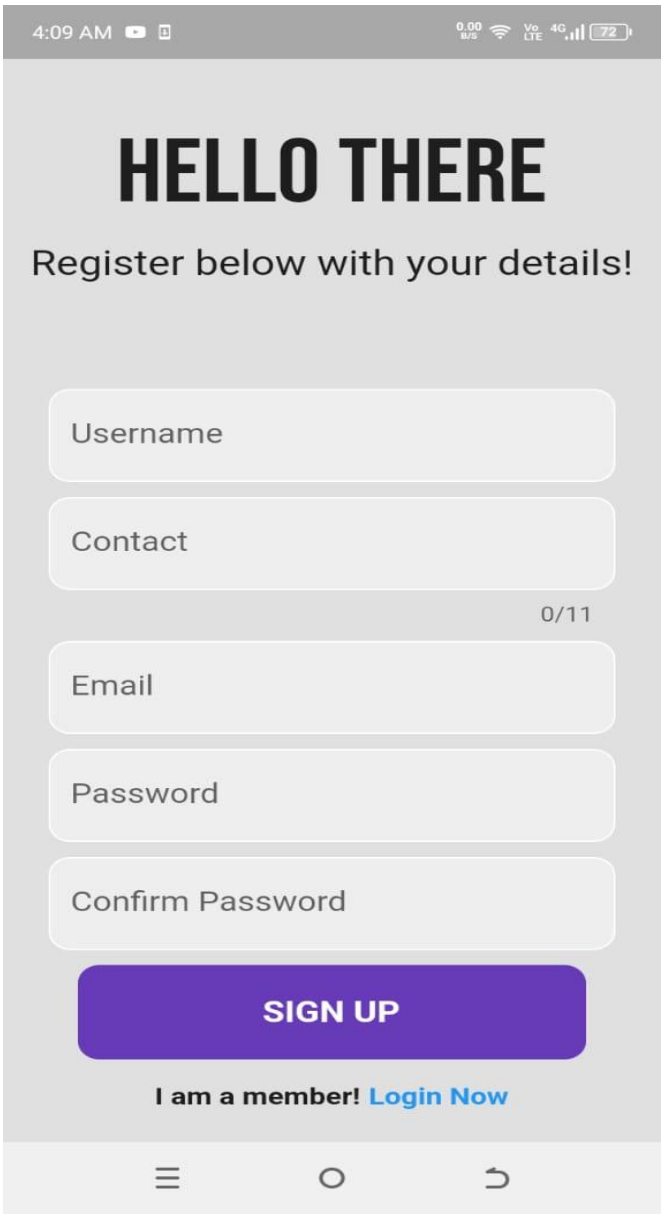


5.5 GUI Design

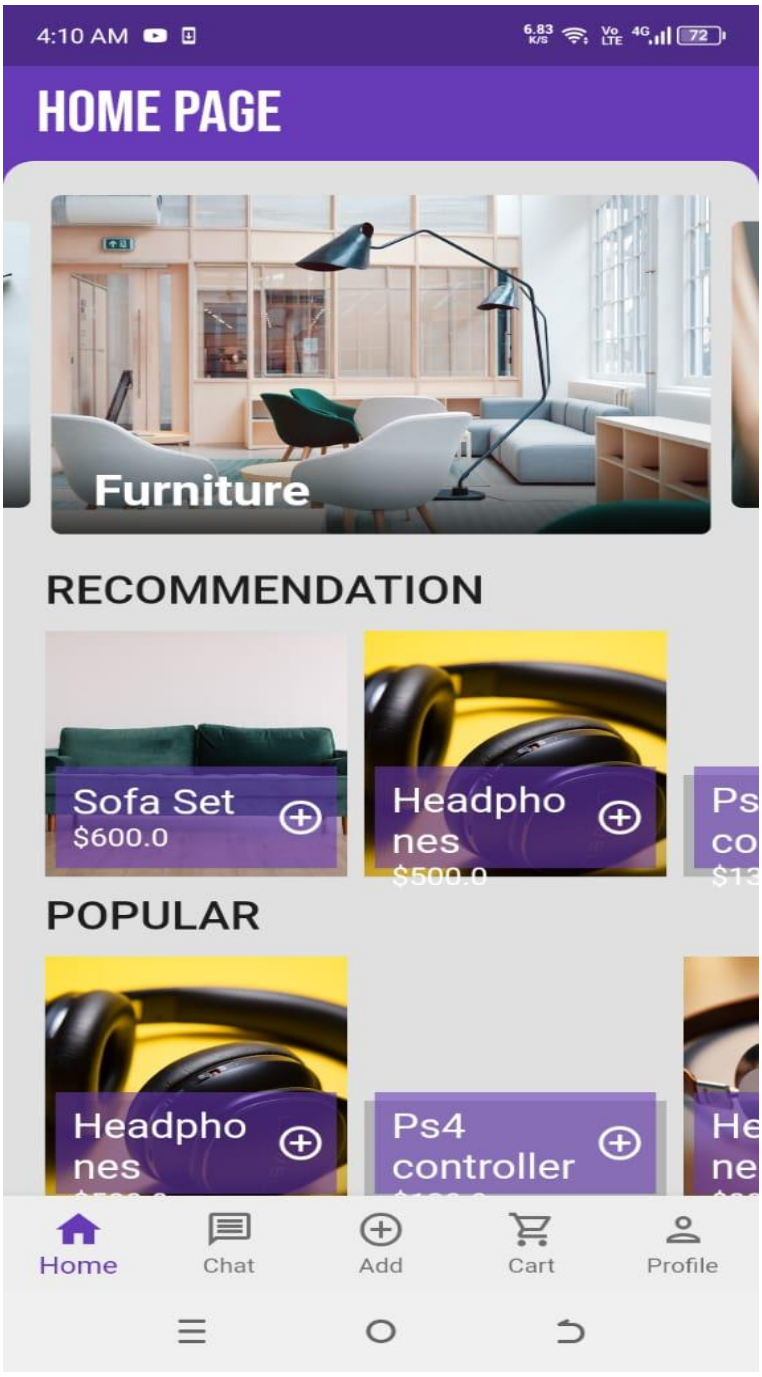
5.5.1 Login Page - Mock Screen 1



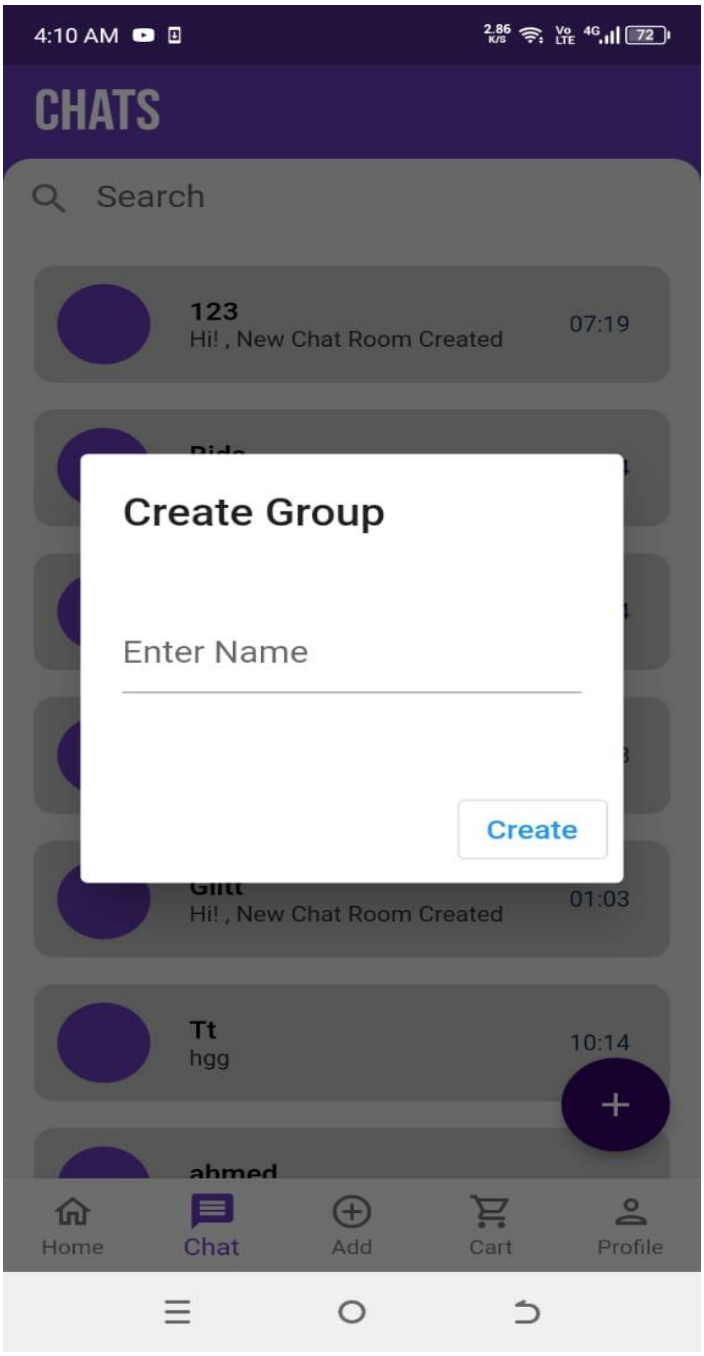
5.5.2 Sign Up Page - Mock Screen 2



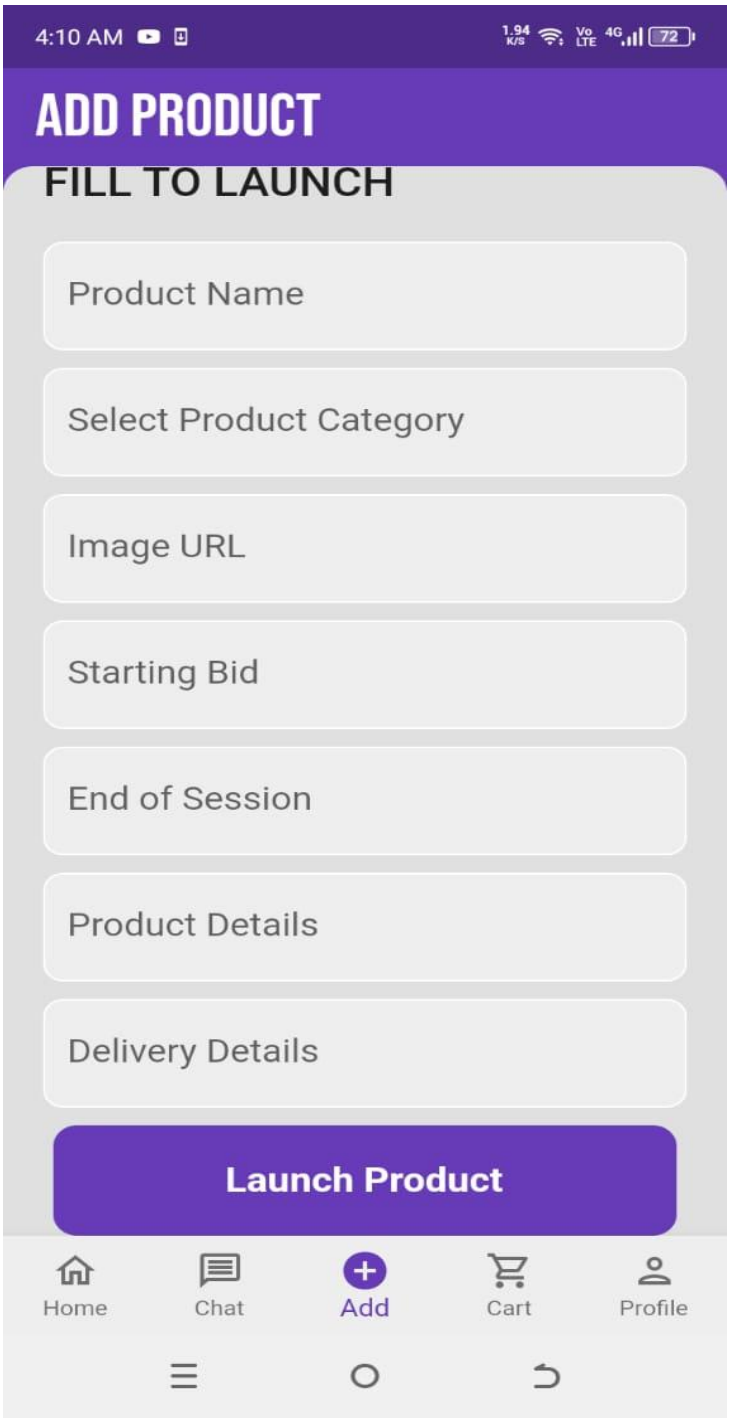
5.5.3 Homepage - Mock Screen 3



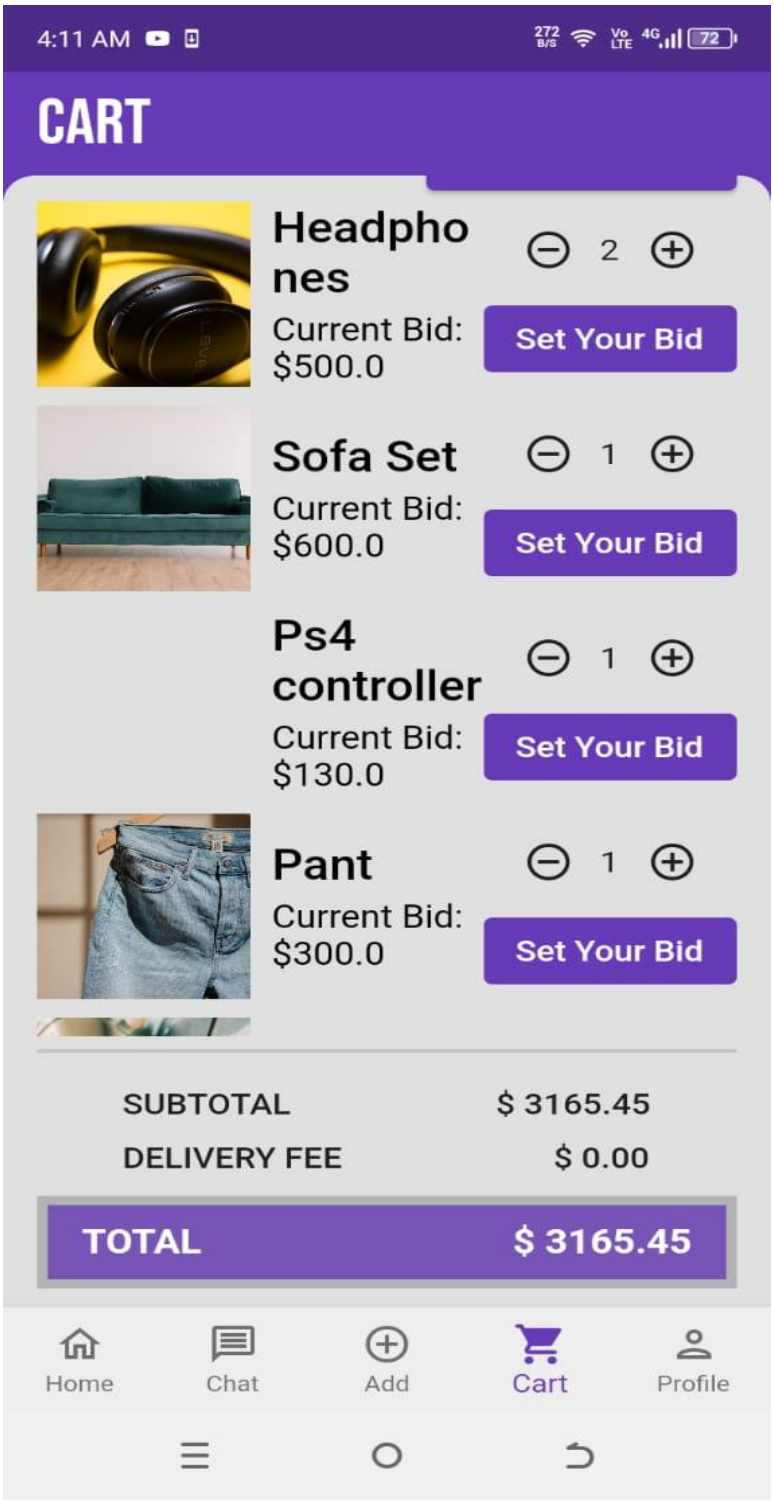
5.5.4 Chat Room - Mock Screen 4



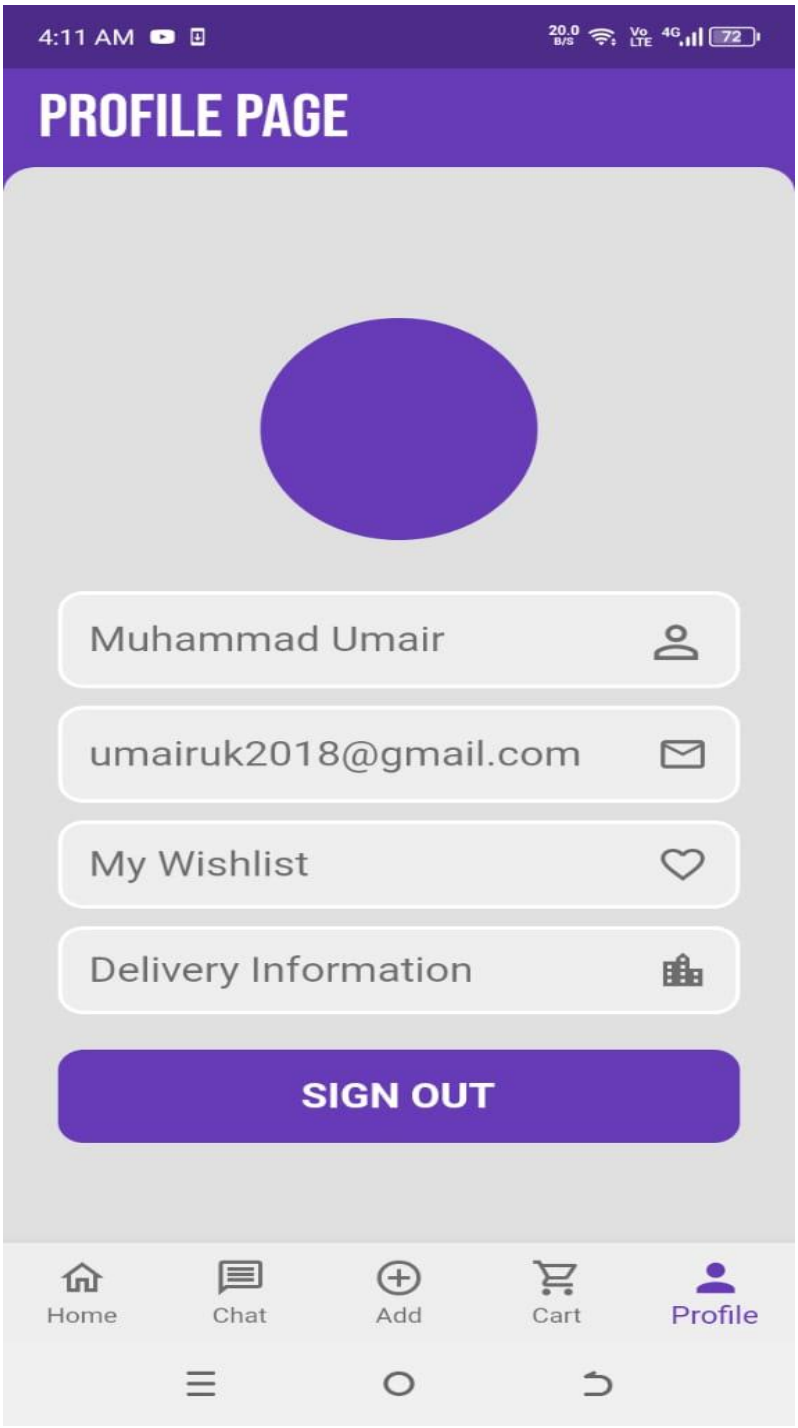
5.5.5 Launch Product - Mock Screen 5



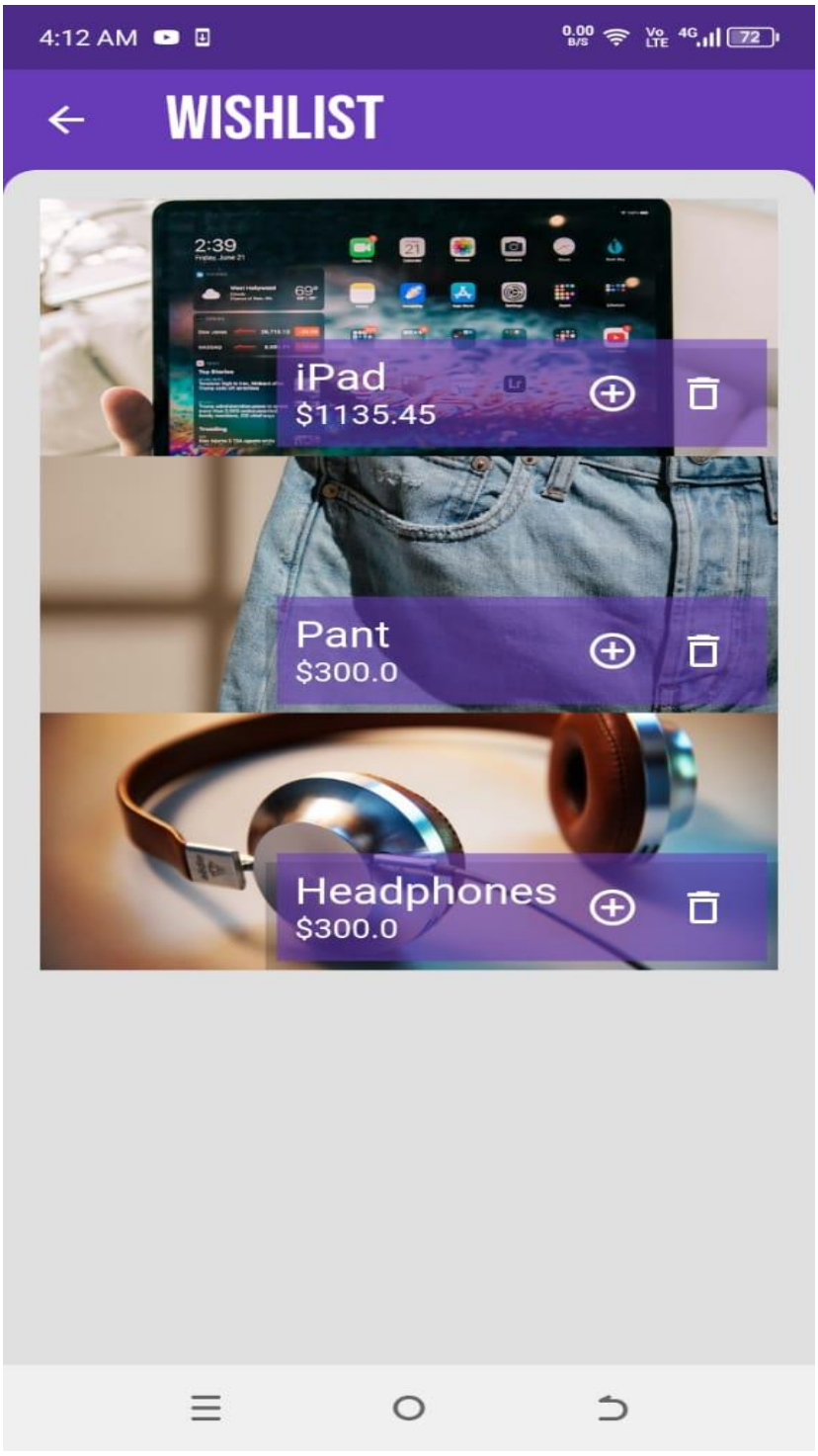
5.5.6 Add To Cart - Mock Screen 6



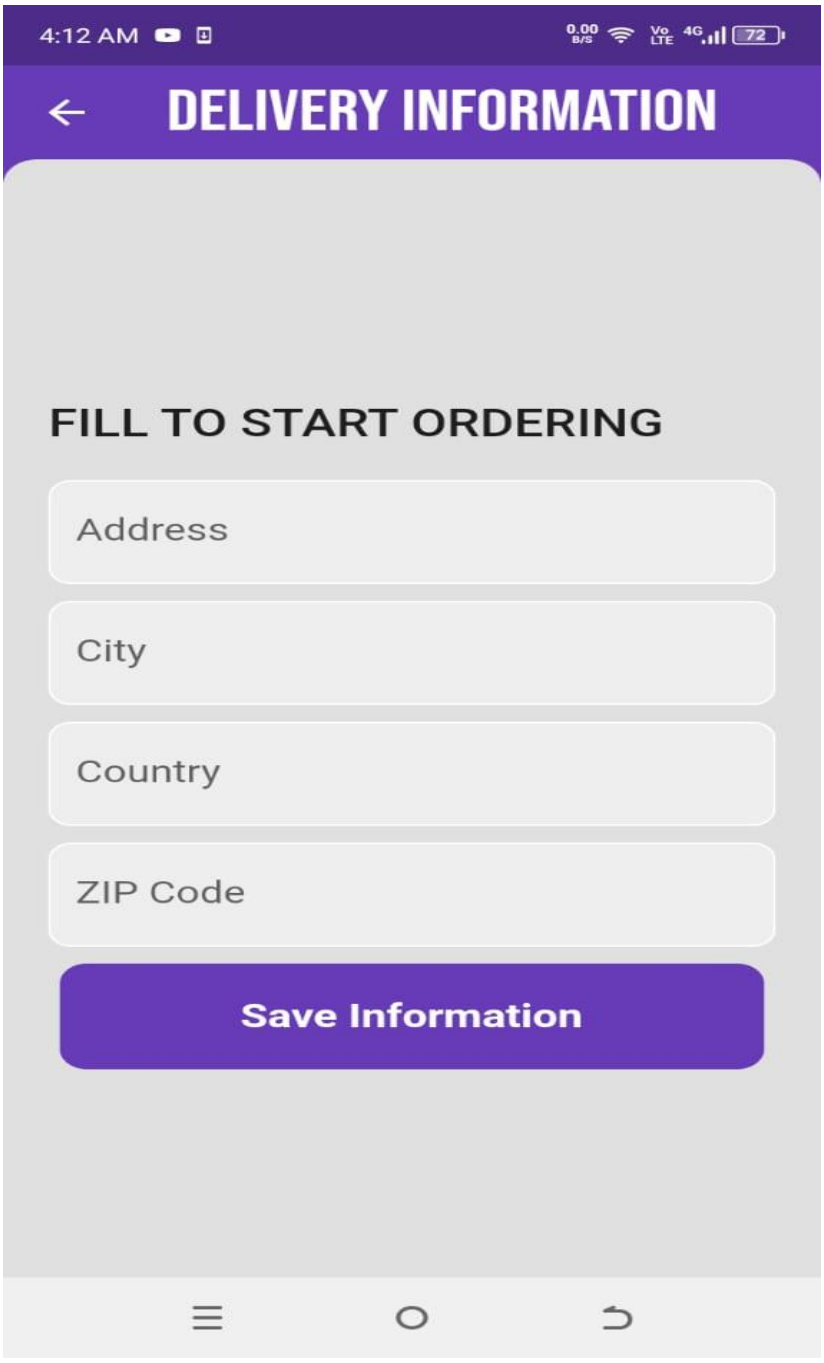
5.5.7 Profile Page - Mock Screen 7



5.5.8 WhishList - Mock Screen 8



5.5.9 Delivery Information - Mock Screen 9



6 References

- D. Gupta, "How to Build an Online Auction Application? Benefits, Features, Costs", 2024.
- G. Patil, "Online Auction App Development: A Complete Guide", India (Delhi), 2022.
- J. Tiwary, "Auction App Development: Unlocking the Future of Online Bidding", India (Mumbai), 30 October, 2023.
- M. Butt, "How to Create an Online Real-Time Bidding/Auction App", Mumbai (bendra), 2022.
- Ajay, "How to Develop an Application for Online Auctions? Features, Advantages, and Costs", Chennai, 2024.

Appendices

[Include supporting detail that would be too distracting to include in the main body of the document.]