ATORI-ATISANAL CORPORATE GIFTING

Project Report

Submitted in fulfilment of the

Requirements for the award of the Degree of

BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

By

Veman Shrinivas Chippa

SEAT NO: - 3023592

&

Mahima DarjeshKumar Choudhary

SEAT NO:- 3023593

Under the esteemed guidance of

Ms.Simin Khan

DEPARTMENT OF INFORMATION TECHNOLOGY

MANSI BHARAT GADA DEGREE COLLEGE OF COMMERCE

(Affiliated to University of Mumbai)

BHIWANDI - 421305

MAHARASHTRA

2024-2025

Abstract

Atori is a specialized corporate gifting Android application designed to streamline the process of discovering, managing, and delivering premium gifts for corporate clients. The platform offers two distinct modules: a **Client App** for users (corporate buyers or employees) and an **Admin App** for backend management.

The **Client App** enables businesses to explore a curated collection of gift products organized by categories such as digital gadgets, eco-friendly items, hampers, lamps, and more. Users can view detailed product information, add items to their cart, and place orders via a multi-step checkout process including shipping address, payment method, and final confirmation. The app supports both card and cash payment methods, and all orders are securely stored in Firebase Firestore.

The **Admin App** empowers administrators to manage the product catalog and oversee incoming orders in real time. Admins can view, edit, add, or delete products, and confirm or reject customer orders. Upon confirmation, the user is notified with an estimated delivery date. If an order is rejected, it is removed from the system and the user is promptly informed. Orders are displayed in reverse chronological order for efficiency.

Built with Kotlin and integrated with Firebase, the Atori app is optimized for performance, scalability, and user experience. It simplifies the complex logistics of corporate gifting by offering a user-friendly interface, streamlined operations, and powerful admin capabilities — making it an ideal solution for companies looking to enhance their corporate gifting workflow.

Acknowledgement

With immense gratitude and heartfelt appreciation, I take this opportunity to acknowledge the unwavering support and encouragement that contributed to the successful completion of my project titled **"Atori-Artisanal Corporate Gifting"**, submitted in partial fulfillment of the degree of **Bachelor of Science in Information Technology (B.Sc. IT).**

I am deeply thankful to **Ms. Simin Khan**, my respected project guide, for her constant guidance, insightful feedback, and motivating words throughout this journey. Her mentorship played a crucial role in shaping the direction of this project.

I also extend my sincere thanks to the **Principal of Mansi Bharat Gada Degree College of Commerce** for providing the infrastructure, resources, and academic environment necessary for the smooth execution of this work.

My heartfelt appreciation goes to **Ms. Praveen Ansari**, Head of the IT Department, for her support and encouragement, and to all the **faculty members and staff** of the department whose cooperation made a difference during challenging phases.

I would especially like to thank my dedicated teammate, **Ms.Mahima Choudhary DarjeshKumar** for their commitment, creativity, and excellent collaboration throughout every stage of this project.

Lastly, I am sincerely grateful to my **friends and family**, who stood by me with their constant encouragement, patience, and positivity.

This journey has been more than just a technical venture—it has been an experience filled with **learning, growth, and inspiration**, teaching me the true essence of teamwork, resilience, and determination.

CERTIFICATE

This is to certify that the project entited

"Exclusive Artisanal Gifting Solutions"

was undertaken at

Atori - The Artisanal Corporate Gifting Company

by

Mr. Veman Shrimvas Chippa and
Ms. Mahima Darjesh Kumar Choudhary

in partial fulfilment of the B.Sc. IT degree (Semester-VI).

This has been successfully carried out under the guidance of

Ms. Karishma Rajan
Co-Founder, Atori

Karishma Rajan

Karishma Rajan
Co-Founder, Atori

atorigifts@gmail.com | +91 8766686229 | Pune, Maharashtra

Oswal Shikshan & Rahat Sangh Sanchalit

MANSI BHARAT GADA DEGREE COLLEGE OF COMMERCE

(Affiliated to University of Mumbai)
(NAAC Accredited 'B' Grade& ISO 9001:2015 certified)

CERTIFICATE

This is to certify that **Mr.Veman Shrinivas Chippa** bearing **Seat. No:3023592** has satisfactorily completed the lab practical's prescribed for the **Subject: Software Testing And Quality Assurance** of **Semester-VI** in partial fulfillment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY (B.Sc. I.T.) from University of Mumbai for the academic year: **2024-2025**

Date:

Signature
Internal Examiner

Signature
HOD (IT)

College Seal

Signature
External Examiner

Signature
I/C Principal

Index

CHAPTER 1

INTRODUCTION

Atori -Artisanal Corporate Gifting

1.Introduction

- Atori is a corporate gifting company focused on providing businesses with high-quality, curated gift options. Corporate gifting plays a vital role in strengthening business relationships, improving client retention, and boosting employee morale. However, the process of selecting, customizing, and delivering thoughtful gifts can often be cumbersome and time-consuming.

- In response to this challenge, the Atori Android application aims to streamline the corporate gifting experience by offering a user-friendly platform where businesses can easily browse, customize, and order gifts tailored to their needs. The app will provide a seamless process for businesses to express their appreciation and build strong relationships with clients, partners, and employees.

- This project involves the design and development of the Atori mobile application, which will enhance the corporate gifting process by providing a simple, efficient, and personalized experience for users..

1.1 Background

- In today's competitive business environment, corporate gifting plays an essential role in building and maintaining relationships with clients, employees, and partners. It is an important gesture that shows appreciation and enhances goodwill. The traditional methods of corporate gifting, however, can be time-consuming and lack personalization. This project aims to develop a mobile application for **Atori**, which specializes in corporate gifting, to simplify the process of selecting, purchasing, and managing gifts for businesses

- The Atori app will provide users with an easy-to-use platform where they can browse through a curated selection of corporate gifts, customize orders based on their branding needs, and track deliveries efficiently.

1.2 Objectives

The main objective of this project is to design and develop an Android application for Atori that allows businesses to:

- Browse and select from a variety of corporate gift items.
- Customize their gift orders according to company preferences.
- Place orders and track them easily through a single platform.
- Provide secure payment options for hassle-free transactions.
- This project aims to improve the efficiency of corporate gifting processes by providing a modern and user-friendly application for Atori's clients

1.3 Purpose, Scope, and Applicability

1.3.1 Purpose

- The purpose of this project is to create a mobile application that addresses the challenges faced by businesses in corporate gifting. By developing an efficient, streamlined platform, Atori will be able to offer better service, enhance customer satisfaction, and provide a wider range of gift options to its clients. The app will make corporate gifting easier and more accessible to businesses of all sizes

1.3.2 Scope

The scope of this project covers the design, development, and deployment of the Atori Android application. Key features include

- A simple user interface for browsing gifts.
- An option for users to customize gift orders.
- A secure payment gateway for processing transactions.
- Real-time tracking and order management functionality.
- Admin tools for managing inventory, processing orders, and viewing reports.
- The app will be developed using JAVA for Android, with a backend supported by **Firebase** for data storage, authentication, and real-time notifications. The payment system will be integrated using **Razorpay** or a similar payment gateway**.**

1.3.3Applicability

- This project is applicable to all businesses that require corporate gifting solutions, from small startups to large corporations. The app can be used for:
- Client appreciation.
- Employee recognition and rewards.
- Celebrating business milestones and holidays such as Diwali, Christmas, and New Year

1.4 Achievements

Upon completion, the project is expected to deliver:

- A fully functional Android app with a user-friendly interface.
- Simplified processes for businesses to order and manage corporate gifts.
- A well-organized backend for managing product listings, order tracking, and payment processing.
- Improved user experience with real-time order updates and notifications.

These achievements will allow Atori to expand its services and reach more businesses, making the corporate gifting process more efficient.

1.5 Organization of Report

This report is organized into the following chapters:

- **Chapter 1: Introduction** – Provides an overview of the project, including the background, objectives, and scope.
- **Chapter 2: Survey of technologies –** Outlines the steps and technologies used in the development of the Atori application.
- **Chapter 3: Requirements and Analysis-**This chapter provides an overview of the requirements and analysis for the Atori app, highlighting the need for an efficient system to manage corporate gifting. By addressing the identified problems and implementing the specified requirements, the app aims to enhance user experience and streamline the gifting process.
- **Chapter4:System Design and Implementation–** Describes the system architecture, user interface design, and coding strategies employed in the project.

CHAPTER2:
SURVEY OF TECHNOLOGIES

2.1 Android Studio

- Android Studio is Android's official IDE. It is purpose-built for Google's Android operating system to accelerate your development and help you build the highest-quality apps for every Android device, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development.
- It is available for download on Windows, macOS and Linux based operating systems or as a subscription-based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development.

2.1.1 History

- Android Studio, the official integrated development environment (IDE) for Google's Android operating system, was first announced at the Google I/O conference in May 2013.
- It is based on JetBrains' IntelliJ IDEA software and replaced Eclipse Android Development Tools (ADT) as the primary IDE for Android app development. Android Studio is designed to offer developers an enhanced experience in building and testing Android applications across a range of devices.

2.1.2 Features

The following features are provided in the current stable version of Android Studio:

- Instant App Run
- Visual Layout Editor
- Fast Emulator
- Intelligence Code Editor
- Addition of New Activity as a Code Template
- Help to Build Up App for All Devices
- Help to Connect with Firebase

2.1.3 Version History

The following is a list of Android Studio's major releases:

Version	Release date
Giraffe (2022.3.1**)**	July 2023
Flamingo (2022.2.1)	April 2023
Arctic Fox (2020.3.1)	July 2021
4.2	May 2021
4.1	Oct 2020
4.0	May 2020
3.6	February 2020
3.5	August 2019
3.4	April 2019
3.3	January 2019
3.2	September 2018
3.1	March 2018
3.0	October 2017
2.3	March 2017
2.2	September 2016
2.1	April 2016
2.0	April 2016
1.5	November 2015
1.4	September 2015
1.3	July 2015
1.2	April 2015
1.1	February 2015
1.0	December2014

2.1.7 Advantages of Android Studio

- Comprehensive IDE
- Advanced Code Editor
- Rich Layout Editor
- Built-in Emulator
- Gradle Build System
- Firebase Integration
- .Version Control Support
- Performance Profiling
- Extensive Documentation and Community Support

2.1.8 Disadvantages of Android Studio

- High System Requirements
- Complexity for Beginners
- Slow Build Times
- Frequent Updates
- Verbose Code
- Limited Customization
- Dependency Management Issues
- Resource Consumption

2.2 Kotlin

- Kotlin is a modern, statically typed, general-purpose programming language that runs on the Java Virtual Machine (JVM) and is fully interoperable with Java. It is concise, expressive, and designed to improve code safety and developer productivity. Kotlin supports both object-oriented and functional programming paradigms, and it's officially supported by Google as a first-class language for Android development. Kotlin aims to reduce boilerplate code and improve readability without compromising performance.

2.2.1 History

- Kotlin was developed by JetBrains and officially released in 2011, with version 1.0 released in 2016. The language was named after Kotlin Island near St. Petersburg, Russia. Kotlin was created to address the shortcomings of Java, particularly verbosity and lack of modern features. In 2017, Google announced official support for Kotlin on Android, making it a popular choice for mobile development.

2.2.2 Features

- Kotlin is known for its modern features, concise syntax, and strong interoperability with Java. Some of its key features include:

  - **Object-Oriented and Functional**: Kotlin supports both OOP and functional styles, offering great flexibility.
  - **Interoperability:** Kotlin is 100% interoperable with Java, allowing developers to use existing Java libraries and frameworks.
  - **Null Safety**: Kotlin helps eliminate the NullPointerException by making all types non-nullable by default.
  - **Concise**: Reduces boilerplate code significantly, leading to shorter and cleaner programs.
  - **Smart Casts**: Kotlin automatically casts types when possible, making code simpler and safer.
  - **Extension Functions**: Allows developers to add new functionalities to existing classes without modifying their source code.
  - **Coroutines:** Kotlin provides built-in support for asynchronous programming using coroutines for better performance.

2.2.3 Advantages

• Concise and Expressive
• Null Safety
• Interoperable with Java
• Functional Programming Support
• Coroutines for Asynchronous Code
• Smart Casts
• Extension Functions
• Modern and Readable Syntax
• Official Android Support by Google
• Easy to Learn for Java Developers

2.2.4 Principles of Kotlin

- **Encapsulation**: Kotlin supports data hiding and access control using classes and visibility modifiers.
- **Inheritance:** Classes can inherit properties and functions from other classes using the open and override keywords.
- **Polymorphism**: Like Java, Kotlin allows method overloading and overriding to provide polymorphic behavior.

- **Abstraction**: Kotlin supports both abstract classes and interfaces to hide complex implementation details.

2.2.5 Limitations

• Slower Compilation Speed in Some Cases Compared to Java

• Smaller Developer Community Compared to Java (but growing rapidly)

• Some Libraries or Tools May Have Limited Kotlin Support

• Requires Learning Curve for Java Developers Adapting to Kotlin's Functional Features

• Still Runs on the JVM, So Can't Directly Access Hardware Resources Like RAM or Processor

2.3 Firebase

- Firebase is a mobile and web application development platform developed by Firebase, Inc. in 2011, then acquired by Google in 2014. As of October 2018, the Firebase platform has 18 products, which are used by 1.5 million apps.

2.3.1 History:

- Firebase evolved from Envolve, a prior startup founded by James Tamplin and Andrew Lee in 2011. Envolve provided developers an API that enables the integration of online chat functionality into their websites. After releasing the chat service, Tamplin and Lee found that it was being used to pass application data that weren't chat messages. Developers were using Envolve to sync application data such as game state in real time across their users. Tamplin and Lee decided to separate the chat system and the real-time architecture that powered it. They founded Firebase as a separate company in September 2011 and it launched to the public in April 2012.

2.3.2 Features

- **Realtime Database**: Firebase allows developers to store and sync data between users in real-time.
- **Firebase Authentication**: Provides easy integration for authentication services using email, phone numbers, Google, Facebook, and more.

- **Cloud Firestore:** A scalable, flexible database solution for apps that require cloud storage.
- **Firebase Analytics:** Offers powerful tracking and reporting tools for understanding user behavior in apps.
- **Firebase Cloud Messaging (FCM):** A service for sending push notifications and messages to users on multiple platforms.

2.3.3 Advantages

- Real-time Database Helps to Store and Synchronize Data
- Firebase has Become Smarter with Google Analytics
- Firebase Offers Facility of Crash Reporting to Fix Bugs Quickly
- Fast and Secured Web Hosting
- Firebase Authentication
- Firebase Allows the Content Storage with Ease
- Developers have the Accessibility of Machine Learning
- Send Notifications and Messages to Targeted Audiences
- Send the App Recommendation through Dynamic Links

2.3.4 Limitations:

- You will need to write your own codes in order to query the data, and the query flexibility is quite limited.
- For example, it is not simple to apply a where condition with order by certain attributes.
- Another limitation (or perhaps strength) is that the data structure is more towards horizontal.
- As a result, you will need to store duplicated data under separate nodes.

2.3.5 Cloud Firestore:

- Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud. Like Firebase Realtime Database, it keeps your data in sync across client apps through realtime listeners and offers offline support for mobile and web so you can build responsive apps that work regardless of network latency or Internet connectivity. Cloud Firestore also offers seamless integration with other Firebase and Google Cloud products, including Cloud Functions.

CHAPTER 3

REQUIREMENTS AND ANALYSIS

3.1 Problem Definition

- The corporate gifting landscape often suffers from inefficiencies, including a lack of personalization, difficulty in selecting appropriate gifts for different occasions, and challenges in managing orders for multiple recipients. Businesses need a streamlined solution that facilitates thoughtful gifting, enhances client and employee relationships, and reflects their brand identity. The Atori Corporate Gifting app aims to address these issues by providing a user-friendly platform that simplifies gift selection, customization, and order management.

Problem Statement

Users face challenges in selecting, personalizing, and delivering gifts efficiently due to a lack of tailored solutions and fragmented processes.

Key Challenges

1. **Overwhelming Choices**:
    - Users often struggle to find the right gift from a vast array of options available online, leading to decision fatigue.
2. **Lack of Personalization**:
    - Many gifting platforms offer limited personalization options, resulting in generic gifts that do not reflect the sender's sentiment or brand identity.
3. **Inefficient Order Processes**:
    - The process of selecting a gift, providing shipping details, and making payments can be cumbersome and time-consuming.
4. **Tracking and Communication Gaps**:
    - Users often lack real-time updates on their orders, leading to uncertainty about delivery statuses.

3.2 Requirements Specification

Functional Requirements

1. **User Registration and Authentication**:
    - Users must be able to create an account and log in securely

2. Gift Catalog Browsing:

- Users can browse gifts categorized by occasion, recipient type, and price range.

3. Personalization Options:

- Users can customize gifts with messages, logos, and packaging options.

4. **Order Management:**

- Users can add items to a cart, place orders, and track delivery status.

5. **Occasion Reminders:**

- The app will send notifications for upcoming gifting occasions.

6. **Corporate Account Features:**

- Businesses can manage multiple users, track budgets, and oversee orders.

7. **Customer Support:**

- Users can access support through FAQs, chat, or email.

Non-Functional Requirements

1. **Performance:**

- The app should load within 3 seconds and handle up to 10,000 simultaneous users.

2. **Usability:**

- The app must provide an intuitive interface, ensuring that users can navigate and complete tasks with minimal training.

3. **Security:**

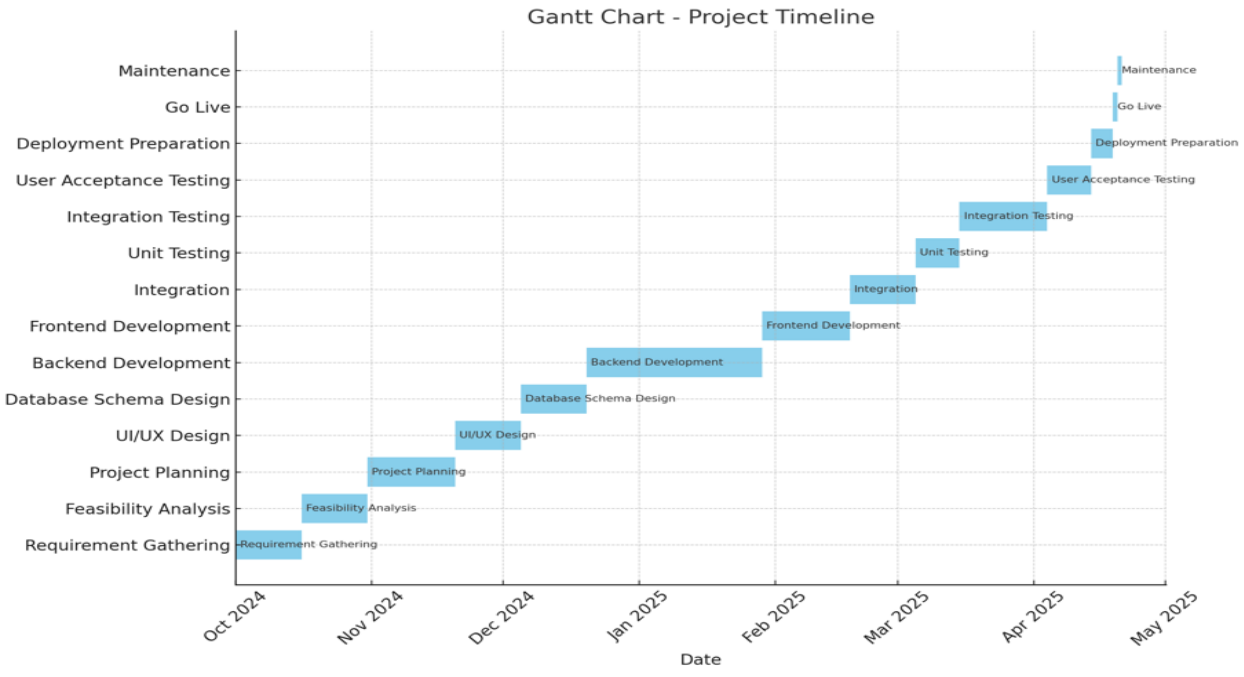
- User data must be encrypted, and secure payment processing must be implemented.

4. **Compatibility:**

- The app should be compatible with Android versions 8.0 (Oreo) and above.

3.3 Planning and Scheduling

Task	Start Date	Duration	Dependencies
Requirement Gathering	Oct 01, 2024	15D	None
Feasibility Analysis	Oct 16, 2024	15D	Requirement Gathering
Project Planning	Oct 31, 2024	20D	Feasibility Analysis
UI/UX Design	Nov 20, 2024	15D	Project Planning
Database Schema Design	Dec 05, 2024	15D	UI/UX Design
Backend Development	Dec 20, 2024	40D	Database Schema Design
Frontend Development	Jan 29, 2025	20D	Backend Development
Integration	Feb 18, 2025	15D	Frontend Development
Unit Testing	Mar 05, 2025	10D	Integration
Integration Testing	Mar 15, 2025	20D	Unit Testing
User Acceptance Testing	Apr 04, 2025	10D	Integration Testing
Deployment Preparation	Apr 14, 2025	5D	User Acceptance Testing
Go Live	Apr 19, 2025	1D	Deployment Preparation
Maintenance	Apr 20, 2025	Ongoing	Go Live



3.4 Software and Hardware Requirements

Software Requirements

- **Operating System:**Android 8.0 (Oreo) and above
- **Development Environment:**Android Studio
- **Programming Language:** Kotlin
- **Database:**Firebase
- **Payment Gateway Integration:**PayPal, Stripe, or similar

Hardware Requirements

- **Devices:** Any Android device running Android 8.0 or above
- **Development Machines:** Minimum specifications:
- **Processor:** Intel i5 or equivalen
- **RAM**: 8 GB
- **Storage:** 256 GB SSD
- ✓ **Other software used for project report: MS Word, Lucidchart**

3.5 Preliminary Product Description

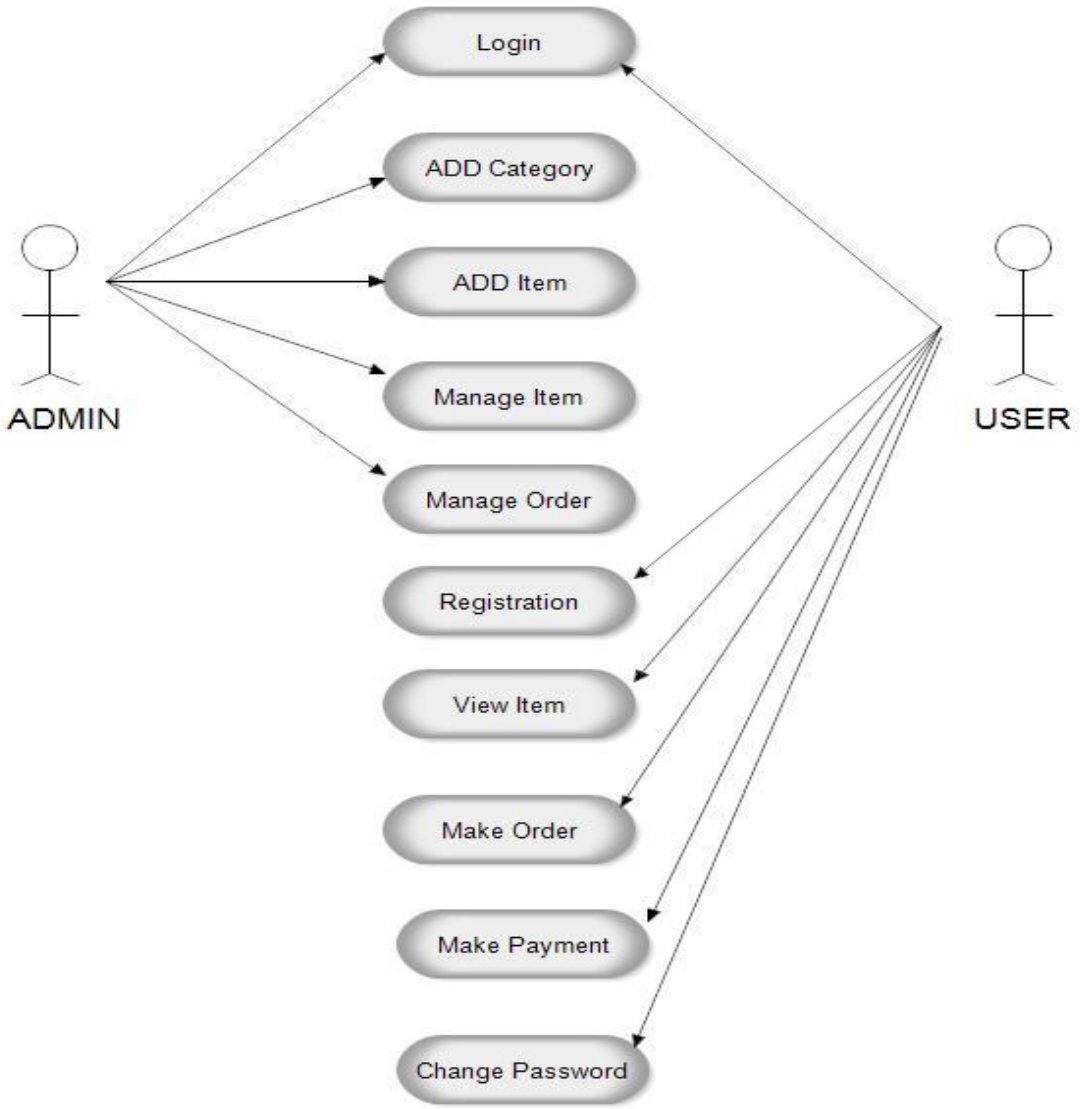
- The Atori Corporate Gifting app is a mobile platform that enables businesses to easily select, customize, and manage corporate gifts.
- The app features a diverse catalog of high-quality gifts, allowing users to filter options based on occasion, recipient, and brand identity. It offers personalization features to ensure that each gift reflects the user's sentiment.
- Users can place orders, track deliveries, and receive reminders for upcoming occasions. The app supports corporate accounts, enabling businesses to manage multiple users and budgets efficiently. With an intuitive interface and robust performance, the Atori app aims to transform the corporate gifting experience.

3.6 Conceptual Models

In the realm of software development, a conceptual model may be used to represent relationships of entities within a database. A conceptual model can easily represent abstract concepts of the relationships between objects in the system, such as Users and their relationships to Accounts.

❖ USECASE DIAGRAM:

The Use Case Diagram for Atori Android app will depict the interactions between the users (actors) and the system (app). It shows the primary functions users can perform within the app and the system's response.

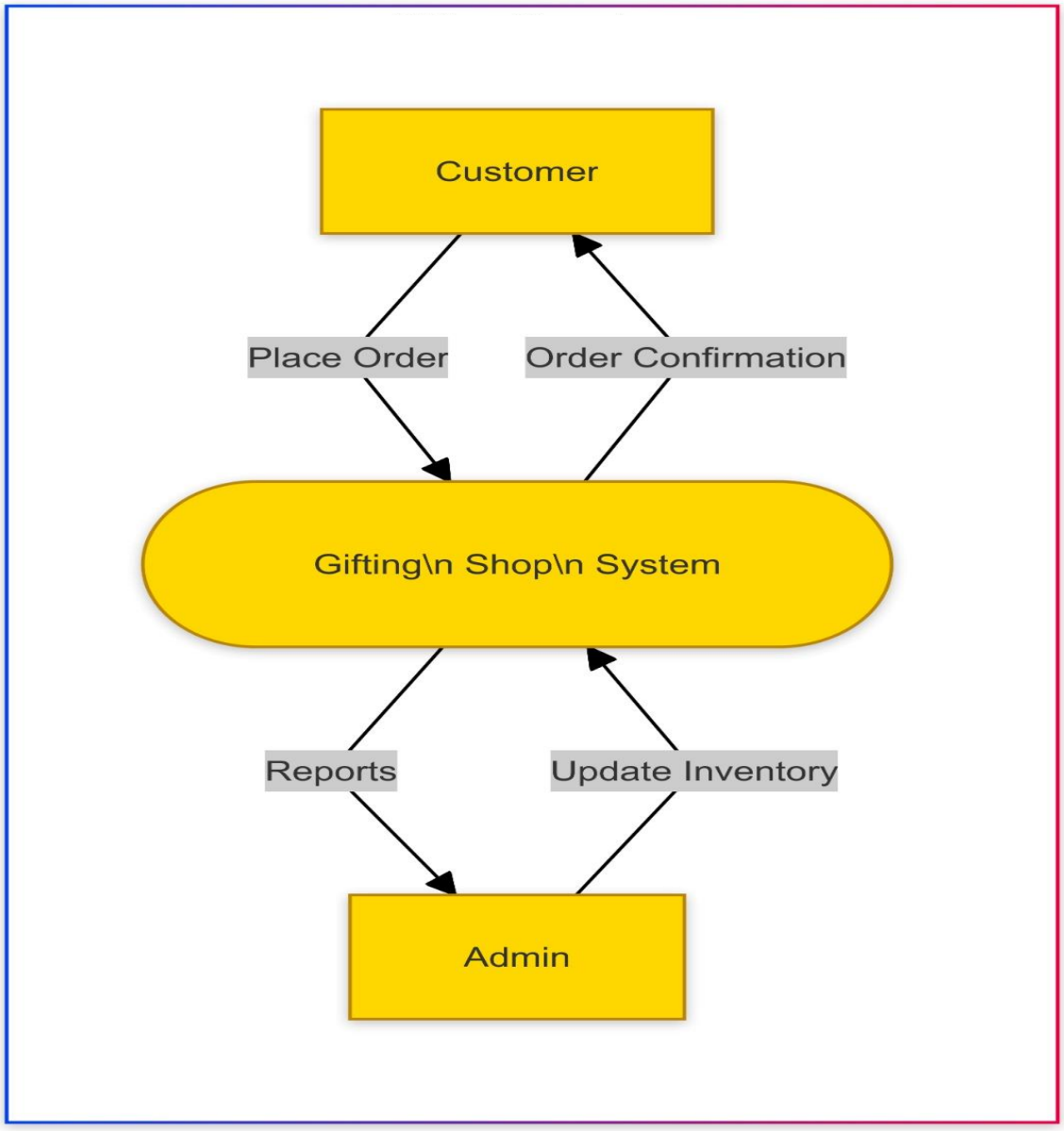


DATA FLOW DIAGRAM:

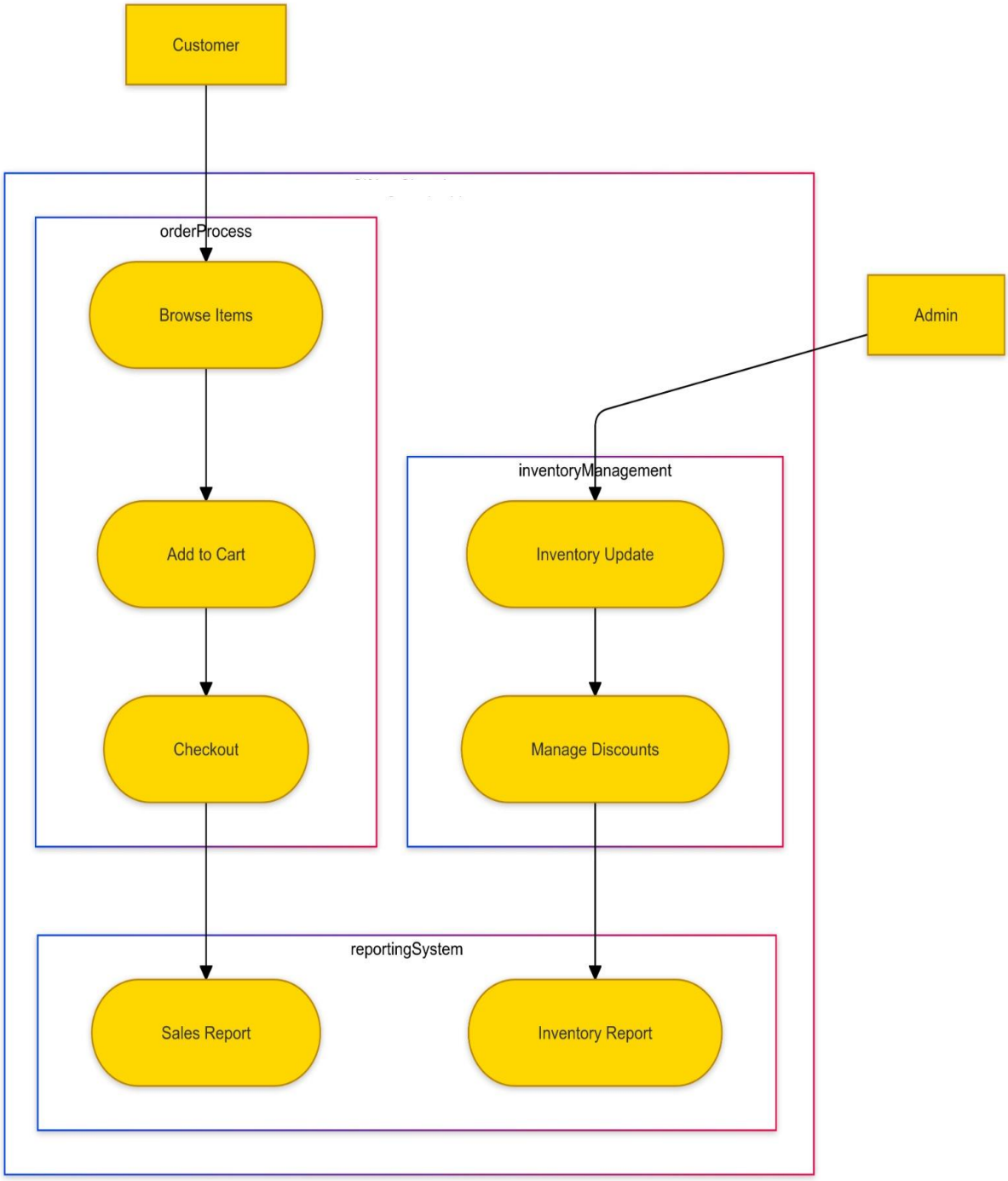
- A data flow diagram shows the way information flows through a process or system. It includes data inputs and outputs, data stores, and the various sub processes the data moves through.

DATA FLOW DIAGRAM

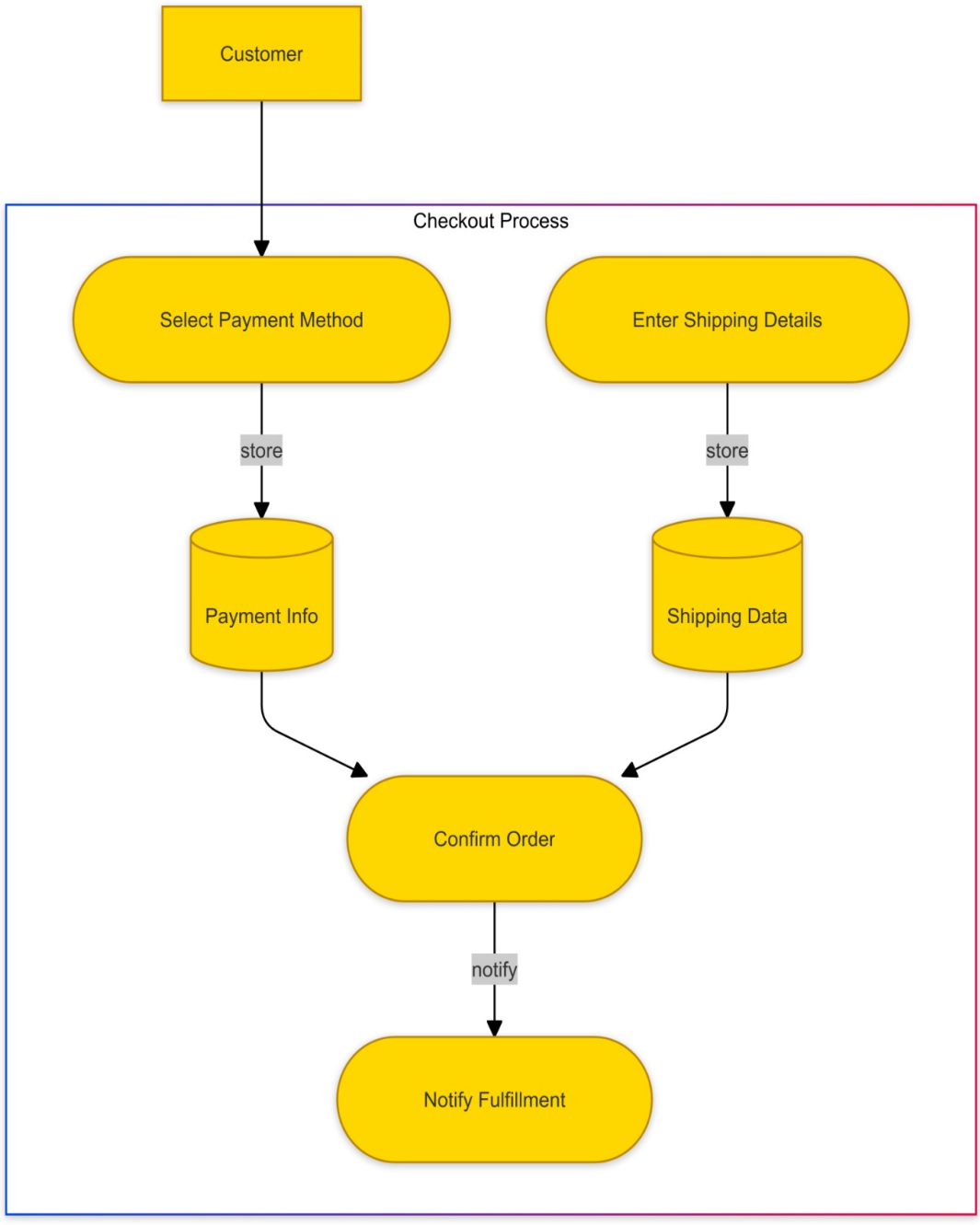
Level -0



Level 1

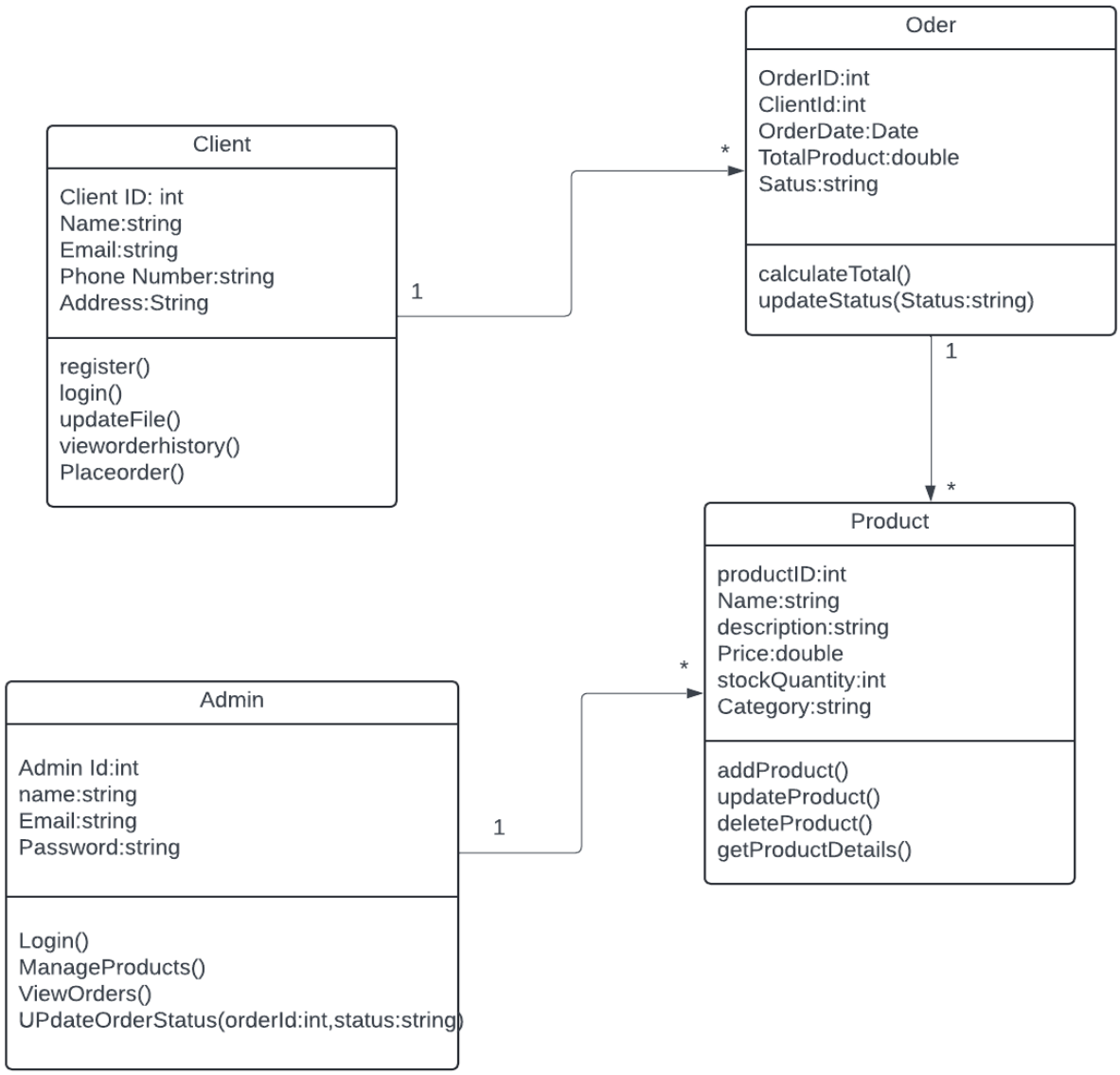

Customer
orderProcess
Browse Items
Add to Cart
Checkout
Admin
inventoryManagement
Inventory Update
Manage Discounts
reportingSystem
Sales Report
Inventory Report

Level 2


Customer
Checkout Process
Select Payment Method
Enter Shipping Details
store
store
Payment Info
Shipping Data
Confirm Order
notify
Notify Fulfillment

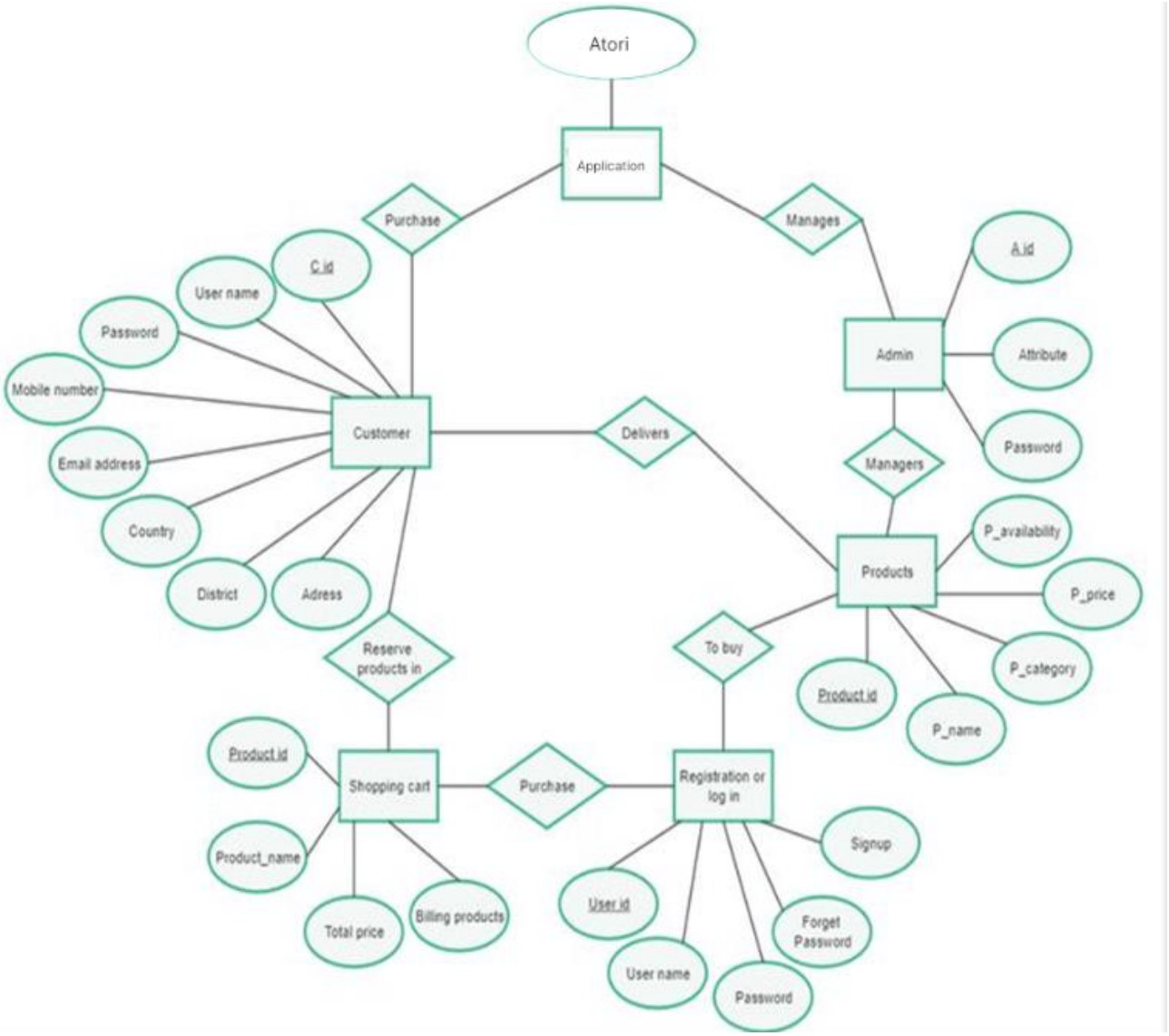
❖ CLASS DIAGRAM :

- Class diagram is a static diagram. It represents the static view of an application.



❖ ER Diagram (Entity Relationship Diagram):

- An **Entity-Relationship Diagram (ERD)** is used to model the database structure of a system by representing entities, their attributes, and relationships between them. For Atori Android app, the ER diagram will show how data entities like Users, Products, Orders, Payments, and others are related to one another.



CHAPTER 4:
SYSTEM DESIGN

Chapter 4: System Design

This chapter outlines the system design for the Atori App, detailing the fundamental modules, data architecture, procedural design, user interface considerations, security measures, and test case design.

4.1 Basic modules

What is module?

- A module is a software component or part of a program that contains one or more routines. One or more independently developed modules make up a program.

Module 1

❖ **Client module**

- The Client Sub module is designed to provide end-users (clients) with a seamless and intuitive experience for browsing, selecting, and managing gifts. It allows clients to personalize gifts, place orders, track their status, and manage their profiles.

Sub-module 1:

➢ **User Registration and Login**

- **Registration Form:** Collects user details (name, email, password).
- **Login Interface:** Allows existing users to log in securely.

Sub-module 2:

➢ **Profile Management**

- **View Profile:**Clients can view their profile information.
- **Edit Profile:**Allows clients to update their information (e.g., name, email, password).

Sub-module3:

- **Product Browsing**
  - **Home Screen**: Displays featured, sale, and new products using multiple adapters.
  - **Category Browsing:** Products loaded dynamically by category from Firestore (with fallback to local JSON).
  - **Product Details:** Tapping a product opens a detail screen with full info.

Sub-module 4:

- **Cart and Checkout**
  - **BagFragment:** Shows cart items with quantity +/− and total calculation.
  - **Checkout Flow:**
    - Step 1: Shipping Address (name, phone, address, city, state, zip).
    - Step 2: Payment Method (Cash on Delivery or Card).
    - Step 3: Order Confirmation (saves full order in Firestore).
  - **Cart Management:** Items added to cart via Room database with support for updates and removal.

Sub-module 5:

- **Order Management**
  - **Add to Cart:** Clients can select gifts and add them to their cart.
  - **Review Cart:** Allows users to view selected items, update quantities, or remove items.
  - **Checkout Process:** Facilitates payment and order confirmation.
    - **Place Order:** Order placed with complete details (user info, product names, amount, address).
    - **Firestore Storage:** Orders saved to orders collection for admin access.

Sub-module 6:

- **Product Video Module**
  - **Video Listing:** Users can view product-related or promotional videos.
  - **Full Screen Player:** Videos are viewable in full screen for better engagement

Sub-module 7:

- **User Favorites**
  - **Add/remove** products to favorite list.
  - Stored locally using SharedPreferences.

Sub-module 8:

- **About Us & Contact**
  - Describes Atori's mission and values.
  - Contact details for **Karishma and Sakshi**.
  - **Clickable phone numbers and email/website buttons.**
  - Static design with ScrollView and LinearLayout.

Sub-module 9:

- **Recommendations**
  - Similar product suggestions based on selected category.
  - Shown in ProductDetailsActivity using local JSON or Firestore.

Module 2

- **Admin module**
  - The Admin module is designed to provide administrators with tools to manage the application effectively. It enables them to oversee user accounts, manage inventory, process orders, and generate reports.

Sub-module 1:

- **Admin Login**

- **Login:** Admin logs in securely to access the backend (Firebase Auth).

Sub-module 2:

- **Product Management**
  - **View Products**: All products from all categories shown in a single RecyclerView.
  - **Add Product:** Add new products with fields (name, price, image, brand, etc.).
  - **Edit Product:** Update product details using Firestore document ID.
  - **Delete Product:** Remove a product permanently from Firestore.

Sub-module .3

- **Order Management**
  - **View Orders:** Admins can view all orders placed by clients.
  - **Update Order Status:** Change the status of orders (e.g., pending, shipped, confirm).
  - **Manage Returns and Refunds:** Process return requests and issue refunds as needed.

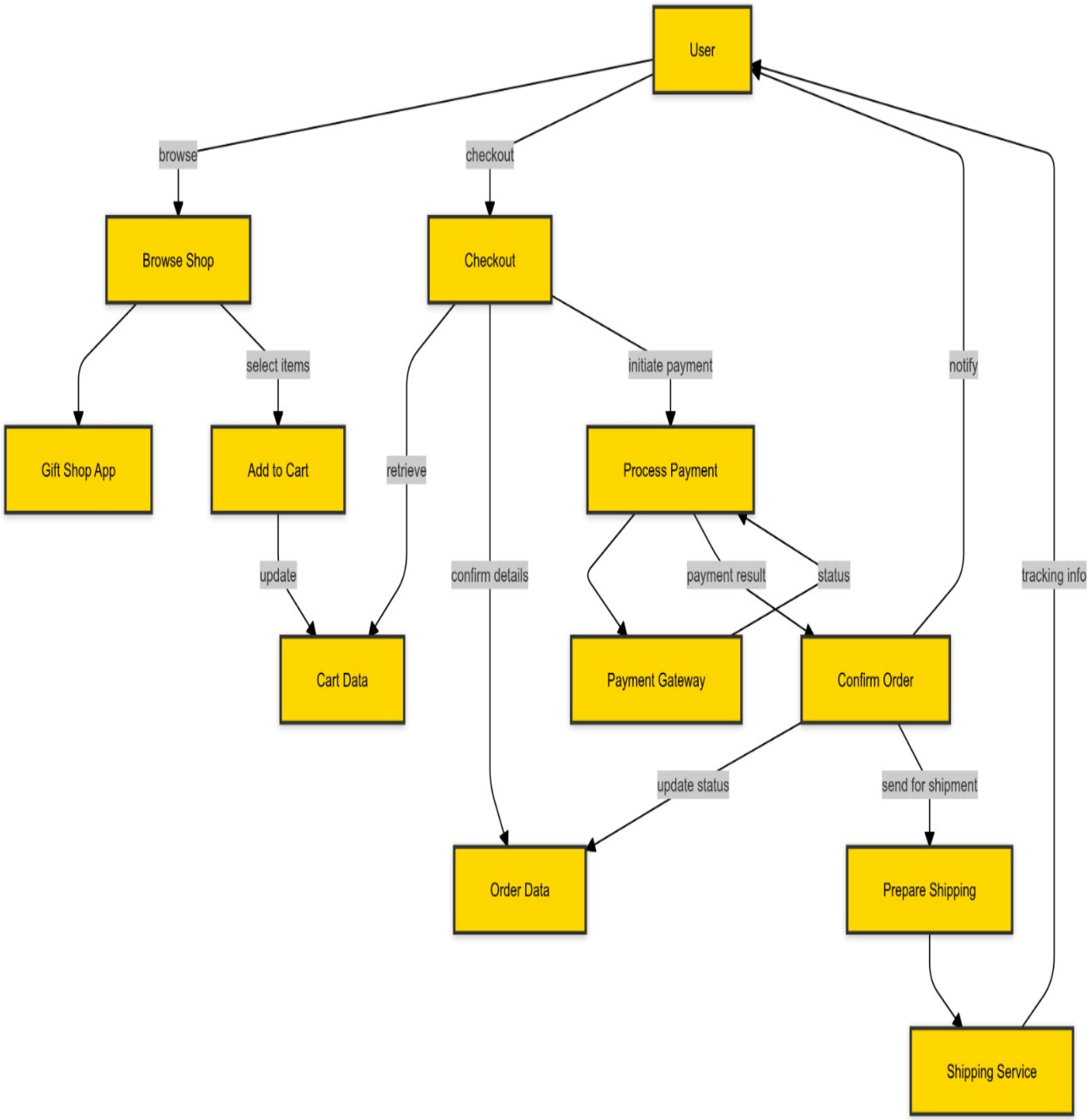
Sub-Module 4

- **User & Order Overview**
  - **View Total Users**: Admin can see a complete list of all registered users.
  - **View Total Orders:** Admin has access to all orders placed by users

4.2 Data Design

Data flow diagram

A Data Flow Diagram (DFD) is a visual representation that illustrates how data moves through a system, showing the interactions between different entities, processes, and data stores

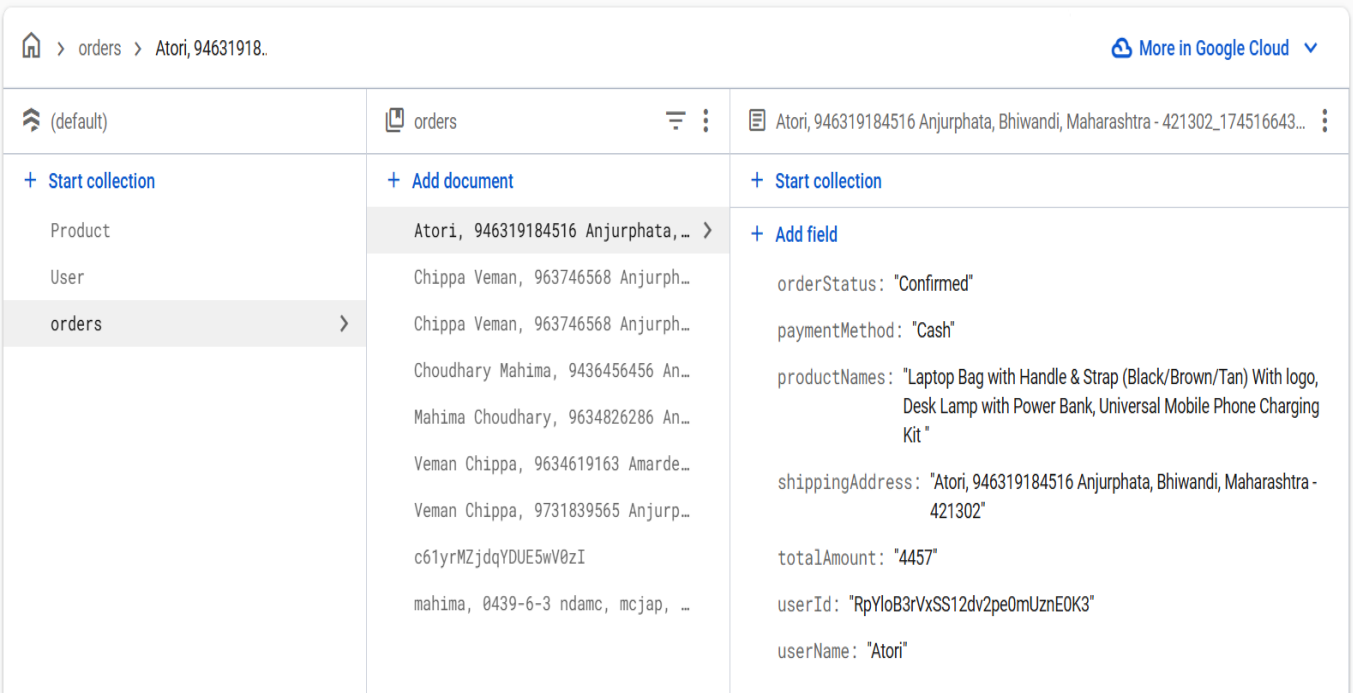


4.2.1 Schema Design

- A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

Structure of Firebase Firestore Database Schema:

- Cloud Firestore is schemaless, so you have complete freedom over what fields you put in each document and what data types you store in those fields. Documents within the same collection can all contain different fields or store different types of data in those fields.



4.2.2 Data Integrity and Constraints

To ensure data integrity, the following constraints will be implemented:

- **Primary Key Constraints:** Ensure uniqueness for identifiers such as `user_id`, `gift_id`, `order_id`, and `payment_id`.

- **Foreign Key Constraints:** Maintain referential integrity between related tables (e.g., linking `user_id` in Orders to `Users` table).

- **Unique Constraints:** Enforce unique values for `username` and `email` in the Users table to prevent duplicates.

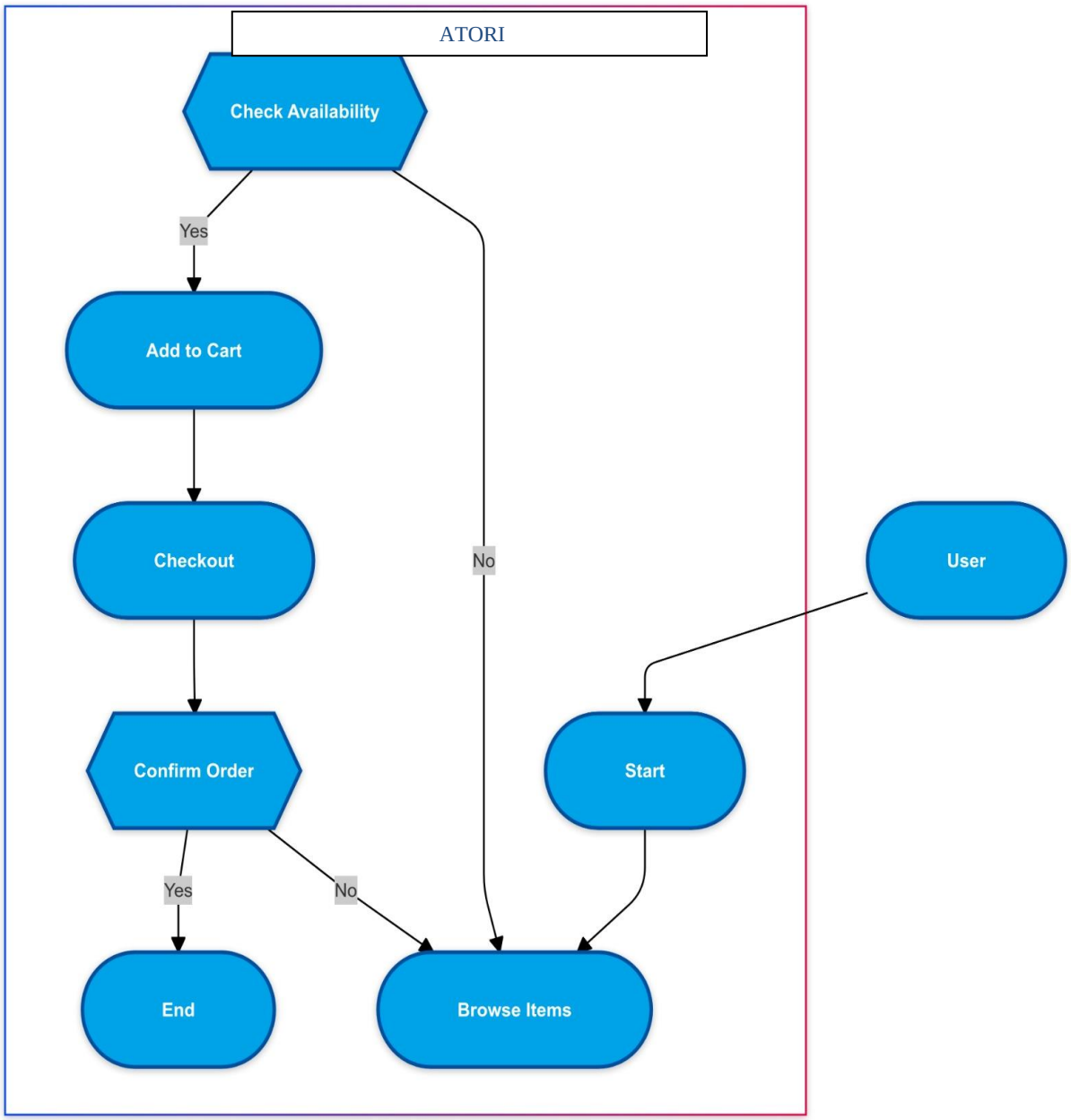
- **Not Null Constraints**: Ensure essential fields like `gift_name`, `price`, and `order_date` are not left empty

4.3 Procedural Design

The procedural design defines the logic and flow of operations within the app.

4.3.1 Logic Diagram

A high-level logic diagram illustrates the interaction between sub modules:



4.3.2 Algorithm design

1. Admin Login

Input:Admin credentials (username and password)
Output: Admin access
Steps:

1. **Start**
2. **Display Login Screen:**
    - **Input fields for username and password.**
3. **On Submit:**
    - **Validate input fields (ensure fields are not empty).**
    - **Send a login request to the server with the admin credentials.**
4. **Receive Response:**
    - **If valid credentials:**
        - **Grant access to the admin dashboard.**
        - **Navigate to Admin Dashboard Screen.**
    - **If invalid credentials:**
        - **Display an error message.**
        - **Allow retry.**

2. Admin Access

Input:Admin actions
Output: Access to admin functionalities
Steps:

1. **On Admin Dashboard:**
    - **Display options: Manage Gifts, View Orders, Manage Users, View Analytics.**
2. **On Selecting an Option:**
    - **Execute the corresponding function:**

- **Manage Gifts: Add, edit, or delete gifts.**
- **View Orders: Display order history and details.**
- **Manage Users: Add, edit, or delete user accounts.**
- **View Analytics: Show sales and engagement statistics.**

3. **Logout Option: Allow admin to log out and return to the login screen.**

3. Client Login

Input: Client credentials (username and password)
Output: Client access
Steps:

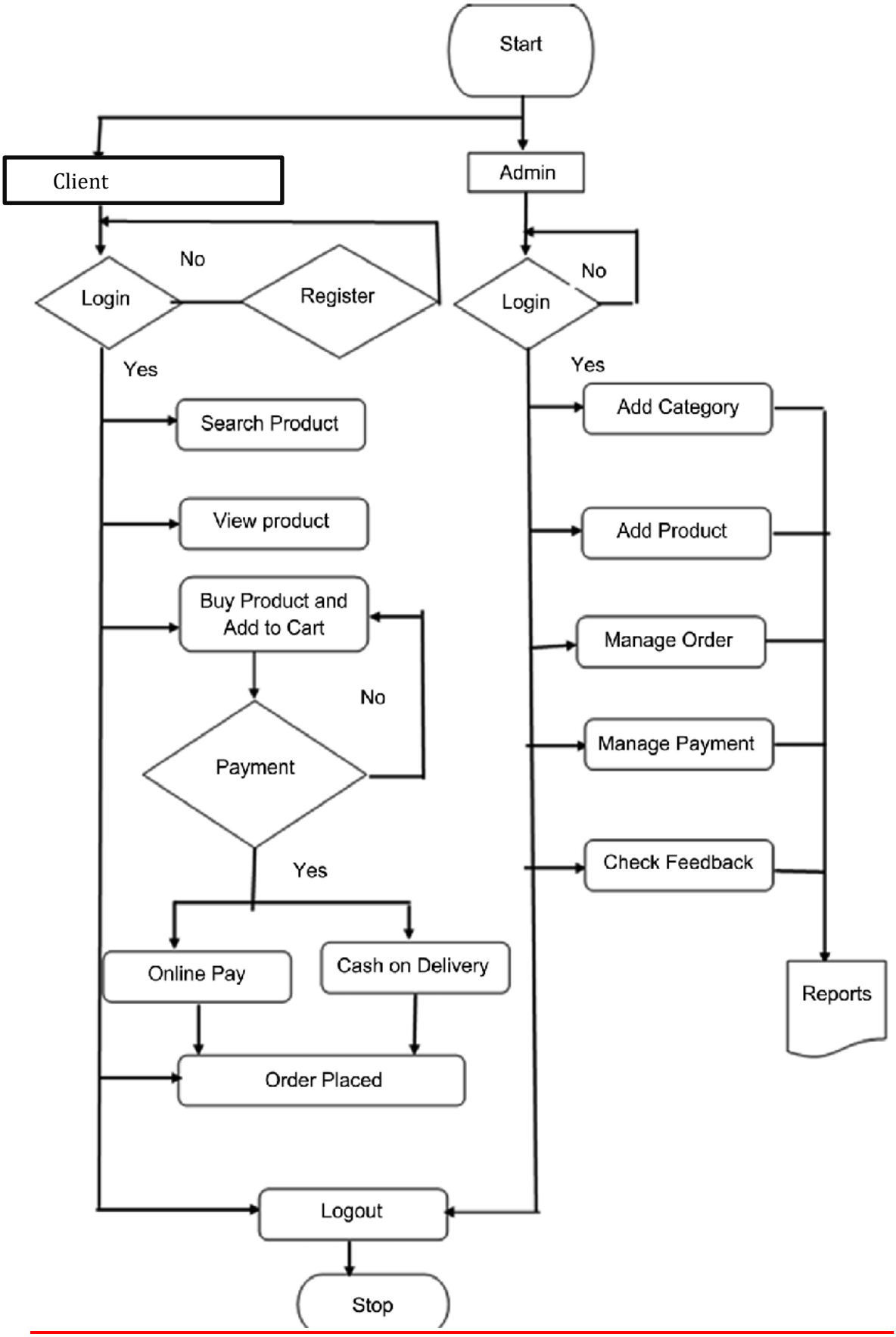
1. **Start**
2. **Display Login Screen:**
   - **Input fields for username and password.**
3. **On Submit:**
   - **Validate input fields (ensure fields are not empty).**
   - **Send a login request to the server with the client credentials.**
4. **Receive Response:**
   - **If valid credentials:**
     - **Grant access to the client dashboard.**
     - **Navigate to Client Dashboard Screen.**
   - **If invalid credentials:**
     - **Display an error message.**
     - **Allow retry.**

4. Client Access

Input: Client actions
Output: Access to client functionalities

Steps:

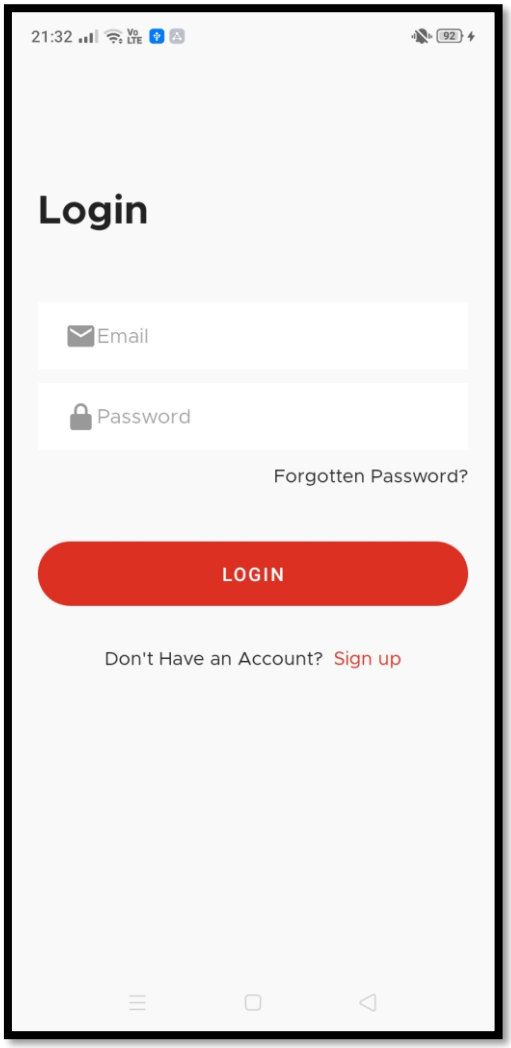
1. **On Client Dashboard:**
    - **Display options: View Gifts, Place Orders, View Order History, Update Profile.**
2. **On Selecting an Option:**
    - **Execute the corresponding function:**
        - **View Gifts: Display available gifts with details.**
        - **Place Orders: Allow clients to select gifts and complete the order.**
        - **View Order History: Display past orders with details.**
        - **Update Profile: Allow clients to update their personal information.**
3. **Logout Option: Allow clients to log out and return to the login screen.**


Start
Client
Admin
Login
No
Register
Login
No
Yes
Yes
Search Product
View product
Buy Product and Add to Cart
Payment
No
Yes
Online Pay
Cash on Delivery
Order Placed
Logout
Stop
Add Category
Add Product
Manage Order
Manage Payment
Check Feedback
Reports

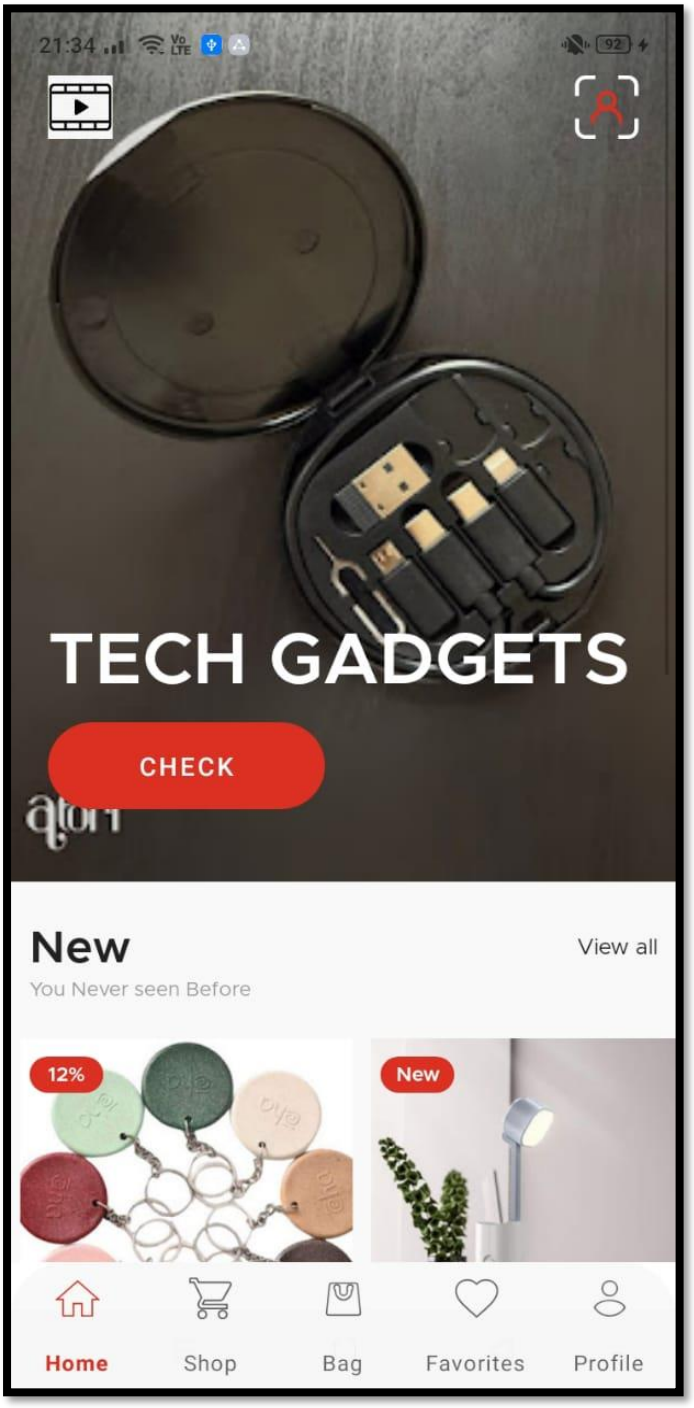
4.4 User Interface Design

Atori SplashScreen

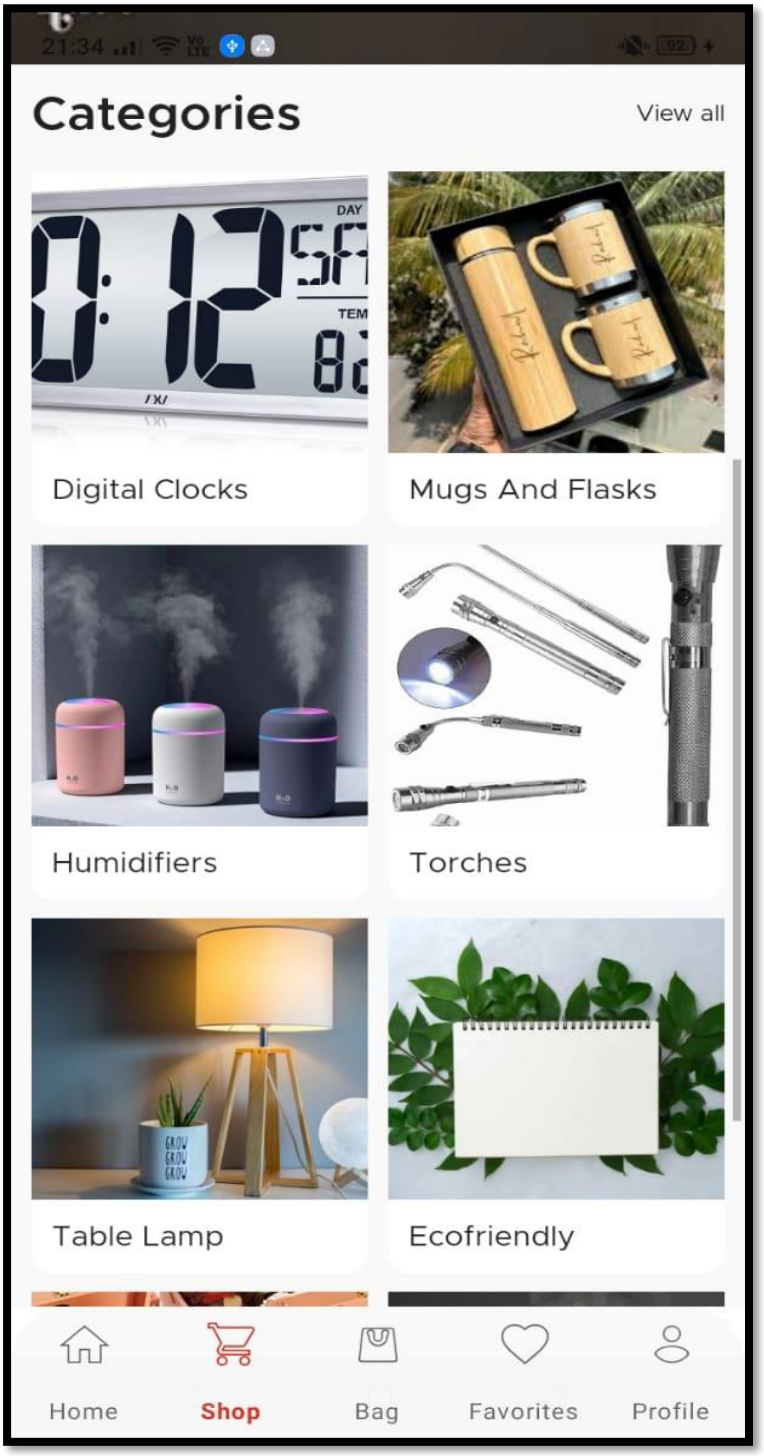
LoginPage



HomePage


21:34
TECH GADGETS
CHECK
New
View all
You Never seen Before
12%
New
Home
Shop
Bag
Favorites
Profile

Shopping page



4.5 Security Issues

- **Security** is critical in protecting user data and transactions. Key measures include:
- **Data Encryption**: Secure sensitive data, such as passwords and payment information, using encryption standards (e.g., AES).
- **Secure Authentication**: Implement multi-factor authentication (MFA) for user accounts.
- **Regular Security Audits**: Conduct routine assessments to identify and mitigate vulnerabilities.
- **Compliance**: Adhere to relevant data protection regulations (e.g., GDPR, PCI-DSS).

4.6 Test Cases Design

1. Functional Test Cases

- **TC01: User Registration**
    - Objective: Verify that a new user can register successfully.
    - Input: Valid user details (name, email, password).
- **TC02: User Login**
    - Objective: Verify that a user can log in with valid credentials.
    - Input: Valid email and password.
- **TC03: Browse Gift Catalog**
    - Objective: Ensure users can browse available gifts.
    - Input: Navigate to the gift catalog.
- **TC04: Place Order**
    - Objective: Verify that a user can place an order for gifts.
    - Input: Select gifts, provide shipping details, and payment information.
- **TC05: Payment Processing**
    - Objective: Check payment processing functionality.
    - Input: Valid payment information.

2. Usability Test Cases

- **TC06: User Interface Navigation**
    - Objective: Ensure that users can navigate through the app easily.
    - Input: Use various navigation buttons.
- **TC07: Gift Search Functionality**
    - Objective: Verify the search feature for gifts.
    - Input: Search for a specific gift.

3. Performance Test Cases

- **TC08: Load Time**
    - Objective: Measure the app's load time during peak usage.

- Input: Open the app during peak hours.

- **TC09: Concurrent Users**
  - Objective: Test the app's performance with multiple concurrent users.
  - Input: Simulate multiple users accessing the app simultaneously.

4. Security Test Cases

- **TC10: Data Encryption**
  - Objective: Verify that sensitive user data is encrypted.
  - Input: Examine the app's data storage.
  - Expected Result: User data should be encrypted and not readable in plain text.

- **TC11: Session Management**
  - Objective: Check session timeout functionality.
  - Input: Log in and leave the app idle.
  - Expected Result: User should be logged out after a predefined period of inactivity.

Test Case Format

Here's a simple format you can use for each test case:

Test Case ID	Description	Preconditions	Test Steps	Expected Result	Actual Result	Status
TC01	User Registration	App installed	1. Open app 2. Click register...	Account created		Pass/Fail
TC02	User Login	User is registered	1. Open app 2. Enter credentials...	User logged in		Pass/Fail

CHAPTER 5
IMPLEMENTATION AND TESTING

5.1 IMPLEMENTATION APPROACHES

Tools and standards used for implementation

The implementation of the Atori app followed a modular approach, dividing the system into client and admin modules. The client module allows users to register, browse gifts, personalize products, and track orders. The admin module provides functionalities for managing users, gifts, and orders. Kotlin was chosen as the primary programming language for its concise syntax and full interoperability with Java. Firebase served as the backend platform, offering real-time database, authentication, and storage services.

Implementation included structured coding using the MVVM architecture. ViewModels separated UI logic from business logic, improving testability and maintenance. Firebase Firestore was chosen for its real-time capabilities, and Firebase Authentication was used for secure user access.

For Atori and Admin app, I have used :

Android Studio

- Official IDE for Android development by Google.
- Based on JetBrains' IntelliJ IDEA.
- Specially designed for building Android applications.
- Provides tools for designing UI, writing code, testing, and debugging.

Kotlin

- Modern, statically-typed programming language developed by JetBrains.
- Fully interoperable with Java.
- Officially supported language for Android development.
- Concise syntax, null safety, and coroutine support make it ideal for mobile development.

XML (Extensible Mark-up Language)

- Used to describe and structure data in Android apps.
- Custom tags are defined by the developer.
- Plays a crucial role in defining UI layouts and resources.

Types of XML Files in Android:

- **Layout XML Files**
  - Define the UI structure of the app.
  - Used in activities and fragments to display content.
- **Manifest XML File (AndroidManifest.xml)**
  - Declares application components: activities, services, broadcast receivers.
  - Lists permissions and the app's package name.
- **Strings XML File (strings.xml)**
  - Contains all the text used in the app.
  - Helps in managing hardcoded text and supports localization.
- **Styles XML File (styles.xml)**
  - Defines custom styles and themes for UI elements.
  - Promotes consistency in design.
- **Drawable XML Files**
  - Define graphic elements like shapes, gradients, selectors, etc.
  - Used as backgrounds or icons.
- **Colors XML File (colors.xml)**
  - Stores color codes used across the app.
  - Helps in maintaining a consistent color scheme.

Firestore Firebase

- A NoSQL cloud database from Firebase and Google Cloud.
- Designed for scalable, real-time data handling in mobile and web apps.

5.2 Coding Details and Code Efficiency:

This section highlights the core logic, structured architecture, and efficient implementation used in the client application. The app uses **Kotlin, Firebase Firestore, RecyclerView, and Gson** to achieve a smooth and functional e-commerce experience.

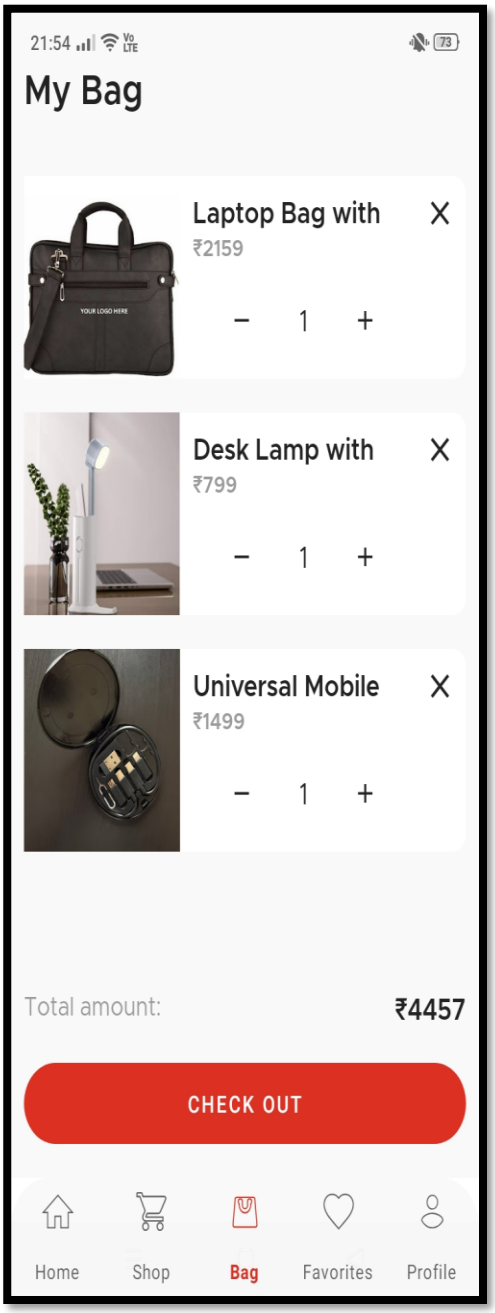
1. Cart and Checkout Flow

The checkout flow includes three steps:
BagFragment → ShippingAddressActivity → PaymentMethodActivity → OrderConfirmationActivity

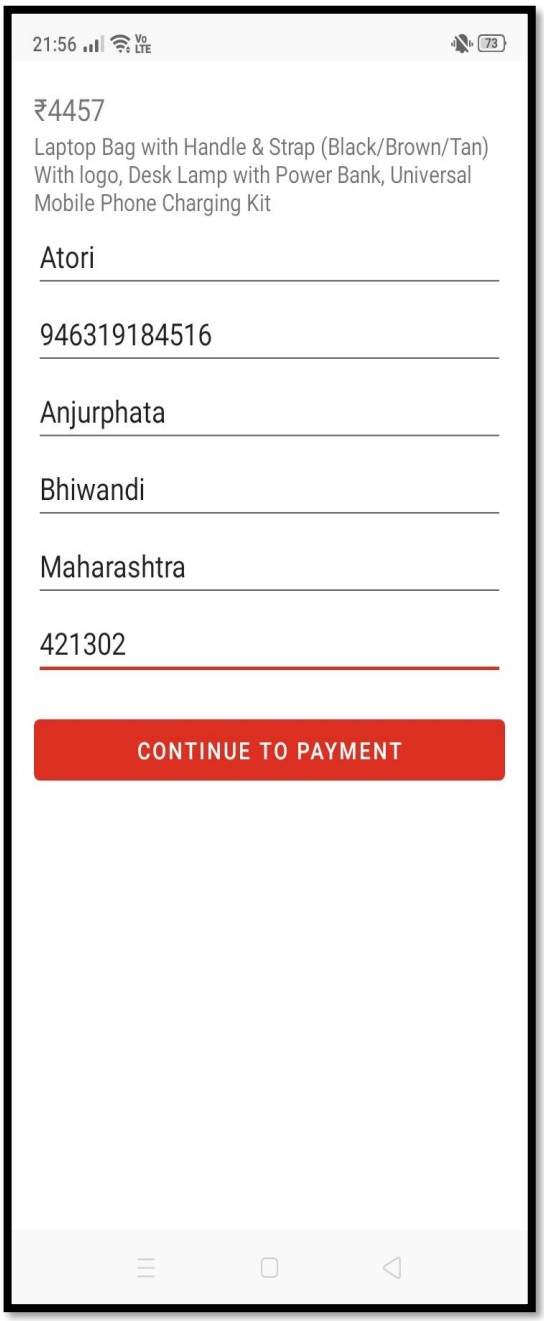
Efficient data passing example (from BagFragment):

```
checkOutBtn.setOnClickListener {
    val intent = Intent(requireContext(), ShippingAddressActiivty::class.java)
    intent.putExtra("TOTAL_AMOUNT", sum)

    val productNames = Item.joinToString(", ") { it.name }
    intent.putExtra("PRODUCT_NAMES", productNames)

    startActivity(intent)
}
```

BagFragment



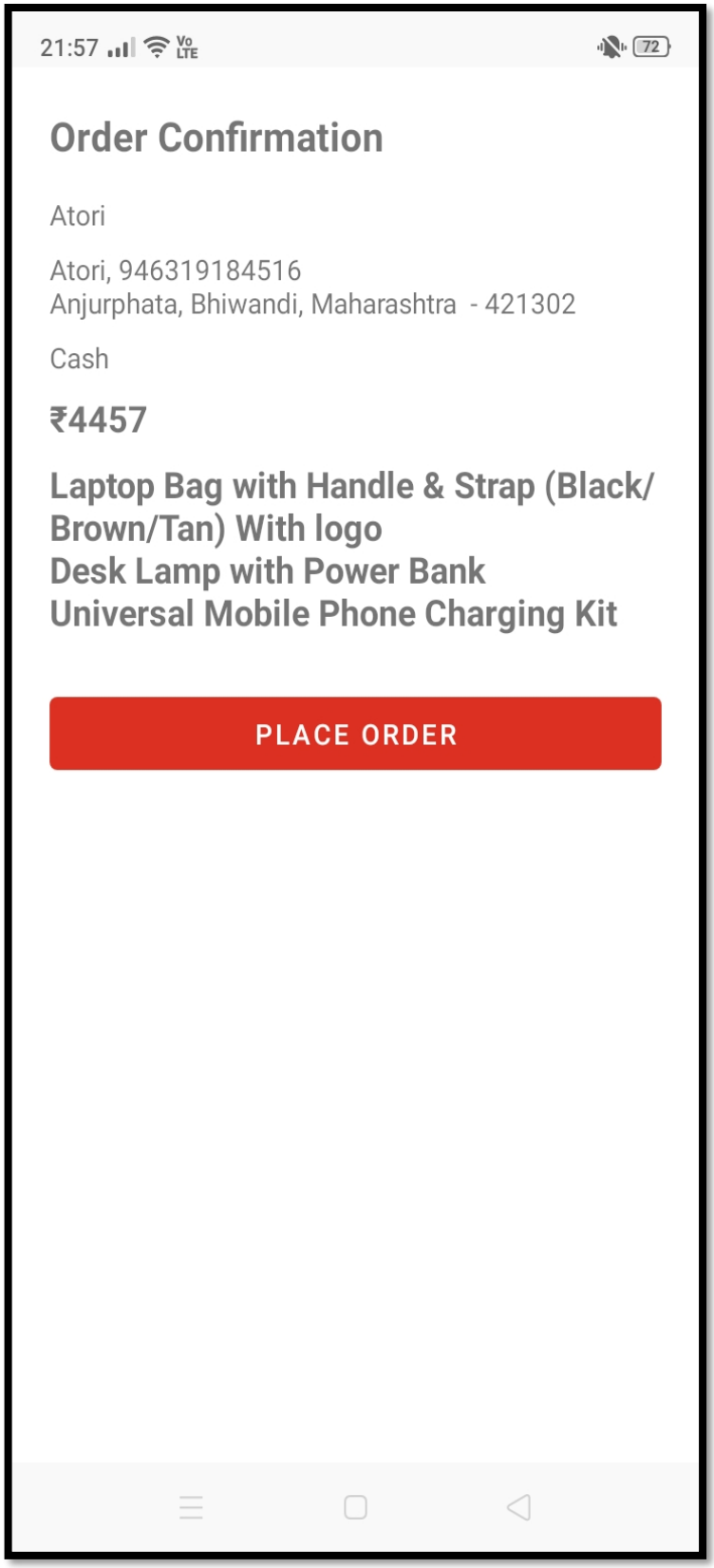
ShippingAddress

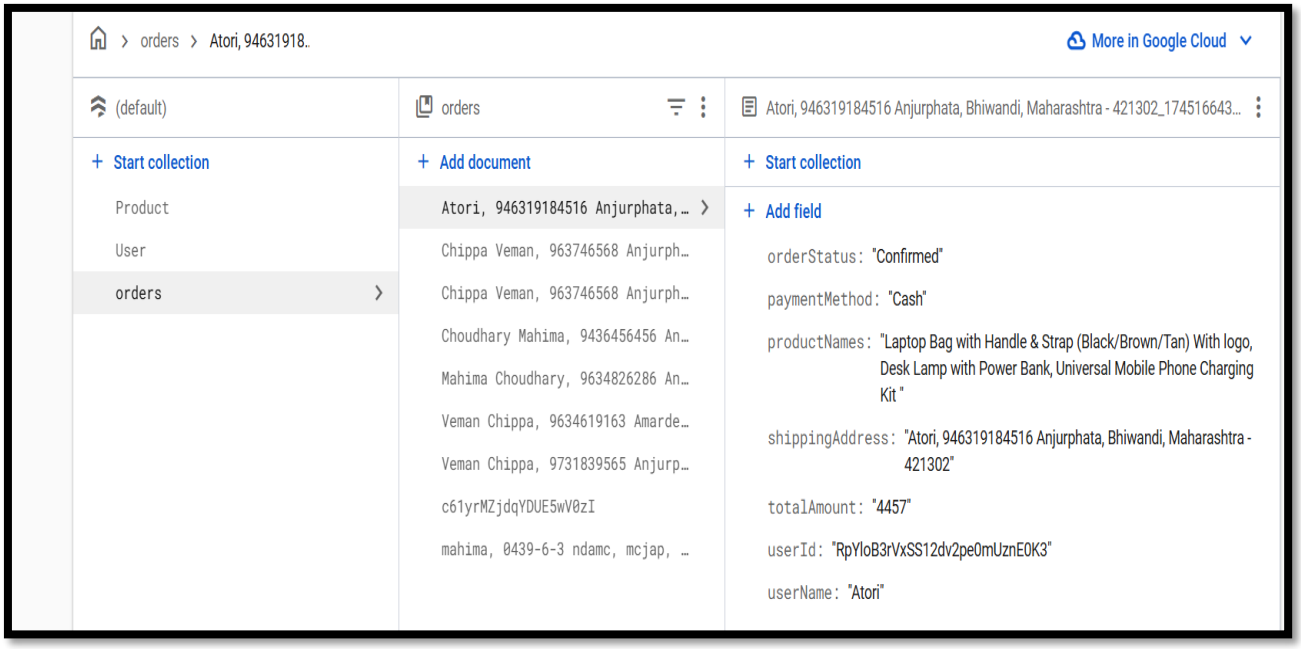


2. Order Placement with Firestore

On confirming the order, all relevant order details are stored in Firestore using the following code:

```
 val order = Order(
        userId = auth.currentUser?.uid ?: "guest",
        userName = userName,
        shippingAddress = shippingAddress,
        productNames = productNames,
        totalAmount =  totalAmount,
        paymentMethod = paymentMethod,
        orderStatus = "Pending"
    )

    val docId = "${shippingAddress}_${System.currentTimeMillis()}"

    db.collection("orders").document(docId).set(order)
        .addOnSuccessListener {
            Toast.makeText(this, "Order Placed Successfully!",
Toast.LENGTH_SHORT).show()
            cartViewModel.deleteAllCart()
            finish()
        }
        .addOnFailureListener {
            Toast.makeText(this, "Failed to place order!",
Toast.LENGTH_SHORT).show()
        }
  }
}
```

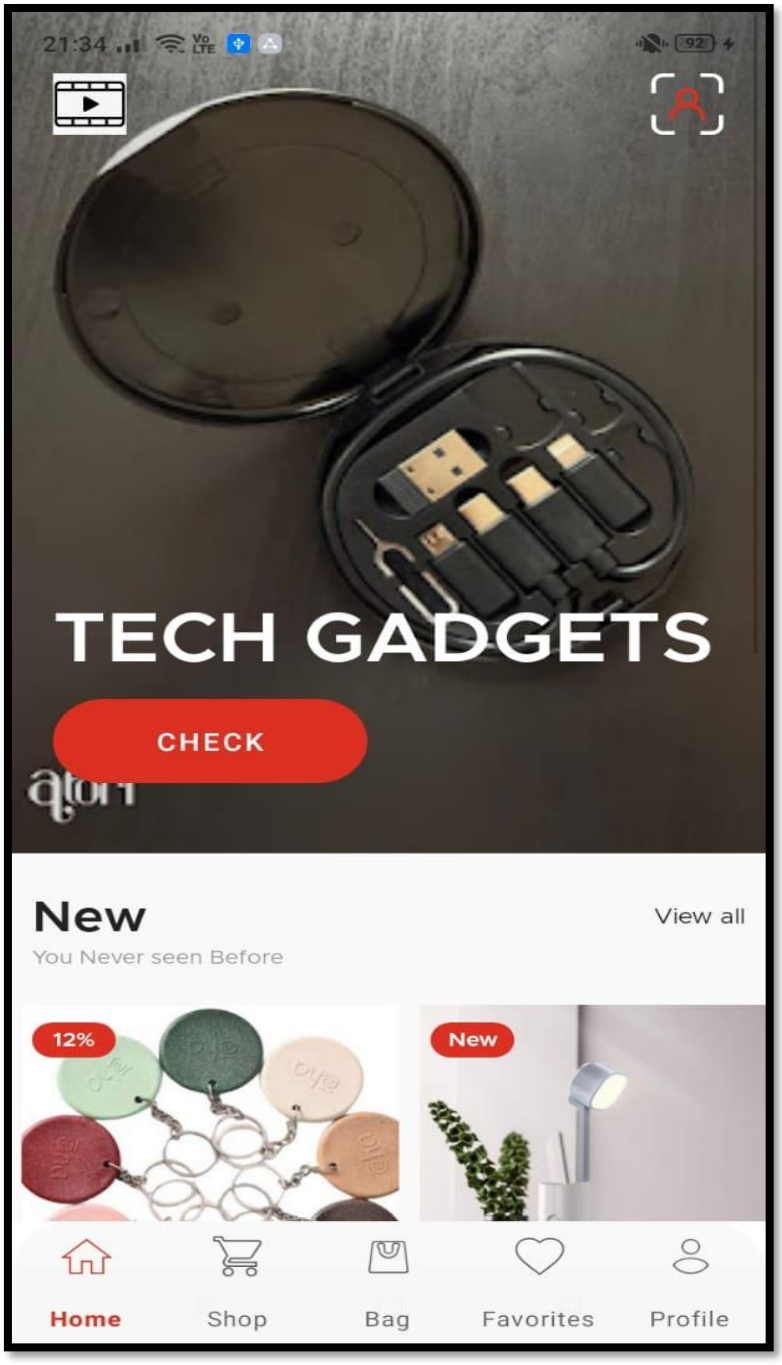

21:57
72
Order Confirmation
Atori
Atori, 946319184516
Anjurphata, Bhiwandi, Maharashtra - 421302
Cash
₹4457
Laptop Bag with Handle & Strap (Black/ Brown/Tan) With logo
Desk Lamp with Power Bank
Universal Mobile Phone Charging Kit
PLACE ORDER



2. Product Display using RecyclerView

Products are dynamically displayed from local JSON or Firestore using ProductAdapter. Gson is used to send the product object between activities:

```
private fun goToDetailsPage(position: Int, product: Product) {
        Log.d("DEBUG_ADAPTER", "Product Clicked - Index: $position,
Category: ${product.productCategory}")

        val intent = Intent(context, ProductDetailsActivity::class.java)
        val productJson = Gson().toJson(product)
        intent.putExtra("product", productJson)
        intent.putExtra("ProductFrom", product.productCategory) // Optional
        context.startActivity(intent)

    }
}
```


21:34
TECH GADGETS
CHECK
New
View all
You Never seen Before
12%
New
Home
Shop
Bag
Favorites
Profile

5.2.1 Code Efficiency

Code efficiency is a comprehensive term used to describe the **reliability**, **speed**, and **programming methodology** used in developing code for an application. Code efficiency is directly related to **algorithmic efficiency** and the **runtime execution speed** of the software. It is a vital element in ensuring high performance.

In **Atori**, code efficiency is attained by:

- Declaring all the variables and variable types specifically
- Using break statements
- Using loops efficiently and tightening loops
- Isolating common expressions
- Setting unused objects to null
- Writing reusable code
- Developing simple and easy-to-understand code with fewer lines

Code Optimization

Code optimization is a technique of modifying code to improve its **quality** and **efficiency**. A program may be enhanced so that it becomes smaller in size, consumes less memory, executes more rapidly, or performs fewer input/output operations.

In **Atori**, code optimization is achieved by:

- Using the strings.xml file to enhance code reusability
- Using XML language as it is easily readable by both humans and machines, and is also scalable and easy to develop
- Inlining small functions and concise code blocks
- Deleting unwanted records, variables, and data
- Designing layouts using XML because it is a lightweight language that prevents layouts from becoming heavy
- Utilizing Gradle-based build support along with Android's refactoring and quick-fix capabilities
- Using the Android Virtual Device (Emulator) to run and debug apps directly within Android Studio

5.3Testing Approach

Testing is vital for the success of any software. Software testing is the process to evaluate a system to test whether it can meet the expectations and fulfill the requirements. It also ensures if the system is free fromall the errors before releasing it in the market. Therefore, it is necessary to test the software in order to providea high quality and reliable system to the customer.

Types of Testing

- **White Box Testing**

  White Box Testing was applied to examine the internal structure, logic, and flow of the application code. The main areas tested include:

- **CartViewModel logic**: Ensured correct quantity updates and price calculations.
- **Order Confirmation logic**: Verified correct data handling before sending orders to Firestore.
- **Navigation checks**: Confirmed activity transitions and data passing between screens (e.g., Shipping → Payment → Confirmation).

- **Black Box Testing**

  Black box testing (also referred to as behavioral, opaque-box, or specification-based testing) focuses on evaluating the functionality of a software application without any knowledge of the internal code or structure. It is based solely on input and output:

- **Client App:** Login, product browsing, cart operations, order placement.
- **Admin App:** Login, user list, product management, order status update.

- **Grey Box Testing**

  Grey Box Testing was performed to ensure proper integration between front-end and back-end (Firestore database).

- Verified whether orders placed by users appear properly in the admin app.
- Checked if **admin updates** (like order status) reflect in user tracking.
- Ensured **Firestore rules** allow only secure and valid access to data.

5.3.1 Unit Testing

Unit Testing is the first level of software testing, where individual **components or modules of the software** are tested in isolation. The goal is to verify that each unit performs as expected. A unit is the **smallest testable part** of a software system, typically a function, method, or procedure.
In procedural programming, units may include standalone programs, functions, or procedures. In object-oriented programming, a unit is often a method belonging to a class—whether base, abstract, or derived.
This testing is primarily conducted using White Box Testing methods and is performed before Integration Testing.

- **CartViewModel**: Addition/removal of items, total price calculation.
- **ProductAdapter:** Binding correct data to RecyclerView.
- **OrderModel** and other data classes: JSON serialization/deserialization using Gson.

Test Approach for Unit Testing:

- Designing and preparing test cases
- Generating test data with appropriate validation checks
- Performing a detailed code review of the module
- Executing manual testing of the code
- Fixing errors identified during testing
- Preparing test result documentation

Aspects Covered During Unit Testing:

- Functional testing of each module/form
- Validation of all user inputs
- Ensuring adherence to coding standards
- Testing with various sets of test data
- Verifying functions involving calculations and logic

5.3.2 Integration Testing

Integration Testing is the second level of testing, conducted after successful unit testing. It involves combining multiple software modules and testing them as a group to verify their interaction. This stage aims to identify issues in the interfaces and communication between **integrated units.**

Modules in the project are typically integrated using the top-down approach. The testing technique can include Black Box, White Box, or Grey Box methods, depending on how a 'unit' is defined.

- **End-to-end checkout flow (cart → shipping → payment → confirmation).**
- **Firestore product updates showing up in both apps.**
- **Synchronization of order status between admin and client.**

Test Approach for Integration Testing:

- Integration of related modules/forms within the system
- Designing and preparing test cases
- Creating test data with complete validation coverage
- Performing manual testing
- Logging all encountered errors
- Modifying and fixing errors after testing
- Documenting test results after corrections

Aspects Covered During Integration Testing:

- Verifying seamless interaction between integrated modules
- Testing newly integrated features that were not unit tested
- Testing database connectivity and queries from modules
- Validating integration with external hardware interfaces
- Ensuring proper exception handling between modules

5.4 TEST CASES DESIGN

Test Case Overview

A test case is a detailed specification that includes input values, execution conditions, testing procedures, and expected results. Each test case is designed to verify that a specific function of the software performs as intended, ensuring that the system meets its requirements and behaves reliably under different scenarios.

Test cases are critical for ensuring:

- The correctness of application functionality
- Compliance with system requirements
- Identification of defects or unexpected behavior
- Quality assurance before deployment

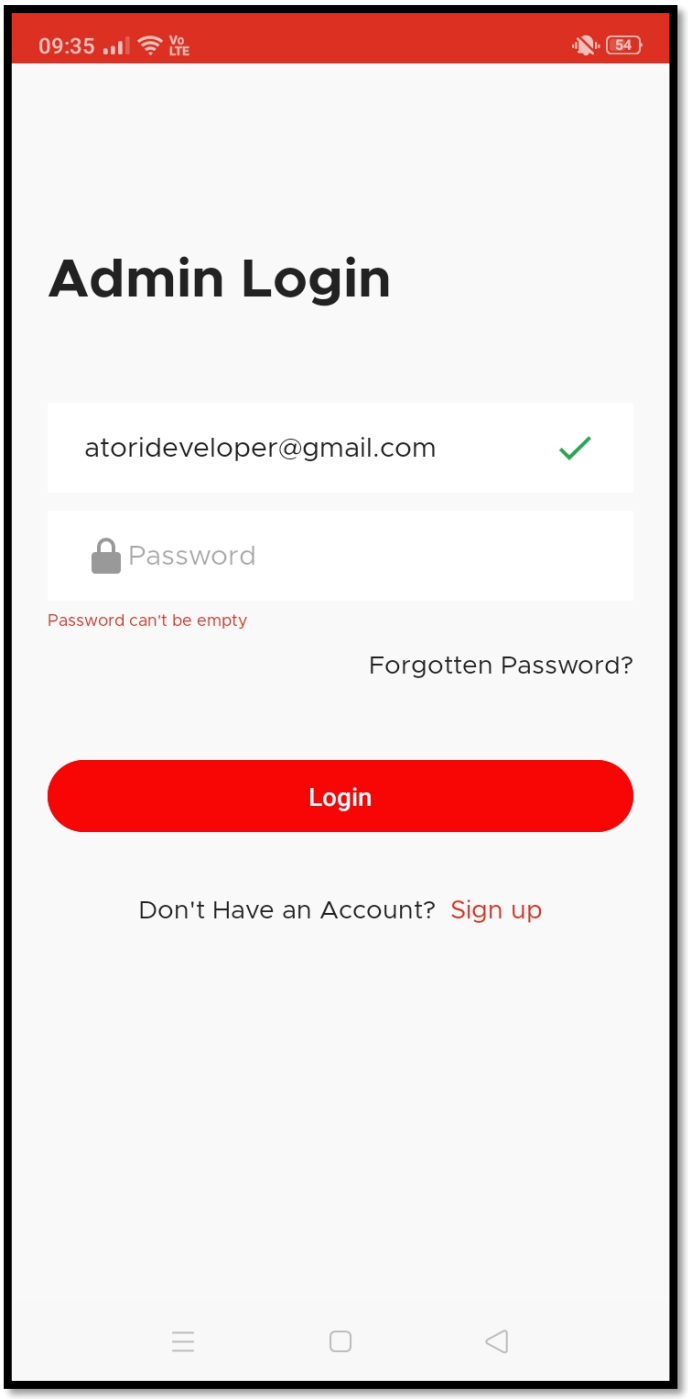
Test Case Design for Atori Admin App/Dashboard

S.No	Test Scenario	Inputs	Expected Output	Actual Output	Test Result	Test Comments
1	Checking Admin Login	Valid login credentials (email & password)	Display of Admin dashboard	Display of Admin dashboard	Pass	Home page appeared
2	Adding/Updating Products	Valid details: Name, Rating, Description, Image/Video URL → Add Click	Product uploaded successfully	Product uploaded successfully	Pass	Product successfully added
3	Deleting Product	Select a valid product → Click Delete	Product deleted	Product deleted	Pass	Product successfully removed
4	Viewing Order List	Navigate to Orders tab	List of all orders displayed	List of all orders displayed	Pass	Orders visible with update option
5	Viewing User List	Navigate to Users tab	List of users with name & email	List of users with name & email	Pass	User list visible

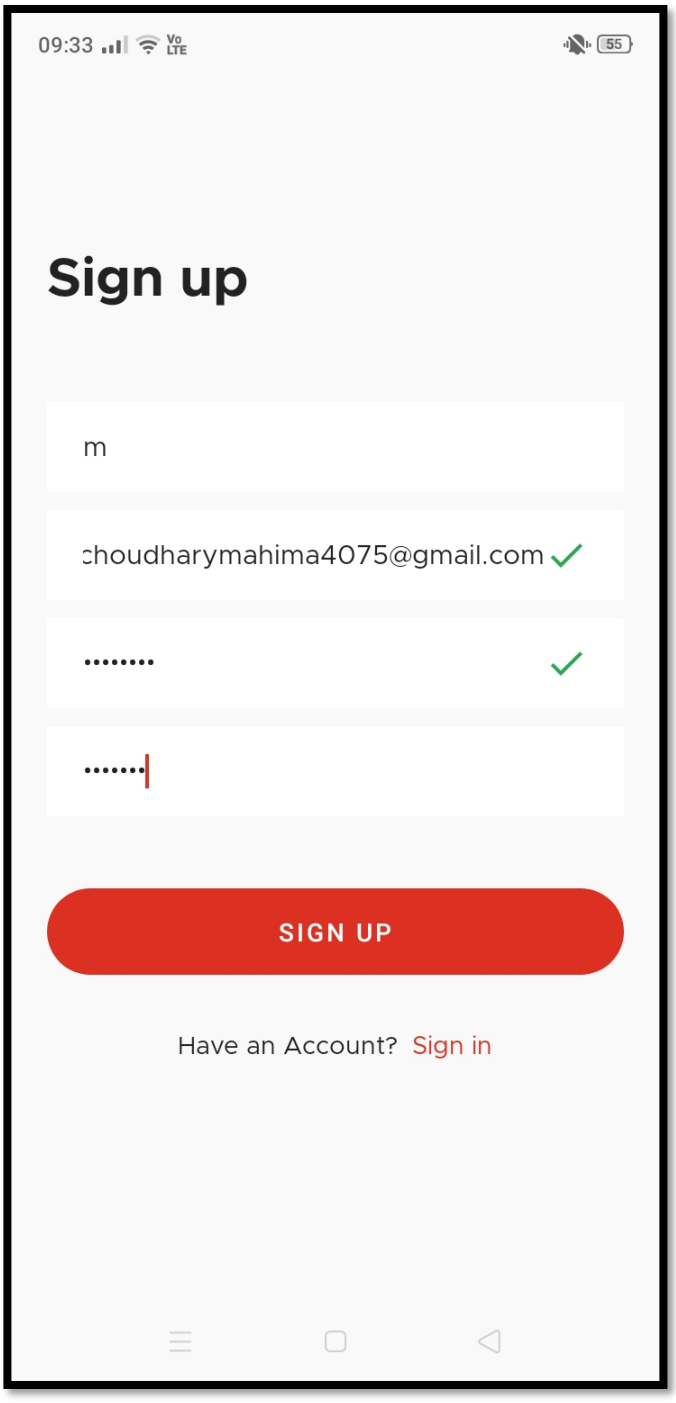
Test Case Design for Atori Client App

S.No	Test Scenario	Inputs	Expected Output	Actual Output	Test Result	Test Comments
1	Checking User Login	Valid login credentials (email & password)	User dashboard appears	User dashboard appears	Pass	Login successful
2	Browsing Products	Navigate to home screen	List of products displayed	List of products displayed	Pass	Products visible on home
3	Adding to Cart	Select product → Click Add to Cart	Product added to cart	Product added to cart	Pass	Cart updated successfully
4	Placing Order	Proceed to checkout with valid details	Order placed successfully	Order placed successfully	Pass	Order stored in Firestore
5	Viewing Orders	Navigate to Orders tab	List of user's orders displayed	List of user's orders displayed	Pass	Order list visible

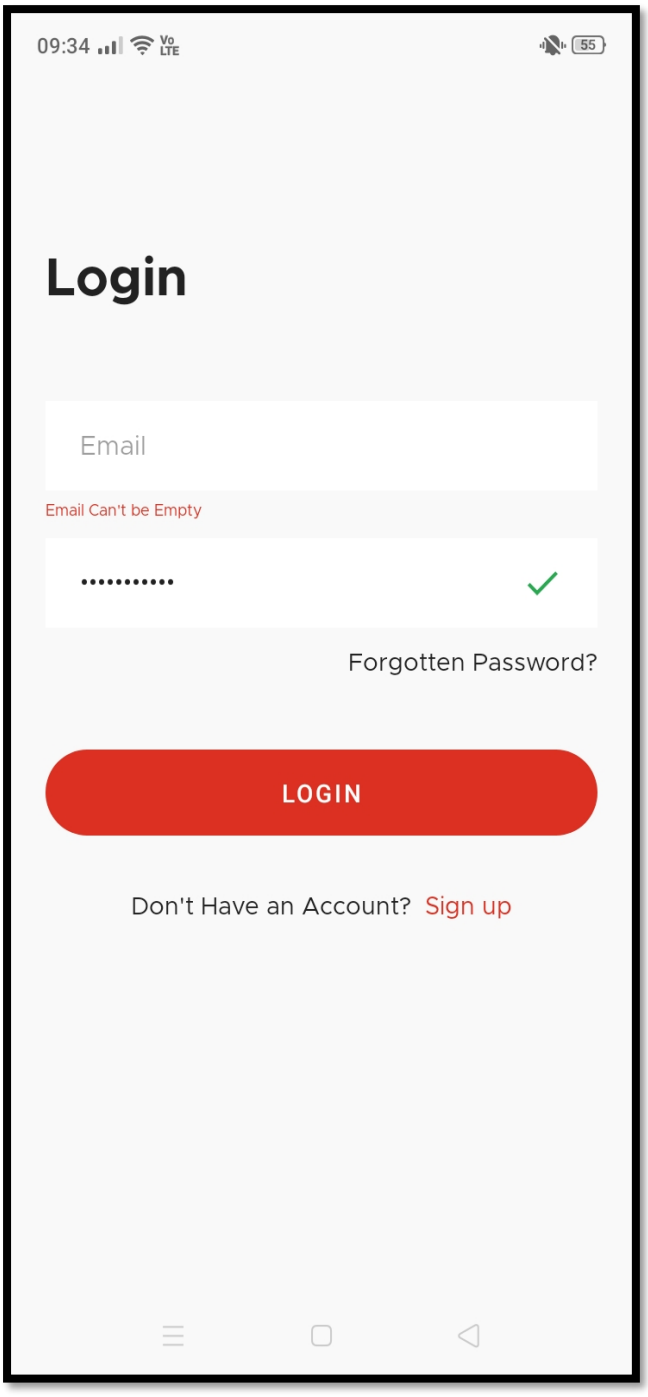
Admin Login


09:35
Admin Login
atoridеveloper@gmail.com
Password
Password can't be empty
Forgotten Password?
Login
Don't Have an Account? Sign up

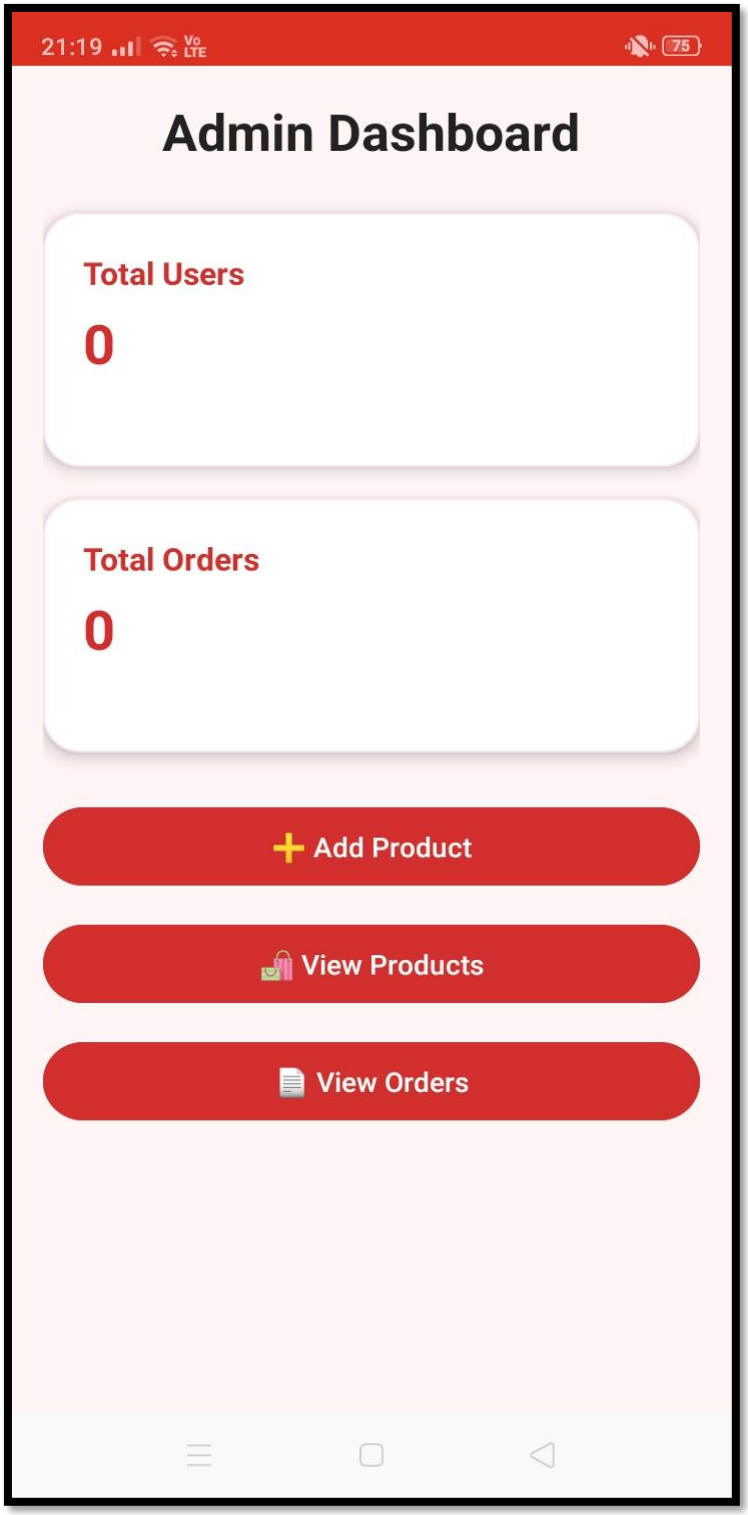
Client Sign Up



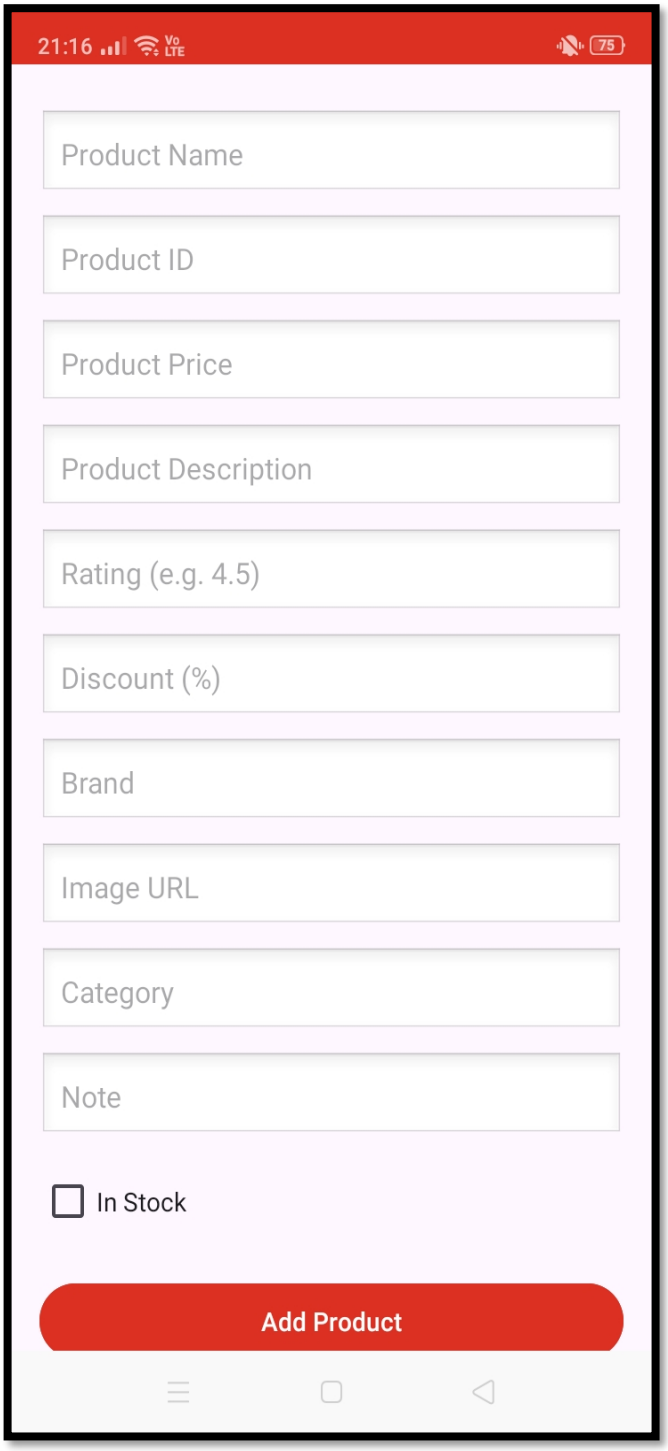
Client Login



Admin DashBoard



Admin –Add Product



5.5 Modifications and Improvements

- The development of the Atori mobile application involved continuous updates and refinements to improve usability, performance, and functionality. The project includes two separate apps—Client App (for users) and Admin App (for Atori's internal team). Below are the key modifications and improvements made during the development process:

Client App

1. **Enhanced UI Design**
   The home screen and navigation structure were redesigned with a more engaging and clean interface to reflect Atori's branding. Font styles, color palette, and spacing were adjusted to improve readability and user interaction.

2. **ShowCase Fragment Implementation**
   A new fragment called **ShowCase** was introduced to feature promotional videos and curated gifting options. This section allows users to view gifting ideas in a visually appealing manner.

3. **Streamlined Navigation**
   The bottom navigation bar was simplified for better accessibility to key sections such as Products, Cart, and Profile. This ensured a smoother user experience.

4. **Firebase Optimization**
   Firebase queries were optimized to improve data retrieval speed and reduce unnecessary reads, making the app more efficient and responsive.

5. **Improved Input Validation**
   Validations were added to user input fields (like name, email, phone number) to minimize errors during data submission and ensure clean data entry.

Admin App

1. **Dashboard Enhancements**
   A clear and organized dashboard was developed, displaying important metrics such as **Total Revenue** and **Total Orders**. Data is presented using card layouts for a better visual overview.

2. **Product Management Module**
   The admin can now add, update, and delete products with fields like **Product Name**, **Rating**, **Description**, and **Blogger Image/Video URL**. The CRUD functionality was implemented to provide complete control over the product catalog.

3. **User List Feature**
   A list of users who added items to the cart is displayed with their **Name** and **Email**. This helps the admin track interested users since the ordering process happens via direct call, not in-app checkout.

4. **Order Management Section**
   Admins can view and update user orders. The status update feature simplifies order tracking and helps keep records organized and up-to-date.

5. **Firebase Project Integration**
   Both apps were connected to the same Firebase project to ensure seamless data synchronization. Proper rules were set up for user authentication and secure data handling.

6. **Consistent Design with Client App**
   The Admin app was styled to match the Client app's theme—same colors, fonts, and layout principles—ensuring consistency and a professional brand identity across both platforms.

   These modifications and improvements ensured that both applications—Client and Admin—were optimized for performance, user experience, and business utility. They also helped in aligning the apps with Atori's goal of providing a seamless and professional gifting experience for corporate clients.

CHAPTER 6
RESULTS AND DISCUSSION

6.1 Test Report

After thorough testing of all modules in both the **Atori Client App** and **Admin App**, the applications were found to be stable, functional, and ready for deployment. The user interface is clean and user-friendly, and all features worked as expected. Testing confirmed that the applications meet the project objectives and handle user interactions and edge cases effectively.

Unit Testing Results:

- Each module was tested individually to ensure correct functionality and performance.
- Common issues like invalid login, incorrect input handling, and display errors were identified and resolved.
- The modular and reusable codebase allowed for easier debugging and future improvements.
- Notable fixes during unit testing included:
    - Login validations and Firebase Authentication
    - Cart functionality improvements
    - Product list loading and video display
    - Admin dashboard metrics logic
    - Order and user data display issues

Integration Testing Results:

- Integration testing verified the smooth communication between modules (e.g., cart → checkout → order confirmation).
- Admin and client sync validated for real-time Firestore data updates.
- User placing an order successfully triggers admin app's order list.
- Editing a product in the admin app updates it in real-time in the client app.

6.2 User Documentation

1.Getting Started

The ATORI mobile application is designed for both clients and administrators. This manual will guide you through using the app efficiently.

- **Client App** (for Atori's customers)
- **Admin App** (for Atori's internal team)

2. Client Application

- **2.1 Browsing Products**
  Clients can browse through various product categories such as Mugs, Humidifiers, Ecofriendly items, etc. Each product listing includes an image, name, price, and rating.

- **2.2 About Us Screen**
  This screen shows Atori's brand description, team members, contact numbers, company address, and links to website/email. It allows users to quickly connect with the company for queries or custom gift requests.

- **2.3 Product Details & Favorites**
  Clicking a product displays detailed information. Users can add products to their favorites for easy access later.

- **2.4 Shopping Cart and Orders**
  Users can add products to the shopping cart, modify quantities, and proceed to checkout using either cash or saved cards. Orders can be tracked through status indicators.

- **2.5 Video Showcase Module**
  Users can view Atori's promotional video which highlights the mission, product line, and client experience. This screen gives an immersive brand feel and may include corporate gifting walkthroughs.

- **2.6 Favorites (Wishlist)**
- Users can **add products to their Favorites** list by tapping the heart icon on any product.
- Favorites are saved locally using **SharedPreferences**, allowing quick access without reloading from Firestore.
- Users can:
    - Add or remove products from their wishlist.
    - Access the **Favorites screen** to view all saved items.
    - Click on any favorite product to view full details or add it to their cart.

3. Admin Application

3.1 User Management

- Admins can view a complete list of registered users.
- Provides options to **edit** or **delete** user profiles directly.
- Helps in managing user access and maintaining system integrity.

3.2 Product Management

- Admins can:
    - **Add new products** to the catalog.
    - **Edit existing products** (update name, price, description, image, etc.).
    - **Delete products** that are no longer available.
- Each product entry includes:
    - Product Name
    - Product Price
    - Image URL
    - Product Category
    - Product Description

3.3 Order Management

- Admins can:
    - View a list of all orders placed by users.
    - See order details such as user name, total amount, and items.
    - **Update order statuses** (e.g., Pending → Confirmed → Shipped → Delivered).

3.4 Dashboard Reports

- Shows summarized stats including:
    - Total number of users.
    - Total number of orders.

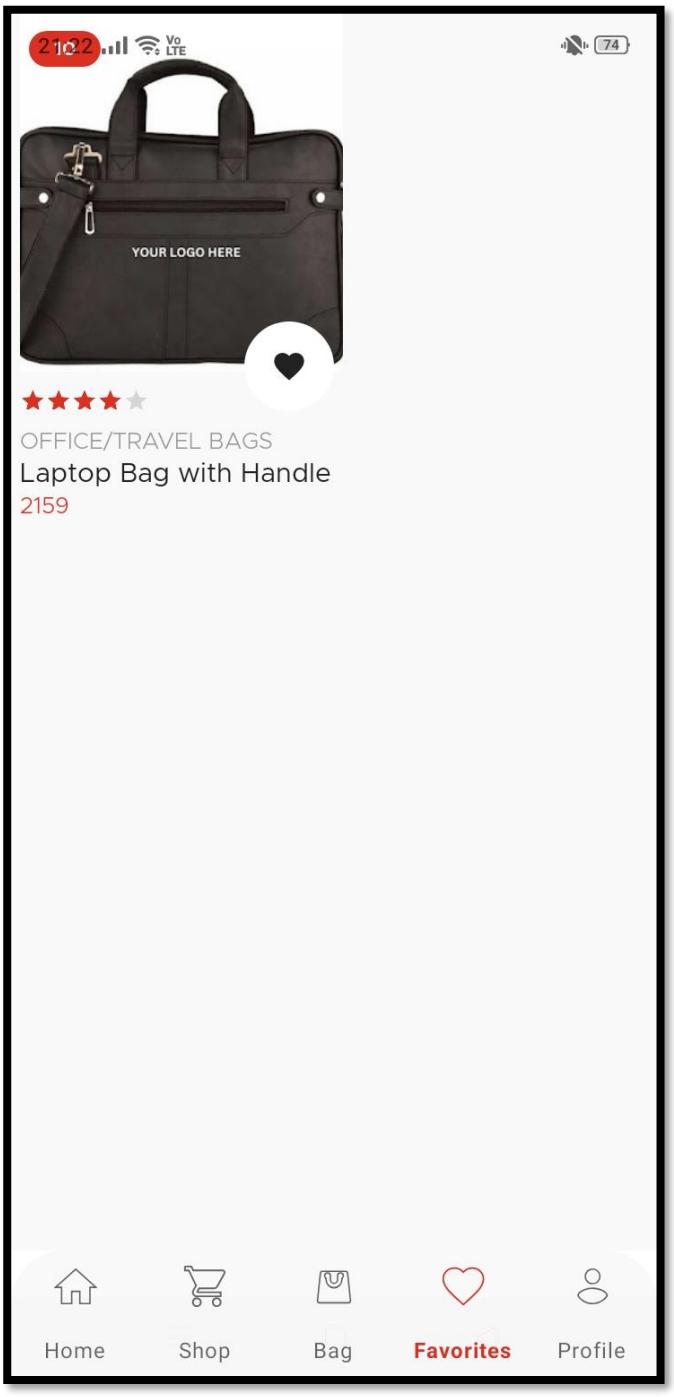
- Helps admin analyze app performance and user activity at a glance

4. Troubleshooting

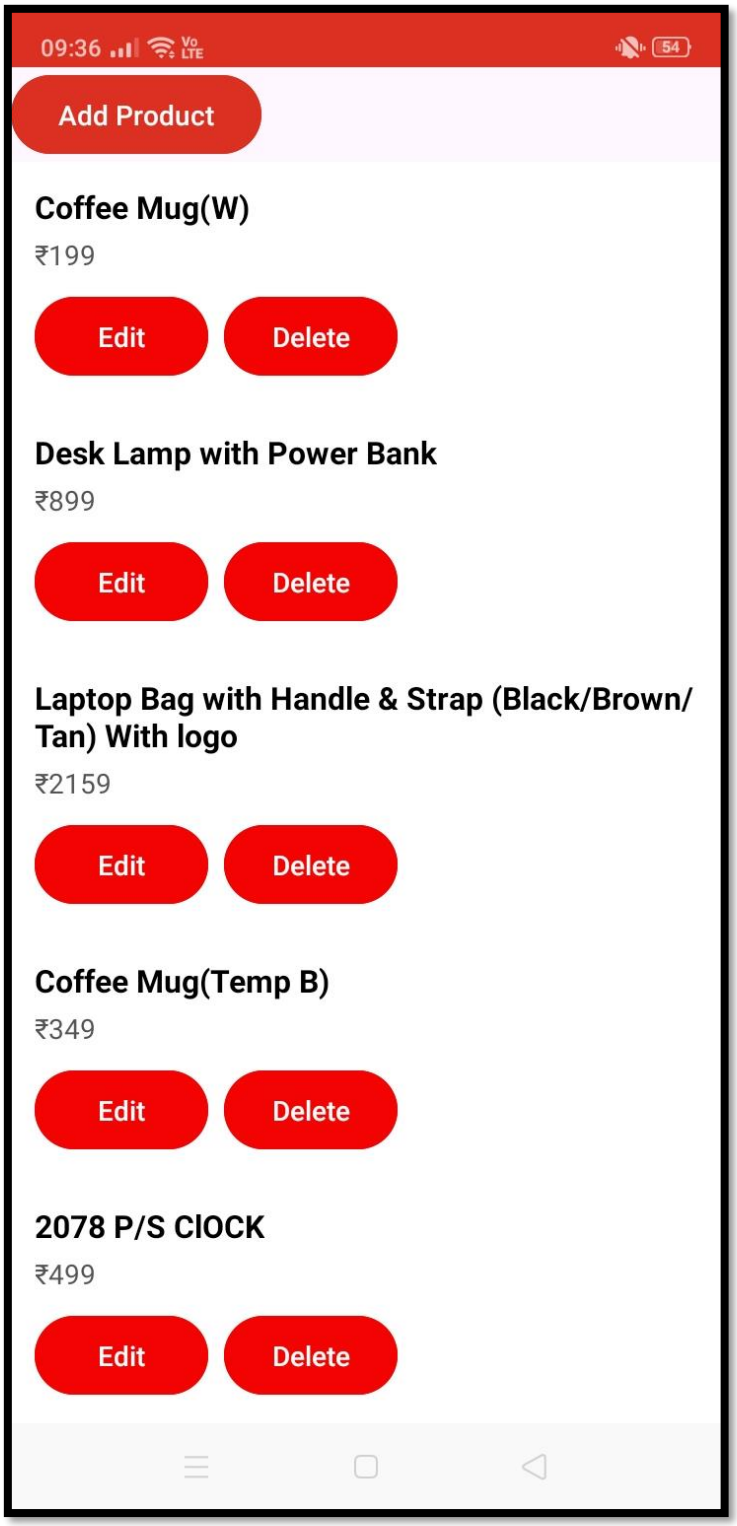
- **Internet Issues**
  Ensure your device is connected to the internet to fetch real-time product and user data from Firestore.
- **App Crashes**
  If the app crashes:
  - Restart the application.
  - Open **Logcat in Android Studio** to inspect crash reports and fix errors.
- **Firestore Rules Access Issues**
  Make sure Firestore rules allow proper read/write access for the current user.

Video Showcase Module

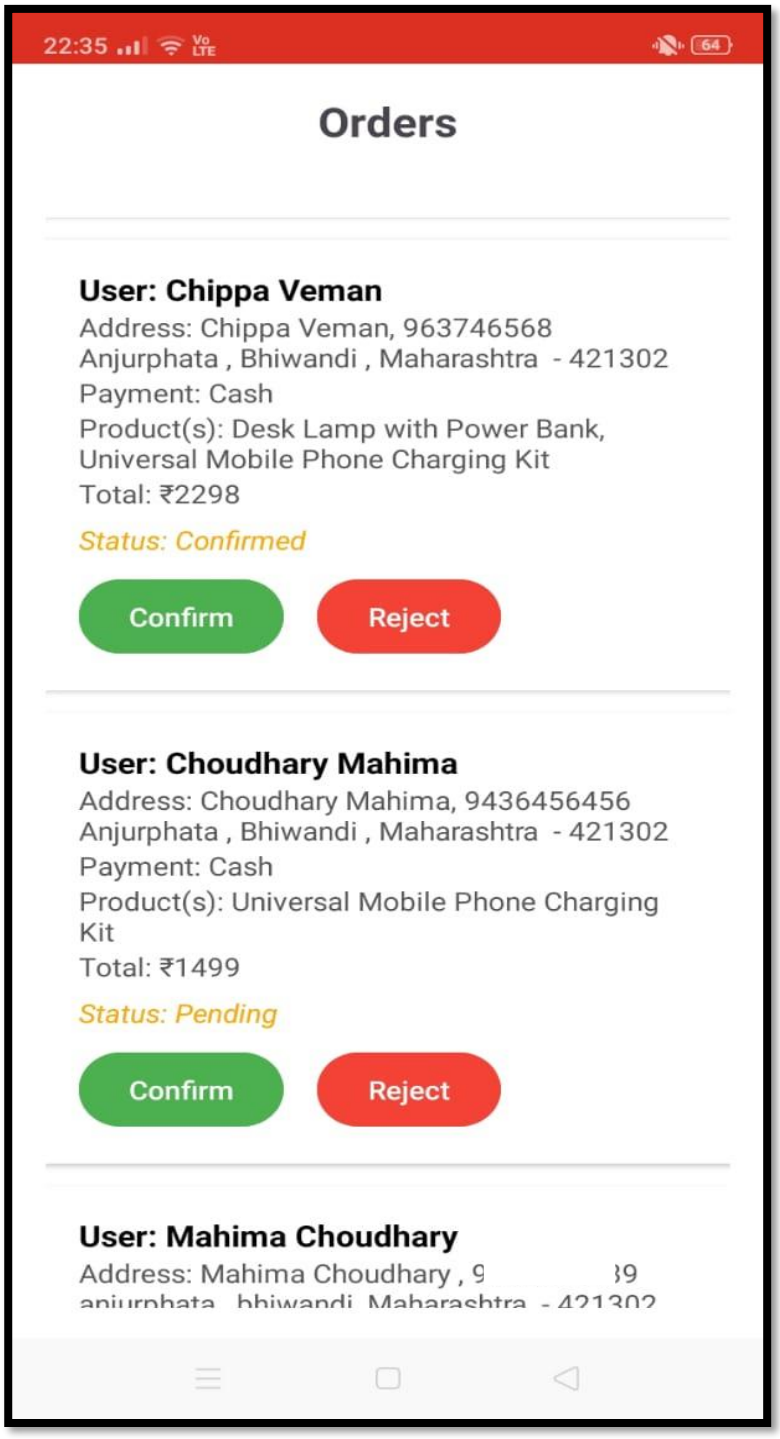
Client- Favorites (Wishlist)



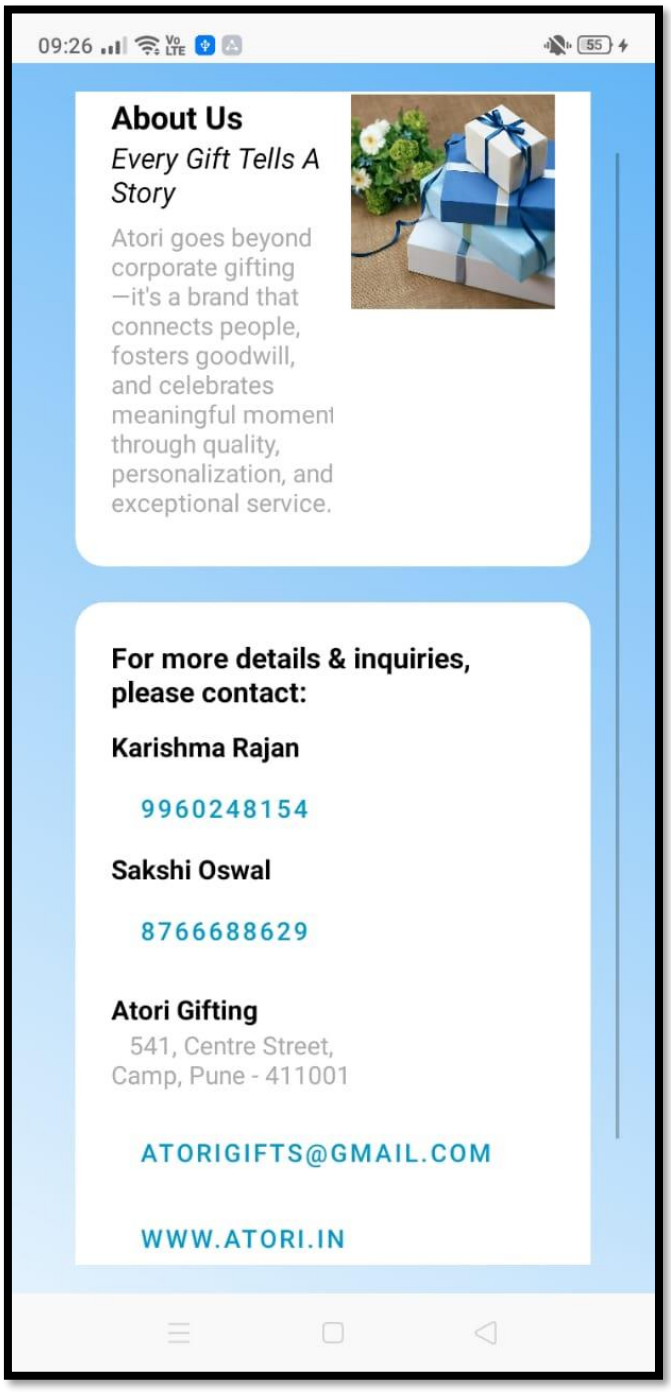
Admin ---View Product



Admin App - Order Details



Client –About Us



CHAPTER 7
CONCLUSION AND FUTURE WORK

7.1 Conclusion

- Technology plays a transformative role in modern business. The Atori mobile application system was developed to improve the efficiency, accessibility, and professional presentation of Atori's corporate gifting services.
  - Through the **Client App**, users can easily log in, browse a curated range of corporate gift products, view promotional videos, and add products to their cart. Rather than a direct checkout, users initiate contact with Atori via call, ensuring a personal and consultative approach to corporate gifting.
  - The **Admin App** empowers Atori's internal team to manage products, track orders, monitor user activity, and maintain the product catalog efficiently. With secure login, real-time dashboard analytics, and Firebase integration, the admin can make immediate updates and manage user engagement effectively.

Key benefits observed:

- Enhanced user experience and customer engagement
- Real-time data sync through Firebase
- Simplified product and order management
- Reduced manual efforts through automation
- Fast and responsive UI with clean navigation

The project fulfills its objective of providing a mobile-first platform that simplifies the gifting process for businesses and gives Atori a competitive edge in the digital space.

7.2 Limitations

Despite its success, a few limitations exist due to time constraints and project scope:

- The app does not currently support push notifications or in-app messaging.
- Orders are not placed directly within the app; users must contact via phone.
- There is no in-app payment integration.
- The Admin app requires further enhancements for bulk data import or analytics reporting.
- The app is not yet deployed on the Google Play Store due to financial constraints.

7.3 Future Scope of the Project

Several enhancements can be made to further improve the Atori application:

- **Push Notification System:** To alert users about new products, updates, or festive gifting ideas.
- **In-App Messaging or Chat Support:** To enhance real-time customer interaction.
- **Order Placement Module:** Enable users to place and track orders directly within the app.
- **Payment Gateway Integration:** Support online transactions securely.
- **Product Rating and Review System:** Allow users to rate or leave feedback on gifts.
- **Admin Analytics Dashboard:** Include detailed reports for sales, category performance, and user behavior.
- **Publish on Google Play Store:** To reach a wider user base and enhance brand visibility.

These improvements would help Atori scale operations, increase customer satisfaction, and align with modern mobile commerce standards.

CHAPTER 8
REFERENCES

1. https://developer.android.com — Official documentation for Android development and Jetpack libraries.

2. https://firebase.google.com/docs — Firebase documentation used for Authentication, Firestore, and Storage integration.

3. https://kotlinlang.org/docs/android-overview.html — Kotlin documentation for Android-specific development.

4. https://www.youtube.com/watch?v=2oJfQ19eBXk — Firebase Firestore integration tutorial by Stevdza San.

5. https://www.youtube.com/watch?v=OvDZVV5CbQo — Firebase Authentication with Kotlin (Coding in Flow).

6. https://github.com/firebase — Firebase sample repositories for CRUD and model design patterns.

7. https://stackoverflow.com — Used for resolving Firebase Firestore errors and runtime crashes.

8. https://fonts.google.com & https://www.color-hex.com — Resources for UI fonts and color schemes.

9. **Android Studio IDE** — Used for complete development of both admin and client Android apps.

10. https://www.tutorialspoint.com/android/index.htm — Comprehensive guide for Android components and Firebase usage.

11. https://www.programming-books.io/essential/android/ — *Essential Android Kotlin Programming* open-source reference.

12. https://www.instagram.com/atori_in?utm_source=ig_web_button_share_sheet&igsh=ZDNlZDc0MzIxNw== — Official Instagram for “Atori – Artisanal Corporate Gifting”.

13. https://www.linkedin.com/in/atori-gifting-66986b302/ — Official LinkedIn page for “Atori – Artisanal Corporate Gifting”.