LiftDrop

Mobile App for Deliveries

Gonçalo Morais, n.º 49502, e-mail: a49502@alunos.isel.pt, tel.: 927468061
João Ramos, n.º 49424, e-mail: a49424@alunos.isel.pt, tel.: 919222551

Orientador: Miguel Gamboa, e-mail: miguel.gamboa@isel.pt

Co-Orientador: Diogo Silva, e-mail: diogo.silva@lyzer.tech

March of 2025

Abstract

The rise of the **gig economy** has significantly transformed industries like food and package delivery. This model, based on flexible short-term work facilitated by digital platforms, is used by companies such as Uber Eats, Glovo, and Bolt Food to enable on-demand urban logistics.

While these platforms offer convenience to customers, they often impose high commission fees on couriers, lack job stability, and introduce logistical inefficiencies. This creates a growing tension between platform profitability and worker welfare, while also affecting service reliability for customers.

LiftDrop addresses these issues by introducing a mobile app that promotes fairness, efficiency, and transparency. It aims to optimize delivery routes, reduce platform fees, and ensure better conditions for couriers—ultimately improving the overall delivery experience for both workers and users.

Acknowledgements

We would like to express our sincere gratitude to our project supervisor, Prof. Miguel Gamboa and Engineer Diogo Silva, for their invaluable guidance, feedback, and support throughout the duration of this project. Their insights and encouragement greatly contributed to the development and completion of this work.

Finally, we would like to thank my peers, friends, and family for their encouragement and support during this journey.

Contents

1 Introduction

The gig economy has revolutionized delivery services, yet most existing platforms prioritize customer convenience at the expense of courier well-being. **LiftDrop** addresses this imbalance by proposing a fairer, more efficient mobile application for last-mile deliveries.

1.1 Motivation

Gig economy platforms like Uber Eats, Glovo, and Bolt Food have become essential to urban logistics, offering convenience to customers and flexible work opportunities to couriers. However, these platforms exhibit systemic issues that disproportionately affect couriers and ultimately degrade overall service quality.

Key Issues in Existing Platforms

- **Unfair Order Assignment:** Many platforms use opaque algorithms that favor highly active or high-performing couriers, reinforcing inequality and disadvantaging new or less available workers.
- **Lack of Transparency:** Couriers typically receive minimal information about how their performance is evaluated, how earnings are calculated, or why specific orders are assigned. This opacity leads to distrust and confusion.
- **Safety Oversights:** Couriers are often routed through unsafe neighborhoods, especially at night, with little to no in-app risk alerts or mitigation strategies.

These challenges point to a systemic flaw: *existing platforms are optimized for customer satisfaction rather than courier well-being.*

LiftDrop aims to correct this imbalance by embedding fairness, transparency, and safety at the core of its delivery model. By redesigning the order assignment logic, integrating safety monitoring, and promoting data transparency, LiftDrop introduces a more ethical and human-centered approach to delivery logistics.

Key Objectives:

- Implement real-time order assignment based on proximity, traffic conditions, and fairness principles.
- Improve courier safety using neighborhood risk ratings and proactive alert systems.
- Develop a scalable Android application supported by two core APIs:
  - **Courier API:** Manages orders and provides real-time updates via WebSockets.
  - **Client Simulation API:** Simulates customer orders for testing and development purposes.

The figure below provides a high-level architectural view of the LiftDrop system. It illustrates the major components: the courier-facing mobile application, the client Web API for simulating orders, the backend server containing the business logic, the central database for persistent storage, and a geospatial API for calculating distances between couriers and delivery locations.

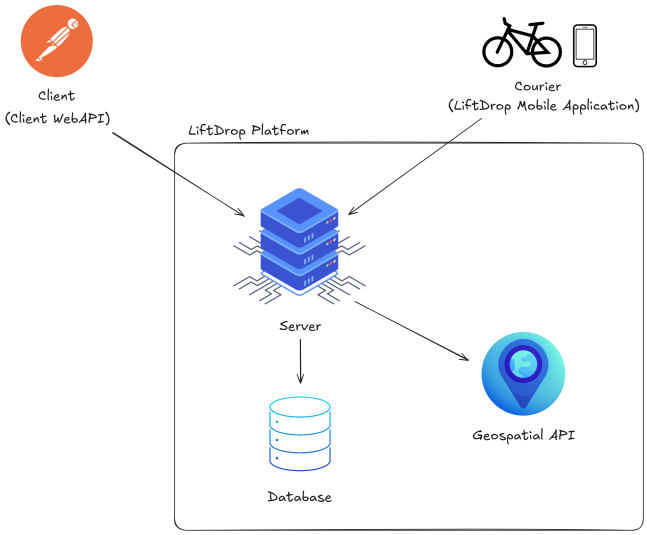


Figure 1: High-level overview of the LiftDrop system architecture

Figure 1 summarizes the system's structure and component interactions without delving into implementation specifics.

1.2 Requirements

Functional Requirements

The following features are essential for the success of the LiftDrop application:

- Orders shall be assigned based on courier proximity, availability, and a fairness algorithm.
- Couriers shall receive visual safety ratings for neighborhoods within the mobile app.
- The Courier API shall provide real-time order status updates using WebSockets, ensuring timely communication.

1.3 Organization

This report is structured as follows:

- **Background Knowledge** provides an overview of the gig economy and logistics frameworks.
- **System Architecture** details the components and data flow within LiftDrop.
- **Interface Design** covers the user-facing aspects of the mobile application.
- **User Journey** demonstrates typical courier and client workflows.
- **Technical Implementation Details** discusses the development process and key technologies used.

2 Background

The rise of the gig economy has transformed the logistics and delivery landscape, with platforms like Uber Eats, DoorDash, and Instacart enabling flexible work for couriers and fast service for customers. However, this flexibility often comes at the cost of courier well-being, with issues such as unfair order distribution, unsafe working conditions, and lack of transparency in pay and performance metrics.

Several studies and reports have highlighted the systemic problems in current delivery platforms:

- **Order Assignment Bias:** Algorithms tend to favor high-performing couriers or those in high-demand areas, reinforcing inequality.
- **Safety Concerns:** Couriers are often sent to unfamiliar or unsafe neighborhoods without adequate information or alerts.
- **System Opacity:** Many platforms offer limited insight into how decisions are made or how to improve courier rankings.

Technologically, modern delivery apps rely heavily on real-time geolocation, traffic data, and cloud-based infrastructure for scalability. Technologies like Google Maps API (for geospatial data) and WebSockets (for real-time communication) enable efficient routing and continuous status updates. Still, few platforms integrate these technologies with a fairness- and safety-first mindset.

LiftDrop aims to address these gaps by incorporating ethical algorithmic design and safety-first features into a scalable, testable delivery platform.

2.1 Comparison with Existing Platforms

Several delivery platforms offer features similar to LiftDrop, but none provide the same balance of fairness, security, and real-time operational support. For example, **Glovo**, **GoBuddy**, and **CTT Now** support order management, provide real-time updates, and implement basic security mechanisms. However, none of these platforms ensure equitable order distribution, as they often rely on opaque or performance-weighted assignment algorithms.

Deliveo is one of the few platforms that incorporates all four capabilities: order management, real-time support, security features, and fair distribution of tasks. However, its market presence and interface scalability are relatively limited when compared to larger competitors.

Vanbu supports basic order distribution but lacks real-time communication features and safety mechanisms, making it less adaptable to fast-paced delivery scenarios.

Takeaway.com Courier performs well in most operational aspects but, like many others, does not address fairness in order allocation—leading to potential workload imbalances for couriers.

In contrast, **LiftDrop** integrates all these essential features into a single platform. It offers robust order and status management, supports real-time communication via WebSockets, implements route-based safety alerts, and—critically—prioritizes fair order distribution among couriers. This combination positions LiftDrop as a more balanced and ethical alternative in the delivery platform ecosystem.

3 Scope

This section defines the functional and technical boundaries of the LiftDrop platform. It outlines the key features available to users, clients, and couriers, along with the strategies employed for real-time data communication, order assignment, and safety. These boundaries establish what the system is designed to accomplish, while also clarifying what lies outside the project's current scope.

3.1 Platform Features

3.1.1 User Features

- Register as either a client or a courier.
- Clients provide personal and contact information during registration.
- Couriers specify their transportation method upon registration.

3.1.2 Client Features

- Place orders by selecting restaurants and desired items.
- Track order status and view estimated time of arrival.

3.1.3 Courier Features

- Accept or decline delivery requests.
- Set availability by entering or exiting "waiting" status.
- Update the delivery status or cancel an accepted order.
- Confirm order delivery upon completion.

3.2 Real-Time Communication Strategy

Managing real-time data updates and courier interactions is central to the platform's success. Three communication models were considered:

- **Polling (Request-Response):** Simple but inefficient, creating excessive network overhead.
- **Server-Sent Events (SSE):** Efficient for unidirectional updates, but limited in interaction complexity.
- **WebSockets:** Enables full-duplex communication and minimizes latency.

WebSockets were selected for their ability to support real-time bidirectional updates, crucial for dynamic interactions like order assignment, courier status changes, and live notifications.

3.3 Order Assignment Strategy

Orders are assigned based on multiple criteria to ensure fairness and efficiency:

- **Proximity:** Couriers closest to the pickup location are prioritized.
- **Fairness:** Distribution considers past assignments to avoid overload.
- **Traffic Conditions:** Live data helps optimize delivery time.

For geospatial data and route optimization, the system integrates the **Google Maps API**, leveraging its routing, distance calculation, and traffic-aware services.

3.4 Courier Safety Measures

To support safe working conditions for couriers, the platform includes weather alerts. Those alerts are made with the use of SSE (Server Side Events) to pass that information to the courier.

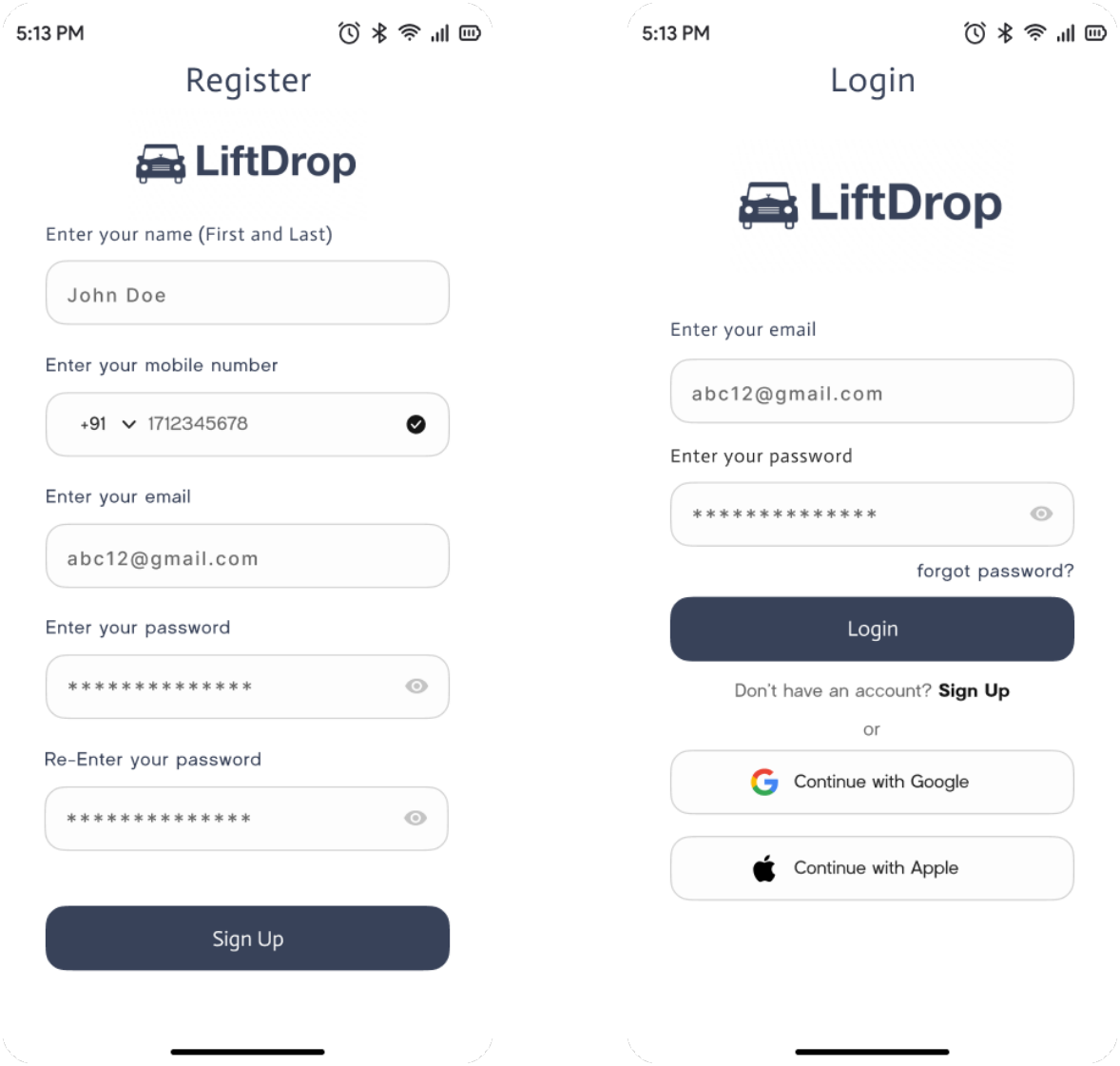
4 Interface Design

4.1 User Flow and Mockups

The LiftDrop application features distinct interfaces for different stages of the delivery process. Below are the key mockups organized by functionality.

4.1.1 Authentication Flow

The authentication flow allows users to securely access the LiftDrop system.



(a) User registration screen

(b) User login screen

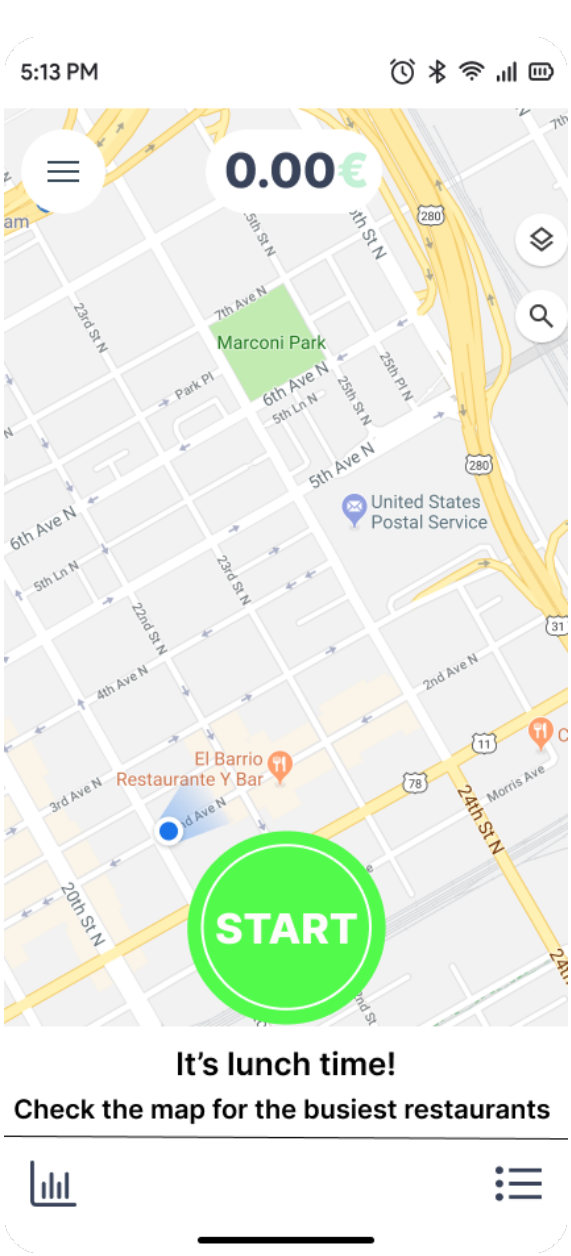
Figure 2: Authentication flow screens for LiftDrop

Registration Screen (2a): Allows new users to create an account by entering essential information such as name, email, and password. This screen initiates access to the LiftDrop platform.

Login Screen (2b): Enables users to securely log in using their credentials. The layout emphasizes simplicity and quick access.

4.1.2 Courier Flow

These screens illustrate the courier's working states, from availability to delivery.



(a) Active status screen while courier is idle

(b) Waiting screen during order availability check

Figure 3: Courier status screens showing different working states

Go Screen (3a): Screen before the courier starts waiting for orders. A central status toggle makes it easy to go online or offline.

Waiting Screen (3b): Displays a waiting state while the system checks for nearby delivery requests. It reassures the user that the system is searching in the background.

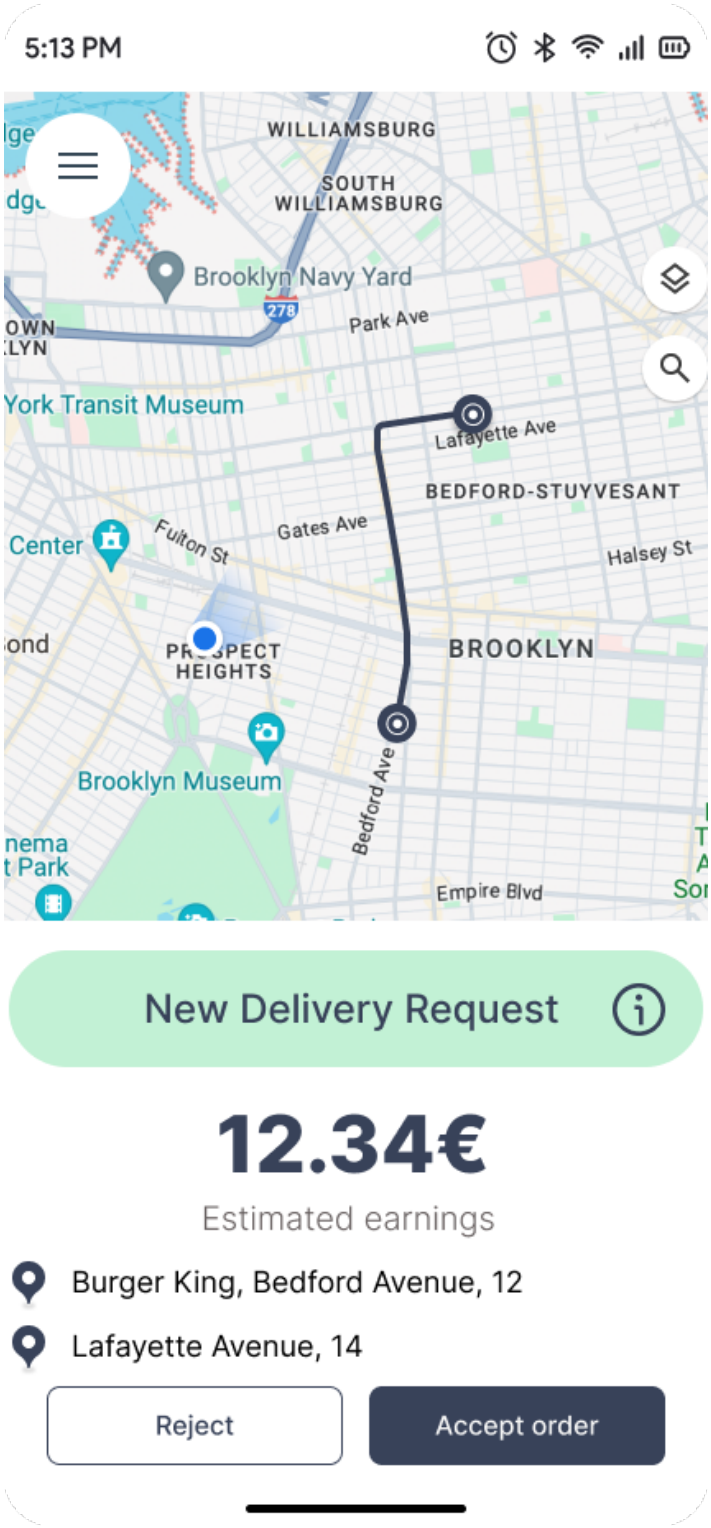
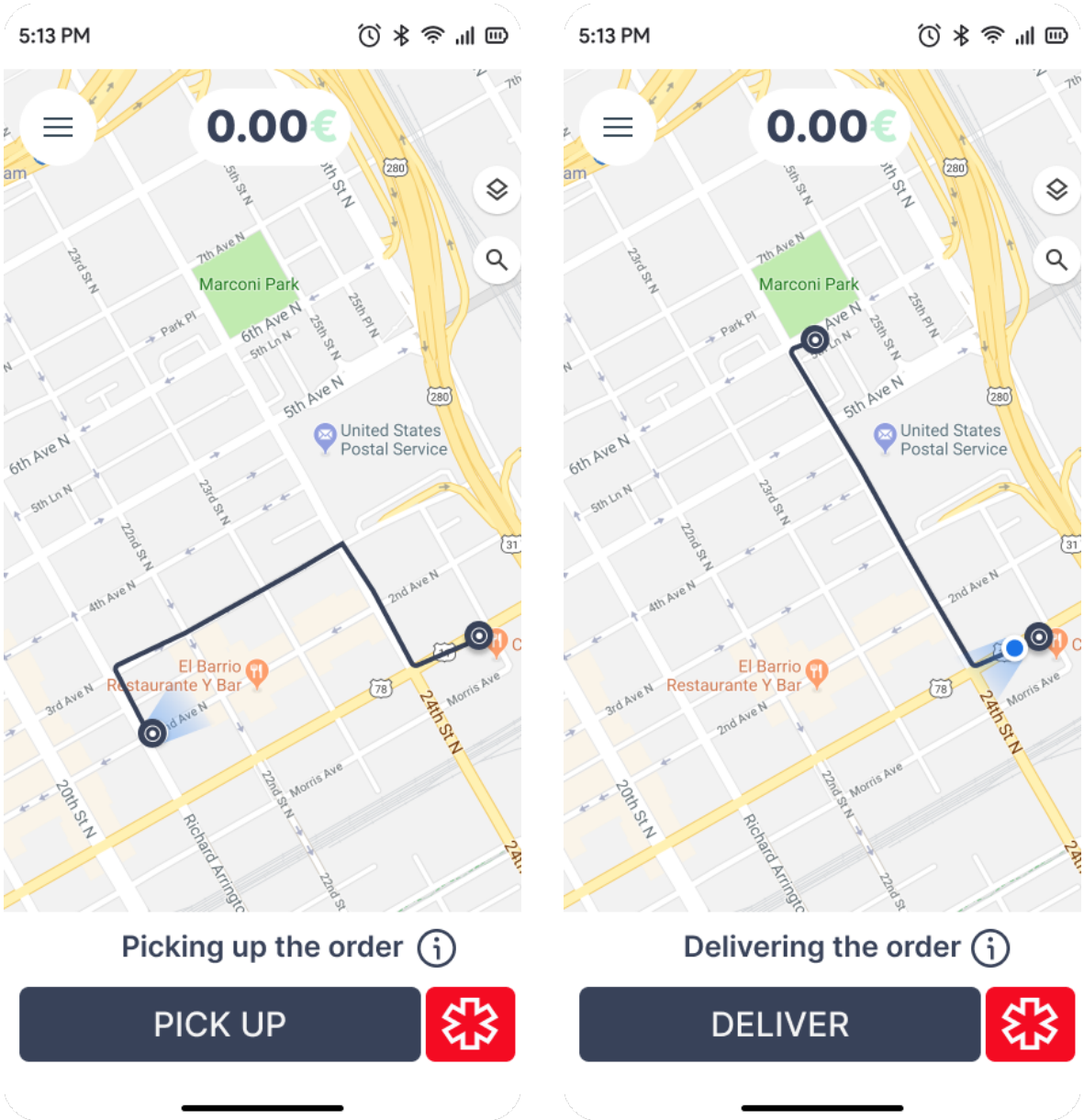


Figure 4: New delivery request notification with order details and action buttons

Delivery Request Notification (4): Shows an incoming delivery request with pickup and drop-off addresses, estimated distance, and action buttons to accept or decline. It's designed for quick decision-making and clarity.



(a) Pickup Confirmation Screen

(b) Delivery Confirmation Screen

Figure 5: Courier delivery journey screens during different phases of the delivery

Pickup Screen (5a): Provides order pickup instructions and customer location. It includes a confirmation action once the parcel is collected.

Delivery Screen (5b): Guides the courier to the drop-off location. Includes proof-of-delivery options like signature or photo, depending on the order.

5 User Journey Diagram

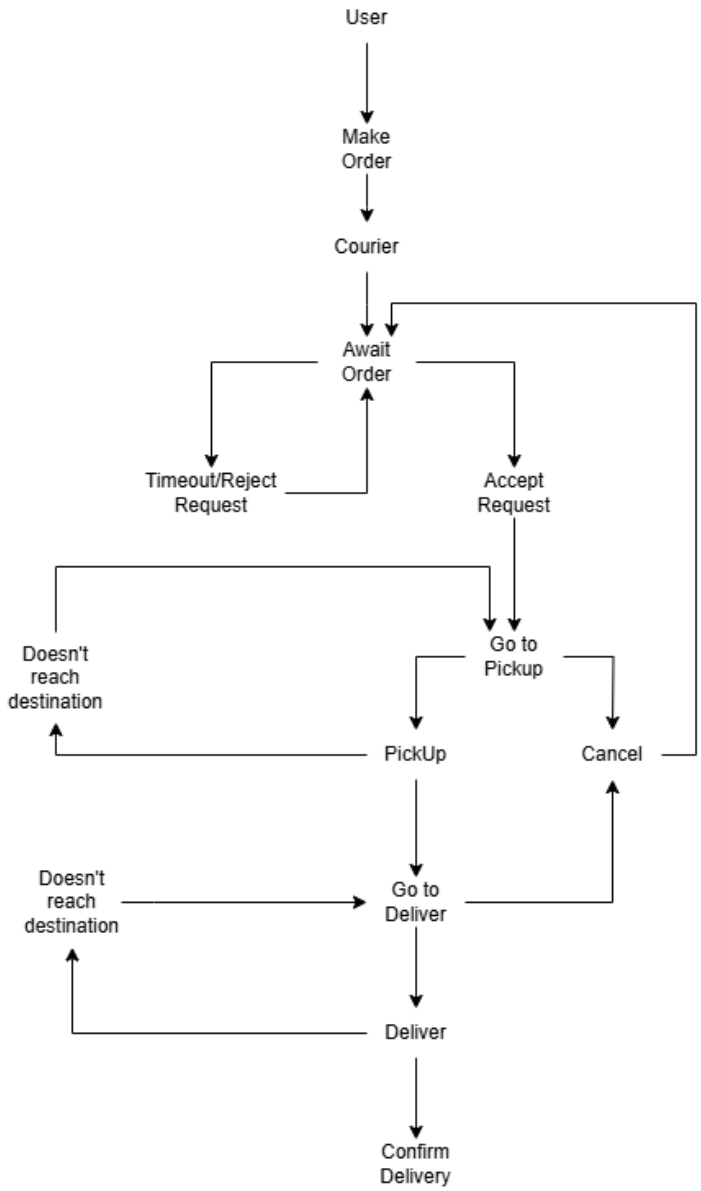


Figure 6: Order journey diagram showing success and failure paths

This diagram illustrates the typical order journey from a user's perspective, covering both successful and failed outcomes. It captures the interactions between clients, couriers, and the platform across different stages: order placement, courier acceptance, delivery progress, and resolution in case of errors or cancellations. The flow ensures that edge cases—such as order rejection or delivery failure—are considered during system design.

6 System Architecture and Data Model

This section presents different architectural views of the system.

6.1 Location-Focused Architecture

This view highlights how physical locations are managed in the system. The `Location` entity serves as a base class, with `PickupSpot` and `DropOffSpot` inheriting its properties. Each pickup location can contain multiple `Item` objects, each with a unique designation.

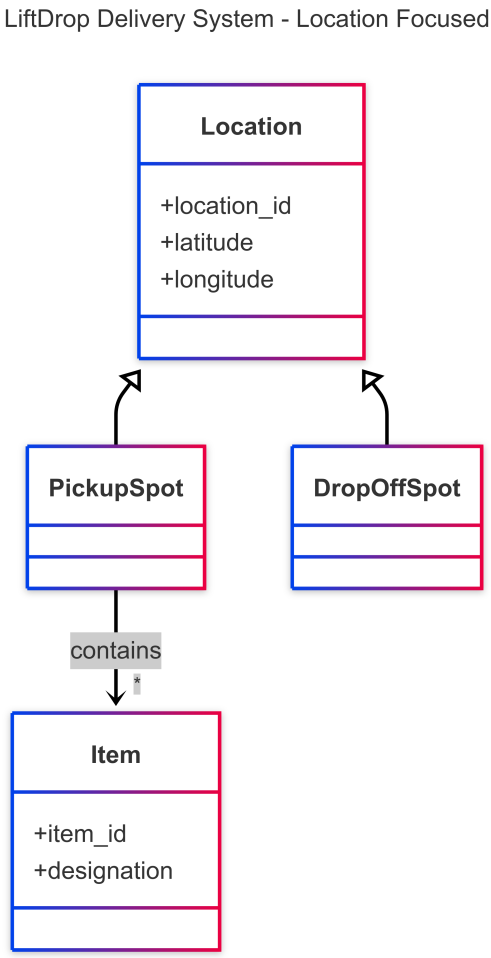


Figure 7: Location-Focused Architecture

6.2 User and Role Simplification

This simplified model captures how different user roles (Client and Courier) interact with the delivery request lifecycle. A `Client` can make multiple `Requests`, while a `Courier` can handle many of them. Every request is linked to a `Delivery`.

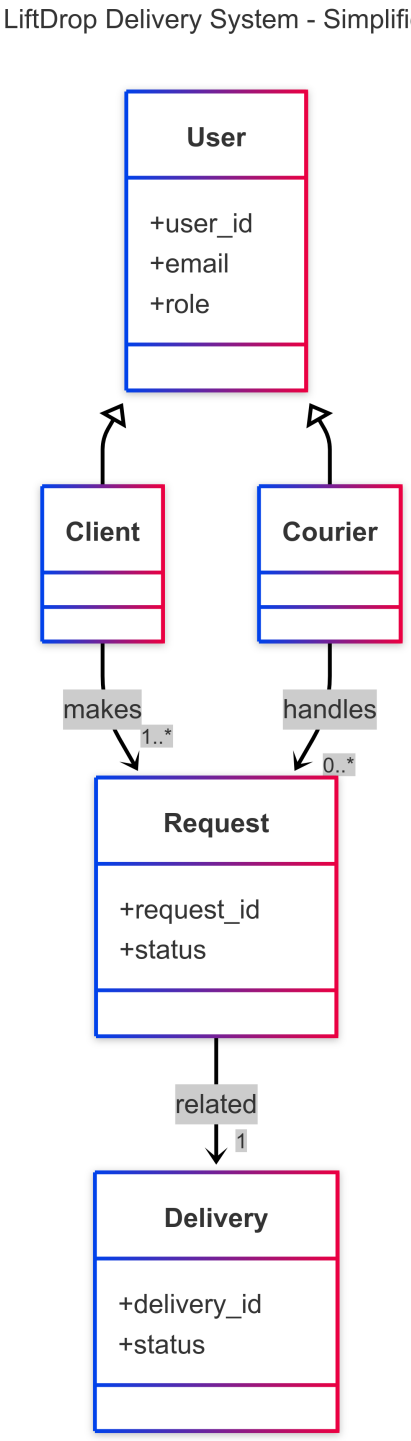


Figure 8: Simplified User and Role Diagram

6.3 Request-Focused View

This view focuses specifically on the request and fulfillment process. Each `Request` has detailed metadata in `RequestDetails` and is associated with exactly one `Delivery`. This structure emphasizes accountability and end-to-end traceability in the delivery chain.

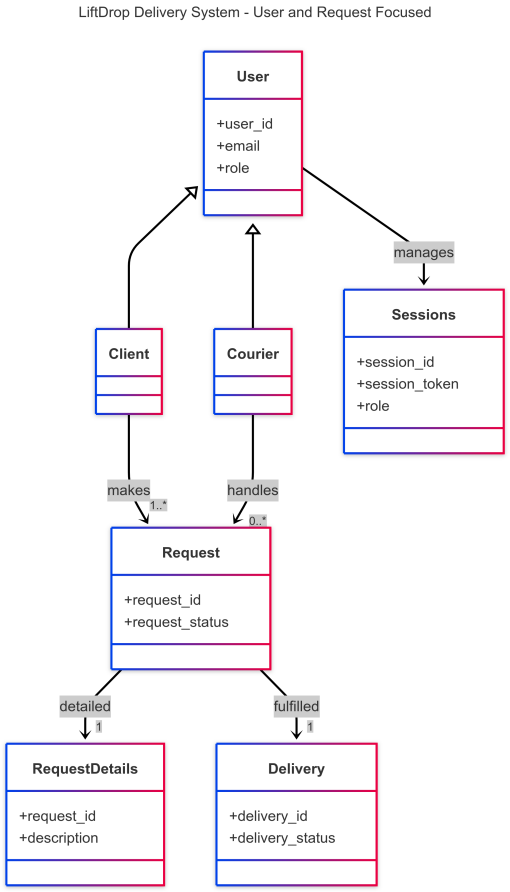


Figure 9: Request-Focused Architecture

6.4 Complete Delivery System

The comprehensive model integrates all key modules including `User`, `Address`, `Location`, `Item`, `Request`, and `Delivery`. It also tracks session information for secure authentication and maps logical entities to physical addresses and locations. This detailed structure supports real-time delivery management with status tracking, ETA calculation, and geographic correlation.

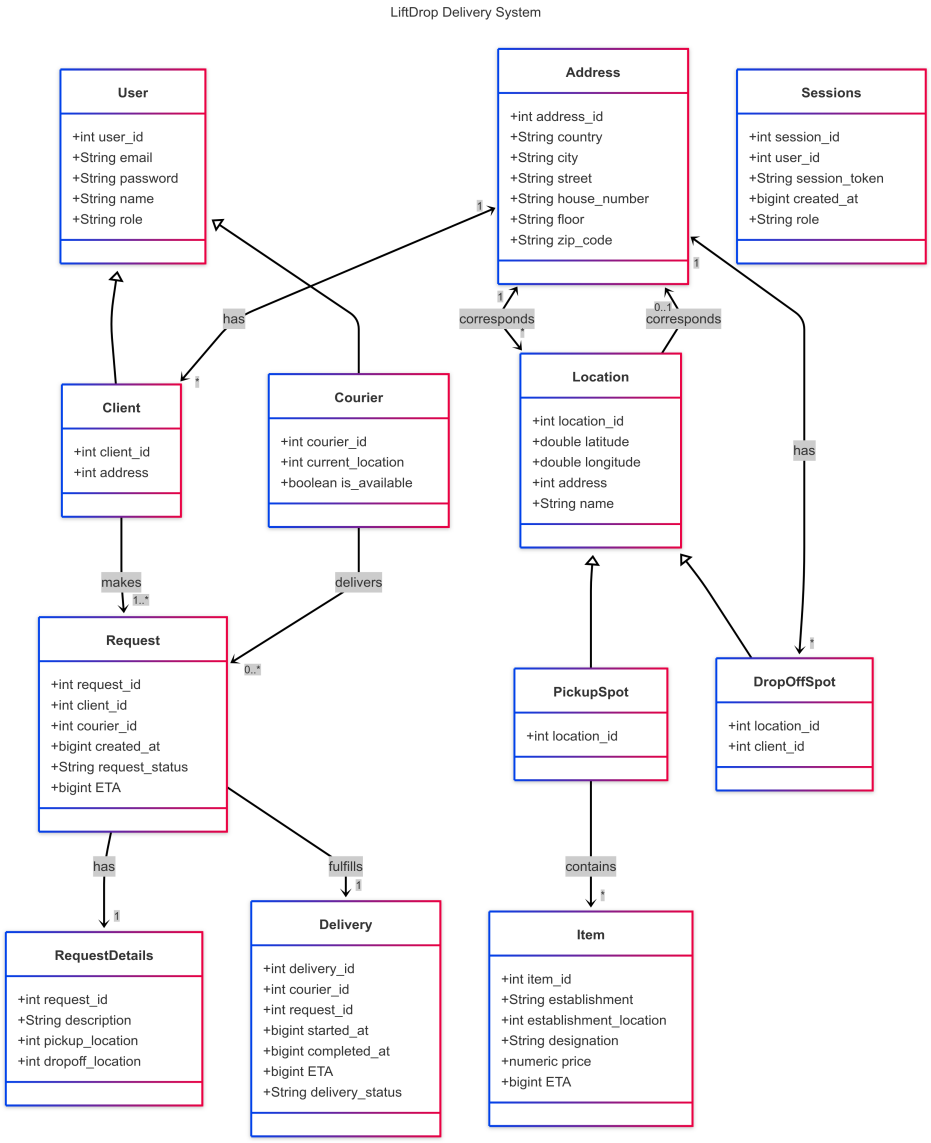


Figure 10: Full System Diagram

7 Technical Implementation Details

This section outlines the core technologies and tools employed in the development of the LiftDrop platform, along with key decisions regarding communication, backend logic, and security.

7.1 Technology Stack

- **Mobile Application**: Developed using Android Jetpack Compose (Kotlin), the app enables couriers to manage order availability, accept or reject requests, and track deliveries in real time.
- **Backend Server**: Built with Spring Web MVC, the server handles business logic, order assignment, and serves API endpoints for communication with the mobile application.
- **Database**: PostgreSQL is utilized for the structured storage of essential data, including user profiles, orders, delivery statuses, and location history.
- **Geospatial Services**: The Google Maps API is leveraged for calculating courier distances, estimating delivery times, and optimizing routes based on live traffic conditions.

7.2 Communication Protocol

To facilitate real-time updates between the mobile application and the backend server, **WebSockets** are used. WebSockets offer several advantages over alternatives like polling and Server-Sent Events (SSE), including bidirectional communication, reduced latency, and minimized network overhead. These benefits ensure a more responsive and interactive experience for couriers, particularly when managing order updates and status changes.

The courier's location is continuously updated using **Foreground Services** on Android. This approach ensures the app maintains an active connection, delivering real-time location data even when running in the background, and avoids being terminated by the operating system.

7.3 Order Assignment Logic

Orders are assigned based on the following factors:

- **Proximity to the pickup point**
- **Fair workload distribution** considering the courier's recent delivery history
- **Live traffic conditions**, using geospatial data to optimize routing

7.4 Security and Authentication

To ensure secure user sessions and API requests:

- **Session-Based Authentication with Cookies**: After a successful login, the server sets an HTTP-only cookie containing a session token. This token is inaccessible to JavaScript, which helps mitigate cross-site scripting (XSS) attacks.

7.5 Development Tools

- **Postman**: Used to test and validate REST API endpoints during backend development.
- **Git and GitHub**: Employed for version control, collaboration, and managing the development workflow.
- **Android Studio**: The primary IDE for building, testing, and debugging the mobile courier application.
- **Intellij**: The primary IDE for building, testing, and debugging the backend.

7.6 Deployment Details

- **Backend Deployment**: The backend is hosted on Render, a cloud platform that streamlines deployment and scaling processes.
- **Database Hosting**: PostgreSQL is managed and hosted on Render, providing a scalable, reliable environment for data storage.