PRINTFLYERS : A PRINTOUT DELIVERY APP

By

ARUL PRASANTH K

2023239001

A PROJECT REPORT

Submitted for the Course

of

XC3451- SOFTWARE ENGINEERING

DEPARTMENT OF MATHEMATICS

ANNA UNIVERSITY

CHENNAI – 600025

APRIL – 2025

PRINTFLYERS

DEPARTMENT OF MATHEMATICS

COLLEGE OF ENGINEERING GUINDY

ANNA UNIVERSITY

CHENNAI – 600025

Name :	ARUL PRASANTH K		
Roll No :	2023239001	**Programme :**	M.Sc.(Integrated)
Branch :	COMPUTER SCIENCE	**Semester :**	IV

Certified to be bonafide record of work done by the above student in the ***SOFTWARE ENGINEERING*** *Theory cum Laboratory Course during the Semester IV of the academic year* ***2024-2025*** *Submitted for the Practical Examination held on* ***30-04-2025***

Lab Course Instructor **Head of the Department**

ABSTRACT :

In today's fast-paced, digitally driven world, convenience and time efficiency are essential. *PrintFlyers* is a mobile-based printout delivery application designed to solve the everyday challenges faced by students, professionals, and the general public when accessing print services. The project primarily aims to eliminate the inconvenience of physically visiting print shops, standing in queues, or dealing with shop closures—particularly in metropolitan areas and educational institutions.

The core idea behind *PrintFlyers* is to enable users to upload documents, customize print settings (such as color options, page orientation, and binding), and get them delivered directly to their doorstep. Designed especially for college students who often face difficulties finding reliable print services during peak hours or emergencies, the application also serves professionals who need urgent printouts for meetings and general users needing bulk document printing.

The system consists of four main user types: **Customers**, **Shopkeepers**, **Delivery Personnel**, and **Admins**. Customers can create accounts, upload files in various formats, select customization options, and track their orders in real-time. Shopkeepers receive and manage incoming orders, update their status, and prepare documents for delivery. Delivery personnel are assigned orders based on proximity and are responsible for delivering them efficiently. Admins oversee the entire system, ensuring smooth operations, resolving disputes, and managing performance and user data.

Key features include free delivery for a minimum of 50 pages, secure payment options (via UPI, cards, and wallets), and real-time order tracking. The app integrates GPS for accurate delivery, file encryption for secure uploads, and OTP-based login for user authentication. Notifications and alerts keep users informed about order status, promotions, and updates.

From a technical perspective, the app is compatible with Android and iOS devices, using APIs for payment processing, location tracking, and messaging. Non-functional requirements ensure performance stability, user data security, and fast processing times even under heavy load. Legal and operational requirements, including compliance with data privacy standards and a 99.9% uptime policy, make *PrintFlyers* a reliable and user-friendly solution.

CHAPTER NO. **TABLE OF CONTENTS** **PAGE NO.**

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ICMR	Indian Council of Medical Research
API	Application Program Interface
SQL	Structured Query Language
UML	Unified Modeling Language
ER	Entity Relationship
UI	User Interface
UX	User Experience
DB	DataBase
OS	Operating System

CHAPTER – 1

1 INTRODUCTION:

1.1 OBJECTIVE OF THE PROJECT

The objective of this project is to develop a printout delivery service aimed at students, professionals, and the general public. This service aims to save time for users, particularly in metropolitan cities, by eliminating the need to wait in queues at local printout centre. For instance, within a college setting, where students often face delays due to temporary closures or long waiting times at printout centres, this app will provide an efficient alternative for students to complete their printing tasks while focusing on other activities. While primarily designed for college students, the service will also be available to the general public.

1.2 OVERVIEW OF THE PROJECT

PrintFlyers is a mobile-based printout delivery application designed to simplify document printing for students, professionals, and the general public. The app allows users to upload files, customize print options (colour, sides, paper type), and get their documents delivered to their doorstep. Targeting metropolitan users who face long queues or shop closures, the app saves time and ensures convenience. Orders are only accepted for 50+ pages and delivered by assigned personnel. Shopkeepers manage incoming orders through their dashboard, while delivery staff track and update order status. Admins oversee app performance, user management, and disputes, ensuring seamless service and operational efficiency.

1.3 ORGANISATION OF THE REPORT

This project report is organized into multiple chapters, each focusing on specific aspects of the PrintFlyers application development:

- **Chapter 1 – Introduction:**

  Describes the background of the project, the motivation behind the app, and outlines the key objectives.

- **Chapter 2 – System Analysis:**

  Discusses how system requirements were collected using surveys and stakeholder feedback. It includes both functional and non-functional requirements.

- **Chapter 3 – System Design:**

  Details the architecture of the application, along with UML diagrams such as use case diagrams, class diagrams, ER diagrams, and sequence diagrams.

- **Chapter 4 – System Implementation:**

  Explains the technologies used, development tools, backend logic, database structure (PostgreSQL), and frontend (React Native) integration. Module-wise functionality is described.

- **Chapter 5 – Conclusion:**

  Summarizes the accomplishments of the project, key outcomes, and possible improvements and extensions for future versions.

- **References:**

  Lists the references used throughout the project and includes survey results, raw data, and supplementary diagrams or screenshots.

CHAPTER – 2

2 SYSTEM ANALYSIS:

2.1 FEASIBILITY REPORT:

Introduction

The project aims to create an application where customers can upload files for printout, choose customization options (black & white/colour, single/double side and have the prints delivered to their location. It is mainly targeted at college students and busy professionals in metropolitan areas who want to save time.

Types of Feasibility

Technical Feasibility

- **Availability of Technology**: The required technologies (Android Studio for mobile app, Figma for UI design, Firebase/MySQL for database) are easily available.
- **Developer Skills**: Developers have knowledge of frontend, backend, and database management.
- **Integration**: Integration of file upload, order management, and delivery tracking is technically achievable using existing APIs and frameworks.

✅ Conclusion: **Technically feasible.**

Economic Feasibility

- **Development Cost**: Low, if built in-house by students or small teams.
- **Operational Cost**: Server hosting charges, SMS/email services for order notifications, minimal promotional budget.
- **Return on Investment**: High potential if targeted properly at college campuses and offices.

✅ Conclusion: **Economically feasible** with manageable investment.

Operational Feasibility

- **User Perspective**: Simple and user-friendly UI ensures customers can easily place orders.
- **Business Perspective**: Print shop owners can easily accept, process, and manage orders using the shopkeeper panel.
- **Delivery Management**: Delivery boys can access simple assigned tasks via the app, reducing confusion.

✅ Conclusion: **Operationally feasible** with clear role segregation (customer, shop owner, delivery boy).

Legal Feasibility

- **Data Protection**: Must ensure privacy for uploaded documents.
- **Terms of Use**: Should be clear that the service is for academic/professional materials only, not for illegal content.

✅ Conclusion: **Legally feasible** with standard disclaimers and privacy policies.

Table 2.1 Team Requirements

Role	**Responsibility**
Front-End Developer	Figma , UI/UX
Back-End Developer	API & Logic
DB Administrator	PostgreSQL DB design
API Handler	Data integration

Table 2.2   Development Timeline

Phase	**Duration**
Communication	3 Weeks
Planning	2-3 Weeks

Designing	4 Weeks
Implementation	8 Weeks
Testing	1 Week

Table 2.3 Risks and it’s solution:

Risk	**Solution**
Late Deliveries	Implement real-time route optimization and allow preferred delivery slot selection
File Upload Issues	Use stable file upload APIs, allow preview, and support retry/resume functionality
Order Mismanagement	Use barcode/QR code for order tracking and verify orders before dispatch
Payment Failures or Fraud	Use trusted payment gateways, offer COD, and securely log transactions
Privacy & Data Breach	Encrypt data, auto-delete files post-processing, and restrict access to staff only
Poor User Experience	Design a user-friendly UI, add help sections, and offer prompt customer support

Summary :

Aspect	**Feasibility**
Technical Feasibility	✓ Achievable
Economic Feasibility	✓ Affordable
Operational Feasibility	✓ User-friendly
Legal Feasibility	✓ Clear compliance

Final Verdict :
✅ The Printout Delivery Application is highly feasible and recommended for development.

2.2 SOFTWARE REQUIREMENTS SPECIFICATION :

INTRODUCTION:

Purpose:

The purpose of this project is to develop a printout delivery service aimed at students, professionals, and the general public. This service aims to save time for users, particularly in metropolitan cities, by eliminating the need to wait in queues at local printout centre. For instance, within a college setting, where students often face delays due to temporary closures or long waiting times at printout centres, this app will provide an efficient alternative for students to complete their printing tasks while focusing on other activities. While primarily designed for college students, the service will also be available to the general public.

Document conventions:

- User – One who uses the application for printout (students, public).
- Shopkeeper- One who process the printing job and keep it out for delivery.
- Admin – One who take care of the app performance, safety ,revenue .
- Delivery personnel – One who delivers the printout from the shop to customers

Intended Audience and Reading Suggestions:

- **Users:** Students (for assignments, projects, and reports), Professionals (for meetings, presentations, and official work), and the General Public (for bulk printouts with home delivery convenience).
- **Shopkeepers** – Print shop businesses that want to expand their services through online orders and delivery.
- **Delivery personnel** - Individuals responsible for picking up print orders from shops and delivering them to users. They will use the app to track assigned deliveries, update order statuses, and navigate routes efficiently.
- **Administrators** – The team managing the app, handling user queries, and ensuring smooth operations.

- **Developers & Designers** – Refer to system requirements, UI/UX design, and security features for better implementation.

Project scope:

The goal of this project is to develop a mobile application that addresses the challenges faced by users in accessing printouts. The app aims to eliminate the need for users to travel to printout centres and wait in long queues, saving valuable time.

The app will provide the following key features:

- **Printout Customization**: Users will be able to choose their printout preferences, including options such as **black and white or colour** and **single or double-sided** printing.
- **Address Selection**: Users can specify a delivery address for the printouts to be sent to, whether on campus or at a different location.
- **Free Delivery**: The app will provide **free delivery** for printout orders, ensuring a cost-effective solution for users.

LIMITATIONS:

- Orders will only be accepted for a minimum of **50 pages** or more.
- Delivery may experience delays due to **traffic** or other logistical factors, particularly in **metropolitan cities**

BENEFITS:

- **Time-saving**: The app reduces the need to travel to printout centres, allowing users to engage in other tasks.
- **Convenience**: Students and professionals can place orders directly from the college or workplace, ensuring timely delivery even when they forget to print documents in advance.

References:

1. Pressman, R.S. “Software Engineering: A Practitioner Approach”, 8th Edition Revised, McGraw Hill, Chennai, 2019.
2. Sommerville, I. “Engineering Software Products”, Global Edition, Pearson Education, 2021.
3. Abrahm Silberschatz, Henry. F. Korth, S. Sudarsan “Database System Concepts”, McGraw Hill, Seventh Edition, Indian Edition, 2021.
4. Balagurusamy, “Object-Oriented Programming with C++”, Eighth Edition, McGraw Hill, 2020.

OVERALL DESCRIPTION:

Product perspective:

A database system stores the following information:

<u>Printout shop details</u>:

It includes the shop location, shop operating hours, services offered by the shop (like binding , black and white or colour), contact information of the shop etc,.

<u>User (or customer) description:</u>

It includes user name, user contact number, user address (to deliver the printout), user time slot(to deliver the product), log out their account, delete their account.

<u>Shopkeeper description:</u>

Shopkeepers can view and manage incoming orders, track deliveries and set availability etc., mark the delivered products, logout their account, delete their account.

<u>Delivery personnel description:</u>

Assigns delivery personnel to orders, tracks deliveries, and updates status.

<u>Admin description:</u>

Admins can monitor app performance, manage user accounts and view analytics on order and revenue, delete shopkeeper or user account.

Product Features:

Customer Features:

- **User Registration & Authentication** – Sign up/login using OTP, email, or phone number.
- **File Upload & Format Support** – Upload PDFs, Word documents, and images.
- **Customization Print** – Choose B/W or colour, single/double-sided, paper type, and size.
- **Minimum Order Requirement** – Orders accepted only for 50+ pages.
- **Order Tracking & Status Updates** – Real-time tracking of order progress.
- **Delivery Address Selection** – Add and save multiple delivery addresses.
- **Free Delivery** – No extra charge for eligible orders.

- **Secure Payment Options** – Pay via UPI, credit/debit cards, wallets, or COD.
- **Notifications & Alerts** – Get updates on order confirmation, delivery, and promotions.
- **Order History & Reordering** – View past orders and reorder with one click.
- **Customer Support** – Live chat and email support for queries.
- **Order Cancellation & Modification** – Modify or cancel orders within a specific time.
- **Order Tracking & Status Updates** – Real-time tracking of order progress, including delivery status updates from the assigned delivery person.

Order Management Features

- **Order Confirmation & Invoice Generation** – Auto-generated order summary and invoice.
- **Order Queue System** – Prioritizes orders based on time and shop availability.
- **Estimated Delivery Time Calculation** – Based on distance, traffic, and shop processing speed.
- **Delivery Slot Selection** – Customers can choose delivery time slots.
- **Live Delivery Tracking** – Real-time updates on package location.
- **Order Packaging Options** – Users can select packaging (flat, binding).
- **Feedback & Ratings** – Customers can rate shops and delivery experience.
- **Delivery Personnel Assignment** – Orders will be assigned to available delivery personnel based on proximity and availability.
- **Delivery Confirmation** – Delivery personnel will update the app upon successful delivery.

Shopkeeper Features

- **Shop Registration & Profile Management** – Register shops and update location, services, and timing.
- **Order Management Dashboard** – View, process, and complete incoming orders.
- **Print Job Queue –** Auto-assigns print jobs based on priority and order time.
- **Pricing Customization** – Set prices for different print services.

- **Shop Availability Status** – Toggle between open/closed to manage workload.
- **Earnings & Transaction History** – Track revenue and payments.
- **Promotions & Discounts** – Offer discounts to attract customers.
- **Customer Reviews & Feedback Management** – View and respond to user ratings.

Admin Features

- **User & Shop Management** – Add, remove, or verify users and shops.
- **Order Monitoring & Analytics** – View order trends, peak times, and revenue reports.
- **Dispute Resolution System** – Handle customer/shopkeeper complaints.
- **Performance Metrics & Feedback Analysis** – Track service quality.
- **Fraud Detection & Security** – Monitor unusual transactions and enforce policies.
- **App Content & Promotions Management** – Control banners, offers, and user notifications.
- Monitor delivery efficiency, performance, and customer feedback.
- Resolve disputes related to late deliveries or misplaced orders.

User classes and characteristics:

Users:

One who uses this app to get printout delivery.

Characteristics:

- Includes college students, office workers, and the general public.
- May need urgent printouts but lack access to a nearby print shop.
- Prefer a convenient and time-saving service instead of waiting in long queues.

Shopkeepers:

Users who process and fulfil print orders

Characteristics:

- Print shop owners or managers looking to expand their customer base.
- Need a system to manage incoming orders efficiently.
- Require a way to track earnings and business performance.

Delivery Personnel:

Users responsible for delivering the printouts to customers.

Characteristics:

- Receive delivery assignments from shopkeepers.
- Update real-time status of the delivery (e.g., "Out for Delivery," "Delivered").
- Use GPS navigation to locate delivery addresses.
- Communicate with shopkeepers or customers in case of delivery issues

Admins:

User who oversee and manages the platform

Characteristics:

- Responsible for maintaining the app, monitoring transactions, and handling disputes.
- Need real-time insights into orders, revenue, and customer activity.

Operating environment:

- Smartphones (Android/iOS) or computers with an internet connection.
- Minimum 2GB RAM and 16GB storage for smooth app performance.
- Smartphones that support **GPS functionality** and are compatible with **4G and 5G networks**.
- Stable internet connection for app usage and real-time order updates.

SYSTEM FEATURES:

User / shopkeeper registration and authentication:

- User / shopkeeper can register the account through their mail id or mobile numbers.
- User / shopkeeper can login into the account using the password they set during registration.
- User / shopkeeper can reset their password if they forgot the old password .

Shop searching:

- User can search or filter the shops by the distance.

Order placement:

- User can upload files (any format)
- Users have print customization offers.
- Users can select **delivery or pickup options.**
- Users can **choose or enter a delivery address** manually or via GPS.
- The system provides an **estimated delivery time** before order confirmation.
- Users receive **order confirmation and status updates** via notifications, SMS, or email.

Shopkeeper features:

- Can update **order status** (e.g., "Printing," "Ready for Pickup," "Out for Delivery,")
- They can view **order history and earnings reports**.
- Can communicate with admin via **in-app messaging or support chat.**
- Access to **customer reviews and ratings** for service improvement

Payment integration:

- UPI (Google Pay, PhonePe, Paytm, etc.)
- Credit/Debit Cards
- Mobile Wallets (Paytm Wallet, Amazon Pay, etc.)
- Net Banking

Delivery personnel:

- Can view assigned deliveries, update status (e.g., "Picked Up," "Delivered").
- Access navigation tools for optimized routes and estimated delivery times.
- Can communicate with customers/admin via in-app chat or call support.
- Track earnings, delivery history, and receive customer feedback for improvement.

Admin dashboard:

- View, edit, and remove customers, shopkeepers, and delivery personnel.
- Manage user authentication and account verification**.**
- Approve or reject shop registrations.
- Monitor shop activity, including order completion rates and ratings.
- Handle disputes related to pricing, service quality, and customer complaints.

- Track delivery personnel activity, including order completion time, delivery performance, and user ratings.
- Resolve delivery-related complaints.

EXTERNAL INTERFACES:

User interfaces:

- Login & Registration Screen
- Home Screen
- File Upload Screen
- Print Customization Screen
- Shop Selection Screen
- Order Confirmation & Payment Screen
- Order Tracking Screen
- Order History Screen
- Profile & Settings Screen

Shopkeeper Interface

- Login & Dashboard
- Order Management Screen
- Pricing & Service Settings
- Earnings & Payment Dashboard
- Customer Communication Screen

Delivery personnel interface

- Login & Dashboard
- Order Pickup List
- Real-Time Route Navigation
- Order Delivery Confirmation Screen
- Earnings & Payment Dashboard

Hardware Interfaces

- The app will function on smartphones and tablets with iOS and Android.
- The app will need GPS functionality for location-based searches and item availability.

Software Interfaces

- Payment Gateway API: Integration with Stripe or PayPal for handling transactions securely.

- Location API: Google Maps or similar API for mapping locations and item availability.
- Push Notifications: Firebase Cloud Messaging or similar services for real-time updates.

OTHER NON-FUNCTIONAL REQUIREMENTS

Performance Requirements

- The system should handle **at least 1000 concurrent users** without performance degradation.
- File uploads should be processed within **5 seconds f**or files up to **50MB.**
- Order placement should not take more than **3 seconds** after confirmation.
- The app should work efficiently even on a **4G network with limited bandwidth**.
- Delivery tracking updates should be reflected in real-time within 2 seconds of status change.

Safety requirements:

- The app should prevent file corruption or loss during upload and processing.
- In case of system failure, order data should be recoverable without user intervention.
- The system should log all critical transactions for auditing purposes.

Security requirements:

- All personal data, files, and transactions must be encrypted using AES-256 encryption.
- Payment transactions should comply with PCI-DSS security standards.
- Shopkeepers should only access their own order data and not see other shops' orders.
- User authentication should be secured with OTP-based login.
- Delivery personnel should only access delivery-related information (order ID, customer address, and contact details) while assigned to an active order.

Usability Requirements

- The app should be easy to navigate for students, with minimal on boarding required.
- Accessibility features, such as screen reader support and high-contrast modes, should be implemented.

OTHER REQUIREMENTS:

Legal Requirements

- Users must agree to the Terms and Conditions and Privacy Policy before using the app.

Operational Requirements

- The system should be available 99.9% of the time, excluding scheduled maintenance periods.
- Regular backups should be conducted to ensure data safety and disaster recovery.
- The customer support team should respond to user inquiries within 24 hours.

2.3 TOOLS AND TECHNOLOGY

Table 2.4 Tools and Technologies

TOOLS AND TECHNOLOGY	DESCRIPTION
Figma	A collaborative interface design tool used to create and prototype the app's user interface for all user roles, generated frontend code snippets were later customized in React Native.
Visual Studio Code	Visual Studio Code, also commonly referred to as VS Code, is a source-code editor developed by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add functionality.

	Visual Studio Code is based on the Electron framework, which is used to develop Node.js web applications that run on the Blink layout engine.
PostgreSQL	PostgreSQL is a powerful, open-source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. PostgreSQL has earned a strong reputation for its proven architecture, reliability, data integrity, robust feature set, extensibility, and the dedication of the open-source community behind the software to consistently deliver performant and innovative solutions. PostgreSQL runs on all major operating systems, has been ACID-compliant since 2001, and has powerful add-ons such as the popular PostGIS geospatial database extender. It is no surprise that PostgreSQL has become the open-source relational database of choice for many people and organisations.
Umbrello	A UML modelling tool used to design system architecture and generate use case, class, and sequence diagrams for the project
React Native	A JavaScript-based open-source framework used for developing mobile applications that work seamlessly on both Android and iOS platforms with shared code.
Express.js	Express.js is a minimal, server-side web application framework for Node.js, released under the MIT License. It provides a thin yet powerful layer of fundamental web features routing, middleware support, and HTTP utility methods, without obscuring the underlying Node.js APIs. Express.js uses a middleware pattern to handle requests and responses, and it can be organized around the MVC paradigm when desired. It encourages the use of web standards such as JSON or XML for data exchange and HTML, CSS, and JavaScript for user interfaces. In addition to its lightweight core, Express.js emphasizes modular design, the DRY principle, asynchronous non-blocking I/O, and seamless integration with ORMs (e.g.,

	Mongoose, Sequelize) or templating engines (e.g., Pug, EJS).
JavaScript	JavaScript is an interpreted, high-level, general-purpose programming language which supports multiple programming paradigms. It was initially created to enable interactive web pages and has since become one of the core technologies of the World Wide Web, powering both front-end and back-end applications. In JavaScript, functions are first-class objects and it uses a prototype-based object model rather than classical inheritance. JavaScript is dynamically typed and features automatic memory management and just-in-time compilation in modern engines. It supports multiple programming paradigms, including event-driven, functional, and imperative programming.

CHAPTER – 3

3 SYSTEM DESIGN

3.1 SYSTEM ARCHITECTURE:

1. Client Layer (Frontend – React Native Mobile App):

- Register/Login
- Profile Management
- Order Placement
- Order History
- Notifications
- Settings/Logout

2. API Gateway Layer (Backend – Node.js + Express):

- Authentication Controller (JWT, bcrypt)
- User Controller (CRUD operations)
- Order Controller (Place/View Orders)
- File Upload Middleware (Multer)
- Notification logic
- Token validation middleware

3. Database Layer (PostgreSQL):

- users table
- orders table

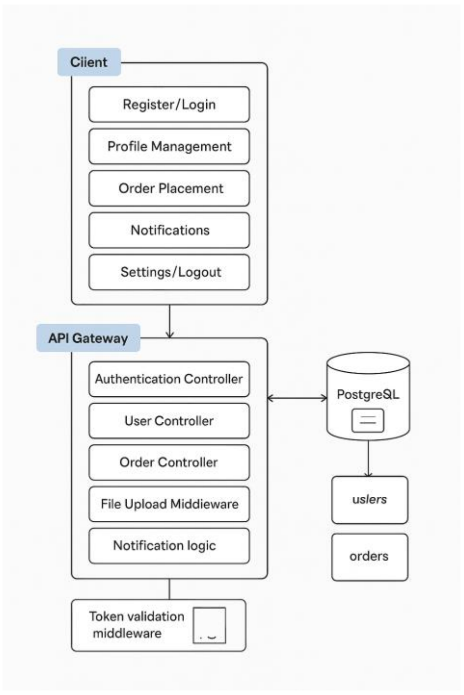


FIGURE 3.1 System Architecture

3.2UML DIAGRAMS :

- o Unified Modelling Language (UML) diagrams serve as a standardized visual representation to model and document software systems and processes.
- o UML offers a versatile set of diagrams, each serving a specific purpose in illustrating different aspects of a system's architecture, behaviour, and interactions.
- o UML diagrams provide a standardized and clear means of communication among developers, stakeholders, and team members.

3.2.1 Use-Case Diagrams:

In the Unified Modelling Diagram (UML), a use case diagram for this project summarizes the details of system's users (also known as actors) and their interactions with the system.

In our project, the main actions (use cases) performed by three types of users in a system—**User**, **Owner**, and **Delivery Man**—likely for a **print and delivery service platform.**

1. USER

These are the end-users (customers) who use the service to upload files for printing and delivery:

- **Sign Up / Log In / Forgot Password / Verify Password**: Basic account management.
- **Upload Files**: Upload documents for printing.
- **Select Print Options**: Choose specifications like number of copies, colour/BW, size, etc.
- **Place Orders**: Submit the order for printing and delivery.
- **Track Orders**: Monitor the status of placed orders.
- **Proceed Payments**: Make payments for the service.
- **Give Feedbacks**: Provide reviews or ratings.

---

2. OWNER

These are the shop owners or service providers managing the print and delivery operations:

- **Manage Orders**: Oversee and process incoming orders.
- **Set Availability**: Indicate when the service is available.
- **Manage Users**: Administer customer accounts or permissions.
- **View Analytics**: Access data and insights on usage, revenue, etc.
- **Print Flyers** (<<include>>): A specific print operation available in the system.

---

3. DELIVERY MAN

Responsible for delivering the printed items to users:

- **Sign Up / Log In / Forgot Password / Verify Password**: Account management for delivery staff.
- **Receive Delivery Notification**: Get alerted about assigned deliveries.
- **Update Delivery Status**: Mark delivery progress (e.g., dispatched, delivered).
- **Navigate to Location**: Use maps or directions to reach the customer.

- **View Assigned Deliveries**: Check all current delivery tasks.
- **Print Flyers** (<<include>>): May also print flyers if needed.

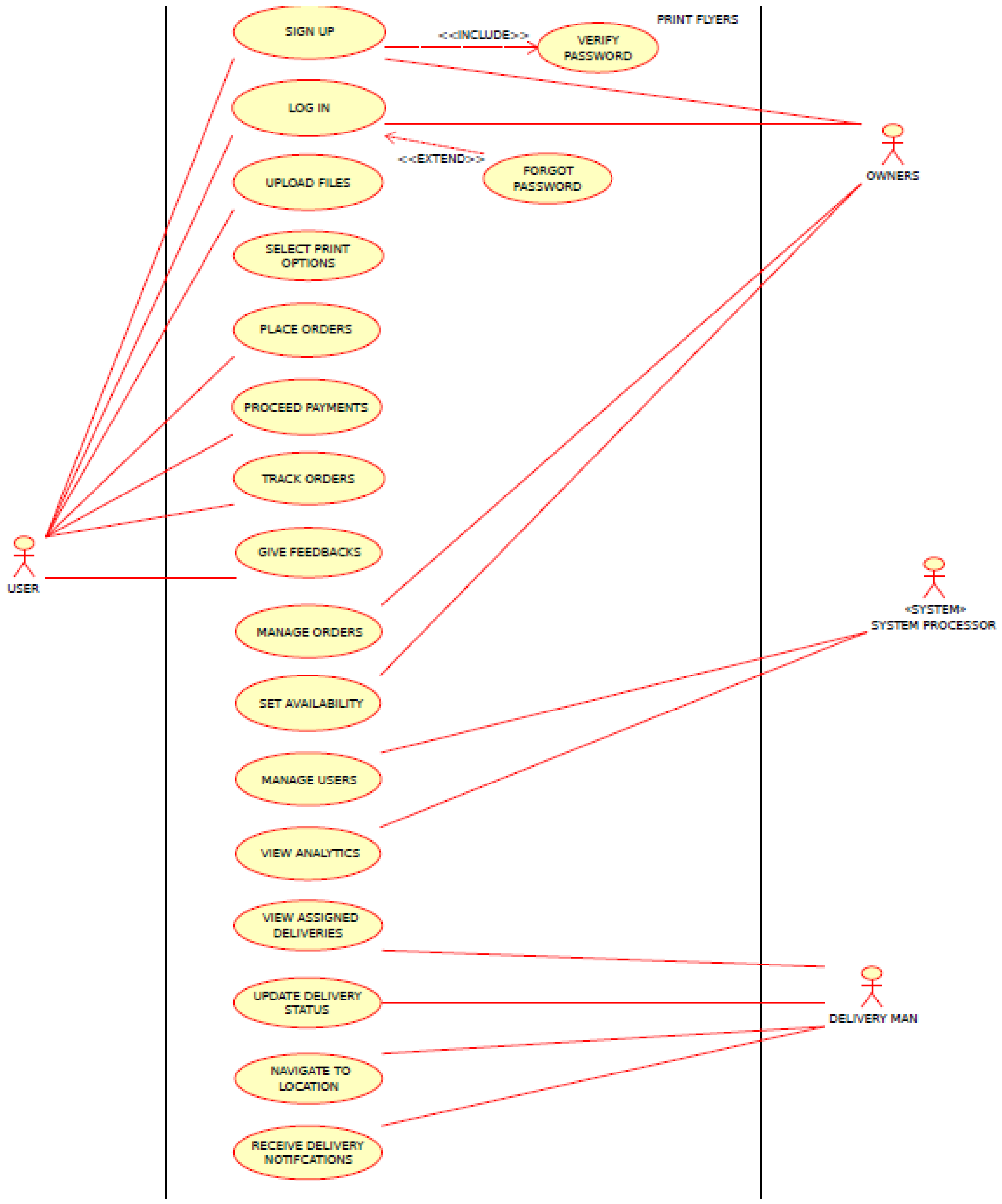


FIGURE 3.2 Use-Case Diagram-1

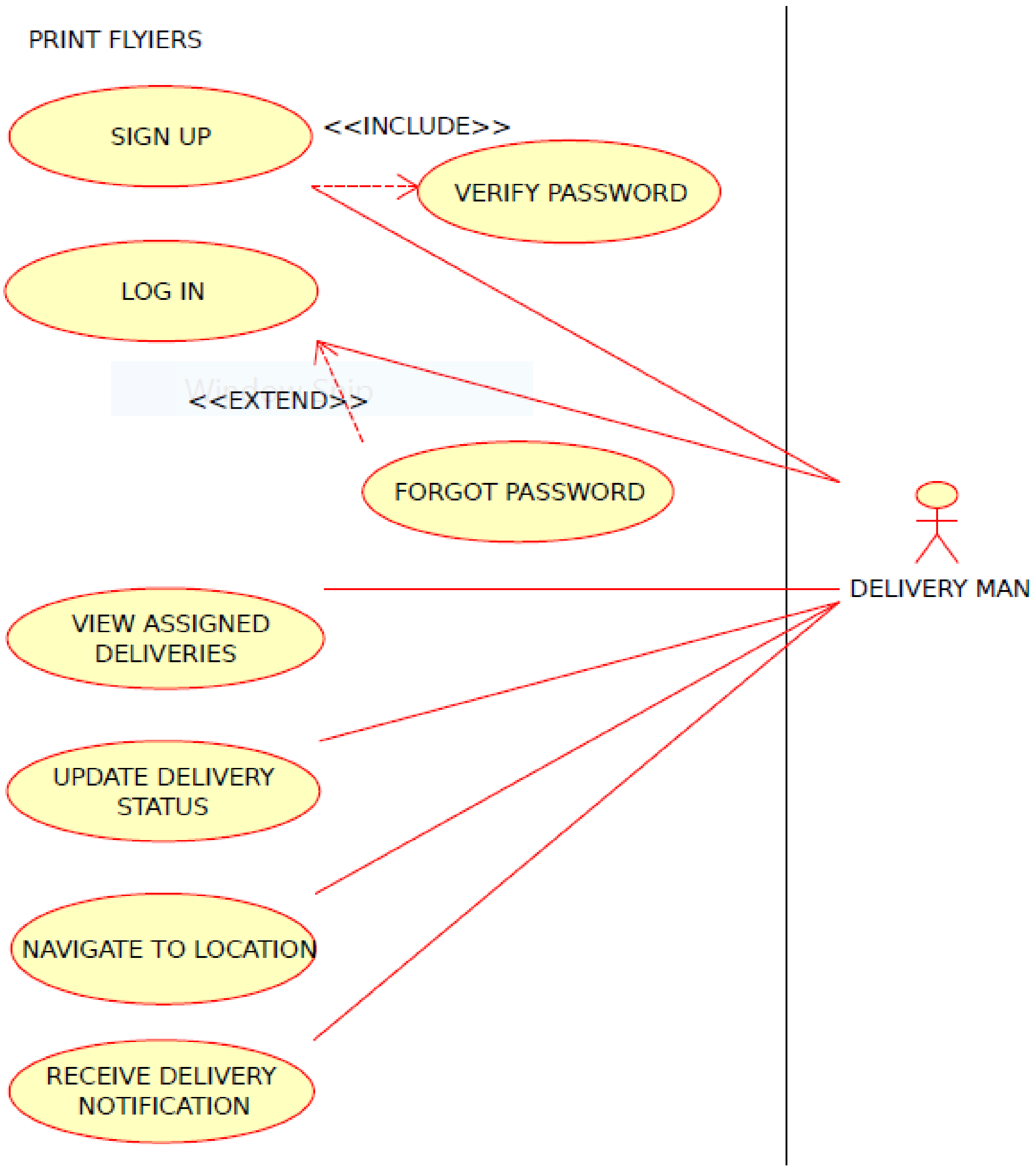


FIGURE 3.3 Use-Case Diagram-2

3.2.2 CLASS DIAGRAM:

Class diagram gives the static view of an application. A class diagram describes the types of objects in the system and the different types of relationships that exist among them. This modelling method can run with almost all Object-Oriented methods. UML class diagram gives an overview of a software system by displaying classes, attributes, and their relationships.

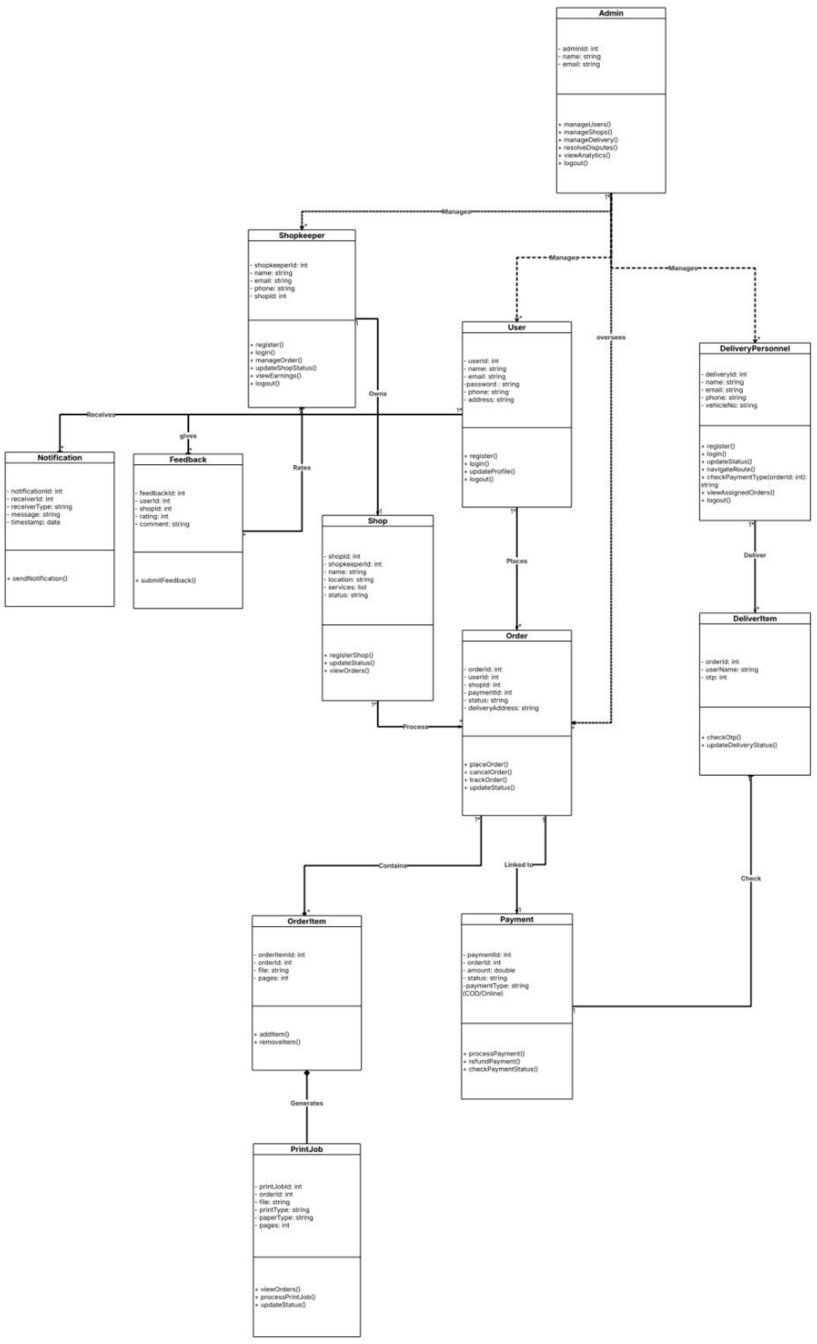


FIGURE 3.4 Class Diagram-2

3.2.3 SEQUENCE DIAGRAM:

Asequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. These diagrams are used by software developers and business professionals to understand the requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

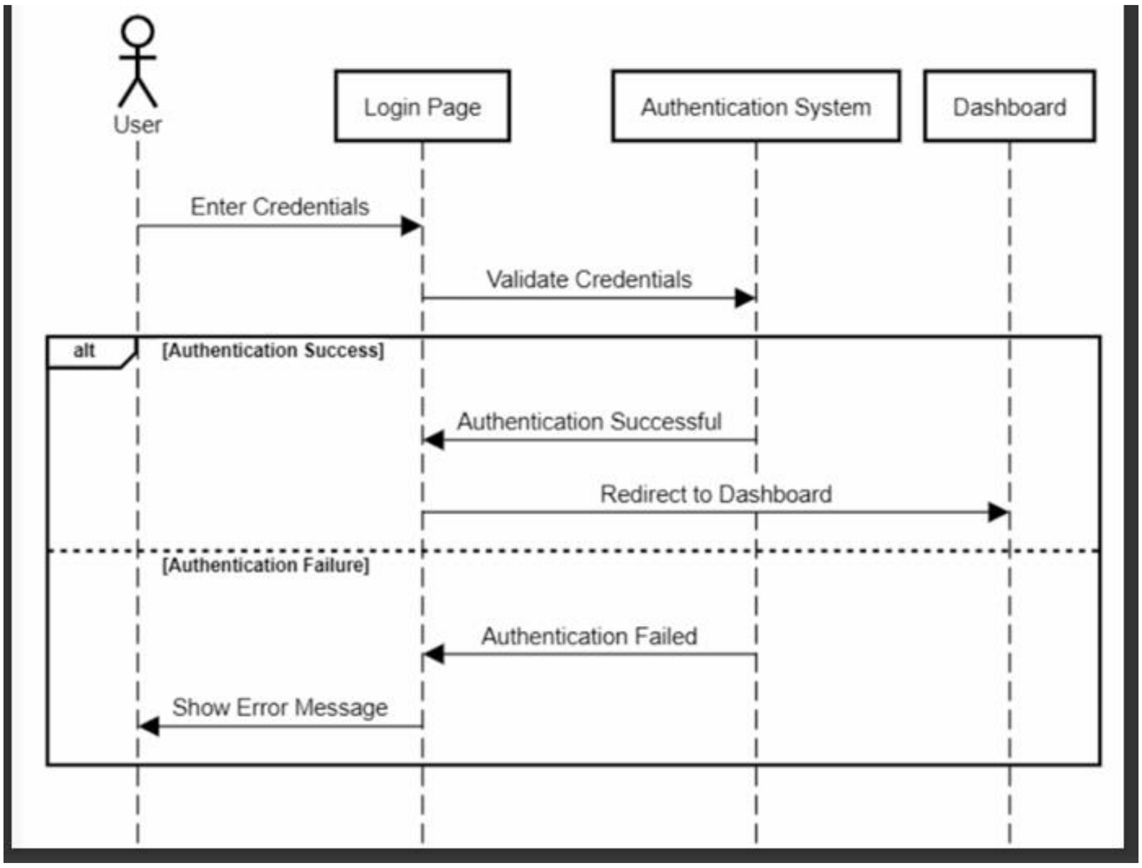


FIGURE 3.5 Sequence Diagram-1

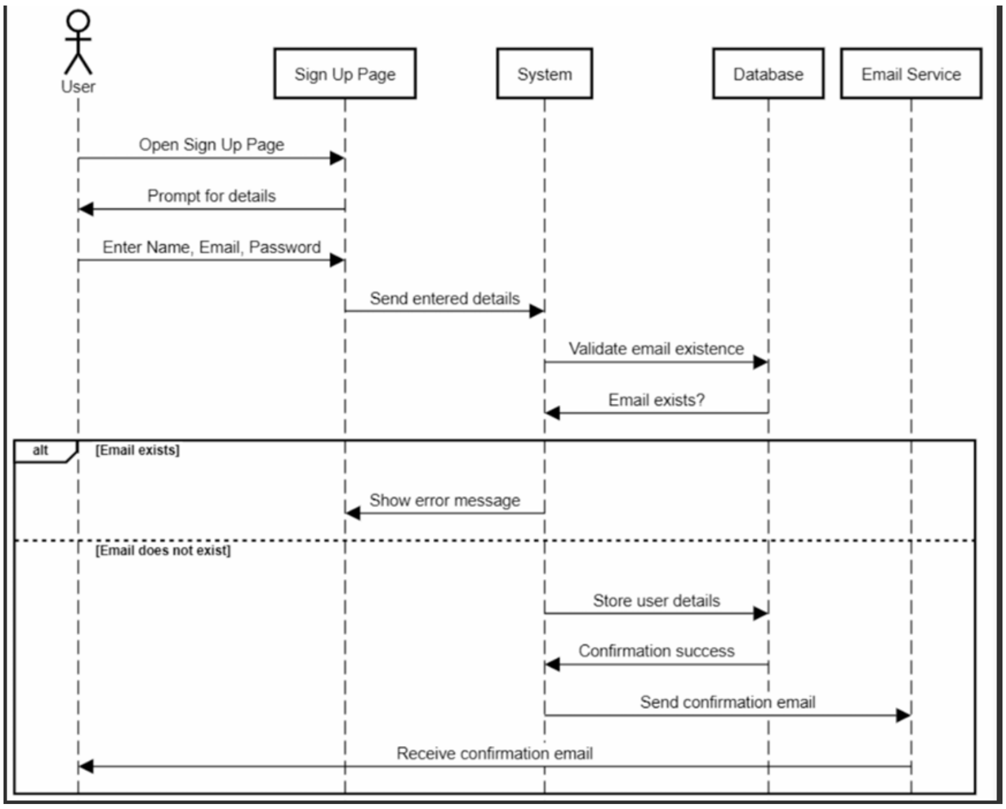


FIGURE 3.6 Sequence Diagram-2

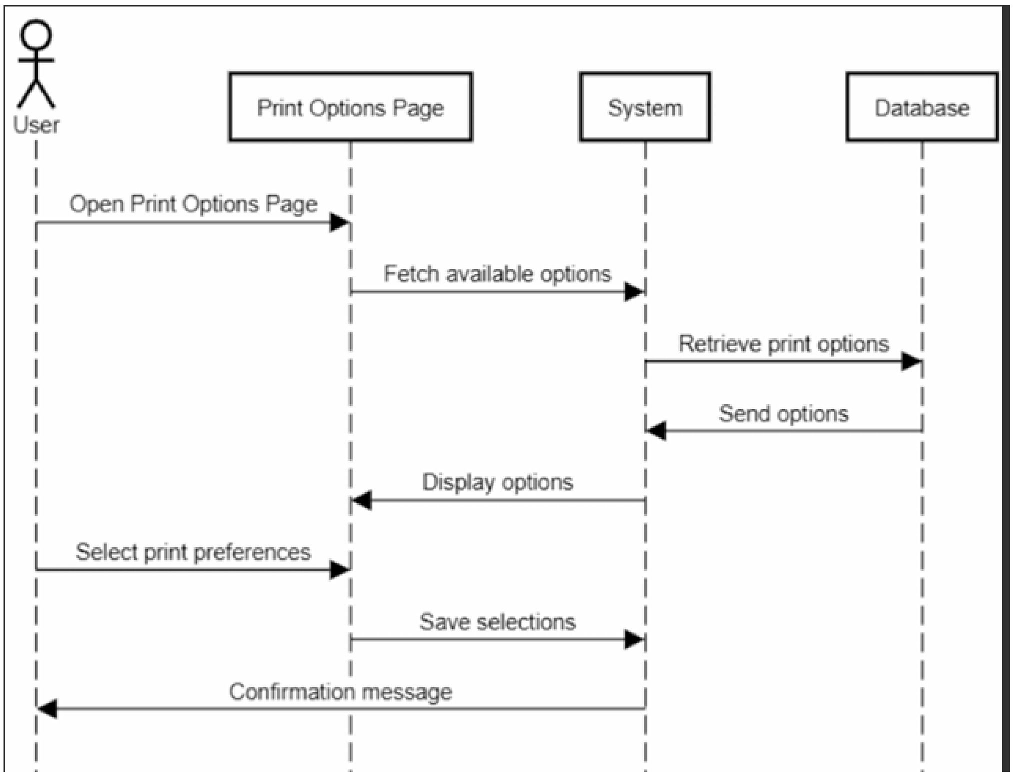


FIGURE 3.7 Sequence Diagram-3

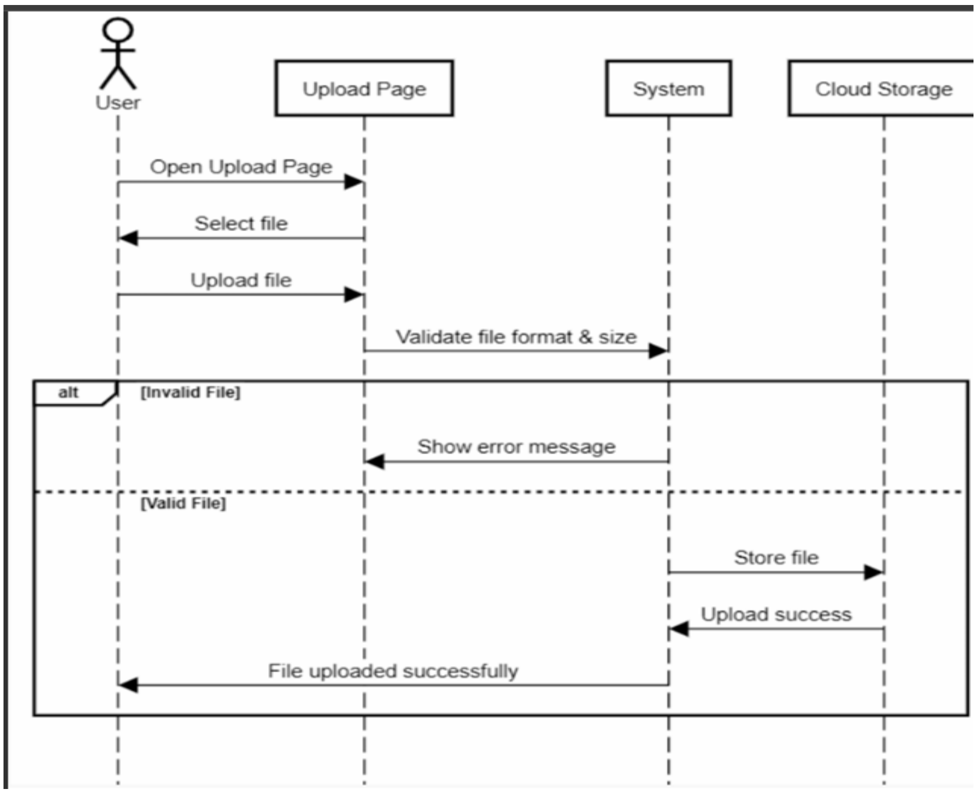


FIGURE 3.8 Sequence Diagram-4

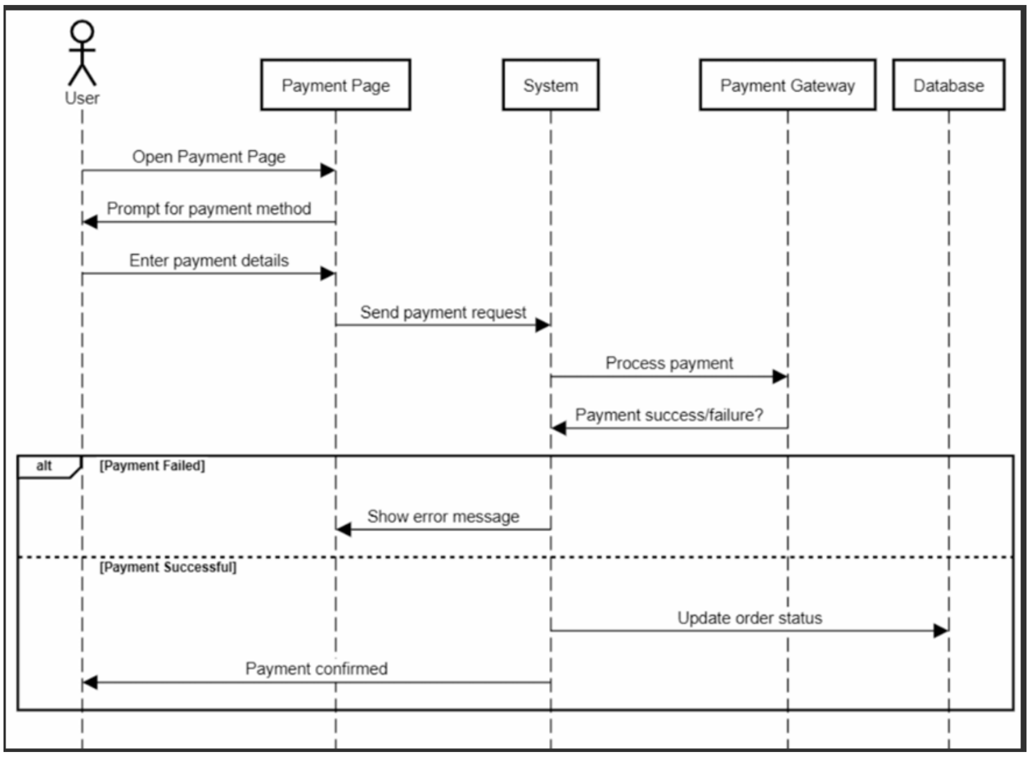


FIGURE 3.9 Sequence Diagram-5

3.3 OTHER DIAGRAMS:

3.3.1 SCHEMA DIAGRAMS

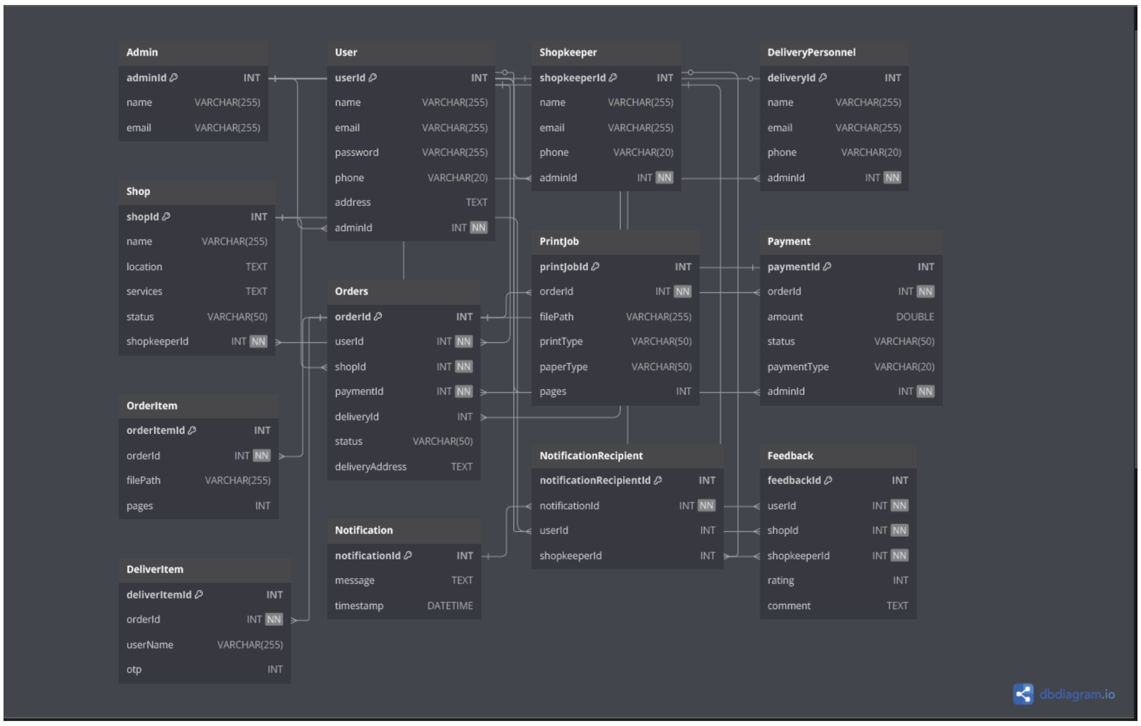


FIGURE 3.10 SCHEMA DIAGRAM-1

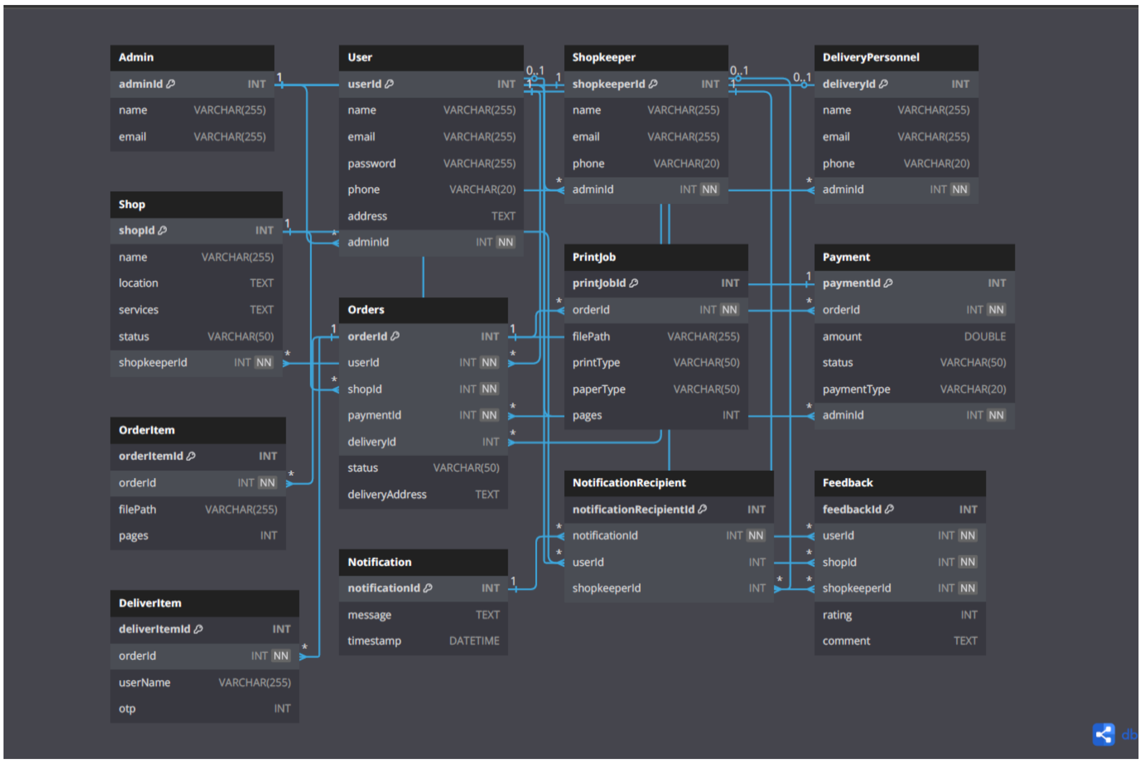


FIGURE 3.11 SCHEMA DIAGRAM-2

CHAPTER - 4

4. SYSTEM IMPLEMENTATION:

System implementation is the process of defining how the information system should be built (i.e.., physical system design), ensuring that the information system meets quality standard (i.e.., quality assurance). Systems implementation is a set of procedures performed to complete the design (as necessary) contained in the approved systems design document.

4.1 SYSTEM MODULES:

In the development of this project, the system was designed in a modular structure to separate responsibilities and ensure scalability, clarity, and maintainability. Each module serves a specific function and interacts with others through well-defined interfaces.

Modules Developed by Me (User-Side)

As part of this project, I was responsible for implementing all core modules that power the **user-side experience**. These modules enable regular users to register, log in, manage profiles, place orders, and interact seamlessly with the platform.

1. User Authentication Module:

I implemented the entire user authentication flow using React Native for the frontend and Node.js/Express for the backend.

- Designed and coded the Register and Sign In screens with proper validation.
- Integrated password visibility toggle using eye icons for better UX.
- Connected the frontend with backend APIs to register and authenticate users.
- On the backend, I developed secure APIs using Express, and added JWT-based login token handling.
- Used bcrypt to hash passwords and prevent plain-text storage.

This module ensures that each user has a secure and seamless login/registration experience.

Relevant Database Table:

Column Name	**Data Type**	**Description**
id	SERIAL	Primary key

Column Name	Data Type	Description
username	VARCHAR	Unique identifier for the user
mobile	VARCHAR	User's contact number
password_hash	TEXT	Encrypted password using bcrypt
profile_image	TEXT (URL)	Path to the uploaded profile image
created_at	TIMESTAMP	Account creation timestamp

2. Profile Management Module:

I designed and developed the complete user profile screen, allowing users to view and update their personal information.

- Built a profile screen that fetches user data via token-authenticated API.
- Enabled profile picture upload using image pickers and connected it to the backend using Multer middleware.
- Provided real-time feedback (e.g., success/error messages) to the user.
- Created backend logic to securely update user data, such as name, mobile number, and location.
- This module gives users full control over their identity and contact information within the system.

3. Order Placement and Tracking Module:

I implemented the entire user-side ordering functionality.

- Designed the **order creation UI** where users can submit new requests (e.g., Xerox delivery or service requests).
- Connected this UI to backend APIs for submitting and storing orders.
- Built the user’s **order history screen**, where users can track the status of each order (Pending, Accepted, Rejected, Completed).
- Integrated real-time updates for tracking order progress via API responses.

This module enables users to easily place and monitor service requests in a user-friendly interface.

Relevant Database Table:

Column Name	Data Type	Description
id	SERIAL	Primary key
user_id	INTEGER	Foreign key referencing users(id)
order_details	JSON / TEXT	Description of the order (e.g. files, pages)
status	VARCHAR	Order status (Pending, Accepted, Completed)
created_at	TIMESTAMP	Timestamp of order placement

4. Notification and Status Update Module (User):

I developed a notification mechanism where users are informed about changes in order status.

- Integrated alert messages and conditional rendering to show users when their order is accepted or completed.
- Used polling mechanisms or API-based refreshes to fetch real-time order updates.
- Ensured that status changes are accurately reflected on the user's order screen.

This improves the user's awareness of their service status and increases platform reliability.

Relevant Database Usage:

- Notifications are indirectly tied to the orders table via the status field and timestamps.

5. Settings and Logout Module:

I created the user settings interface and logout functionality.

- Implemented a **logout feature** that clears the user's session token and redirects to the login screen.
- Designed UI controls for future additions like **dark mode** or notification toggles.
- Ensured smooth transition and state reset upon logout.

This module handles user preferences and ensures secure session termination.

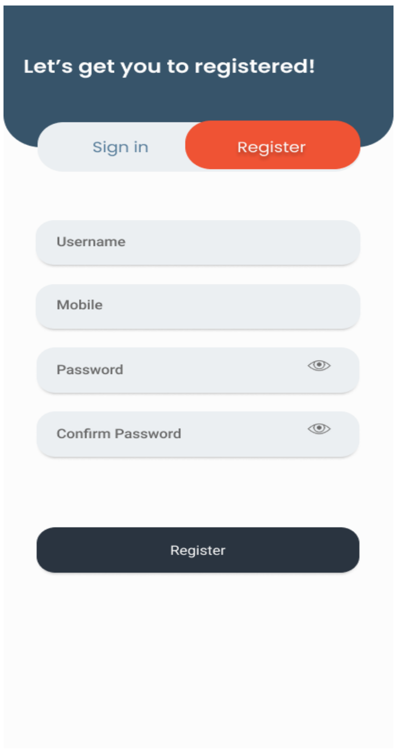
6. API Communication and Integration Module

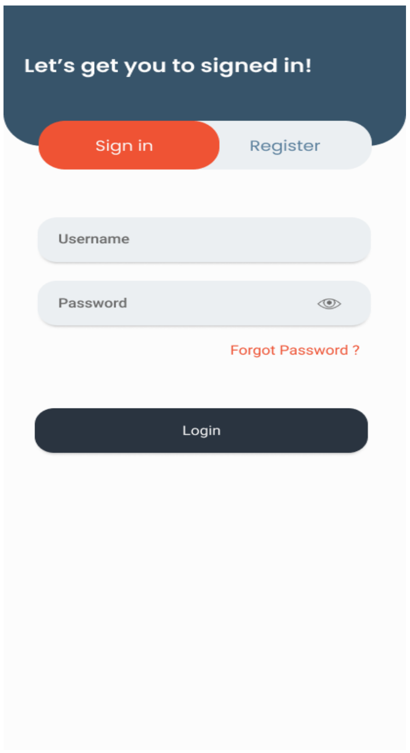
I built reusable functions to connect the frontend to the backend securely and efficiently.

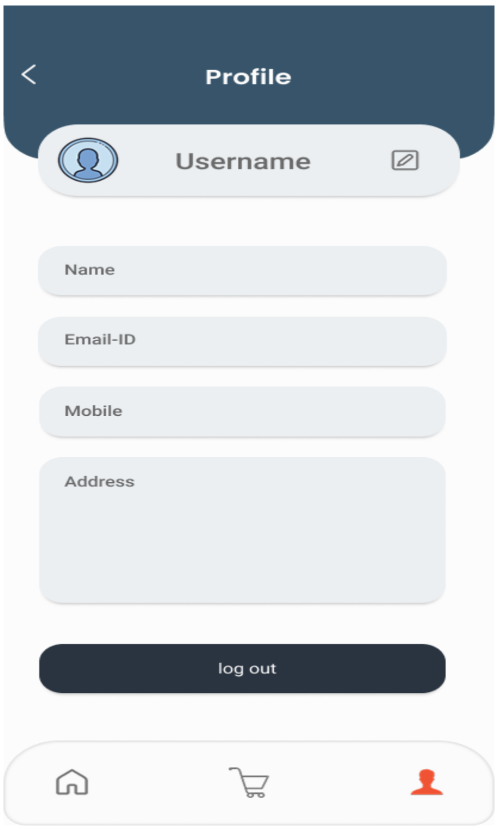
- Developed **API service layers** using fetch to send/receive data from Express backend.
- Handled all authentication headers, error responses, and loading states manually.
- Ensured robust communication between user screens and backend modules like orders and authentication.

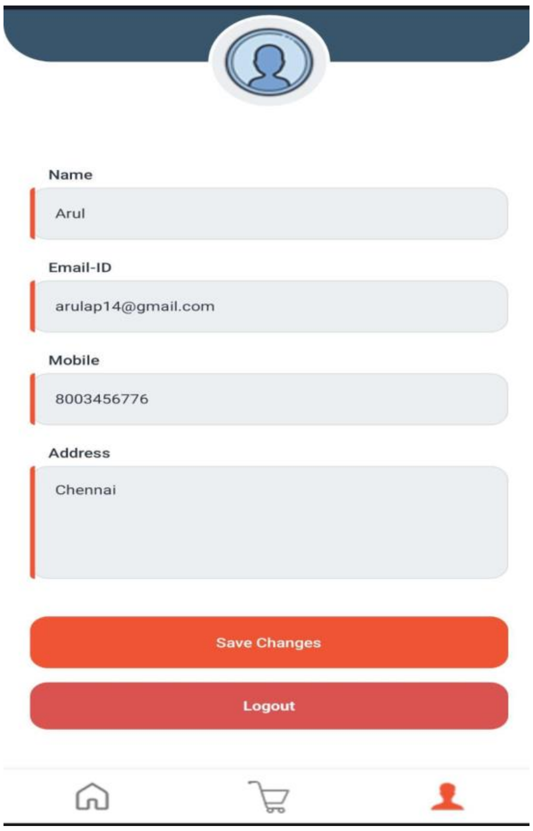
This allows the user interface to function seamlessly with the backend system.

4.2 SCREEN SHOTS:





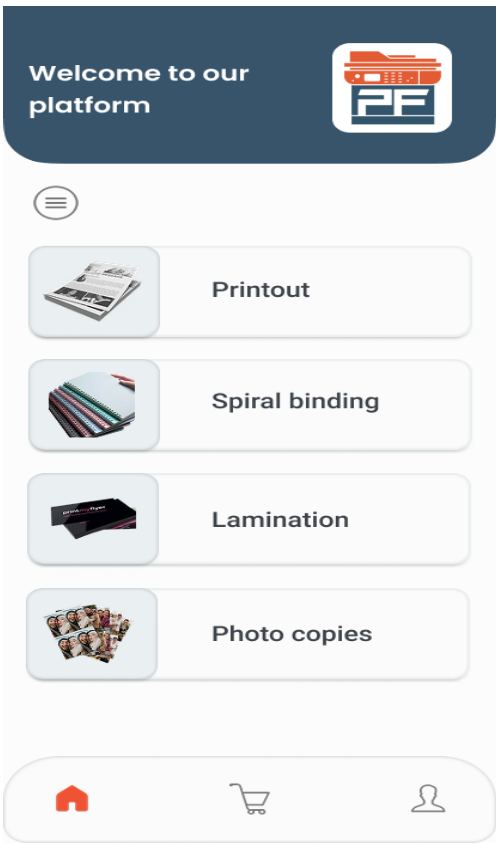


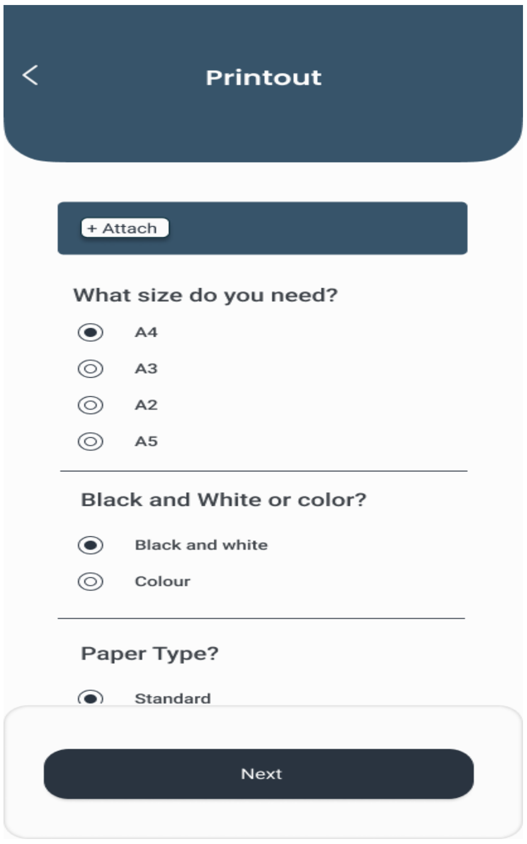

Name: Arul

Email-ID: arulap14@gmail.com

Mobile: 8003456776

Address: Chennai





What size do you need?

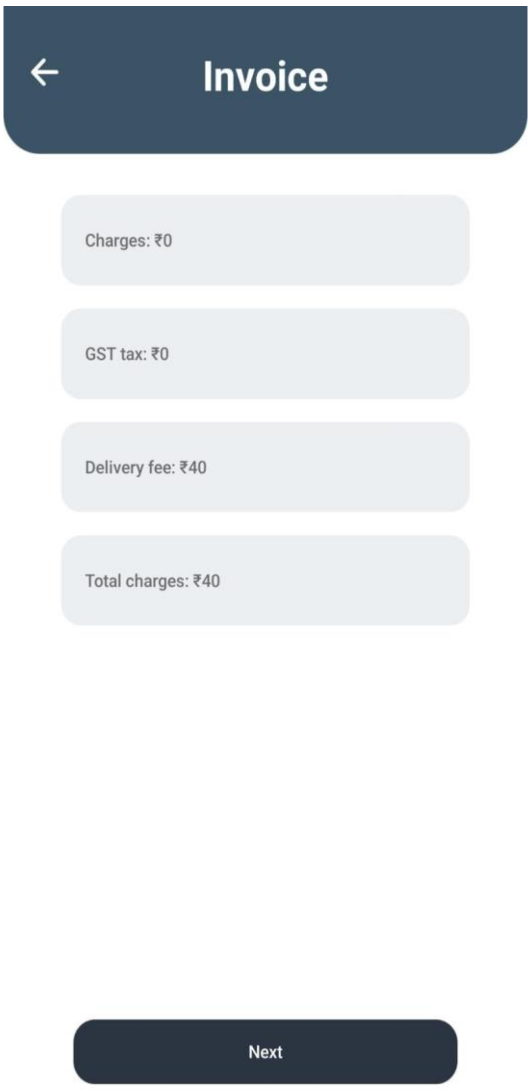
[✓ A4] A3 A2 A5

Black and White or color?

[✓ Black and white] Colour

Paper Type?

[✓ Standard]



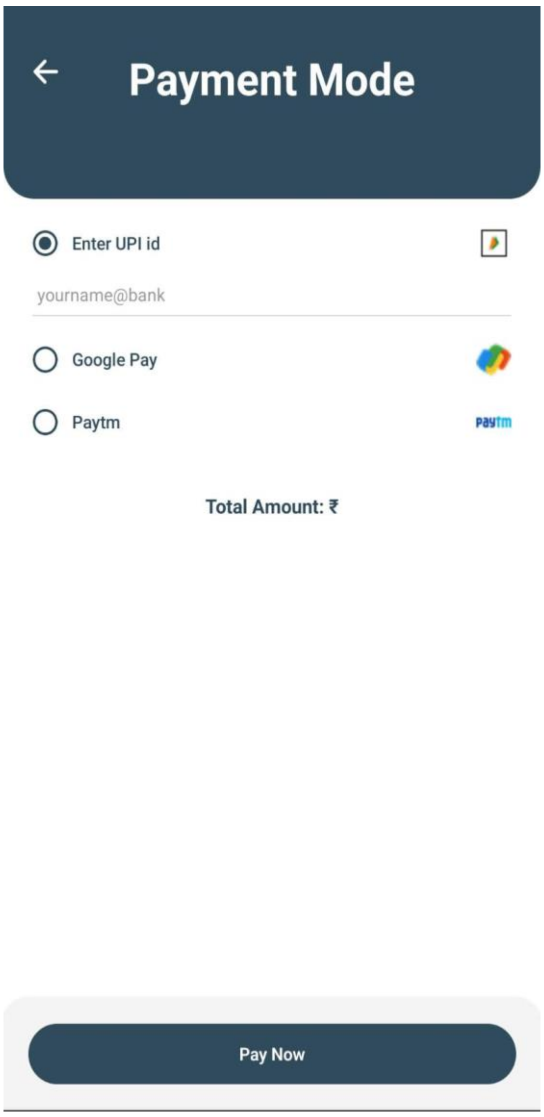
Invoice

Charges: ₹0

GST tax: ₹0

Delivery fee: ₹40

Total charges: ₹40



Payment Mode

[✓ Enter UPI id]

yourname@bank

Google Pay

Paytm

Total Amount: ₹

CHAPTER - 5

5. CONCLUSION:

In conclusion, our printout delivery app offers a smart and convenient solution for students and professionals who need bulk printing services without the hassle of visiting a physical shop. By allowing users to upload documents, customize print options (such as black & white or colour, single or double-sided), and choose delivery addresses, the app simplifies the entire printout process. It ensures time-saving and stress-free service, especially beneficial in metropolitan cities where travel and queues can be a burden. With a minimum order requirement of 50 pages, the app is designed to handle only bulk orders efficiently, helping streamline operations. Overall, the app bridges the gap between users and printing services, making document delivery accessible, timely, and highly user-friendly.

5.1 LIMITATIONS:

- Only accepts orders of **50 pages or more**, which may exclude users with smaller print needs.

- Delivery may be delayed due to **traffic or external factors**, especially in metropolitan areas.

- Currently operates only in **specific locations**, not available in rural or less-connected regions.
- The app requires a **stable internet connection** to upload files and place orders.
- Users cannot not be able to **track their delivery** in real time .

5.2 FUTURE ENHANCEMENTS:

- Add live tracking so users can monitor the delivery status of their printouts.
- Allow orders below 50 pages with additional charges or during off-peak hours.
- Provide the app interface in multiple languages to reach a broader user base.
- Integrate a live helpdesk or chatbot for real-time customer assistance.

REFERENCES:

- Swiggy. (n.d.). *How Swiggy Works*. Retrieved from https://www.swiggy.com
- Dunzo. (n.d.). *Delivery Model Overview*. Retrieved from https://www.dunzo.com
- Figma. (n.d.). *Collaborative Interface Design Tool*. Retrieved from https://www.figma.com
- PostgreSQL Global Development Group. (n.d.). *PostgreSQL Documentation*. Retrieved from https://www.postgresql.org/docs/
- Pressman, R.S. "**Software Engineering: A Practitioner Approach**", 8th Edition Revised, McGraw Hill, Chennai, 2019.
- Sommerville, I. "**Engineering Software Products**", Global Edition, Pearson Education, 2021.
- Abrahm Silberschatz, Henry. F. Korth, S. Sudarsan "**Database System Concepts**", McGraw Hill, Seventh Edition, Indian Edition, 2021.