CAMPUS CART

Minor Project-I

(ENSI152)

Submitted in partial fulfilment of the requirement of the degree of

MASTER OF COMPUTER APPLICATIONS

to

K.R Mangalam University

by

Ayaan Ur Rehman (Roll: 2401560036)
Riya Srivastav (Roll: 2401560047)
Harshvardhan Singh (Roll: 2401560064)
Manish Yadav (Roll: 2401560045)

Under the supervision of

Ashwini Kumar
(Assistant Professor SOET)

Department of Computer Science and Engineering

School of Engineering and Technology

K.R Mangalam University, Gurugram- 122001, India

April 2025

CERTIFICATE

This is to certify that the Project Synopsis entitled, "**CAMPUS CART**" submitted by "**Ayaan Ur Rehman (2401560036), Riya Srivastav (2401560047), Harshvardhan Singh (2401560064) and Manish Yadav (2401560045)"** to **K.R Mangalam University**, **Gurugram, India,** is a record of bonafide project work carried out by them under my supervision and guidance and is worthy of consideration for the partial fulfilment of the degree of **Master Of Computer Applications** of the University.

Type of Project (Tick One Option)

Industry/Research/University Problem

Signature of Internal supervisor
Ashwini kumar (Assistant Professor)

Signature of Project Coordinator

Date: 30th April 2025

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to all those who supported and guided us throughout the development of our project, ***Campus Cart.***

We are deeply thankful to **Mr. Ashwini Kumar**, our internal guide, for his valuable guidance, constant encouragement, and constructive feedback, which played a vital role in the successful completion of this project. We also extend our heartfelt thanks to **Mr. Saransh Sharma**, our external mentor, for his expert advice, insights, and unwavering support at every stage of the project.

We are grateful to the **School of Engineering and Technology** for providing us with the opportunity and resources to work on this project, as well as for fostering an environment that encourages innovation and learning.

Lastly, we would like to acknowledge the teamwork, dedication, and contributions of each member of our group — Ayaan, Harshvardhan, Manish, and Riya — without whom this project would not have been possible.

We are truly thankful to everyone who directly or indirectly contributed to the success of *Campus Cart*.

Ayaan Ur Rehman (Roll: 2401560036)

Riya Srivastav (Roll: 2401560047)

Harshvardhan Singh (Roll: 2401560064)

Manish Yadav (Roll: 2401560045)

INDEX

ABSTRACT

The *Campus Cart* project aims to develop a mobile application that simplifies the shopping experience for college students and local customers by enabling them to order essential products online and schedule store pickups. Unlike traditional e-commerce platforms focused on home deliveries, Campus Cart offers a cost-effective, efficient solution by eliminating the need for delivery logistics. Built using Flutter for the frontend and Firebase Firestore for backend services, the app ensures secure online transactions through Razorpay integration. Customers can browse products, manage their cart, make secure payments, and select a convenient pickup slot, thereby reducing wait times and in-store congestion.

The application also offers features like group orders and purchase history dashboards, helping store owners personalize offers and improve service. Through this innovative approach, Campus Cart enhances customer convenience, supports local businesses, and optimizes store operations for a better shopping experience.

KEYWORDS: E-commerce, Mobile Application, Flutter, Firebase Firestore, Razorpay Integration, Scheduled Pickup, Campus Shopping, Digital Store Management, Group Orders, Student Convenience

CHAPTER: 1

1. Introduction

1.1 Background

In today's digitally connected world, the way people shop for goods and services has undergone a massive transformation. With the rise of e-commerce platforms, customers now expect convenience, speed, and seamless service at their fingertips. However, most traditional e-commerce models focus primarily on home delivery services, which may not always be practical or efficient for all customers — especially students living in hostels or campuses. These delivery systems also add logistical complications and costs for local stores that do not have a large delivery infrastructure.

College students, in particular, often face challenges like time constraints, limited access to transportation, and the need for frequent, small essential purchases. Visiting physical stores repeatedly during a busy academic schedule can be both time-consuming and inconvenient. Recognizing these problems, there was a need for a platform that offers the ease of online shopping combined with the convenience of self-pickup from nearby stores, thereby eliminating delivery delays, reducing operational costs, and saving precious time for students.

Campus Cart was conceptualized to address this very gap. It focuses on providing a quick and efficient shopping experience by allowing users to browse products, make online payments, and schedule pickups according to their convenience, all through a user-friendly mobile application.

1.2 Purpose of the Project

The primary purpose of the *Campus Cart* project is to create a mobile application that simplifies the process of shopping for essential items for college students, hostellers, and local residents. The application aims to bridge the gap between traditional in-store shopping and digital convenience by offering a scheduled pickup system without involving third-party delivery services.

The project seeks to benefit both customers and local businesses:

- **For customers**, it offers an easy-to-use platform where they can place orders remotely, pay securely, and pick up their items at a time that suits them best — thus saving time, effort, and unnecessary trips to crowded stores.
- **For store owners**, it provides a digital platform to manage inventory, receive orders, handle payments, and better serve their customers without needing to invest in costly delivery infrastructure.

Additionally, the app is built with a focus on scalability, ensuring that it can later accommodate various product categories and support multiple stores as needed. It also aims to minimize failed payment issues by integrating secure gateways and providing real-time updates to users about their orders.

1.3 Scope of the Project

The scope of the *Campus Cart* project is centered around designing, developing, testing, and deploying a mobile application for Android and iOS platforms using Flutter and Firebase technologies. Key functionalities included within the scope are:

- **User Module:**
    - User registration/login using Firebase Authentication.
    - Browsing products across different categories like groceries, toiletries, snacks, and stationery.
    - Adding products to the shopping cart and placing orders.
    - Online payment integration through Razorpay.
    - Choosing scheduled pickup slots to avoid crowding and waiting.
- **Admin Module:**
    - Store owners can add, edit, and delete product listings.
    - Manage orders and view customer details.
    - Track pickup schedules to optimize store workflow.
- **System Features:**
    - Real-time updates for order confirmation and status changes.
    - Digital invoice generation post successful payments.
    - Push notifications for important updates like order ready for pickup.

However, the project does not cover:

- Delivery to home addresses (it is exclusively a pickup-based system).
- Third-party store integrations beyond the initial local setup.

- Large-scale e-commerce features like dynamic pricing, warehouse management, or global shipping.

In the long term, the project can be expanded to include partnerships with multiple stores, loyalty programs, offer-based marketing, and additional modules like group ordering or custom recommendations based on purchase history.

CHAPTER: 2

2. Problem Statement

2.1 Existing Issues in the Current System

In the current retail landscape, particularly in campus and hostel environments, traditional shopping methods continue to present several inefficiencies for both customers and store owners. Students often need to physically visit multiple stores to fulfill their everyday requirements such as groceries, toiletries, stationery, and snacks. This process is not only time-consuming but also inconvenient, especially during busy academic schedules.

Moreover, most local stores lack a streamlined digital system to facilitate pre-ordering and scheduled pickups. Customers are forced to stand in long queues, resulting in significant wait times, especially during peak hours. This congestion also creates challenges in maintaining store operations efficiently and can negatively affect the overall customer experience.

Payment-related issues are also common in existing setups. Many campus stores face network disruptions, leading to payment failures, which further delay the purchase process and cause frustration among customers. In many cases, students are forced to either carry cash or retry transactions multiple times, creating unnecessary delays.

Furthermore, there is limited use of customer data to personalize shopping experiences in local stores. Unlike major e-commerce platforms, small businesses lack access to digital tools that can analyze buying behavior or optimize inventory based on demand patterns. This restricts the potential for offering targeted promotions or loyalty rewards to frequent customers.

Another significant limitation is that most available digital shopping solutions are designed primarily for delivery models, not pickup-based operations. This makes them unsuitable for local stores that cannot manage delivery logistics due to resource constraints. As a result, both students and store owners are left without an efficient, campus-specific shopping platform.

2.2 Need for a New System

Given the limitations of the current system, there is a clear need for a new, specialized digital solution that addresses the specific shopping habits and challenges faced by students and local customers. An efficient mobile application that enables online browsing, secure payment, and scheduled pickups can significantly transform the traditional shopping experience.

Campus Cart proposes to fill this gap by offering a platform that allows customers to order products remotely, pay securely online, and choose a convenient pickup time, eliminating the need for long wait times and multiple store visits. It reduces the dependency on cash transactions and network connectivity at the store by enabling online payments from anywhere, making the entire purchase process more reliable.

The system will provide store owners with an easy-to-use admin panel to manage their products, track orders, and analyze customer behavior, allowing for smarter inventory management and personalized marketing. By digitizing the store's operations, *Campus Cart* empowers small businesses to enhance their competitiveness against larger e-commerce platforms.

Furthermore, the system can improve overall campus life by reducing congestion at local stores, promoting social distancing when necessary, and helping students better manage their limited time. Over time, the application can evolve to support multiple stores, expand into new product categories, and offer features like group orders, loyalty rewards, and personalized deals, ensuring its long-term relevance and scalability.

Thus, the development and deployment of *Campus Cart* is essential to meet the growing need for a smarter, faster, and more flexible local shopping solution that is especially suited for campus environments.

CHAPTER: 3

3. Objectives

3.1 Major Goals

The primary goal of the *Campus Cart* project is to design and develop a mobile application that simplifies the shopping experience for college students and hostel residents by enabling easy product ordering and scheduled in-store pickups. The project seeks to provide a cost-effective, efficient, and user-friendly digital platform that benefits both customers and local store owners.

By removing the dependency on traditional delivery models and reducing in-store congestion, the app aims to create a smarter and faster shopping solution tailored to the unique environment of college campuses and their surrounding areas.

Additionally, the project aims to empower small and medium-sized businesses by providing them with modern digital tools for order management, inventory control, and customer engagement, ultimately improving operational efficiency and customer satisfaction.

3.2 Specific Objectives

- **Develop a Flutter-based Mobile Application:**

  Build a cross-platform app (Android and iOS) using Flutter with a clean, intuitive, and user-friendly interface suitable for students and general customers.

- **Enable Product Browsing and Cart Management:**

  Allow users to browse through different categories of products, add items to the cart, and modify their selections before checkout.

- **Implement Secure Online Payment System:**

  Integrate Razorpay to facilitate secure, quick, and reliable online payment processing, reducing dependency on cash transactions and minimizing in-store payment failures.

- **Provide Scheduled Pickup Slots:**

  Offer customers the option to select convenient pickup times, thereby reducing waiting time, managing store crowding, and improving the overall shopping experience.

- **Create an Admin Panel for Store Management:**

Develop an admin dashboard that enables store owners to easily add, edit, and manage products, track orders, and monitor pickup schedules.

- **Ensure Real-Time Data Updates:**

  Use Firebase Firestore for real-time synchronization of orders, inventory updates, and order status notifications, ensuring smooth communication between users and store owners.

- **Enhance Customer Convenience:**

  Enable customers to place orders from the comfort of their homes or hostels, thereby saving travel time and ensuring hassle-free shopping.

- **Offer Purchase History and Insights:**

  Maintain customer purchase histories to help store owners personalize offers and recommend frequently bought items, enhancing user engagement.

- **Ensure Scalability and Future Expansion:**

  Design the app architecture in a way that supports easy expansion to multiple stores, different product types, group ordering features, and potential loyalty programs in future versions.

CHAPTER: 4

4. Relevance of the Study

4.1 Importance for Students

Students, particularly those living in hostels or off-campus housing, often face significant challenges when it comes to purchasing daily essentials like groceries, snacks, toiletries, and stationery. Their hectic academic schedules, time constraints, and limited transportation options make it difficult to frequently visit multiple stores. The traditional shopping process requires students to allocate extra time for store visits, stand in queues, and sometimes deal with payment failures due to network issues, all of which add unnecessary stress to their busy lives.

The *Campus Cart* mobile application offers a highly relevant solution to these problems. By enabling students to browse a wide variety of products online, place orders from the comfort of their rooms, make secure payments digitally, and select convenient pickup slots, the app helps save valuable time and effort. Students no longer need to waste time traveling between stores or standing in long lines, making their shopping experience far more efficient and stress-free.

Additionally, the platform's real-time updates and easy-to-navigate interface ensure that students are always informed about their order status, minimizing confusion and wait times. By streamlining the shopping process, *Campus Cart* contributes to better time management for students, allowing them to focus more on their academic and personal development. In environments where maintaining social distancing is necessary, scheduled pickups also promote safer, contactless shopping experiences.

Thus, *Campus Cart* addresses a critical need among students by offering them a fast, reliable, and convenient shopping solution tailored to their unique lifestyle.

4.2 Importance for Local Businesses

For local businesses, especially small store owners operating near colleges and universities, traditional sales methods often come with their own set of challenges. Managing in-store customer flow, handling cash transactions, dealing with long queues, and ensuring customer satisfaction without the aid of modern digital tools can limit their growth and efficiency. Additionally, competing against large e-commerce platforms that offer the convenience of online shopping becomes a daunting task for local vendors with limited resources.

Campus Cart presents an invaluable opportunity for local businesses to modernize their operations without the need for heavy investments in logistics or delivery infrastructure. By providing a

digital storefront where store owners can list their products, track orders in real time, manage inventory, and receive online payments securely, the app significantly boosts operational efficiency. It allows businesses to serve more customers in less time by facilitating scheduled pickups, reducing in-store congestion, and minimizing order management errors.

Moreover, with access to customer purchase histories and insights, store owners can better understand buying patterns and offer personalized promotions or loyalty rewards. This ability to tailor services to customer preferences helps foster stronger relationships, ensuring better customer retention and increasing overall revenue.

By adopting *Campus Cart*, local businesses gain a competitive edge by embracing digital transformation. They can expand their customer base, improve customer satisfaction, and streamline store operations, all while maintaining affordability. In the long term, this digital adaptation can lead to sustainable business growth and stronger integration within the campus community.

CHAPTER: 5

5. Methodology

The development of Campus Cart followed a structured and modular approach to ensure smooth functionality, scalability, and user-friendliness. A combination of modern technologies and best practices was utilized to design, develop, test, and deploy the application effectively. Each component of the project was carefully planned to deliver a seamless experience for both customers and store owners.

5.1 Frontend Development (Flutter)

The frontend of the Campus Cart mobile application was developed using Flutter, an open-source UI software development kit (SDK) by Google. Flutter was chosen for its capability to build highly responsive, attractive, and cross-platform applications with a single codebase for both Android and iOS devices.

Key features of the frontend development process included:

- **User Interface (UI) Design:**
  The app was designed using Flutter's Material Design components, ensuring a clean, modern, and user-friendly interface that is intuitive even for first-time users.
- **Authentication System:**
  Firebase Authentication was integrated to manage user login and registration securely. Users can sign in using email/password authentication methods, ensuring data security and privacy.
- **Navigation and Responsiveness:**
  The app's layout was designed to provide smooth navigation across different sections like product browsing, cart management, order history, and user profile.
- **Real-Time Order Status:**
  Users receive real-time updates on the status of their orders, such as "Order Placed", "Order Ready for Pickup", and "Order Completed"

By leveraging Flutter's flexible widget system, the application ensures consistent performance across different device sizes and operating systems.

5.2 Backend Development (Firebase Firestore)

For the backend, Firebase Firestore was utilized as the primary database service due to its scalability, real-time capabilities, and seamless integration with Flutter applications.

The backend responsibilities included:

- **Product and Inventory Management:**
  Storing product listings, categories, prices, and availability statuses within Firestore collections.
- **User and Order Data Management:**
  Handling user profiles, order history, and pickup schedules through secure, structured Firestore documents.
- **Real-Time Data Synchronization:**
  Firestore's real-time update feature ensures that changes in orders, product availability, or pickup slots are instantly reflected on the users' devices.
- **Security and Data Protection:**
  Firebase's security rules were applied to ensure that users could only access their own information and that sensitive data remained protected.

This setup provided a robust, scalable, and highly available backend architecture, essential for handling the dynamic needs of the application.

5.3 Payment Integration (Razorpay)

A crucial part of the application was ensuring secure and efficient payment processing. Razorpay, one of India's leading online payment gateways, was integrated into the app to manage all financial transactions.

Key features of the payment integration included:

- **Secure Online Payments:**
  Razorpay's SDK allows customers to complete payments via UPI, debit/credit cards, net banking, and digital wallets, providing flexibility and security.
- **Transaction Validation:**
  Each transaction is validated to ensure that payments are successfully processed before order confirmation.
- **Digital Invoice Generation:**
  Upon successful payment, customers receive a digital invoice for their purchase, enhancing transparency and building trust.
- **Error Handling:**

In case of payment failures, appropriate error messages are displayed, and users are guided to retry the transaction without losing their order data.

This integration greatly enhanced the overall user experience by providing a seamless, fast, and secure payment process.

5.4 Testing and Deployment

After development, the application underwent thorough testing to ensure that it was reliable, secure, and user-friendly across various devices and conditions.

The testing process included:

- **Functional Testing:**
  Ensuring that all features like user authentication, product browsing, cart operations, payment processing, and scheduled pickup selection worked flawlessly.
- **Usability Testing:**
  Testing the app's user interface to confirm that it was intuitive, easy to navigate, and met the needs of the target audience.
- **Compatibility Testing:**
  Verifying that the application functioned properly on multiple devices with different screen sizes, operating systems (Android and iOS), and system versions.
- **Security Testing:**
  Ensuring that user data and transactions remained protected from any vulnerabilities.
- **Deployment:**
  Once thoroughly tested, the application was prepared for deployment. Plans for publishing the app on the Google Play Store and Apple App Store were outlined to make it accessible to a wide user base.

Continuous monitoring and updates are planned to maintain app performance and add new features based on user feedback.

CHAPTER: 6

6. System Analysis

The system analysis phase is crucial for evaluating the feasibility and practicality of implementing the *Campus Cart* application. The study ensures that the proposed solution is both technically viable and economically and operationally sustainable. Additionally, it helps in identifying potential challenges and provides a framework for addressing them.

6.1 Feasibility Study

A feasibility study assesses whether the *Campus Cart* project can be successfully developed and deployed within the existing constraints. The feasibility study focuses on three key aspects: **Technical Feasibility**, **Economic Feasibility**, and **Operational Feasibility**.

6.1.1 Technical Feasibility

Technical feasibility evaluates whether the current technology can support the proposed system and meet the project's requirements.

Technology Stack:

Campus Cart leverages modern and robust technologies such as Flutter for frontend development, Firebase Firestore for backend services, and Razorpay for secure payment integration. These technologies are widely used and have proven scalability and reliability in real-world applications. Flutter, being a cross-platform framework, ensures that the app works seamlessly on both Android and iOS devices with minimal code duplication, reducing development time and costs.

System Integration:

Integration between the frontend (Flutter), backend (Firebase), and payment gateway (Razorpay) has been designed to be straightforward, with ample documentation and developer support available. Real-time data synchronization via Firebase ensures smooth user interaction with minimal delays.

Scalability:

The system is designed to scale with ease. As the app grows in terms of the number of stores, products, and users, Firebase Firestore can handle the increased load, and Flutter allows for additional features and performance enhancements.

Security:

Firebase provides robust security measures like user authentication, secure database access, and HTTPS communication for payment processing, ensuring that sensitive user data and payment details remain protected.

Thus, the technical feasibility of the Campus Cart system is highly favorable, given the availability of reliable technologies, proven scalability, and secure infrastructure.

6.1.2 Economic Feasibility

Economic feasibility examines whether the project can be completed within the available budget and if it will provide a good return on investment (ROI) in the long term.

Development Costs:

Since Flutter is an open-source framework and Firebase provides a free tier, the initial costs for development will be relatively low. Additionally, the integration with Razorpay offers affordable pricing options for payment processing. Development time and resources are further reduced because the same codebase is used for both Android and iOS.

Operational Costs:

Post-launch costs will primarily include server maintenance (Firebase offers a pay-as-you-go plan), payment gateway transaction fees, and periodic updates. The app can be deployed on Google Play Store and Apple App Store with minimal cost, aside from the respective platform fees.

Revenue Generation:

The app can generate revenue through various channels, such as:

- **Subscription fees** for premium features (for stores with high inventory or customer base)
- **Transaction fees** for each payment processed via Razorpay
- **Advertisement or promotional offers** for businesses to market their products within the app

The economic feasibility of the project is promising, with low initial costs, manageable operational expenses, and several potential revenue streams, making the project financially viable.

6.1.3 Operational Feasibility

Operational feasibility assesses whether the Campus Cart system can function effectively within the organization's operating environment and meet the needs of users.

User Acceptance:

The app is designed with students in mind, offering a user-friendly interface that requires minimal training or onboarding. Given the familiarity of mobile shopping platforms among students, adoption rates are expected to be high.

Store Owners:

The admin panel of Campus Cart is simple to use, allowing local businesses to efficiently manage product listings, view orders, and track customer interactions. Training and support for store owners will be provided to ensure smooth operational workflows.

Maintenance and Updates:

Firebase and Flutter's strong developer communities will help ensure continued support for the app in terms of updates and troubleshooting. Regular updates to improve features, security, and performance can be easily rolled out via the app stores.

Scalability:

The app's backend is built to handle a growing number of users and stores. Firebase's real-time database and Flutter's architecture provide the flexibility needed to scale the system as the app grows.

Considering ease of use, strong user and store-owner engagement, and a smooth maintenance process, the operational feasibility of the project is very high.

6.2 SWOT Analysis

A SWOT analysis helps identify the internal strengths and weaknesses of the Campus Cart project, as well as external opportunities and threats that could impact its success.

Strengths

- **User-Friendly Interface:** The app's simple and clean design, developed using Flutter, provides an excellent user experience for both students and store owners.
- **Cost-Effective:** Low development and operational costs due to the use of open-source technologies and cloud-based infrastructure.
- **Scalability:** The system is designed to handle growth in users, stores, and features, which will allow the app to scale easily as demand increases.
- **Security and Reliability:** Integration with Firebase and Razorpay ensures secure transactions and reliable data storage and management.

Weaknesses

- **Initial User Acquisition:** Gaining initial traction among students and local stores may require targeted marketing and incentives, as the app's success depends on its adoption.
- **Dependency on Internet Connectivity:** Users need stable internet connectivity to use the app effectively, which could be a challenge in areas with poor internet access.

Opportunities

- **Expanding Product Categories:** The app can be extended to include a wide range of products beyond the initial focus on student essentials, such as electronics, clothing, and food delivery.
- **Partnerships with More Stores:** As the app grows, there is an opportunity to partner with multiple stores, creating a marketplace and increasing the app's value to both users and store owners.
- **Loyalty Programs and Offers:** The integration of personalized offers based on purchasing behavior can drive customer loyalty and increase repeat business.

Threats

- **Competition from Larger Platforms**: Major e-commerce platforms (like Amazon, Flipkart, etc.) already have significant market share and may introduce similar features that could overshadow Campus Cart.
- **Technological Issues:** Potential challenges with maintaining app performance and security as the user base grows.

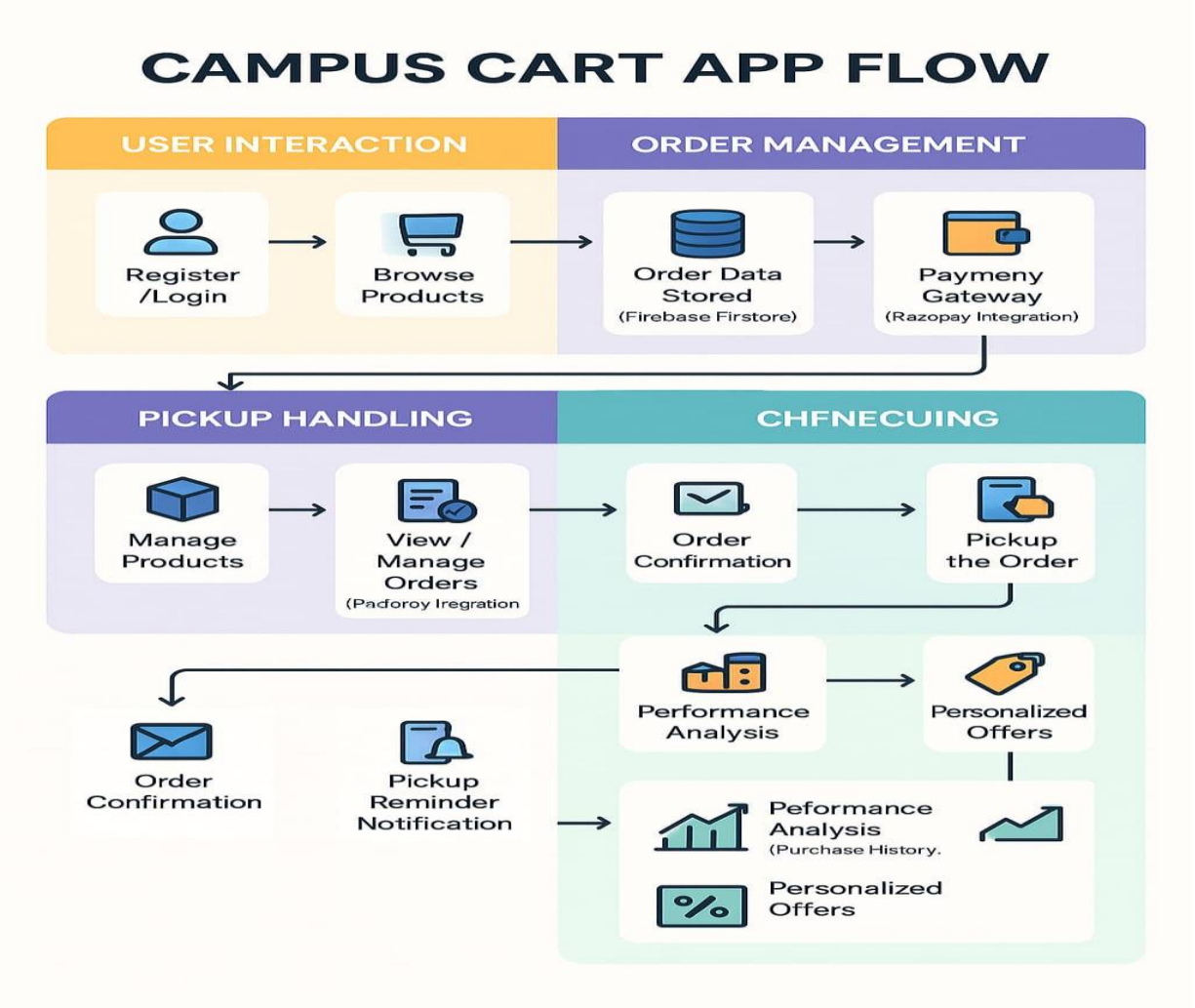
CHAPTER: 7

7. System Design

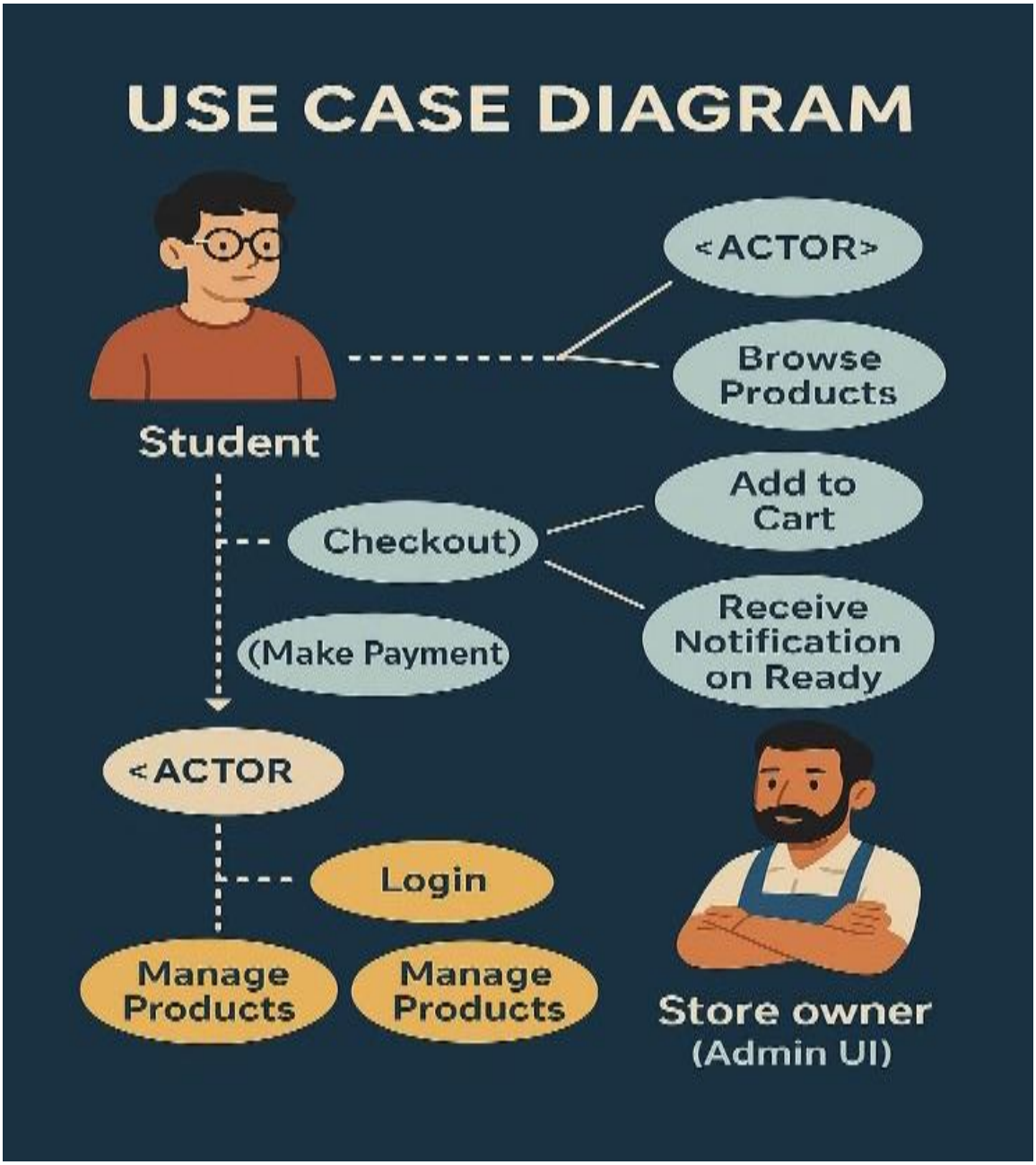
7.1 System Architecture Diagram

Description:

The Campus Cart system is composed of a cross-platform Flutter mobile app, an admin-side web panel, Google Firebase (Authentication + Firestore) as the backend, Razorpay for payments, and a Notification Service (e.g. Firebase Cloud Messaging). Users and store owners interact via their UIs, data flows into Firestore, payments go through Razorpay, and order updates trigger push notifications.



7.2 Use Case Diagram



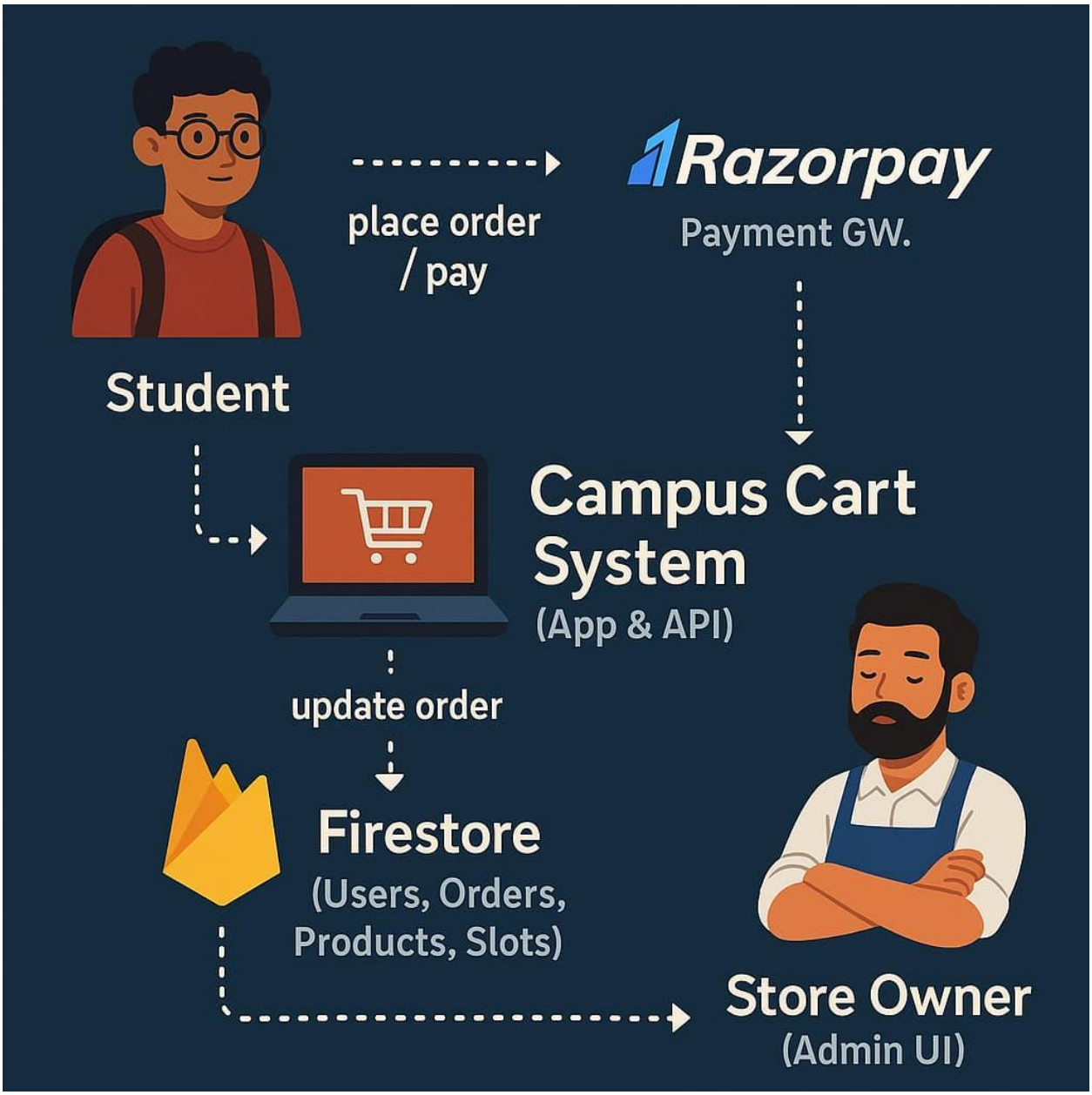
Two primary actors interact with the system:

- **Student (Customer)**
- **Store Owner (Admin)**

Use cases cover user management, product browsing, order processing, and admin maintenance.

7.3 Data Flow Diagram (Level 0)

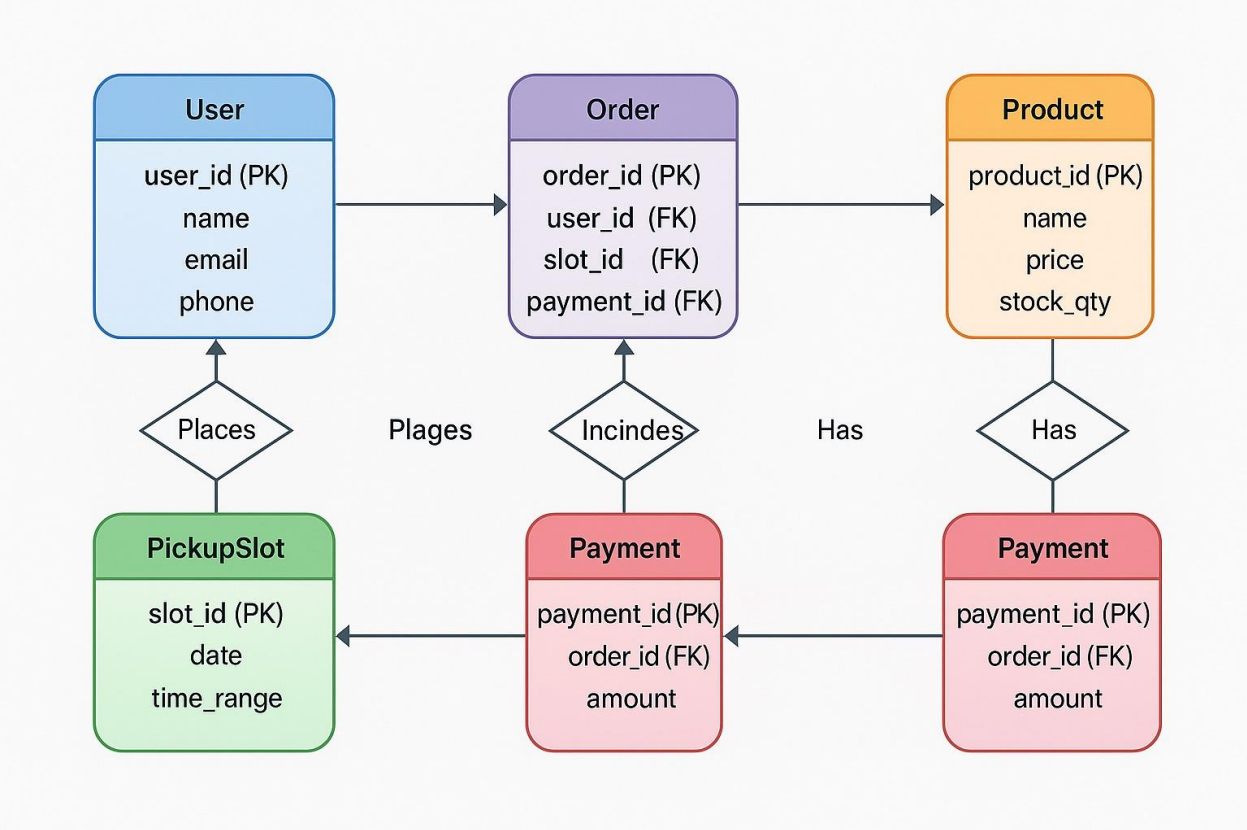
Description:
Shows high-level data flows between external entities (Student, Store Owner, Payment Gateway), the Campus Cart system, and its data stores.



7.4 Entity–Relationship Diagram (ERD)

Description:
Defines the key entities, their attributes, and relationships.

- **User–Order:** One user can place many orders.
- **Order–Product:** Many-to-many (modeled via a junction table in implementation).
- **Order–Payment:** One-to-one.
- **Order–PickupSlot:** One-to-one.



CHAPTER: 8

8. Implementation

The *Campus Cart* app combines a well-thought-out frontend with a robust backend to provide a seamless shopping experience for students and local store owners. The app's core features are built with Flutter for the frontend, Firebase for the backend, and Razorpay for payment processing, ensuring a smooth, efficient, and secure system. Below are the details of the app's implementation.

8.1 Frontend Features

The frontend of Campus Cart is designed with Flutter, enabling a smooth, user-friendly experience across both Android and iOS platforms. The mobile app's interface follows Material Design principles, ensuring that users can easily navigate and interact with the app. Key frontend features include:

User Authentication:

The app provides secure user login and registration using Firebase Authentication. Students can sign up using their email address and password, with the option to log in again securely via Firebase's authentication system.

Product Browsing:

The app displays a catalog of products, categorized by type (e.g., groceries, snacks, toiletries, stationery). Users can easily search for items, view product details, and add them to the shopping cart.

Cart Management:

Once a user has added products to the cart, they can modify the cart by adding or removing items. The cart shows the total price, including taxes, and allows users to review the items before proceeding to checkout.

Pickup Slot Selection:

Users can select a preferred pickup time from available slots, which helps manage store traffic and minimizes wait times. The app allows users to see the available pickup slots and choose the one that fits their schedule.

Order Confirmation & Payment:

After selecting a pickup slot, the user proceeds to payment. The Flutter app integrates Razorpay for secure payment processing (detailed in the next section). Upon successful payment, the user is shown a confirmation screen and is notified about the status of their order.

Real-Time Notifications:

Using Firebase Cloud Messaging (FCM), the app sends real-time notifications to users about their order status, such as "Order Placed", "Order Ready for Pickup", or "Order Completed."

8.2 Backend Features

The backend of the Campus Cart app is powered by Firebase, providing real-time data synchronization, user authentication, and secure data storage. Firebase Firestore serves as the database for storing user, order, product, and pickup data. Key backend features include:

User Management:

Firebase Authentication handles the login and registration process. Users can sign in securely, and their session data is maintained using Firebase's secure authentication protocols.

Product and Inventory Management:

The product data, including descriptions, prices, and availability, is stored in Firebase Firestore. Store owners can easily update the product catalog to reflect stock changes. Products are organized in collections for easy retrieval.

Order Management:

When a user places an order, it is saved in the Firestore database. The backend tracks the order's progress, including payment status and pickup slot selection. The system can also send real-time updates to the user's app via Firebase.

Real-Time Data Synchronization:

Firebase Firestore provides real-time synchronization of data between the user app and the backend. When an order is updated (e.g., "Order Ready for Pickup"), the data is immediately reflected in the app without needing to refresh.

Pickup Slot Scheduling:

The backend tracks available pickup slots. When a user selects a slot, it is reserved in the database, ensuring that other users cannot book the same slot. This helps in managing store capacity efficiently.

Transaction Handling:

Firebase stores transaction data and integrates seamlessly with Razorpay to track payment statuses. After successful payment, the backend updates the order status, triggering notifications to the user.

8.3 Admin Panel Description

The Campus Cart app includes an Admin Panel (a web-based interface) designed for store owners to manage their products, orders, and customer interactions efficiently. Key features of the admin panel include:

Product Management:

The admin can add, edit, or delete products from the catalog. They can modify prices, stock quantities, and descriptions. This allows the store to keep the inventory updated in real-time.

Order Management:

The admin can view all customer orders in real time. They can track the status of each order (e.g., pending, confirmed, ready for pickup). Admins can also confirm the pickup once the product is ready for collection.

User Management:

Admins can view user profiles, including their order history and preferences. This allows store owners to analyze buying behavior and personalize future offers or promotions for loyal customers.

Pickup Slot Control:

The admin has the authority to define available pickup slots for customers. The system will automatically reserve and prevent users from selecting already-booked slots.

Analytics and Reports:

The admin panel provides basic analytics such as total sales, most popular products, and customer demographics. This feature helps store owners understand customer needs and optimize their product offerings.

Order Notifications:

Admins can send notifications to customers to confirm order details, remind them about pickups, or inform them of any changes in availability.

8.4 Payment Gateway Integration

The integration of Razorpay for secure payment processing is a critical component of the Campus Cart system. Razorpay provides a seamless, secure, and user-friendly payment experience for both customers and store owners.

Payment Options:

Razorpay supports a variety of payment methods, including:

- Debit/Credit Cards
- UPI (Unified Payments Interface)
- Net Banking
- Digital Wallets (e.g., Google Pay, PhonePe)

Payment Flow:

- The user proceeds to checkout after reviewing their cart and selecting a pickup slot.
- Razorpay's secure payment page is shown to the user, where they can select their preferred payment method.
- Once the user makes a payment, Razorpay processes the transaction and provides a success or failure status.
- If the payment is successful, the system updates the order status and notifies the user that the order has been confirmed and is ready for pickup.
- If the payment fails, the user is prompted to retry the payment or choose an alternative method.

Transaction Security:

Razorpay uses SSL encryption to secure payment data and two-factor authentication (2FA) for added security. All financial transactions are securely processed and monitored to prevent fraud.

Post-Payment Actions:

After payment is successfully completed, Razorpay sends a callback to the backend, updating the order status. The system also generates a digital receipt for the user, ensuring transparency and accountability.

8.5 Code

```
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'screens/auth_check.dart';
// import 'dart:io' show Platform;
// import 'package:flutter/foundation.dart' show kDebugMode;
import 'utils/logger_util.dart';
import 'services/auth_service.dart';
import 'services/connectivity_service.dart';
import 'widgets/no_internet_dialog.dart';
// import 'screens/login_screen.dart';

void main() async {
  // Ensure Flutter is initialized
  WidgetsFlutterBinding.ensureInitialized();

  try {
    LoggerUtil.info('Initializing Firebase...');
    await Firebase.initializeApp(
      options: const FirebaseOptions(
        apiKey: 'AIzaSyDCu_xwxG4_grimF6C5UX0E4QcJrizuLnQ',
        appId: '1:563465632369:android:bf99bf9472ae89b9ee39d0',
        messagingSenderId: '563465632369',
        projectId: 'campuscart-473e1',
        storageBucket: 'campuscart-473e1.appspot.com',
      ),
    );
    LoggerUtil.info('Firebase initialized successfully');

    // Create admin user if it doesn't exist
    try {
      LoggerUtil.info('Attempting to create admin user...');
      final authService = AuthService();

      // First check if admin exists
      final adminExists = await authService.isAdmin();
      LoggerUtil.info('Admin exists check: $adminExists');

      if (!adminExists) {
        LoggerUtil.info('Creating new admin user...');
        await authService.createAdminUser(
          'admin@store.com',
          'Admin@123',
          'Store Admin',
```

```
        );
        LoggerUtil.info('Admin user created successfully');
      } else {
        LoggerUtil.info('Admin user already exists');
      }

      LoggerUtil.info('Admin user setup completed');
    } catch (e) {
      LoggerUtil.error('Error in admin setup', e);
      LoggerUtil.error('Detailed admin creation error: ${e.toString()}');
    }
  } catch (e) {
    LoggerUtil.error('Error initializing Firebase', e);
    LoggerUtil.error('Firebase initialization error: ${e.toString()}');
  }

  runApp(const MyApp());
}

class MyApp extends StatefulWidget {
  const MyApp({super.key});

  @override
  State<MyApp> createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  late ConnectivityService _connectivityService;
  bool _showNoInternetDialog = false;
  bool _isRetrying = false;

  @override
  void initState() {
    super.initState();
    _connectivityService = ConnectivityService();
    _checkInitialConnection();
    _listenToConnectivityChanges();
  }

  Future<void> _checkInitialConnection() async {
    final hasConnection = await _connectivityService.checkConnection();
    setState(() {
      _showNoInternetDialog = !hasConnection;
    });
  }
```

```
void _listenToConnectivityChanges() {
  _connectivityService.connectivityStream.listen((hasConnection) {
    setState(() {
      _showNoInternetDialog = !hasConnection;
      if (hasConnection) {
        _isRetrying = false;
      }
    });
  });
}

Future<void> _retryConnection() async {
  setState(() {
    _isRetrying = true;
  });

  // Wait for 2 seconds
  await Future.delayed(const Duration(seconds: 2));

  final hasConnection = await _connectivityService.checkConnection();

  if (mounted) {
    setState(() {
      _showNoInternetDialog = !hasConnection;
      _isRetrying = false;
    });
  }
}

@override
void dispose() {
  _connectivityService.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return MaterialApp(
    title: 'Campus Cart',
    debugShowCheckedModeBanner: false,
    theme: ThemeData(
      // This is the theme of your application.
      //
      // TRY THIS: Try running your application with "flutter run". You'll see
      // the application has a purple toolbar. Then, without quitting the app,
      // try changing the seedColor in the colorScheme below to Colors.green
```

```
      // and then invoke "hot reload" (save your changes or press the "hot
      // reload" button in a Flutter-supported IDE, or press "r" if you used
      // the command line to start the app).
      //
      // Notice that the counter didn't reset back to zero; the application
      // state is not lost during the reload. To reset the state, use hot
      // restart instead.
      //
      // This works for code too, not just values: Most code changes can be
      // tested with just a hot reload.
      colorScheme: ColorScheme.fromSeed(seedColor: const Color(0xFF2D3250)),
      useMaterial3: true,
    ),
    home: Stack(
      children: [
        const AuthCheck(),
        if (_showNoInternetDialog)
          Positioned.fill(
            child: Container(
              color: Colors.black54,
              child: Center(
                child: NoInternetDialog(
                  onRetry: _retryConnection,
                  isRetrying: _isRetrying,
                ),
              ),
            ),
          ),
      ],
    ),
  );
 }
}
```

CHAPTER: 9

9. Testing

Testing is a critical phase in the development lifecycle of any application. The purpose of testing is to ensure that the *Campus Cart* app functions as expected, is secure, user-friendly, and free of defects. A thorough testing strategy has been adopted to identify and resolve any issues early,

ensuring that the app is reliable and robust before deployment. Below are the key testing components.

9.1 Testing Strategy

The testing strategy for the *Campus Cart* application follows a structured approach, incorporating various testing methodologies to evaluate both functionality and performance.

Types of Testing:

Unit Testing:

Each individual module (such as product browsing, cart management, and payment integration) was tested to ensure that they work correctly in isolation. These tests were conducted using the Flutter testing framework, ensuring that each function performs as intended.

Integration Testing:

This testing checks if different modules of the app work together as expected. For example, the interaction between the frontend (Flutter), backend (Firebase Firestore), and payment system (Razorpay) was thoroughly tested to ensure data flows correctly between them.

Functional Testing: The functionality of the app was verified by simulating real-world usage scenarios. This type of testing ensures that all features like user registration, cart management, order placement, and payment processing work seamlessly.

UI/UX Testing:

The user interface was tested to confirm that it is intuitive and user-friendly. The navigation flow was also verified to ensure that users can perform actions like browsing products, checking out, and selecting pickup slots with ease.

Security Testing:

Security testing was conducted to ensure that sensitive information such as user credentials, payment details, and personal data are protected. We tested for potential vulnerabilities such as SQL injection, cross-site scripting (XSS), and session hijacking.

Performance Testing:

The app's performance was tested under different loads, including how well the app responds when multiple users are browsing, placing orders, or making payments simultaneously.

Usability Testing:

Usability testing was done to ensure that the app provides a good experience for students and store owners, with easy navigation, clear instructions, and minimal friction.

End-to-End Testing:

The complete user flow from registration to payment and order pickup was tested end-to-end to ensure the app works as expected in all scenarios.

9.2 Test Cases

Below are some of the key test cases executed during the testing phase:

Test Case 1: User Registration and Login

- **Objective:** Ensure that users can register and log in successfully.
- **Steps:**
    1. Open the app.
    2. Click on the "Sign Up" button.
    3. Enter valid user details (email, password, etc.).
    4. Click "Submit".
    5. Verify that the user is registered successfully and logged in.
- **Expected Result:** The user should be able to register and be redirected to the home screen.

Test Case 2: Product Browsing

- **Objective:** Verify that users can browse products correctly.
- **Steps:**
    1. Open the app and log in.
    2. Browse product categories.
    3. Select a product.
    4. Check product details such as price, description, and availability.
- **Expected Result:** Products should be displayed correctly with all relevant details.

Test Case 3: Cart Management

- **Objective:** Ensure that products can be added to, removed from, and updated in the cart.
- **Steps:**
    1. Select a product and add it to the cart.
    2. Open the cart and verify the item is present.
    3. Remove the item from the cart.
    4. Verify that the cart is updated.
- **Expected Result:** The cart should be updated correctly after each action.

Test Case 4: Payment Processing

- **Objective:** Verify that payments are processed correctly using Razorpay.
- **Steps:**
    1. Proceed to checkout.
    2. Select a payment method (e.g., UPI, Credit Card).
    3. Complete the payment.
    4. Verify that the order is confirmed after successful payment.
- **Expected Result:** Payment should be processed successfully, and the order status should be updated to "Order Confirmed."

Test Case 5: Admin Panel Functionality

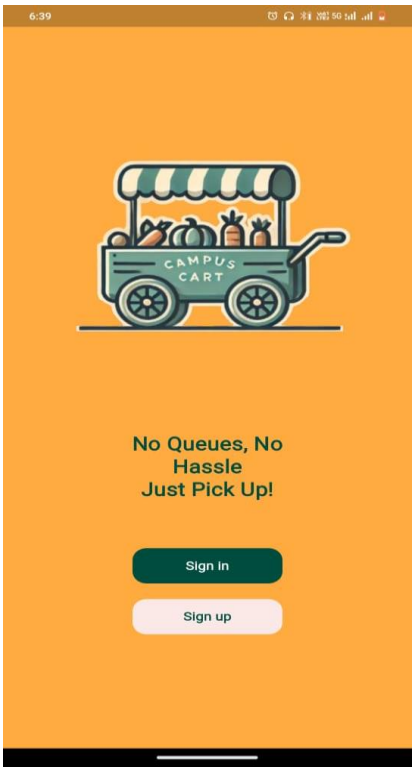
- **Objective:** Verify that store owners can manage products and view orders via the admin panel.
- **Steps:**
    1. Log in to the admin panel.
    2. Add a new product to the catalog.
    3. View customer orders.
    4. Confirm the pickup for an order.
- **Expected Result:** The admin should be able to successfully manage products, view orders, and update the order status.

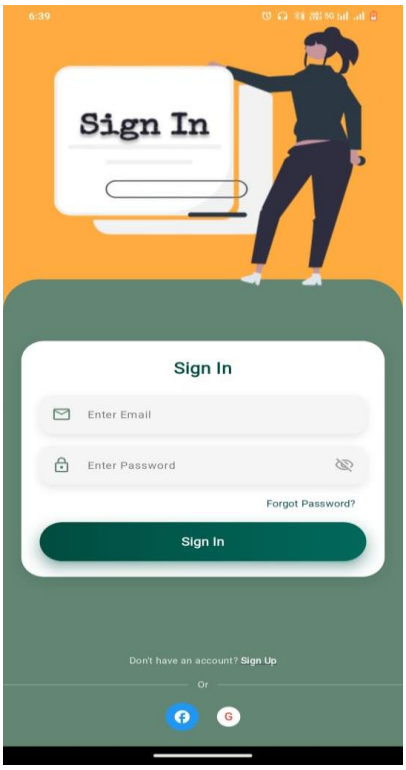
9.3 Test Results

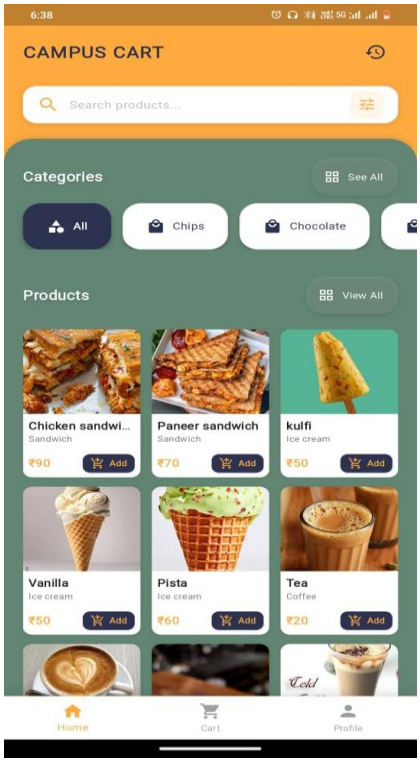
Here are the results for the key test cases that were executed during testing:

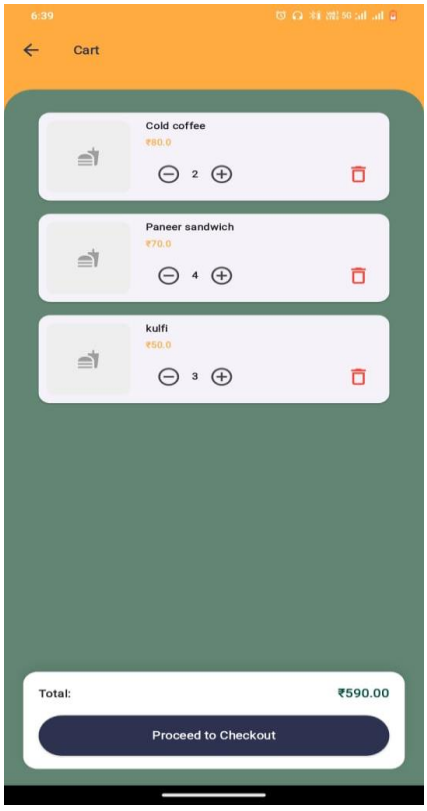
Test Case ID	Test Case Description	Result	Comments
TC001	User Registration and Login	Pass	User registered and logged in successfully.
TC002	Product Browsing	Pass	Products displayed correctly with details.
TC003	Cart Management	Pass	Items added/removed correctly from the cart.
TC004	Payment Processing	Pass	Razorpay integration successful.
TC005	Admin Panel Functionality	Pass	Admin successfully added products and managed orders.

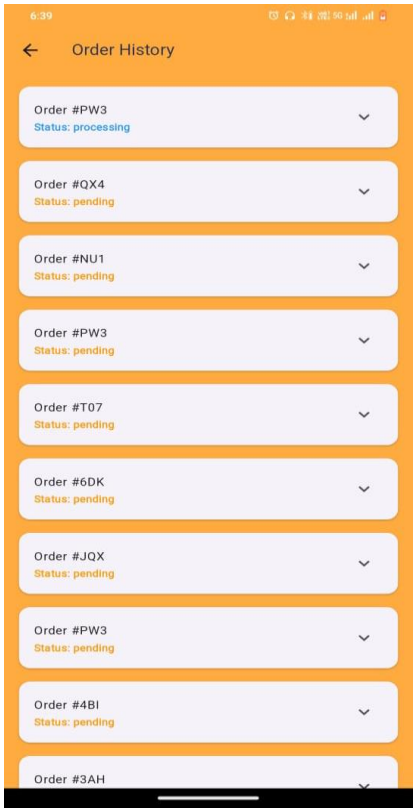
CHAPTER: 10 OUTPUT











CHAPTER: 11

11. Challenges and Limitations

11.1 Technical Challenges

- **Cross-Platform Consistency:** Ensuring uniform performance across Android and iOS required platform-specific tweaks in Flutter.
- **Real-Time Synchronization:** Managing live data updates with Firebase Firestore presented challenges in handling network latency and data conflicts.
- **Payment Integration:** Securely integrating Razorpay for payments involved adhering to strict security protocols and ensuring smooth transaction flows.
- **Notification Delivery:** Implementing Firebase Cloud Messaging (FCM) for timely order updates required handling retries and managing user subscriptions efficiently.

11.2 Financial Constraints

- **Development Costs:** Firebase and Flutter are cost-effective for small-scale projects, but scaling up, including storage and transaction fees, incurs additional costs.
- **App Store Fees:** Publishing on both Google Play and the Apple App Store requires one-time and annual fees.
- **Marketing Costs:** Reaching users requires investment in advertising and incentives to drive app adoption.

11.3 User Engagement Challenges

- **Initial Adoption:** Attracting both students and store owners to the platform requires clear value propositions and incentives.
- **Retention:** Keeping users engaged with personalized offers and incentives to maintain app usage.
- **Trust:** Ensuring reliable transactions and order fulfillment to build and maintain user trust.

11.4 Future Improvements

- **Group Orders & Split Payments:** Enabling multiple users to share orders, particularly for groups or roommates.
- **Loyalty Programs:** Implementing rewards or discounts based on purchase history to increase engagement.
- **AI Recommendations:** Using user data for product suggestions and personalized offers.
- **Offline Mode & PWA:** Supporting offline browsing and creating a Progressive Web App for better accessibility.
- **Multi-Store Support:** Expanding the app to support multiple stores and services within the campus.

CHAPTER: 12

12. Expected Outcomes

12.1 Benefits for Students

The *Campus Cart* application is tailored to address the unique needs of college students and hostel residents, delivering significant advantages in terms of convenience, time management, and overall satisfaction. By enabling students to place orders remotely and select precise pickup slots, the app eliminates the need to navigate busy store aisles or wait in long queues, thereby freeing up valuable time for academic pursuits and extracurricular activities. The digital interface also allows users to compare prices and products at their own pace, ensuring they make informed choices without the pressure of in-store crowds.

Secure online payment integration reduces the risk of transaction failures caused by poor network connectivity on-site and minimizes the necessity of carrying cash, enhancing both safety and peace of mind. Furthermore, the ability to schedule pickups around class schedules or study sessions helps students maintain a balanced routine, reducing last-minute rushes and stress. Over time, regular use of Campus Cart can lead to better budget management, as students can review their purchase history, monitor spending trends, and take advantage of promotions, all within a single, unified platform.

12.2 Benefits for Store Owners

For local store owners, Campus Cart provides a turnkey digital solution that streamlines daily operations and drives revenue growth. The admin panel offers real-time visibility into inventory levels and order volumes, allowing proprietors to proactively restock popular items and avoid stockouts that could disappoint customers. By automating order entry and payment reconciliation, the system reduces manual errors and administrative workload, liberating staff to focus on customer service and in-store experiences.

Scheduled pickups help smooth customer flow, minimizing peak-hour congestion and creating a safer, more manageable environment. Store owners can also leverage built-in analytics—such as most-purchased products and peak ordering times—to tailor stock and staffing levels more effectively. The platform's capacity to collect user data (with respect for privacy) enables targeted marketing campaigns, such as personalized discounts or bundled offers, which in turn can boost average order value and foster customer loyalty. Ultimately, the app empowers small and medium-sized businesses to compete with larger e-commerce players by offering a modern, efficient, and locally focused solution.

CHAPTER 13

13. Applications

13.1 Everyday Usage

Campus Cart is designed as a one-stop solution for routine shopping needs, allowing users to purchase essentials—such as groceries, toiletries, stationery, and snacks—without ever leaving their dorm rooms or lecture halls. The intuitive product catalog, complete with search and filter options, ensures that customers can quickly locate items and add them to their virtual cart. With clearly displayed pickup slots and real-time inventory updates, students can plan their shopping trips in advance, reducing interruptions to their daily schedules. Push notifications alert users when orders are ready, further streamlining the process and making everyday shopping as effortless as possible.

13.2 Group Orders

Recognizing that students often live and shop in groups, Campus Cart includes a group ordering feature that simplifies bulk purchases for shared accommodations or study sessions. One user can create a shared cart, invite roommates or friends via a link or code, and collectively select items and pickup times. The app automatically calculates each participant's share of the total cost, and split payments can be made seamlessly through individual Razorpay transactions. This collaborative approach not only fosters community but also unlocks bulk-purchase discounts and reduces per-person delivery costs, making group shopping both economical and social.

13.3 Personalized Offers through Purchase History

By maintaining a detailed record of each customer's past purchases and preferences, Campus Cart enables store owners to deliver highly relevant promotions and recommendations. The app's dashboard highlights frequently bought products, seasonal trends, and spending patterns, which can be used to generate personalized coupon codes, "frequently bought together" suggestions, and loyalty rewards. Customers receive targeted notifications—such as a discount on their most-purchased snack or an alert that a preferred item is back in stock—enhancing engagement and driving repeat business. Over time, this data-driven personalization fosters stronger relationships between students and their campus stores, creating a win-win scenario for both users and owners.

CHAPTER: 14

14. Conclusion

The *Campus Cart* project successfully demonstrates how modern mobile and cloud technologies can be harnessed to create an efficient, user-friendly shopping platform tailored to the needs of college students and local store owners. By leveraging Flutter for a consistent cross-platform user interface, Firebase Firestore for real-time data management, and Razorpay for secure payment processing, the application provides a seamless workflow from product browsing through payment and scheduled pickup.

Through rigorous system analysis, design, and testing, *Campus Cart* addresses key challenges such as long store queues, payment failures, and inventory management inefficiencies. The introduction of scheduled pickup slots and real-time notifications reduces congestion, enhances customer convenience, and streamlines store operations. The admin panel further empowers small businesses with analytics and order management tools, enabling data-driven decision-making and personalized marketing.

Looking ahead, the foundational architecture of *Campus Cart* supports future enhancements such as group ordering, loyalty programs, AI-driven recommendations, and multi-store integrations. These extensions will deepen user engagement, increase average order values, and broaden the app's applicability beyond campus environments. With ongoing maintenance, user feedback integration, and strategic partnerships, *Campus Cart* is well positioned to evolve into a comprehensive, locally focused e-commerce ecosystem.

In summary, *Campus Cart* not only simplifies everyday shopping for students but also equips local retailers with the digital capabilities needed to compete in today's marketplace. The project exemplifies how targeted technological solutions can bridge gaps between digital convenience and community-based commerce, ultimately delivering mutual benefits to consumers and businesses alike.

References

1. Flutter. (n.d.). *Flutter Documentation*. Retrieved from https://flutter.dev/docs
2. Firebase. (n.d.). *Cloud Firestore Documentation*. Retrieved from https://firebase.google.com/docs/firestore
3. Razorpay. (n.d.). *Razorpay Payment Gateway Integration Guide*. Retrieved from https://razorpay.com/docs/
4. Google. (n.d.). *Google Home*. Retrieved from https://www.google.co.in/
5. Material Design Guidelines – https://material.io/design
6. Firebase Authentication Guide – https://firebase.google.com/docs/auth
7. Firebase Cloud Messaging Guide – https://firebase.google.com/docs/cloud-messaging
8. Firebase Cloud Functions – https://firebase.google.com/docs/functions
9. Progressive Web Apps (PWA) – https://developer.mozilla.org/docs/Web/Progressive_web_apps
10. Statista. (2024). *Mobile commerce (m-commerce) share of e-commerce worldwide*. Retrieved from https://www.statista.com/statistics/806336/mobile-commerce-share-e-commerce-worldwide
11. Laudon, K. C., & Traver, C. G. (2022). *E-commerce 2022: Business, Technology, and Society* (16th ed.). Pearson.
12. Kotler, P., & Keller, K. L. (2016). *Marketing Management* (15th ed.). Pearson.
13. Kumar, V., & Reinartz, W. (2018). *Customer Relationship Management: Concept, Strategy, and Tools* (3rd ed.). Springer.
14. Shankar, V., Urban, G. L., & Sultan, F. (2002). "Mobile Marketing in the Retailing Environment." *Journal of Interactive Marketing*, 16(4), 2–21.