الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

جامعة سعيدة د. مولاي الطاهر

كلية الرياضيات و الإعلام الآلي و الاتصالات السلكية

و اللاسلكية

قسم: الإعلام الآلي

Mémoire de Master en informatique

Spécialité : MICR

T h è m e

Swaply
Peer-to-Peer Delivery Platform
Powered by AI

- **Présenté par :**
  - Mr. Abderrahmane Tayeb FARHI
  - Mr. Mohammed El Amine FARHANE
- **Dirigé par :**
  - Dr. Mohamed Elhadi RAHMANI
  - Dr. Hadj Ahmad BOUARARA

Année universitaire 2025-2026

ملخص

يقدم مشروع "Swaply"، وهو منصة لوجستية رقمية تشاركية تهدف إلى ربط العملاء بسائقين مستقلين لتسهيل عمليات نقل البضائع والطرود بكفاءة وموثوقية عالية. يعتمد النظام على بنية معمارية متكاملة توفر واجهات مخصصة لكل من العميل، السائق، والإدارة، مما يضمن تنظيم سير العمل بشكل آمن وسلس. تقدم المنصة ميزات متقدمة تشمل التتبع اللحظي للرحلات، التسعير الديناميكي، والتواصل المباشر. وتبرز قوة المشروع في اعتماده المعمق على تقنيات الذكاء الاصطناعي لحل التحديات الأمنية والتشغيلية؛ حيث يشتمل على نظام أمني ذكي للتحقق من هويات السائقين ومطابقتها، وخوارزمية دقيقة لتصفية التقييمات الوهمية لضمان شفافية المنصة، بالإضافة إلى مساعد افتراضي مدمج لدعم المستخدمين آلياً. ختاماً، يجسد هذا المشروع حلاً برمجياً متطوراً يخلق بيئة نقل لوجستية آمنة، ذكية، وعالية الموثوقية.

Abstract

This project presents **"Swaply"**, a collaborative digital logistics platform aimed at connecting customers with independent drivers to facilitate the transport of goods and parcels with high efficiency and reliability. The system relies on an integrated architecture that provides dedicated interfaces for the customer, the driver, and the administration, ensuring a secure and seamless workflow. The platform offers advanced features, including real-time trip tracking, dynamic pricing, and direct communication. The core strength of the project lies in its deep integration of Artificial Intelligence (AI) to address security and operational challenges; it encompasses a smart security system for driver identity verification and matching, a precise algorithm for filtering fake reviews to ensure platform transparency, and a built-in virtual assistant for automated user support. Ultimately, this project embodies a sophisticated software solution that establishes a secure, intelligent, and highly reliable logistics environment.

Résumé

Ce projet présente **"Swaply"**, une plateforme logistique numérique collaborative conçue pour connecter les clients à des chauffeurs indépendants afin de faciliter le transport de marchandises et de colis avec une efficacité et une fiabilité élevées. Le système repose sur une architecture intégrée offrant des interfaces dédiées pour le client, le chauffeur et l'administration, garantissant ainsi un flux de travail sécurisé et fluide. La plateforme propose des fonctionnalités avancées comprenant le suivi des trajets en temps réel, la tarification dynamique et la communication directe. La force principale du projet réside dans sa profonde intégration des technologies d'Intelligence Artificielle (IA) pour relever les défis sécuritaires et opérationnels ; il intègre un système de sécurité intelligent pour la vérification et la correspondance des identités des chauffeurs, un algorithme précis de filtrage des faux avis pour assurer la transparence de la plateforme, ainsi qu'un assistant virtuel intégré pour le support automatisé des utilisateurs. En conclusion, ce projet incarne une solution logicielle sophistiquée qui crée un environnement de transport logistique sécurisé, intelligent et hautement fiable.

Acknowledgment

First and foremost, we would like to express my sincere gratitude and deepest appreciation to our supervisors

Dr. Mohamed Elhadi RAHMANI and Dr. Hadj Ahmad BOUARARA for their valuable guidance, continuous support, constructive feedback, and encouragement throughout the completion of this research. Their expertise and insightful comments have greatly contributed to the success of this study.

Our appreciation is further extended to all the teachers who generously shared their experiences and perspectives throughout the years of study in the university.

Contents

LIST OF TABLES

LIST OF FIGURES

GENERAL INTRODUCTION

The rapid growth of digital platforms has fundamentally changed the way people interact, exchange services, and conduct transactions. In the logistics sector, traditional courier systems remain expensive, inflexible, and often inaccessible for small or individual shipments. At the same time, many individuals travel daily between cities with unused carrying capacity, representing an untapped opportunity for collaborative and cost-effective delivery.

Swaply addresses this gap by connecting users who need to send packages with travelers already moving along compatible routes. Rather than relying on a centralized courier service, the platform facilitates direct peer-to-peer coordination, reducing costs and intermediaries while providing tools to manage trust, security, and communication between parties who may not know each other.

The platform integrates identity verification through CNN-based facial recognition, an AI-powered chatbot for user assistance, a star-based reputation

system, and an administrative supervision layer. Together these components form a complete and coherent solution to the challenges of peer-to-peer package delivery.

CHAPTER 1: PROJECT CONTEXT AND PROBLEM STATEMENT

1.1 Introduction

The logistics sector has undergone significant transformation over the past decade, driven by the rise of e-commerce and on-demand delivery services. However, most available solutions target businesses rather than individuals, and the cost of personal shipments between cities remains disproportionately high relative to the actual transport effort involved.

Peer-to-peer digital platforms have demonstrated in other domains such as accommodation and ride-sharing that direct user-to-user coordination is viable and often preferred by users due to lower costs and greater flexibility. The challenge in applying this model to package delivery lies in managing trust, security, and operational coordination when strangers exchange physical goods.

1.2 Problem Statement

The central problem addressed by Swaply is formulated as follows:

How can a digital platform facilitate secure, trustworthy, and efficient peer-to-peer package delivery by connecting senders with travelers, while minimizing fraud risk, identity uncertainty, and workflow complexity through the integration of artificial intelligence and modern web technologies?

This problem encompasses several interconnected challenges including trust and identity assurance, route coordination, real-time communication, platform moderation, and intelligent user support.

1.3 Existing Limitations

Traditional delivery platforms present several limitations:

• High cost: Traditional courier fees are disproportionate for small personal shipments.
• Inflexibility: Fixed schedules and fixed routes do not adapt to individual needs.
• No peer coordination: Existing platforms do not leverage the carrying capacity of fellow travelers.
• Weak identity assurance: Most C2C platforms rely only on email verification with no real identity check.
• No intelligent assistance: Most logistics platforms offer no AI-based user support or guidance.

1.4 Project Objectives

The main objectives of the Swaply project are:

1. Design and implement a full-stack web platform for peer-to-peer package delivery coordination.
2. Provide identity verification through camera-assisted face comparison against an uploaded ID image.
3. Integrate an AI chatbot to assist users within the platform interface.
4. Support trust-oriented decision-making through a star-based reputation system and administrative review.
5. Build a modular and maintainable system suitable for academic evaluation and future extension.

1.5 Conclusion

This chapter has presented the context, problem, and objectives of the Swaply project. The following chapters will detail the technical and functional solutions designed and implemented to address these challenges.

Chapter 2: State of the Art

2.1 Introduction

The rapid evolution of modern digital technologies and increasing popularity of mobile connectivity have led to the emergence of new service platforms based on the principles of the sharing economy. Among the latest innovations emerging in this environment, there has been a new generation of P2P delivery platforms that provide community-based solutions for the transportation of parcels on dedicated routes. Such platforms are based on the idea that people traveling along particular routes in their cars or bags can be engaged in transporting packages for others in exchange for a payment. The use of these solutions implies a radical shift from traditional logistics models based on dedicated courier companies with large fixed infrastructures to a much more flexible, cost-efficient model.

This chapter provides a thorough literature review of the current platforms, technologies, and studies relevant to the development and operation of the Swaply P2P delivery platform. Specifically, it focuses on four major topics: existing peer-to-peer delivery platforms and their architectural characteristics, facial recognition technologies and identity verification techniques for P2P platforms, conversational artificial intelligence and large language models for chatbot-based customer support, and machine learning-based systems for detection of fake reviews on online platforms.

2.2 Peer-to-Peer Delivery Platforms

2.2.1 Overview and Evolution

Crowdshipping (or crowdsourced delivery) is a logistics innovation inspired by the broader sharing economy that appeared back in the early 2010s with such landmark innovations as Uber and Airbnb. The core idea is simple: millions of individuals traveling along predictable routes each day have spare capacity in their cars or bags. By providing a connection between travelers who can transport parcels along compatible routes and those who require parcel delivery, crowdshipping platforms achieve remarkable cost savings compared to traditional logistics service providers while minimizing the environmental impact of delivering shipments.

Several research papers have validated the concept of crowdshipping as a cost-efficient logistics solution. In one study, researchers estimated the costs of crowdshipping to be 40 to 60 percent less than the costs of last-mile urban delivery performed by dedicated logistics companies. The primary source of these cost savings lies in the fact that there is no need to purchase and maintain a fleet of delivery vehicles or build warehouses for storing parcels or paying drivers salaries. [1]

2.2.2 Uber Connect

One of the largest and most popular platforms leveraging crowdsourcing is Uber Connect, a crowdsourced delivery service operated through the Uber ride-sharing app. The service operates in many global markets allowing Uber users to deliver packages by connecting them to individual couriers available through the Uber app. The main strengths of the platform lie in the fact that it is built on top of a mature and proven infrastructure for tracking drivers' location in real time, generating prices for each trip dynamically, and managing a vast user base.

Still, there are a number of security vulnerabilities in the system. First of all, the courier's identity is verified exclusively via regular account verification. In other words, the courier does not need to pass any biometric identity verification to be able to deliver packages to customers. Thus, in theory, it is possible for an unauthorized user to deliver a package without being checked in terms of his identity against official identification documents. Secondly, the fraud detection system implemented in Uber Connect relies mainly on reactive customer reporting, which means that it requires constant monitoring by the user. Finally, the system lacks a built-in intelligent customer support service. [2]

2.2.3 Yassir Delivery

Yassir is a superapplication developed in North and sub-Saharan Africa. It includes several services, including ridesharing, food delivery, and parcels delivery. For parcels delivery, the service uses a mobile app-based platform that facilitates communication between the sender of the shipment and the independent courier selected to deliver the package. The platform has a very intuitive interface, allows users to track the status of deliveries in real time, and provides convenient in-app chat for communicating with the delivery driver.

Although it is quite easy to work with this service, there are some architectural and functional problems. First of all, there is no biometric identity verification process for couriers. Their identity can be verified only via document verification performed manually by customer service. There is no mechanism to automatically check the identity of couriers using facial recognition or any other technology. In addition, the trust management system implemented in Yassir Delivery relies mainly on ratings entered by customers. There is no way to detect fake reviews automatically. Finally, the service is equipped with a rather simple chatbot without natural language processing capabilities. [3]

2.2.4 Glovo

Glovo is a European delivery service operating in over twenty countries. It provides services for the delivery of products of different types, including groceries, medications, food, and general parcels. The couriers performing deliveries are independent workers, who use the Glovo app to receive delivery orders from clients. The company invested heavily into developing the logistics infrastructure that includes advanced route optimization algorithms and real-time tracking capabilities.

There are several problems that make Glovo less reliable than Swaply despite the use of quite advanced technology. Firstly, the system does not publicly disclose any information about how it detects fake reviews. Second, even third-party analysis shows that there are still cases of reviews tampering and coordination by competitors. Third, identity verification performed by the company relies exclusively on manual document verification without any automated biometric checks. Fourth, the chatbot provided by Glovo lacks the ability to respond to customers' queries via natural language. Instead, it works via a decision tree. [4]

2.3 Identity Verification and Facial Recognition Technologies

2.3.1 The Role of Identity Verification in P2P Platforms

Building trust is one of the core issues of peer-to-peer service platforms. Unlike the delivery services run by institutions, the delivery of parcels in a peer-to-peer setting requires users to place implicit trust in the identity of others. This leads to the question of how to reliably verify that the courier delivering the shipment is indeed who he claims to be and that the document he uploaded during registration indeed belongs to him.

Traditionally, the verification process included a number of manual steps. These steps included manual document checking, government ID number verification through a third-party service, and telephone verification via one-time passwords sent to couriers' phones. All these procedures were relatively simple but also quite ineffective because they were subject to forgery and abuse by couriers. It was possible for a courier to submit real documents but let a friend or someone else perform deliveries using the same account.

2.3.2 Using Convolutional Neural Networks for Face Verification

In recent years, computer vision technology experienced a revolution due to the introduction of deep neural networks. Specifically, convolutional neural networks proved to be highly efficient for performing image classification tasks, achieving human-level performance or even surpassing it. The general principle is fairly simple: a convolutional network learns to map images to points in a high-dimensional space. Points closer together represent images of the same individual, whereas farther points represent images of different individuals.

The latest face recognition systems based on the CNN architecture can extract numerical descriptors called embeddings from faces. Comparing two faces involves calculating a similarity score, which can take the form of a Euclidean distance or a cosine similarity value. After applying a threshold to this score, a binary prediction is made: the two faces either belong to the same person or not. This method is widely used in highly sensitive applications like border controls, law enforcement, and finance. [5]

2.4 Conversational AI and LLM-Based Customer Support

2.4.1 From Rule-Based Dialog Systems to LLMs

Customer support automation had gone through three distinct generations since the earliest days of the Internet. Early dialogue engines worked using predetermined decision trees or rules and required exact matching of keywords. The next generation of solutions utilized statistical algorithms trained on labeled examples. These systems managed to identify user intent but still needed extensive training and often failed in processing queries that were out of the training set distribution.

The appearance of language modeling neural networks in the mid-2010s revolutionized the domain. Transformer-based language models trained on huge amounts of data could now understand natural language input, answer complex multilingual questions, and provide detailed multi-sentence answers that were hard to distinguish from those generated by human agents.

2.4.2 Groq API and Inference Acceleration

Groq API is a major step towards the implementation of LLMs on a large-scale in practice. Conventional inference pipelines of language models are subject to memory bandwidth limitations inherent in contemporary GPU hardware, which often leads to excessive response latency incompatible with the requirements of customer support. With its Language Processing Unit, Groq resolves this issue using an alternative hardware design tailored specifically to the matrix multiplications typical for transformer inference and capable of much faster tokenization.

This performance increase makes customer support applications based on delivery platforms more responsive. Users expect prompt replies when interacting with in-app chatbots similar to instant messaging services. Inference speed provided by Groq API allows for providing such a response rate without compromising the capabilities of state-of-the-art language models. [9]

2.5 Fake Review Detection

2.5.1 Review Manipulation Problem

Review systems are indispensable tools for building trust in platform marketplaces. The reliability of users' ratings and reviews determines the level of trust people have toward couriers when assigning tasks. However, due to economic motivations, some couriers have incentives for manipulating reviews to improve their ratings or ruin those of their competitors. Thus, detection of such reviews becomes a necessity for any platform's marketplaces.

A number of research papers have found some distinctive characteristics of fake online reviews that may facilitate their detection. Such reviews tend to be shorter and generic in comparison with real ones, containing superlative adjectives but little experiential data. They have a higher concentration in rating boundaries, with most of them rated with five stars. They

are submitted by newly created user accounts with little to no previous review activity, and they are distributed temporally within a very narrow timeframe following account creation or delivery completion. Also, fake reviews contain less diverse lexical elements, have higher word repetition rates, and exhibit greater polarity values than real reviews. [17]

2.5.2 Machine Learning Approach

Supervised machine learning has been extensively used in the research of fake reviews. Various ensemble learning algorithms have shown themselves to perform particularly well in this regard because of their ability to capture non-linear relationships between different input features like linguistic, behavioral, and temporal.

Feature extraction is the basis of successful fake review detection with ML algorithms. Linguistic features extracted from review texts include word count, sentence length, diversity measures, punctuation and other syntactic properties, as well as sentiment polarity scores. On the other hand, behavioral features include temporal and account-related characteristics, such as account age, submission rate, and account verification.

2.6 Comparative Analysis and Research Gaps

The following table provides an overview of the differences between various existing platforms and Swaply:

Table 2.1 : Feature comparison of delivery platforms

Feature	Uber Connect	Yassir	Glovo	Swaply
P2P Route Matching	✓	✗	✗	✓
Facial Identity Verification	✗	✗	✗	✓ CNN
Fake Review Detection	✗	✗	✗	✓ RF · AUC 0.946
AI Chatbot Support	✗	✓ Basic	✗	✓ Groq LLM
Admin Verification Panel	✓	✓	✗	✓ Full
In-App Sender/Courier Chat	✓	✓	✗	✓
Traveler Earnings Tracking	✓	✓	✓	✓
Document Upload & AI Review	✓	✓	✗	✓ AI-Verified
Crowd-Sourced Traveler Model	✓	✗	✗	✓

From the above discussion, it is clear that even though existing systems have shown the economic viability of peer-to-peer delivery services, they suffer from common architectural deficiencies in the following three areas: biometric authentication, smart natural language assistance, and automatic fake review detection. These three deficiencies are the key scientific and technical innovations of the Swaply system.

2.7 Conclusion

This chapter has discussed the latest advancements in four fields that have direct applications for the Swaply system, namely peer-to-peer delivery systems, facial recognition software, conversational AI, and fake review detection. It has been demonstrated that while significant advances have been made in each field independently, existing commercial delivery platforms have not yet incorporated these innovations to create a unified artificial intelligence-powered platform design. Uber Connect, Yassir, and Glovo offer robust delivery coordination but depend on antiquated methods of biometric verification, rudimentary customer assistance, and manual or nonexistent systems for review validation. Swaply solves these shortcomings through the use of CNN-based facial recognition, Groq API large language model conversational agents, and a Random Forest fake review detector based on 40,000 actual reviews.

CHAPTER 3: SYSTEM REQUIREMENTS AND ANALYSIS

3.1 Introduction

The foundation of a robust software platform lies in a clear understanding of its operational workflows and user needs. This chapter defines the core system requirements for the application. It outlines the primary actors interacting with the platform, details the specific functional requirements that dictate what the system must accomplish, and establishes the non-functional requirements ensuring usability and security. The analysis is based on the platform's global layout, distinct workspaces for passengers and drivers, administrative control mechanisms, and integrated artificial intelligence systems.

3.2 System Actors

The platform categorizes users into three primary actors, each with a tailored workspace and specific permissions determined by a smart routing system upon login:

1. **The Client (Passenger):** The end-user who utilizes the platform to request transportation for various loads. Their interaction focuses on quick onboarding, setting precise locations, customizing order details, tracking the journey, and providing post-trip evaluations.
2. **The Driver (Carrier):** An independent contractor who utilizes the platform to find and fulfill delivery requests. The driver requires a specialized, distraction-free interface to scan for nearby orders, accept suitable trips, update the operational status of the delivery, and communicate securely with the passenger.
3. **The Administrator (Admin):** The operational controller of the platform. The admin manages user statuses, oversees the strict verification gateway, monitors live logistical operations, reviews platform feedback, and handles customer support.

3.3 Functional Requirements

The functional requirements define the specific behaviors and features the system must provide to the actors, categorized by their respective workspaces:

3.3.1 Global Layout & Access Control

- **Role-Based Smart Routing:** The system must identify the user's role upon authentication and automatically direct them to their designated workspace while hiding unauthorized tools.
- **Distinct Registration Flows:** The system must provide a fast, seamless registration process for clients, while enforcing a progressive, document-heavy registration path for drivers (requiring vehicle data and legal documents).

- **Dynamic Navigation:** The system must provide a top navigation bar for notifications and profile management, and role-specific sidebars that adapt into a "Floating Pill" interface on mobile devices to prevent screen clutter.

3.3.2 Passenger Hub (Client Operations)

- **Interactive Mapping:** The system must allow clients to set pickup and drop-off locations using smart street search, direct map interaction, or dynamic pin dragging with automatic area name detection.
- **Order Customization:** Clients must be able to specify load types (e.g., furniture, construction materials, electronics, parcels), enter exact cargo weight, and request extra helpers.
- **Instant Pricing Calculation:** The system must calculate and display a detailed price breakdown (base fare + distance fees + weight fees + extra services) before order confirmation.
- **Journey Tracking (Focus Mode):** Upon order acceptance, the system must display live journey statuses (On the way, Arrived for loading, Delivering) along with a Driver Identity Card showing the driver's name, photo, rating, and vehicle type.
- **In-App Communication:** The system must provide a floating chat window allowing the client to message the driver in real-time.
- **Post-Trip Review and Rating:** Upon completion of a journey, the system must prompt the client to submit a numerical rating (1 to 5 stars) and optional textual feedback regarding their experience to ensure accountability.

3.3.3 Driver Workspace (Driver Operations)

- **Interactive Mission Dashboard:** Drivers must be able to toggle their availability ("Online" or "Offline"). The system must present available requests as detailed "Mission Cards" displaying load type, weight, distance, proposed price, and helper requirements.
- **Comprehensive Deliveries Log:** The system must maintain an archive of the driver's operational history, documenting start/arrival times, routes taken, and a summary of each successful shipment.
- **Reputation Management:** The dashboard must reflect the driver's accumulated rating based on client reviews, motivating them to maintain high service standards.
- **Profile and Compliance Management:** Drivers must be able to manage their public profile photo and access a digital documentation center to upload sensitive legal documents (ID, Driver's License).
- **Verification Status Tracking:** The system must notify the driver of their current document approval status (Pending, Approved, or Needs Correction).

3.3.4 Admin Panel (Operations Room)

- **Analytics Dashboard:** The system must display a real-time overview of the application's health, including trip statistics and user growth.

- **Granular User Management:** The system must categorize users into precise statuses: *PENDING PROFILE*, *PENDING VERIFICATION*, *Active*, *Rejected*, and *Suspended*.
- **Strict Verification Gateway:** Admins must be able to evaluate accounts in the *PENDING VERIFICATION* state. If rejected, the system must force the admin to attach a reasoned text note (e.g., "License photo is unclear"), which is immediately reflected on the driver's dashboard.
- **Feedback & Reputation Monitoring:** Admins must have access to a dedicated interface to monitor all passenger reviews and ratings, allowing them to track driver performance metrics and intervene in case of disputes or consistent negative feedback.
- **Rides and Support Management:** Admins must have tools for live monitoring of logistical operations, reviewing historical orders, and handling user complaints and inquiries.

3.3.5 Integrated AI Systems

- **AI Facial Verification & Matching:** During registration, the system must capture the driver's ID photo and a live selfie. It must extract facial features, calculate a similarity score, and provide this metric (e.g., >85% match) to assist the admin's approval decision.
- **AI Virtual Assistant:** The system must feature a chatbot for Tier 1 support that understands contextual queries, answers FAQs regarding pricing or vehicle requirements, and assists users in tracking orders without human intervention.
- **AI Fake Review Detection:** The system must automatically analyze reviews submitted by clients using a machine learning model trained to identify suspicious or fraudulent reviews. The model must evaluate linguistic patterns, delivery-related context, sentiment indicators, and behavioral features to generate a fraud probability score. Reviews classified as suspicious must be flagged for administrative moderation before publication, helping maintain the reliability and trustworthiness of the platform's rating system.

3.4 Non-Functional Requirements

To ensure the platform delivers a professional and reliable experience, it must adhere to the following non-functional constraints:

- **Usability and Accessibility:** The public main page must clearly communicate the platform's value propositions. The user interface must be modern, responsive, and utilize space-saving elements (like the Floating Pill) to ensure a seamless experience across all devices.
- **Security and Trust:** The platform must enforce strict verification protocols, ensuring that no driver can operate without administrative approval and AI facial matching validation. Furthermore, keeping textual reviews private for administrative review while displaying only aggregated star ratings protects drivers from malicious public comments.

- **Real-Time Responsiveness:** Features such as the interactive map, instant pricing calculator, live journey tracking, and the in-app chat window must operate in real-time with minimal latency to facilitate efficient coordination.
- **Transparency:** The system must ensure complete operational transparency, giving drivers clear upfront details on missions and clients clear breakdowns of pricing.

3.5 Use Case Overview

The system's core interactions can be summarized by the following primary workflows:

- **Client Workflow:** A client registers instantly, interacts with the map to customize a load, reviews the instant price, broadcasts the order, tracks the journey, chats with the assigned driver, and finally submits a rating and review upon completion.
- **Driver Workflow:** A driver uploads legal documents and a selfie. Once approved by the admin, they go "Online," preview incoming mission cards, accept a trip, execute the delivery, and review their updated rating in their deliveries log.
- **Admin Workflow:** An admin logs into the operations room, reviews a driver's pending documents against the AI similarity score, approves the account, monitors live platform analytics, oversees user reviews and ratings, and handles escalated support tickets.

3.6 Conclusion

This chapter has thoroughly established the requirements of the platform, defining the distinct roles of the Client, Driver, and Admin. By detailing the functional capabilities—ranging from smart routing and dynamic mapping to strict verification gateways, transparent review systems, and AI-integrated support—and outlining the non-functional expectations, a comprehensive blueprint has been created. These requirements serve as the direct foundation for the system architecture and technical implementation discussed in the subsequent chapters.

CHAPTER 4: SYSTEM ARCHITECTURE

4.1 Introduction

Transitioning from the conceptual requirements and use cases outlined in the previous chapter, this chapter provides the technical blueprint of the Swaply platform. The system architecture defines how various software components, databases, and third-party services interact to form a cohesive, scalable, and secure application. This chapter details the architectural patterns chosen, the separation between the client and server sides, the structural data model, and the rationale behind the selected technology stack.

4.2 Architectural Overview

Swaply is built utilizing a modern, decoupled Client-Server architecture. This approach strictly separates the user interface (Frontend) from the business logic and data management (Backend).

The communication between these two primary layers is facilitated through two distinct protocols:

1. **RESTful API:** Utilized for standard, stateless operations such as user authentication, profile updates, fetching historical delivery logs, and handling multipart form data for document uploading.
2. **WebSocket Protocol (WSS):** Utilized for stateful, full-duplex communication. This is critical for features requiring minimal latency, specifically the in-app messaging system and the live trip status progression (e.g., broadcasting when a driver changes the status to "Arrived").

By decoupling the architecture, the system achieves high modularity, allowing the frontend and backend to scale independently and making future expansions (such as native mobile applications) highly feasible.

4.3 Frontend Structure (Client-Side)

The user interface is built as a Single Page Application (SPA) using React and Vite. The frontend architecture is characterized by a component-based design that minimizes code duplication and simplifies debugging. It incorporates dynamic routing mechanisms to ensure that users (Clients, Drivers, and Admins) are restricted to their designated workspaces securely. [10]

The structural foundation of the frontend repository is organized to maximize maintainability and scalable routing, as detailed in the following table:

Table 4.1: Frontend (Client-Side) Directory and Module Structure

Directory / Module	Description	Key Examples from Codebase
src/pages/	Contains the primary view components mapped to specific application routes. Each file represents a complete screen.	Home.jsx, PassengerDashboard.jsx, DriverDashboard.jsx, AdminUsers.jsx
src/components/	Houses reusable, isolated UI elements utilized across different pages to maintain visual consistency.	Navbar.jsx, FloatingChat.jsx, PassengerSidebar.jsx, ProtectedRoute.jsx
src/context/	Manages global application states using the React Context API, preventing prop-drilling across the component tree (e.g., managing user sessions).	AuthContext.jsx
src/layouts/	Structural wrappers that define the skeleton (e.g., permanent sidebars) for specific user roles.	AdminLayout.jsx
Root Assets	Core initialization files for styles, environment configurations, and the main React DOM injection.	App.jsx, main.jsx, index.css

4.4 Backend Structure (Server-Side)

The backend acts as the central intelligence and data broker of the Swaply platform. It is engineered using Python and the FastAPI framework. The core logic is built on an asynchronous framework, which is a critical design choice allowing the server to maintain active WebSocket connections smoothly without blocking the main execution thread. [11]

The directory structure follows a logical Domain-Driven Design (DDD) approach, separating database configurations, API endpoints, and utility functions:

Table 4.2: Backend (Server-Side) Directory and Module Structure

Directory / Module	Description	Key Examples from Codebase
app/routes/	The core API endpoints logically grouped by domain. These files handle incoming HTTP requests and map them to business logic.	auth.py, orders.py, ai.py, admin.py, rides.py
app/utils/	Helper modules containing shared logic, security operations (e.g., JWT validation), and complex protocols.	security.py, websocket_manager.py
app/ (Root)	The entry points and structural definitions for the database connection, data schemas, and the FastAPI application instance.	main.py, database.py, models.py
uploads/	A dedicated local storage directory handling all inbound physical media files submitted by users during operations.	Driver Licenses, ID Cards, Profile Selfies (.jpg, .png)
temp_ai/	A temporary buffer directory utilized by the backend to process files before passing them to machine learning models for facial matching.	Temporary image extractions

4.5 Data Model

To accommodate the dynamic and hierarchical nature of logistics data, the system employs a NoSQL, document-oriented database model. This allows for flexible schemas where data is stored in BSON (Binary JSON) format, aligning perfectly with the JSON payloads exchanged via the API. [20]

The primary data collections in the database include:

- **Users Collection:** Stores credentials, hashed passwords, role identifiers, and nested objects containing driver verification documents and precise operational states (e.g., *PENDING VERIFICATION*, *Active*).
- **Orders Collection:** Functions as the ledger for all logistical transactions. It stores pickup/drop-off locations, cargo configurations (weight, helpers), calculated pricing, the assigned driver ID, and an array of status updates.
- **Reviews Collection:** Isolates passenger feedback and star ratings, preventing the main user documents from becoming bloated while allowing the administrative dashboard to aggregate platform metrics efficiently.

4.6 Technology Stack

The technology stack was carefully selected to prioritize execution speed, modern development standards, and robust community support:

- **Frontend Technologies: React.js** for building the interactive UI due to its efficient Virtual DOM. Styling is managed via utility-first CSS frameworks (Tailwind CSS) to ensure responsive design. **Vite** is used as the build tool for optimized performance.
- **Backend Technologies: Python** coupled with **FastAPI** powers the server. FastAPI was selected due to its exceptional asynchronous performance, native WebSocket support, and seamless integration with Python's AI libraries.
- **Database: MongoDB** serves as the primary database. Its flexibility handles the varied structures of user profiles and complex order details effortlessly. [12]

4.7 System Dynamics (Sequence Diagrams)

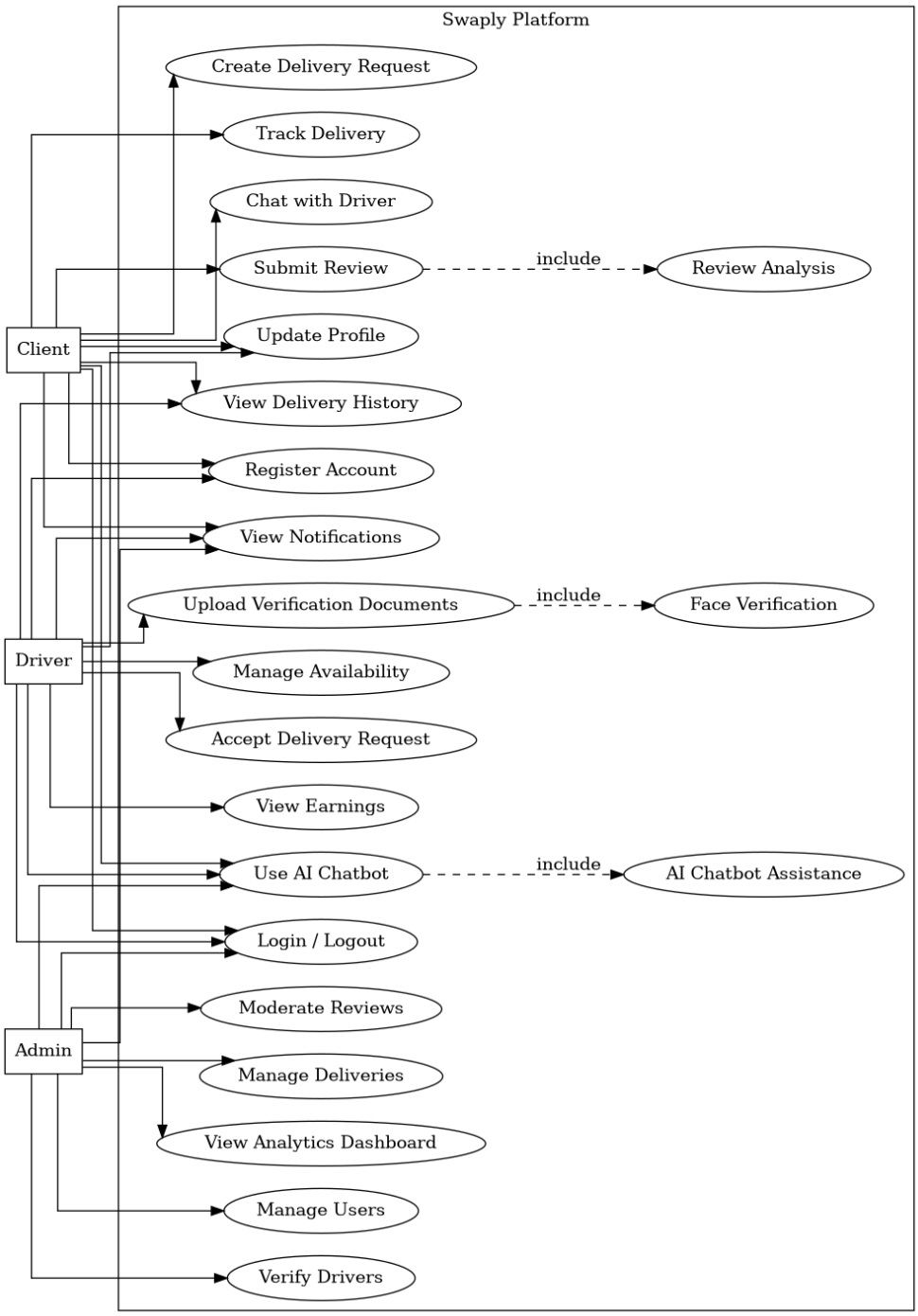

Figure 4.1 -Use Case Diagram of the Swaply Platform

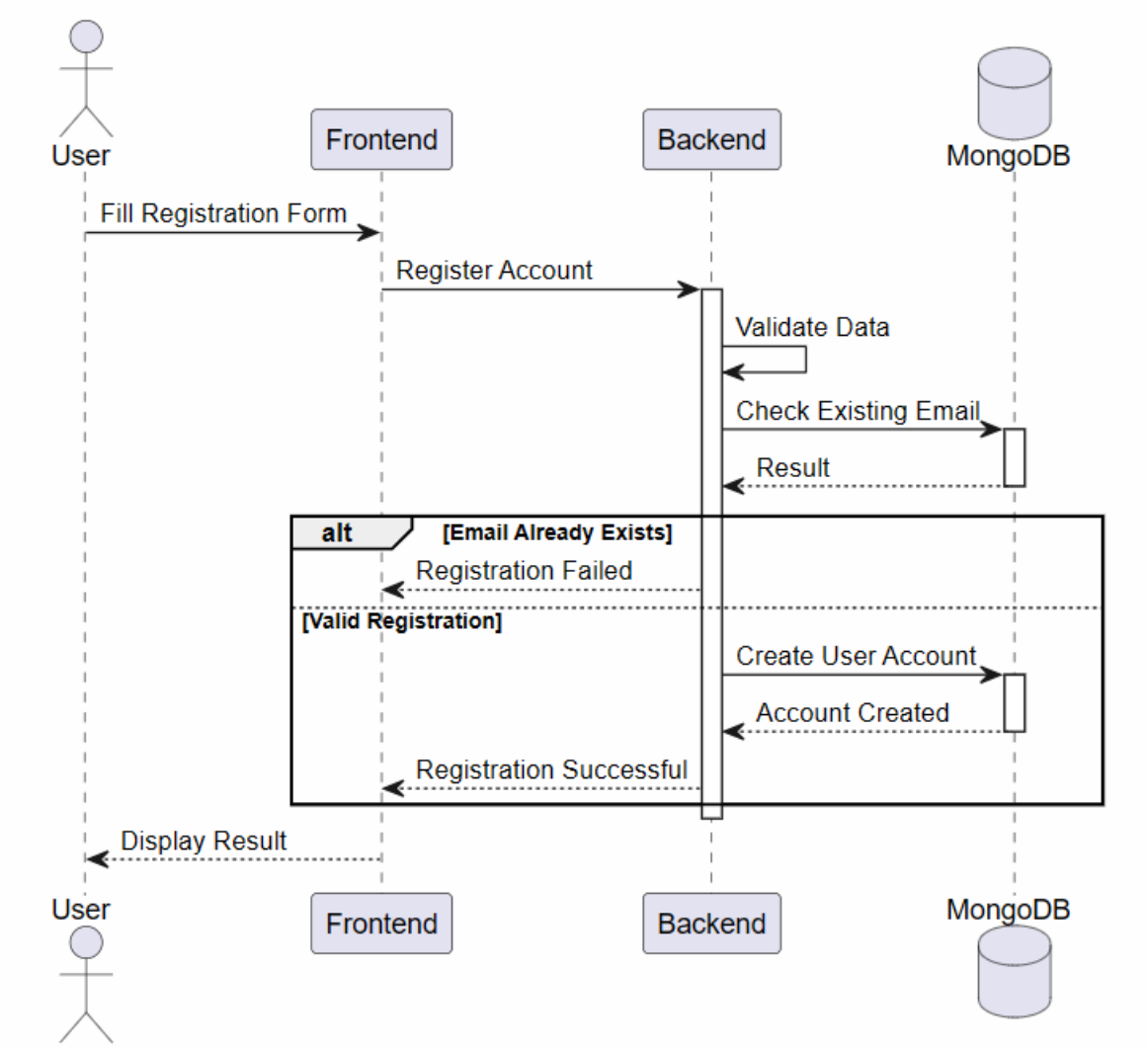

Figure 4.2 - User Registration Sequence Diagram

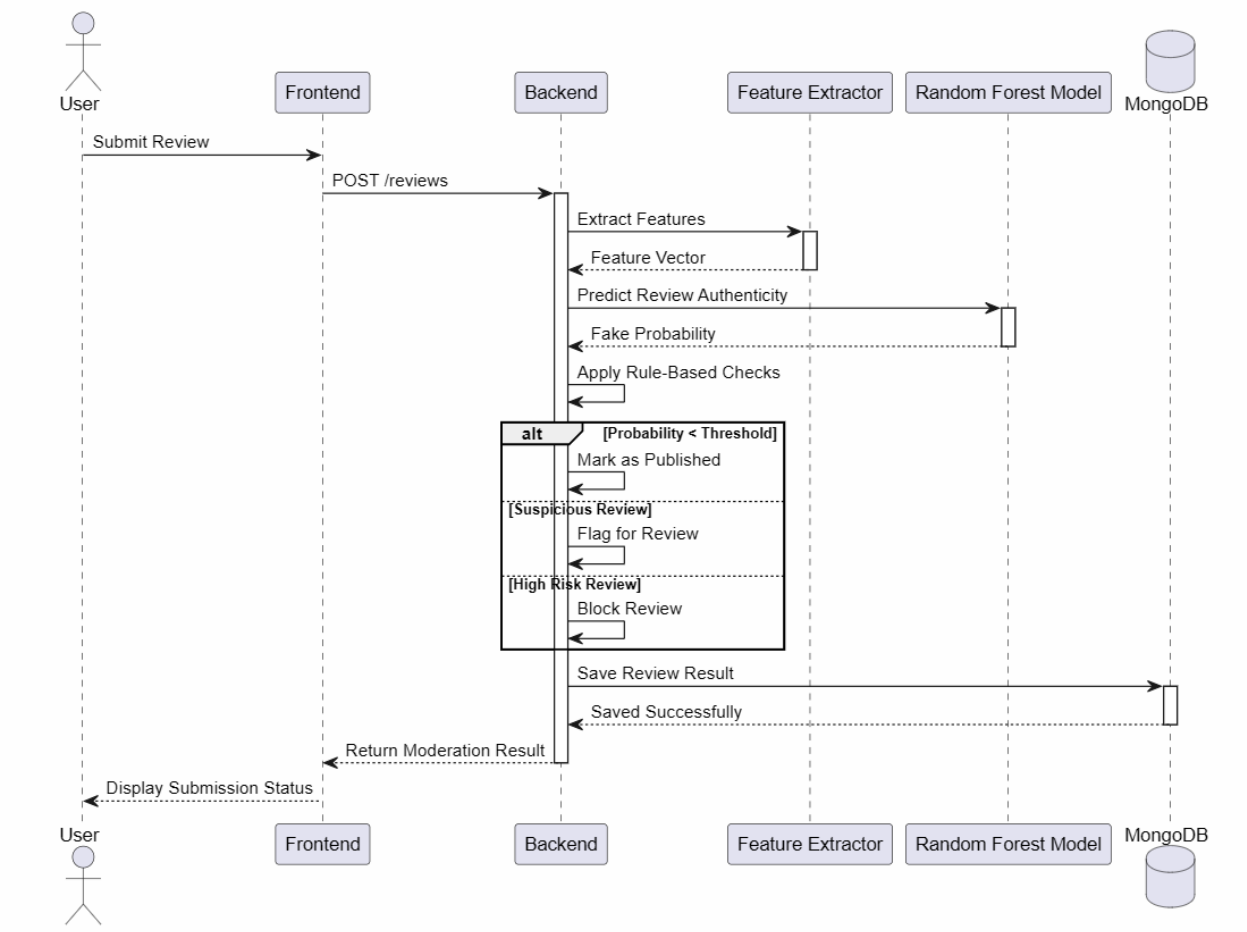

Figure 4.3 - Fake Review Detection Sequence Diagram

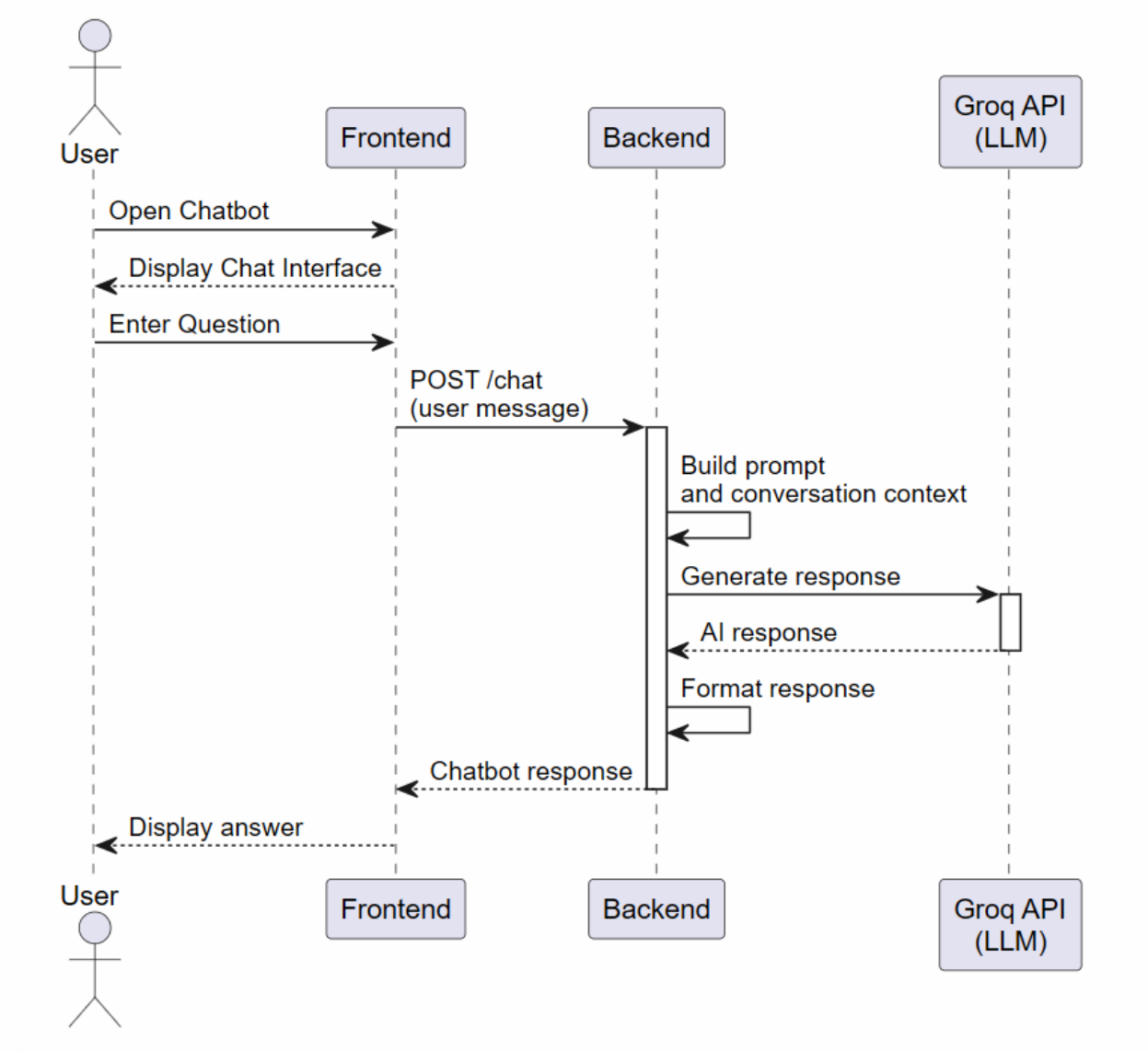

Figure 4.4 - Chatbot Interaction Sequence Diagram

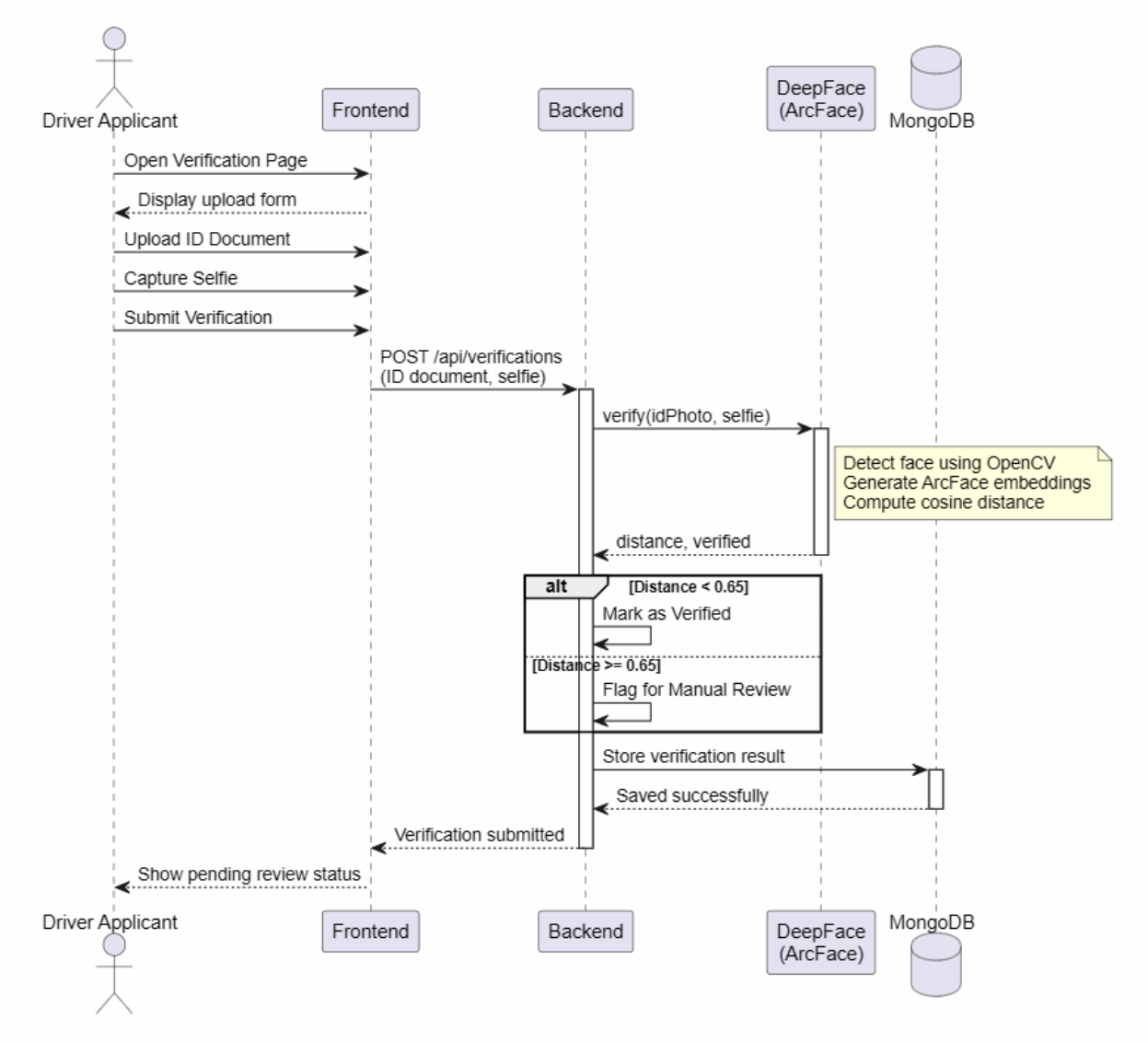

Figure 4.5 - Face Verification Sequence Diagram

4.8 Conclusion

The architectural design of Swaply provides a highly responsive, secure, and scalable foundation. By leveraging React for a dynamic user interface, Python and FastAPI for high-performance backend logic and AI integration, and MongoDB for flexible data storage, the system is optimally structured to meet the rigorous demands of a real-time logistics network. The modular separation of files clearly delineates responsibilities, facilitating the smooth execution of the implemented modules that will be discussed in the following chapter.

CHAPTER 5: IMPLEMENTED MODULES

5.1 Introduction

Following the architectural design detailed in the previous chapter, this chapter showcases the practical realization of the Swaply platform. It presents the graphical user interfaces (GUI) and the functional modules implemented using the React-FastAPI stack. By leveraging utility-first styling and a component-based architecture, the interfaces are designed to be intuitive and responsive. The focus is on demonstrating how the theoretical requirements are translated into interactive features for three distinct user roles: Passengers, Drivers, and Administrators.

5.2 Authentication and Smart Routing

The entry point to the system is a secure authentication gateway. This module manages user identities through dedicated **Login and Registration interfaces**. Upon successful authentication, the application utilizes a smart, protected routing mechanism. This router acts as a strict traffic controller: it directs passengers to their dashboard, grants access to administrators, and crucially, intercepts newly registered drivers who have not yet completed their verification, redirecting them to a mandatory onboarding flow.

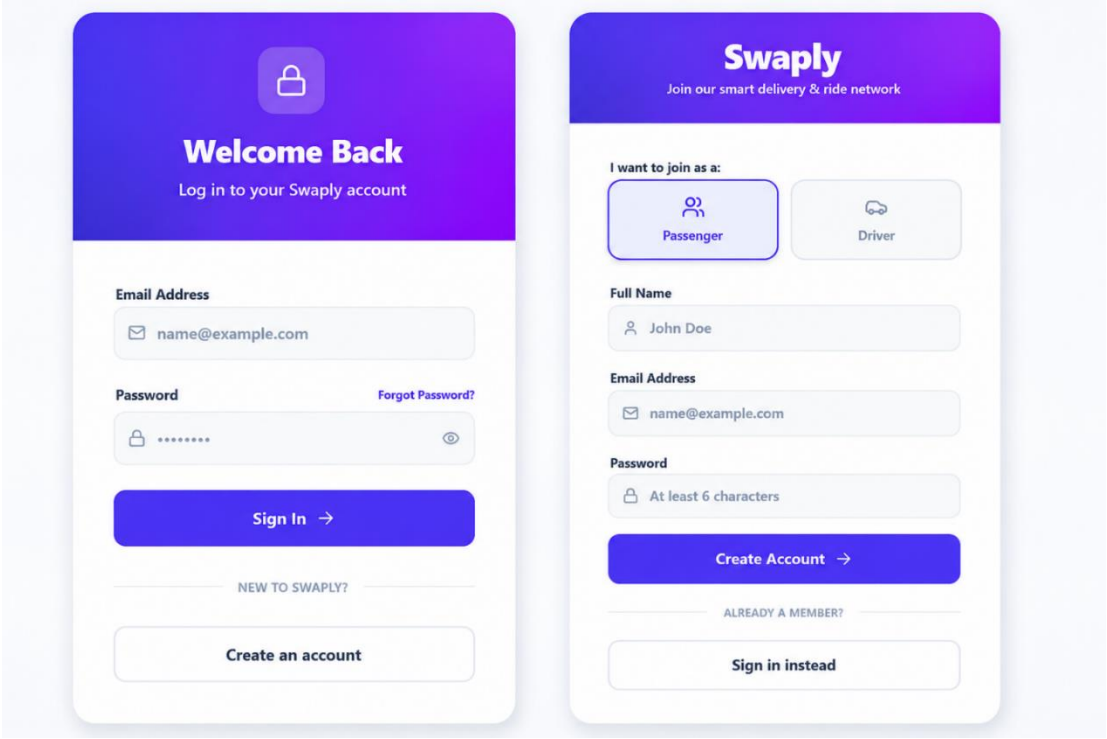

Figure 5.1 - User Login and Registration Interfaces

5.3 Delivery Management Module (Passenger Side)

The **Passenger Dashboard interface** serves as the primary workspace for creating logistics requests. This module integrates interactive mapping components that allow clients to pinpoint specific pickup and drop-off locations. Once the spatial coordinates and cargo details (weight, type, helpers) are set, the system dynamically calculates the distance and provides an instant price breakdown before the user broadcasts the order to the network. [14]

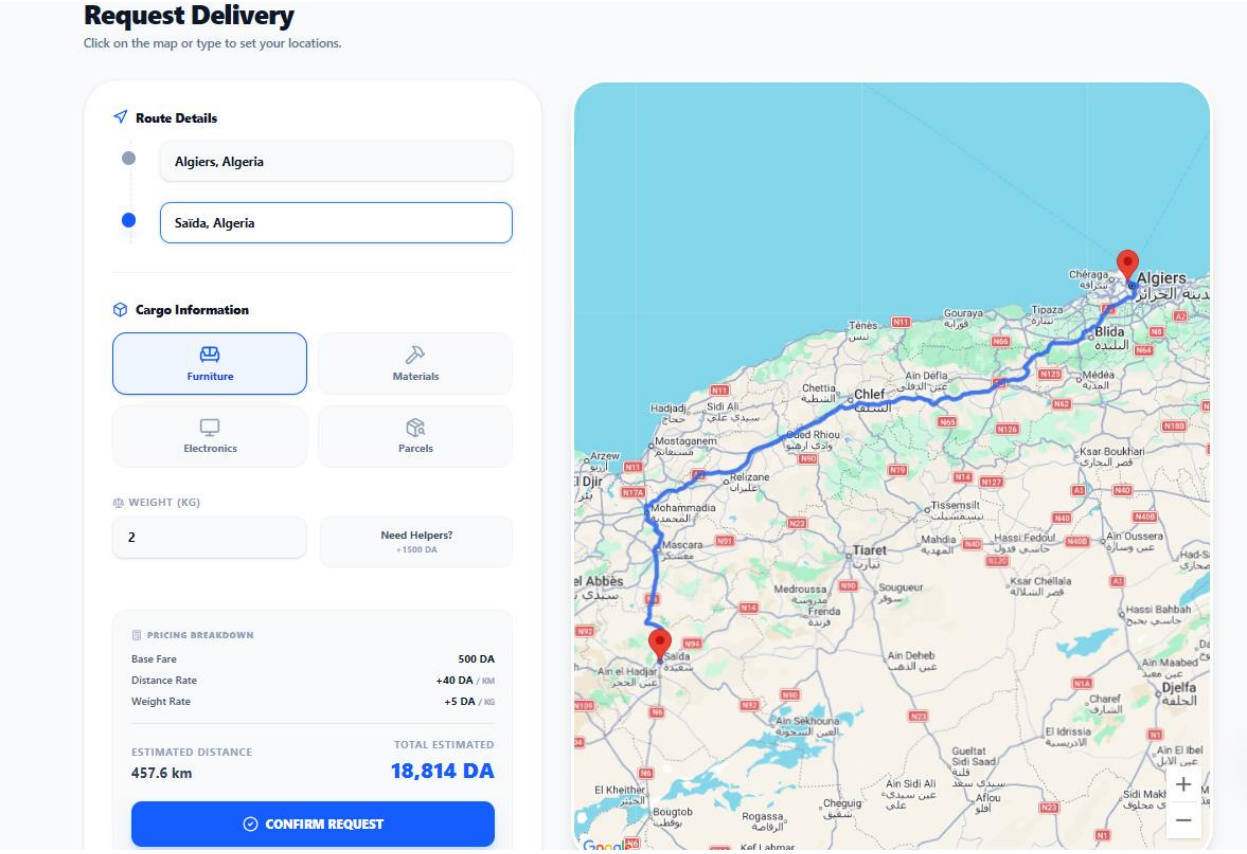

Figure 5.2 - Passenger Dashboard and Order Creation Flow

5.4 Trip Management Module (Driver Side)

For fully verified drivers, the **Driver Dashboard component** is the main operational hub. This module features a status toggle that allows drivers to switch between "Online" and "Offline" availability. When online, the interface dynamically populates with "Mission Cards" representing nearby requests, displaying essential logistical data such as load weight, total distance, and potential earnings, enabling autonomous decision-making.

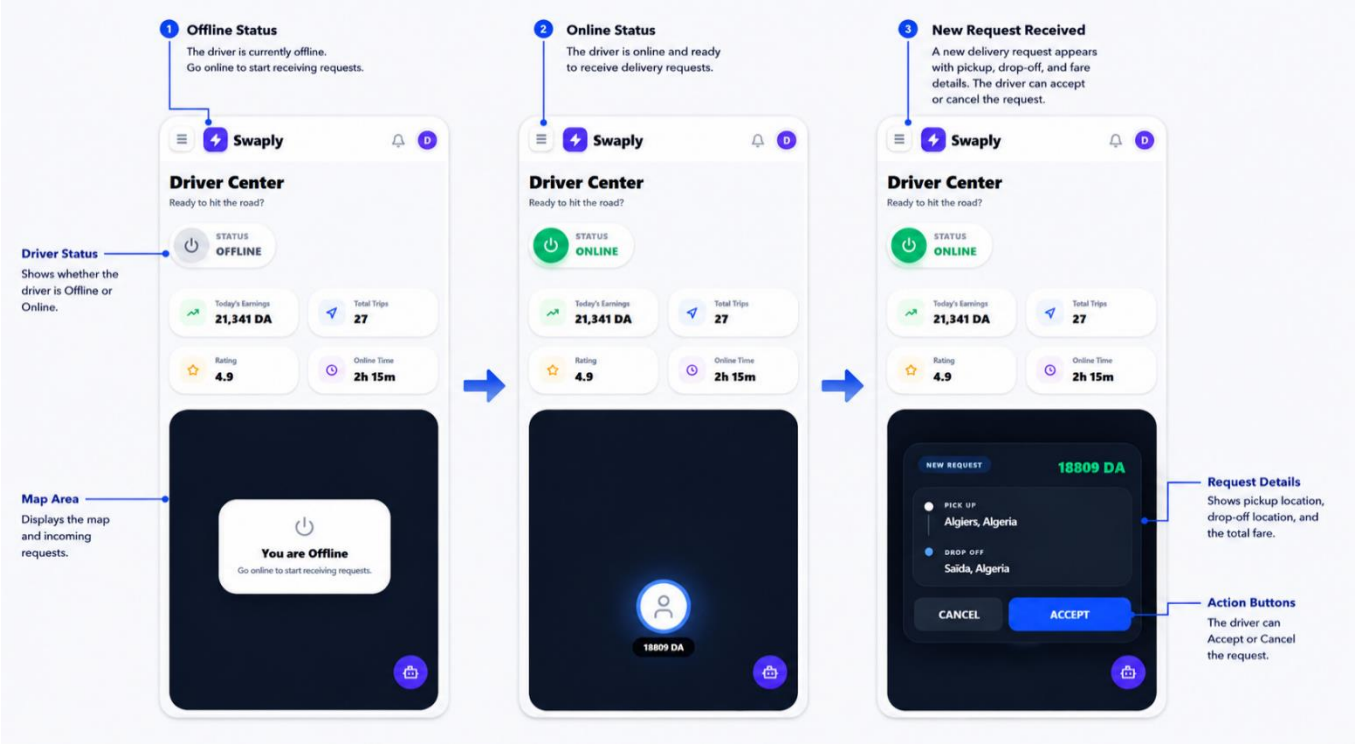

Figure 5.3 - Driver Dashboard and Active Mission Cards

5.5 Real-Time Trip Lifecycle and State Management

Once a driver accepts a mission, the system transitions both the passenger and the driver into a synchronized "Focus Mode." This module meticulously manages the lifecycle of the delivery through distinct, sequential states: Assigned, Arrived (at pickup), In Transit, and Completed. The interface utilizes a dynamic progress stepper and real-time map visual updates. any state change triggered by the driver (e.g., swiping to confirm pickup) instantaneously reflects on the passenger's dashboard, ensuring absolute transparency and seamless tracking throughout the delivery process.

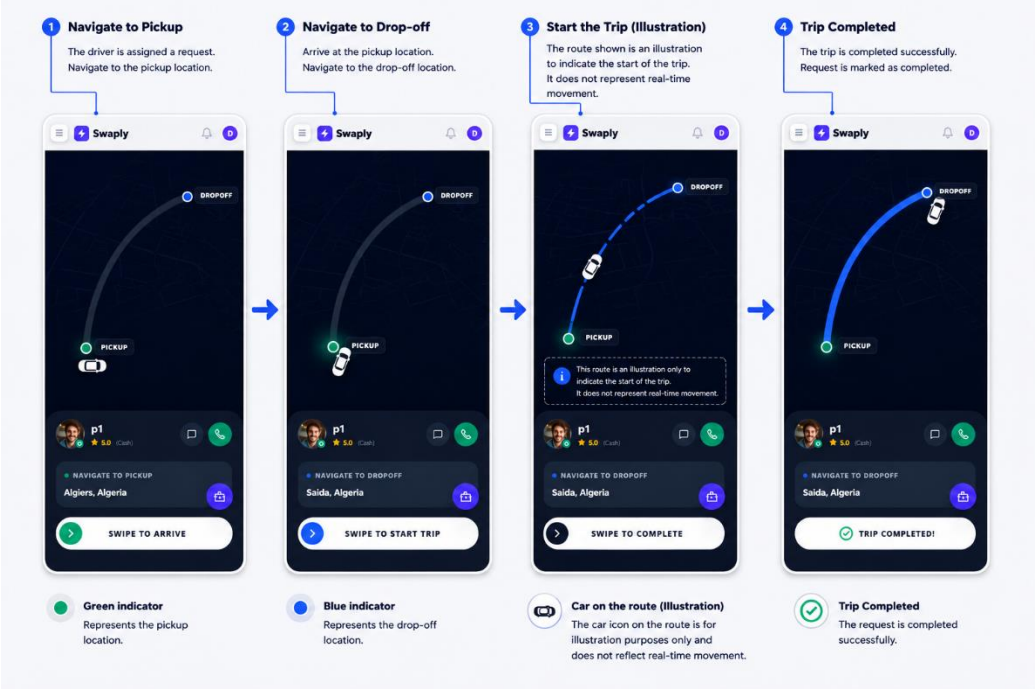

Figure 5.4 - Real-Time Trip State Tracking and Focus Mode Interfaces

5.6 Driver Onboarding and Verification Gateway

Security and trust are strictly enforced at the platform's entry level. The driver onboarding process is a gated workflow. Upon initial registration, a driver is completely restricted from accessing the main application.

1. The system automatically redirects them to a **Profile Completion interface**, where they must upload their identity cards, driver's licenses, and a live selfie.
2. Once submitted, the driver's account state changes, and they are transitioned to a dedicated **Waiting Approval view**. They remain locked in this interface until the administrative review is concluded.

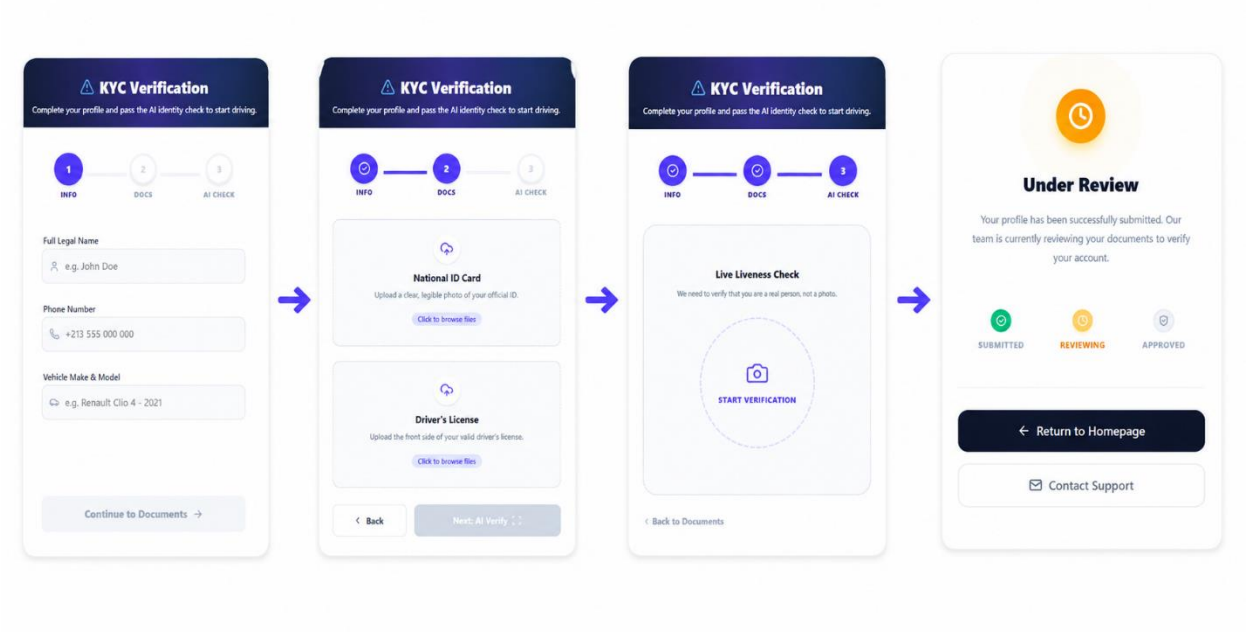

Figure 5.5 - Driver Profile Completion and Waiting Approval Interfaces

5.7 Admin Dashboard & Verification Control Room

The administrative module acts as the central command. Operating within a specialized Administrative layout, administrators can monitor platform-wide activity. Crucially, the Verification Control Room serves as the core of identity management, resolving the driver verification queue. For every driver in the "Waiting Approval" state, the interface abstracts the raw image data by directly presenting the AI Similarity Score on the screen—a metric autonomously calculated in the backend by comparing the uploaded ID document with the live selfie. This streamlined approach allows the administrator to exercise efficient "Human-in-the-Loop" oversight. By relying on the AI's rapid feature extraction and matching score, the admin retains the ultimate decision-making power to swiftly Approve the driver (granting them dashboard access) or Reject the profile, without the need to manually cross-examine the raw photos.

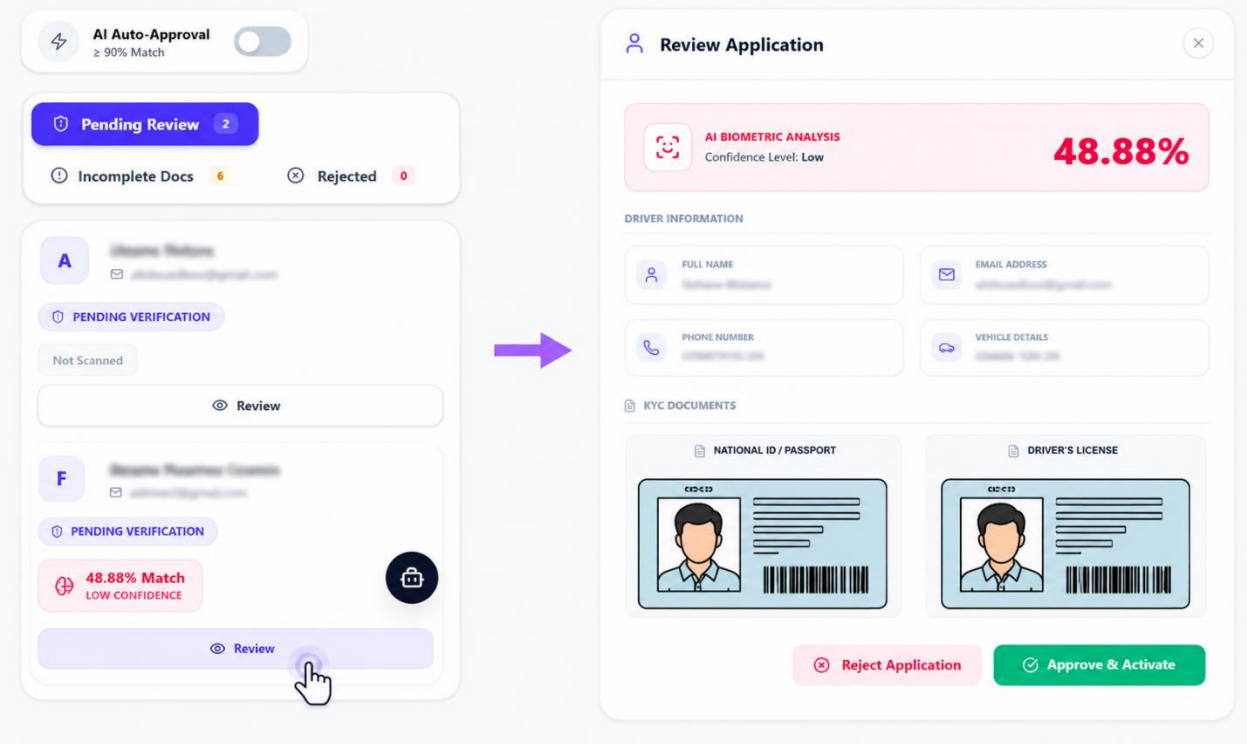

Figure 5.6 - Admin Identity Verification Interface with AI Similarity Results

5.8 Virtual Assistant Interface (AI Chatbot)

To enhance user support and reduce the administrative burden of manual ticket handling, the platform integrates an AI-powered Virtual Assistant. Users can access this support module via a dedicated conversational interface. This frontend component allows users to input natural language queries regarding platform usage, pricing rules, or policy guidelines. The interface displays the AI's instant, context-aware responses in a familiar chat-bubble layout, providing seamless 24/7 automated support.

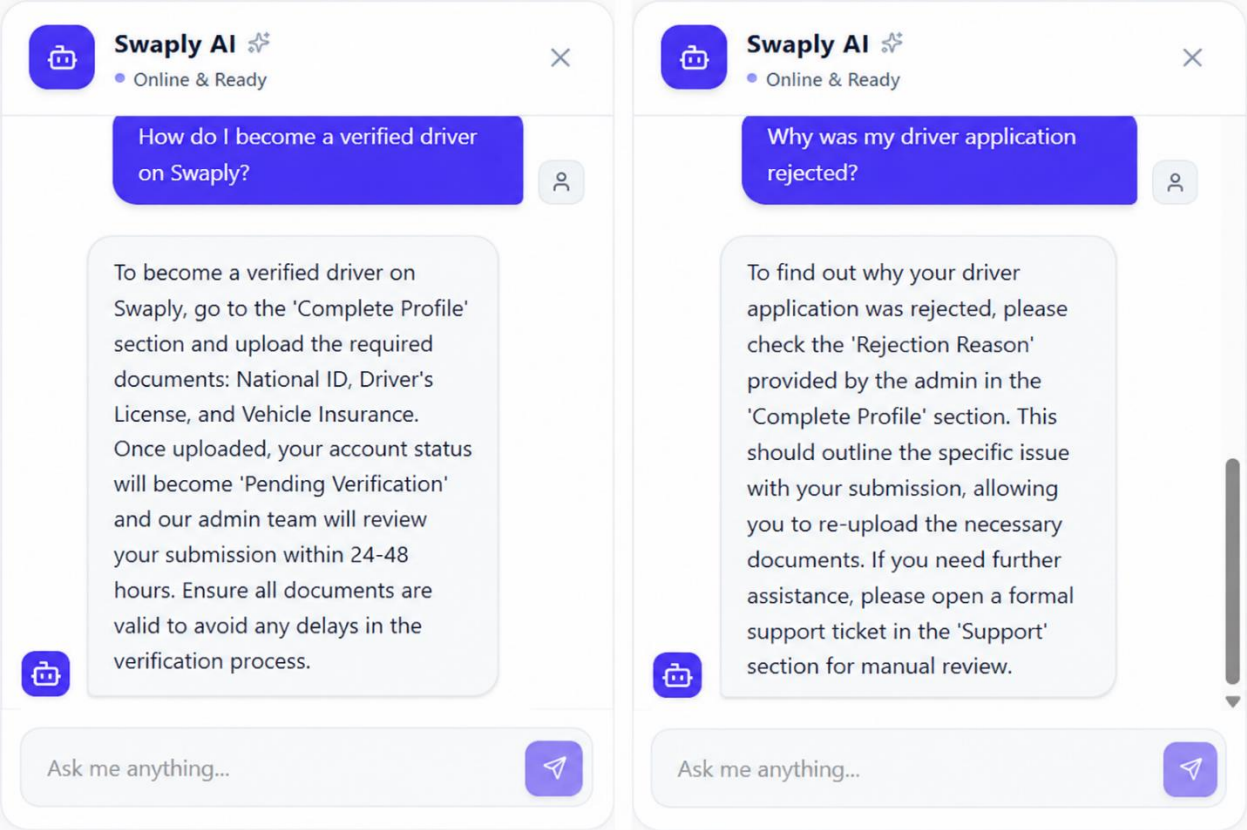

Figure 5.7 - AI Virtual Assistant / Chatbot Interface

5.9 Trust, Rating, and AI Review Filtering

To maintain high service standards, a mandatory feedback loop triggers post-trip, allowing passengers to evaluate drivers via a 5-star rating and a text review. To protect platform integrity, this module strictly integrates with an AI backend.

Before publication, text reviews undergo real-time AI analysis to detect fake or inappropriate content, generating a "Fake Probability" score. Low-risk reviews are automatically marked as Published and immediately update the driver's trust score. Conversely, high-risk reviews are intercepted as Pending Admin and routed to the Admin Dashboard for manual human verification. This filtering ensures that a driver's reputation is built solely on authentic user experiences.

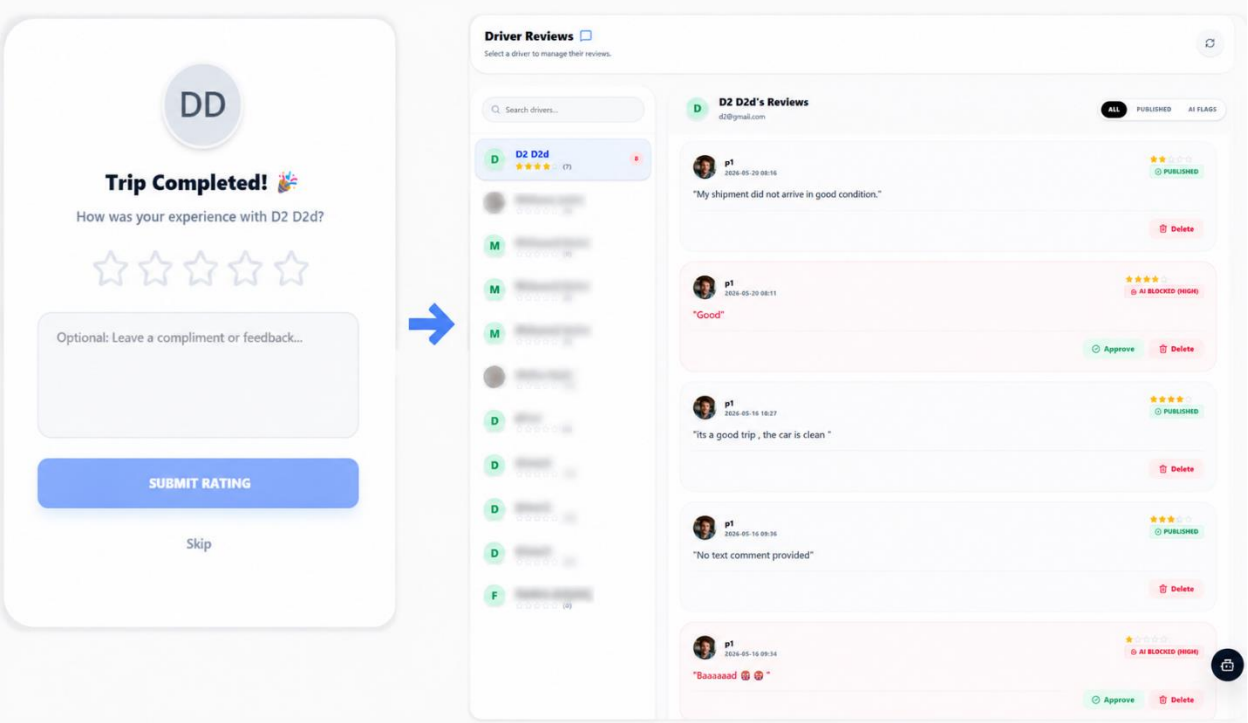

Figure 5.8 - Rating and Feedback Modal

5.10 Conclusion

The implementation phase successfully transformed the architectural blueprints into a fully functional and secure logistics ecosystem. The strict state-based routing ensures that only verified drivers can interact with passengers, while the modular design of the React frontend provides a seamless user experience across all roles. The following chapter will delve deeply into the advanced machine learning integrations that power the facial verification and the intelligent virtual assistance functionalities highlighted in this chapter.

Chapter 6: AI Systems — Integration and Evaluation

6.1 Introduction

The Swaply platform integrates three distinct artificial intelligence systems, each addressing a specific operational challenge within the platform ecosystem. These systems operate on fundamentally different paradigms and were selected based on their individual suitability for the tasks they perform within the platform architecture.

The first system, the face verification module, employs a pre-trained deep learning model to perform biometric identity comparison between a traveler's uploaded identity document and their live selfie. The second system, the fake review detector, is a supervised machine learning classifier that was trained from scratch on a real-world dataset of 40,432 labeled reviews. The third system, the customer support chatbot, leverages a transformer-based large language model accessed through the Groq API to provide intelligent and context-aware assistance to platform users.

This chapter is organized around each of these three systems. For each system, the technical architecture is described, the integration methodology within the Swaply platform is explained, and the performance evaluation is presented. Where a system uses pre-trained models, the published benchmark results from the original authors are cited and the justification for not performing additional training is clearly explained. Where a system was trained as part of this project, the training procedure, dataset, and evaluation results are presented in full detail.

The combination of these three systems establishes a comprehensive artificial intelligence layer within the Swaply platform that addresses three distinct challenges simultaneously: identity security through biometric verification, platform integrity through automated review moderation, and user experience through intelligent conversational assistance.

6.2 AI System 1 — Face Verification Module

6.2.1 Overview and Purpose

The face verification module is responsible for validating the identity of drivers during the registration and verification process. Before a driver account can be approved, the system compares the facial image extracted from the uploaded identity document with a selfie provided by the applicant.

In a peer-to-peer delivery platform such as Swaply, identity verification is a critical security requirement. Drivers are entrusted with transporting packages on behalf of other users, making it essential to ensure that the individual submitting registration documents is the legitimate owner of those documents.

The module performs face verification rather than face recognition. Face verification is a one-to-one biometric comparison task that determines whether two facial images belong to the same individual. Unlike face recognition systems, which attempt to identify an unknown person from a database of known identities, the Swaply verification module only determines whether the submitted selfie matches the photograph contained in the uploaded identification document.

6.2.2 Technical Architecture and Verification Pipeline

The face verification system is implemented using the DeepFace framework [6], an open-source facial analysis library that provides access to state-of-the-art face recognition models and verification pipelines. The system performs verification through a sequence of automated processing stages.

Stage 1 — Image Acquisition:

Two images are provided as input to the verification process:

• The photograph extracted from the uploaded identity document.

• A live selfie captured by the applicant during registration.

Both images are temporarily stored and passed to the verification module for analysis.

Stage 2 — Face Detection:

The OpenCV face detection backend is used to locate facial regions within both images. This step identifies the face area and prepares it for feature extraction. To improve robustness in real-world registration scenarios, face detection enforcement is disabled, allowing the system to continue processing even when image quality is imperfect. [7]

Stage 3 — Facial Representation Extraction:

The ArcFace deep learning model generates a numerical representation of each detected face. ArcFace is a modern face recognition architecture designed to produce highly discriminative facial embeddings while maintaining robustness to variations in pose, illumination, and facial expression.

Stage 4 — Similarity Computation:

After extracting facial embeddings from both images, the system computes the cosine distance between the two representations. Cosine distance measures the similarity between embedding vectors and provides a numerical indication of whether both images belong to the same person.

Stage 5 — Verification Decision:

A decision threshold of 0.65 is applied to the computed cosine distance. If the distance is below the threshold, the system considers both images to represent the same individual and returns a successful match result. Otherwise, the verification is considered unsuccessful and the application is flagged for administrative review.

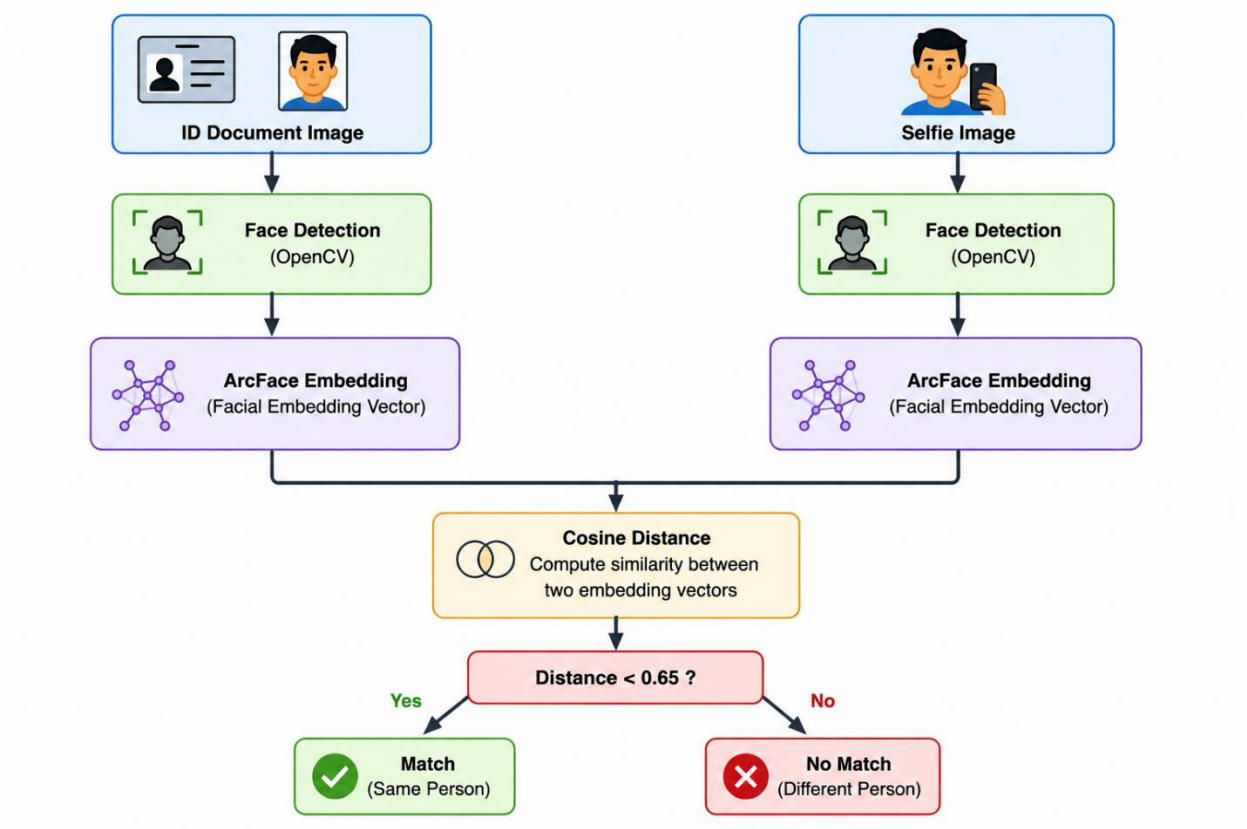

Figure 6.1: Face Verification Processing Pipeline

6.2.3 System Components

The face verification module consists of several interconnected components that collectively perform the identity verification process. Each component contributes a specific function within the verification pipeline, from facial detection to similarity evaluation and final decision making.

Table 6.1: Components of the Face Verification System

Component	Primary Function	Technology Used	Role in Swaply
OpenCV Face Detector	Face Detection	OpenCV	Detects and extracts facial regions from uploaded images
ArcFace Model	Facial Representation	ArcFace	Generates facial embeddings for identity comparison
Cosine Distance	Similarity Measurement	Cosine Metric	Measures similarity between ID photo and selfie
Verification Threshold	Decision Making	Threshold = 0.65	Determines whether both images belong to the same person

The combination of these components enables an efficient and reliable verification process capable of detecting identity mismatches while maintaining a smooth registration experience for legitimate users.

6.2.4 Integration with the Swaply Platform

The face verification module is integrated into the driver onboarding workflow. During registration, applicants are required to upload their identity documents and complete a selfie verification step.

The backend service receives both images and forwards them to the verification module. The module performs face detection, feature extraction, and similarity analysis before generating a verification result.

Applications that satisfy the verification threshold proceed to the next stage of the approval workflow, while applications that fail the verification process are marked for manual administrative review. This approach combines automated verification with human oversight, improving both security and reliability.

6.2.5 Evaluation and Model Justification

The face verification system relies on the ArcFace model through the DeepFace framework. ArcFace is a widely adopted facial verification architecture designed to produce highly discriminative facial embeddings suitable for identity verification tasks.

Because the model is pre-trained on large-scale facial datasets and integrated through the DeepFace framework, additional training was not required for the Swaply platform. Instead, the project leverages the model's established performance and robustness in real-world face verification scenarios.

The use of a pre-trained model significantly reduces development complexity while providing a reliable and practical solution for identity verification within the platform.

6.3 AI System 2 — Fake Review Detector

6.3.1 Overview and Purpose

The fake review detection system was designed to protect the integrity of the Swaply reputation system by automatically analyzing submitted reviews and identifying those that exhibit patterns associated with fraudulent, artificially generated, or manipulated content. In peer-to-peer delivery platforms, the star-based rating and review system serves as the primary mechanism through which users establish trust and make decisions about which couriers or senders to engage with. The presence of fake reviews, whether posted by users attempting to inflate their own ratings or to damage the reputation of competitors, directly undermines this trust mechanism and degrades the quality of the platform as a whole.

Unlike the face verification module, the fake review detector was trained from scratch as part of this project using a publicly available labeled dataset. This system therefore constitutes the primary original machine learning contribution of the project and is the subject of a complete training and evaluation cycle.

6.3.2 Dataset Description

The model was trained on a publicly available fake reviews dataset containing 40,432 labeled reviews. The reviews were originally collected from Amazon product listings and provide binary labels indicating whether each review was authored by a genuine human reviewer or generated by an automated computer system for the purpose of manipulating product ratings.

The dataset is perfectly balanced between the two classes, eliminating the need for oversampling or class weighting techniques during training. The reviews span multiple product categories including home goods, electronics, books, and clothing, providing sufficient linguistic diversity for the trained model to generalize across different writing styles and domains.

Although the training dataset consists of product reviews rather than delivery service reviews, the features engineered for the classifier were specifically designed to capture universal indicators of fake content that are domain-independent, such as lexical diversity, [4] Authentication extremity, and writing style patterns. Additionally, a set of sixteen delivery-specific features were added to the feature engineering pipeline to improve the model's sensitivity to the types of fake reviews most likely to appear on a delivery platform.

6.3.3 Feature Engineering

The classifier operates on a set of 41 hand-engineered features extracted from review text rather than on raw text sequences. This feature-based approach was selected over deep learning text classification alternatives because it provides full interpretability of the model's decision-making process, requires significantly fewer computational resources for training and inference, and produces highly competitive performance on this type of classification task.

6.3.4 Model Selection and Architecture

A Random Forest ensemble classifier was selected as the machine learning algorithm for this task. The Random Forest algorithm constructs a large number of decision trees during training, each trained on a random subset of the training data and a random subset of the available features. At inference time, each tree independently classifies the input and the final prediction is determined by majority vote across all trees. [13]

This algorithm was selected over alternative approaches for several reasons. Random Forest classifiers are highly robust to overfitting due to the inherent diversity introduced by random subsampling of data and features. They handle heterogeneous feature sets naturally, requiring no feature scaling or normalization. They provide direct interpretability through feature importance scores, which allow the most informative detection signals to be identified and analyzed. They also train efficiently on datasets of the size used in this project without requiring GPU infrastructure.

6.3.5 Training Procedure

The full dataset of 40,432 reviews was divided into three subsets using stratified splitting to preserve the class distribution in each subset.

Feature extraction was applied to all 40,432 reviews prior to splitting to ensure consistent feature computation across all subsets. The Random Forest classifier was then trained on the 32,345

training samples. No extensive hyperparameter optimization was performed beyond the initial configuration, as preliminary experiments demonstrated that the selected parameters produced stable and consistent performance across both validation and test sets.

Table 6.2: Dataset Distribution for Fake Review Detection

Class	Label	Count	Percentage
Real Reviews	OR	20,216	50.0%
Fake Reviews	CG	20,216	50.0%
Total		40,432	100.0%

6.3.6 Feature Groups

Group 1 — Linguistic and Text Features (18 Features)

- Word count
- Character count
- Average word length
- Sentence count
- Average sentence length
- Exclamation mark count
- Question mark count
- Period count
- Comma count
- Capitalization ratio
- All-capitals word count
- Maximum word repetition count
- Sentiment polarity score
- Sentiment subjectivity score
- Unique word ratio
- Emoji count
- URL count
- Numeric token count

Group 2 — Derived Binary Indicator Features (7 Features)

- Very short review flag (fewer than 10 words)

- Very long review flag (more than 100 words)
- High word repetition flag
- Excessive exclamation flag (more than 3 exclamation marks)
- URL presence flag
- All-capitals flag
- Low lexical diversity flag

Group 3 — Delivery-Specific Features (16 Features)

- Delivery keyword count and ratio
- Timing keyword count and ratio
- Service quality keyword count and ratio
- Package condition keyword count and ratio
- Product vocabulary presence flag
- Generic spam phrase count
- Mention of a person flag
- Mention of a location flag
- Mention of communication flag
- Delivery context presence flag
- Generic content flag
- Product language presence flag

Table 6.3: Random Forest Configuration Parameters

Parameter	Value
n_estimators	300 decision trees
max_depth	20
min_samples_split	5
min_samples_leaf	2
max_features	sqrt(n_features)
class_weight	balanced
random_state	42
n_jobs	-1 (all CPU cores)

Table 6.4: Dataset Split for Training, Validation, and Testing

Subset	Split	Count	Purpose
Training Set	80%	32,345	Model fitting
Validation Set	10%	4,043	Hyperparameter tuning
Test Set	10%	4,044	Final evaluation

5.3.7 Evaluation Results

The trained model was evaluated on the held-out test set of 4,044 reviews that were not seen during training. The evaluation results are summarized in the table below.

Table 6.5: Evaluation Metrics of the Fake Review Detector

Metric	Score
Accuracy	87.20%
Precision	86.97%
Recall	87.51%
F1-Score	87.24%
AUC Score	0.9468

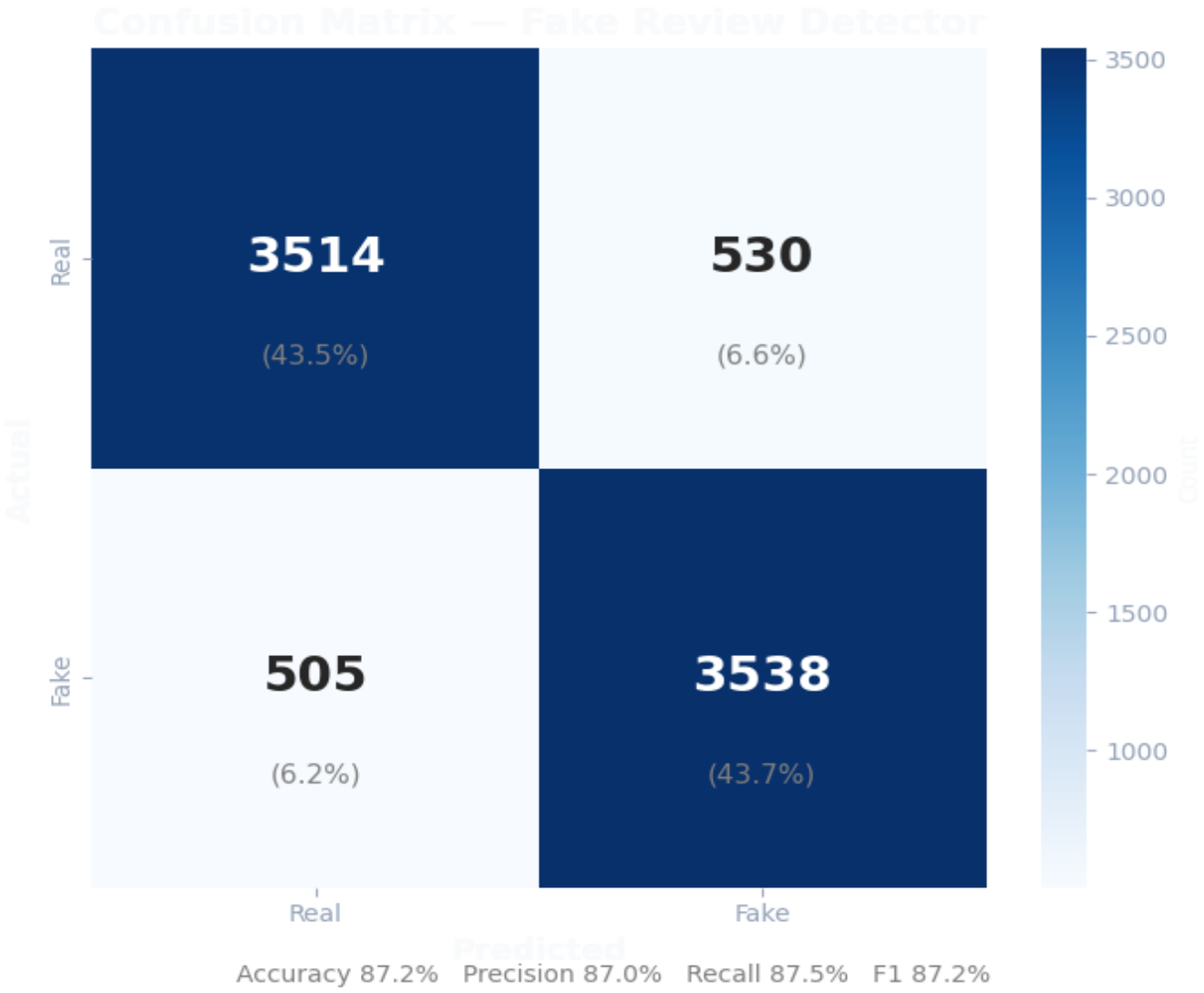

Figure 6.2: Confusion Matrix of the Fake Review Detector

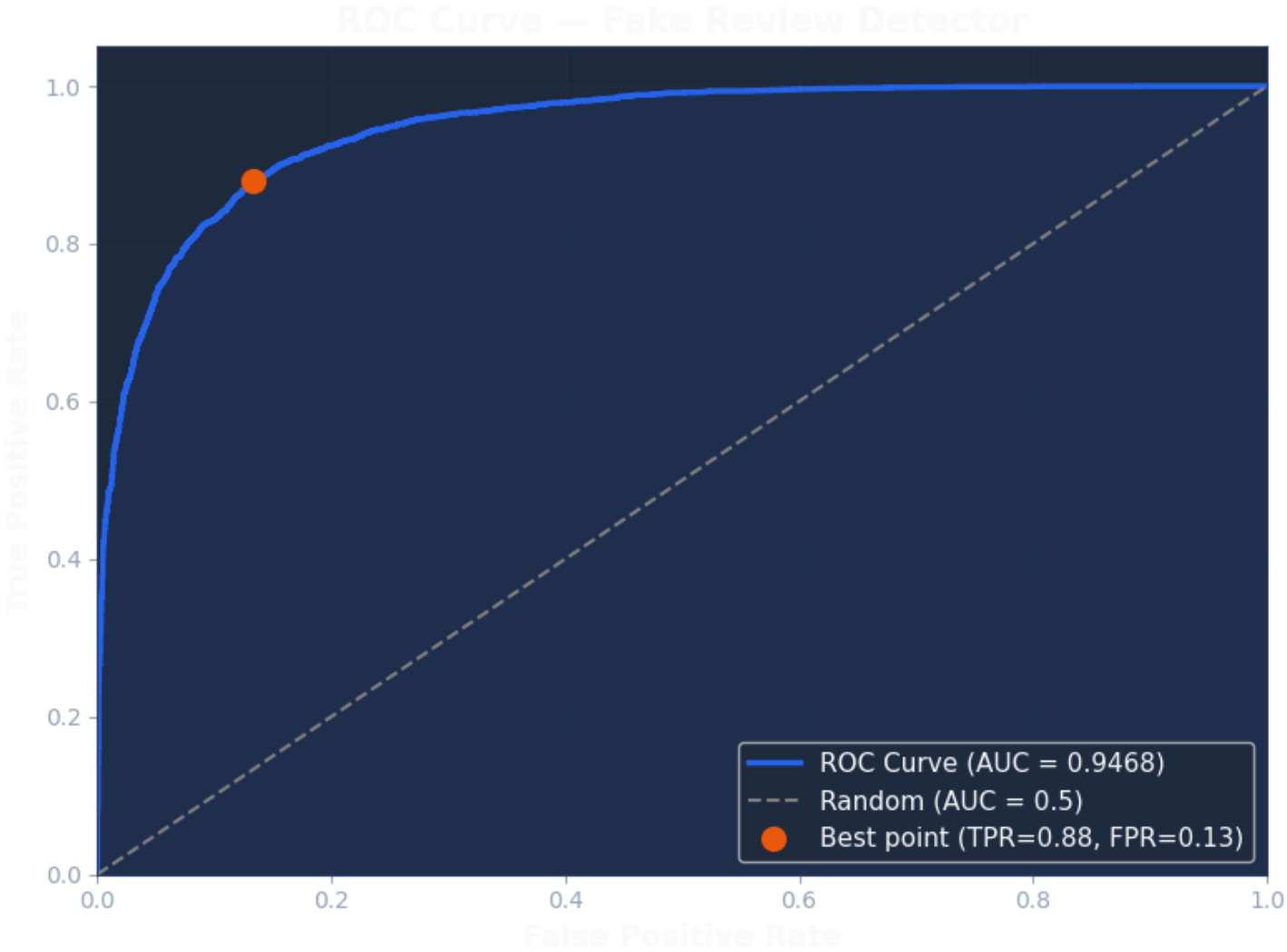

Figure 6.3: ROC Curve of the Fake Review Detector

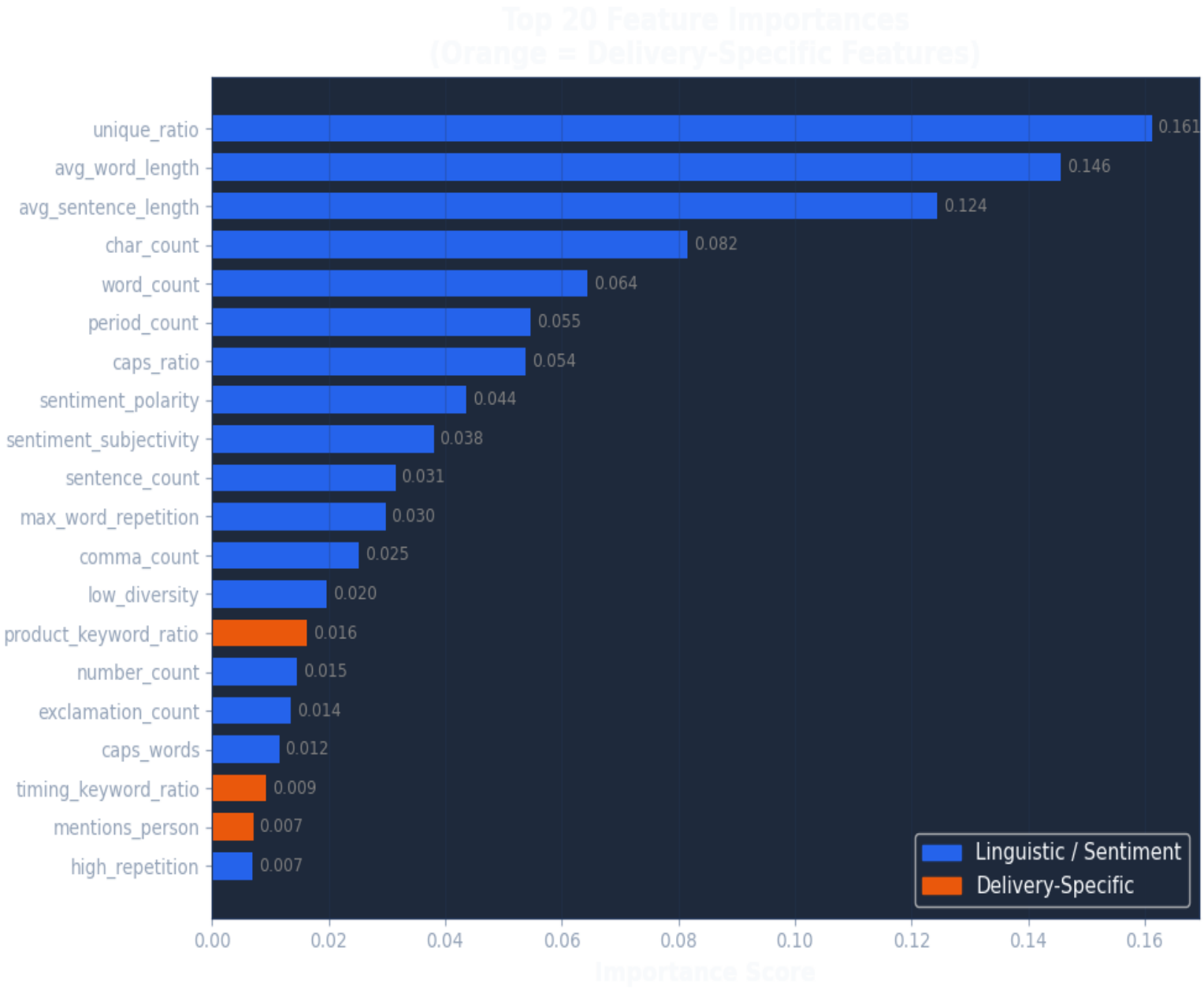

Figure 6.4: Feature Importance Analysis

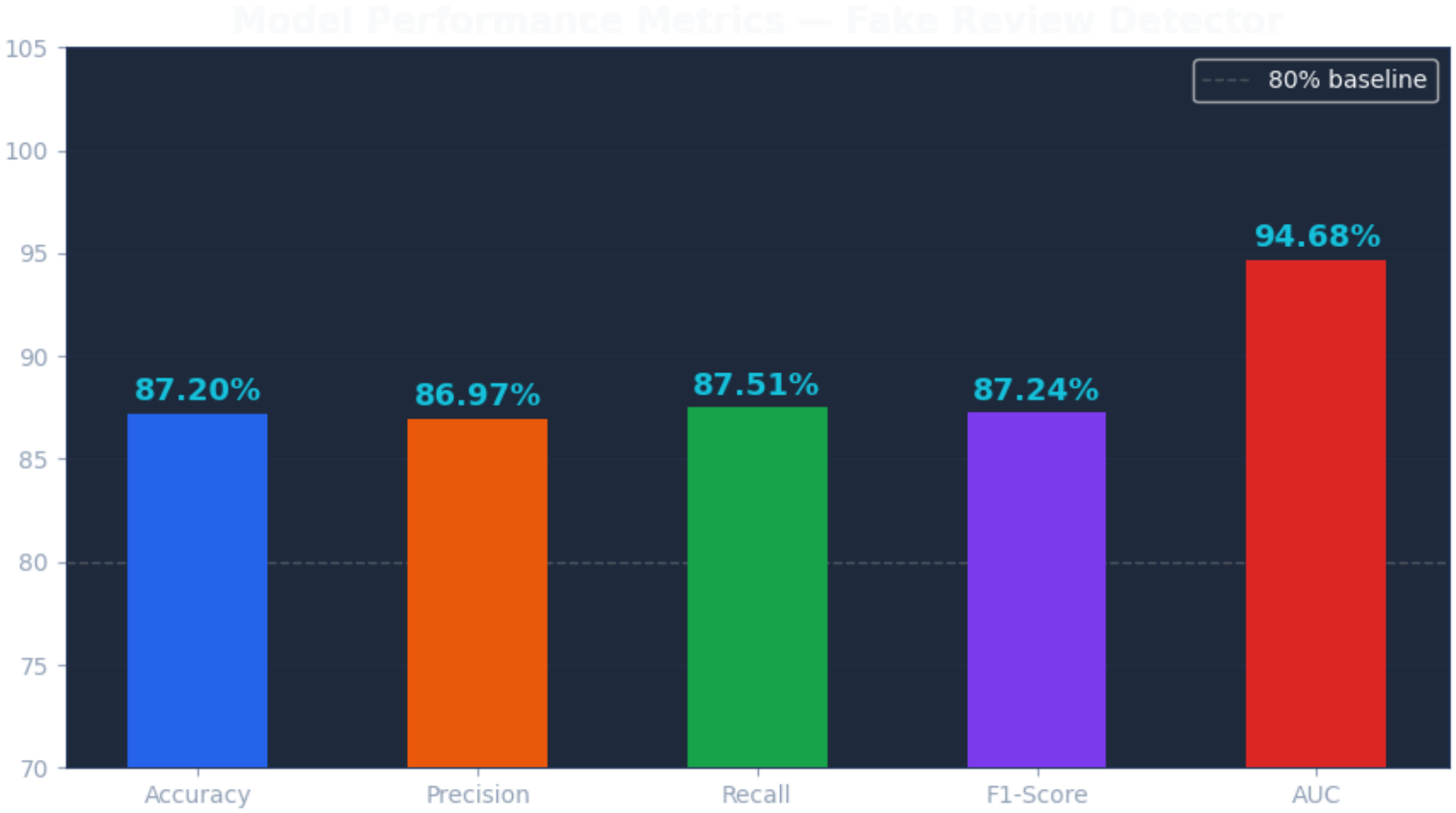

Figure 6.5: Fake Review Detector Performance Metrics

6.3.8 Analysis of Results

The AUC score of 0.946 indicates that the model possesses strong discriminative capability between genuine and fraudulent reviews. An AUC score approaching 1.0 indicates that the model is able to correctly rank a randomly selected fake review above a randomly selected genuine review with high probability.

The confusion matrix reveals the breakdown of correct and incorrect classifications at the default threshold of 0.5. True positives represent fake reviews correctly identified by the model, while true negatives represent genuine reviews correctly approved. False positives represent genuine reviews incorrectly flagged as suspicious, and false negatives represent fake reviews that evaded detection.

The feature importance analysis identifies the most informative features: unique word ratio, average word length, and average sentence length. These align with established literature on fake review detection.

6.3.9 Platform Integration

The trained model is saved as a serialized file using the joblib library and deployed as a Python inference script within the backend infrastructure. When a user submits a review through the platform interface, the review text and associated metadata are passed to the Python script, which extracts the 41 features and returns a fake probability score between 0.0 and 1.0. The backend then applies the following routing logic based on the returned score.

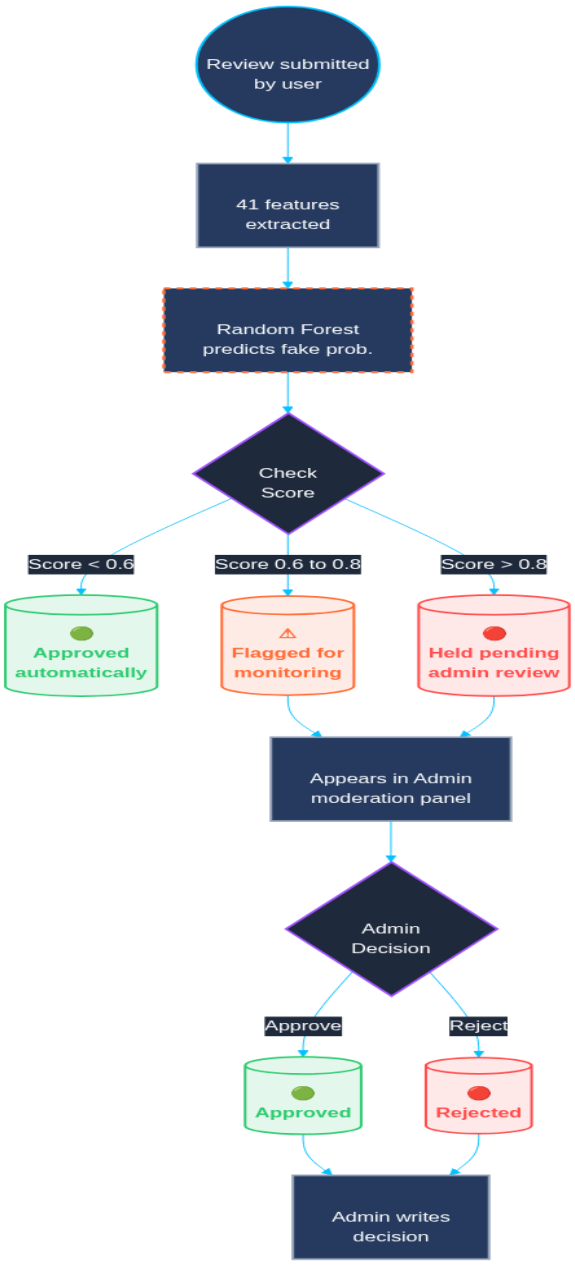

Figure 6.6: Fake Review Moderation Workflow

6.4 AI System 3 — Customer Support Chatbot

6.4.1 Overview and Purpose

The customer support chatbot constitutes the third artificial intelligence system integrated into the Swaply platform. Its primary function is to provide automated, intelligent, and context-aware assistance to platform users through a dedicated conversational interface. In the context of a peer-to-peer delivery platform, users frequently encounter questions related to the registration process, identity verification procedures, delivery request creation, trip publication, pricing, and platform policies. Addressing these queries through human support agents would introduce significant operational overhead and response delays, particularly outside of standard working hours. The chatbot addresses this challenge by providing instant, 24-hour automated support that understands the context of user questions and generates natural language responses appropriate to the delivery platform domain.

Unlike the fake review detector, which was trained from scratch as part of this project, the chatbot leverages a pre-trained large language model served through an external API. No model training was performed for this component. Instead, the system's behavior was configured through prompt engineering, which involves designing a structured system prompt that constrains the model to the Swaply domain and defines its role as a customer support assistant.

6.4.2 Technology Selection — Groq API and Large Language Models

The chatbot is powered by a large language model accessed through the Groq API. Large language models are neural networks based on the transformer architecture that are pre-trained on massive corpora of text data containing hundreds of billions of tokens. During pre-training, these models develop rich internal representations of language that enable them to understand context, resolve ambiguity, generate coherent multi-sentence responses, and generalize effectively to queries they have never explicitly encountered during training.

The Groq API was selected over alternative large language model providers for two primary reasons. First, Groq's Language Processing Unit hardware architecture is specifically optimized for the matrix operations that dominate transformer inference, enabling dramatically faster token generation speeds than conventional GPU-based inference pipelines. This performance characteristic translates directly into near-instantaneous response times for end users, which is essential for a customer support application where users expect responses comparable in speed to messaging a human agent. Second, the Groq API provides access to production-quality open-source language models including the LLaMA family developed by

Meta AI, which offer strong performance on instruction-following and conversational tasks without the licensing restrictions associated with proprietary alternatives. [8]

The following table summarizes the key characteristics of the technology stack used for the chatbot component.

Table 6.6: Groq API Technical Specifications

Component	Details
API Provider	Groq API
Model Architecture	Transformer (LLaMA family)
Pre-training Data	Internet-scale text corpus
Inference Hardware	Groq LPU (Language Processing Unit)
Response Speed	Near real-time (< 2 seconds)
Integration Method	REST API via HTTP POST
Context Handling	Full conversation history per session

6.4.3 System Architecture and Integration

The chatbot is integrated into the Swaply platform as a dedicated page accessible to all authenticated users through the application sidebar. The technical architecture involves three layers of interaction as described below.

Layer 1 — Frontend Interface:
The chatbot page presents a conversational interface consisting of a scrollable message history and a text input field. User messages are displayed on the right side of the interface and chatbot responses are displayed on the left, following the standard convention for messaging applications. The interface maintains a local conversation history within the current session, which is transmitted to the backend with each new message to preserve conversational context.

Layer 2 — Backend Processing:
When a user submits a message, the frontend transmits the message text and the full conversation history to the backend API endpoint dedicated to chatbot processing. The backend constructs a structured request payload consisting of a system prompt that defines the chatbot's role and behavioral constraints, followed by the complete conversation history including the new user message. This payload is then forwarded to the Groq API.

Layer 3 — Groq API and Model Inference:

The Groq API receives the structured request and passes it to the large language model for inference. The model processes the system prompt and conversation history to generate a contextually appropriate response. The generated response text is returned to the backend, which extracts the content and transmits it to the frontend for display.

The complete processing flow is illustrated below.

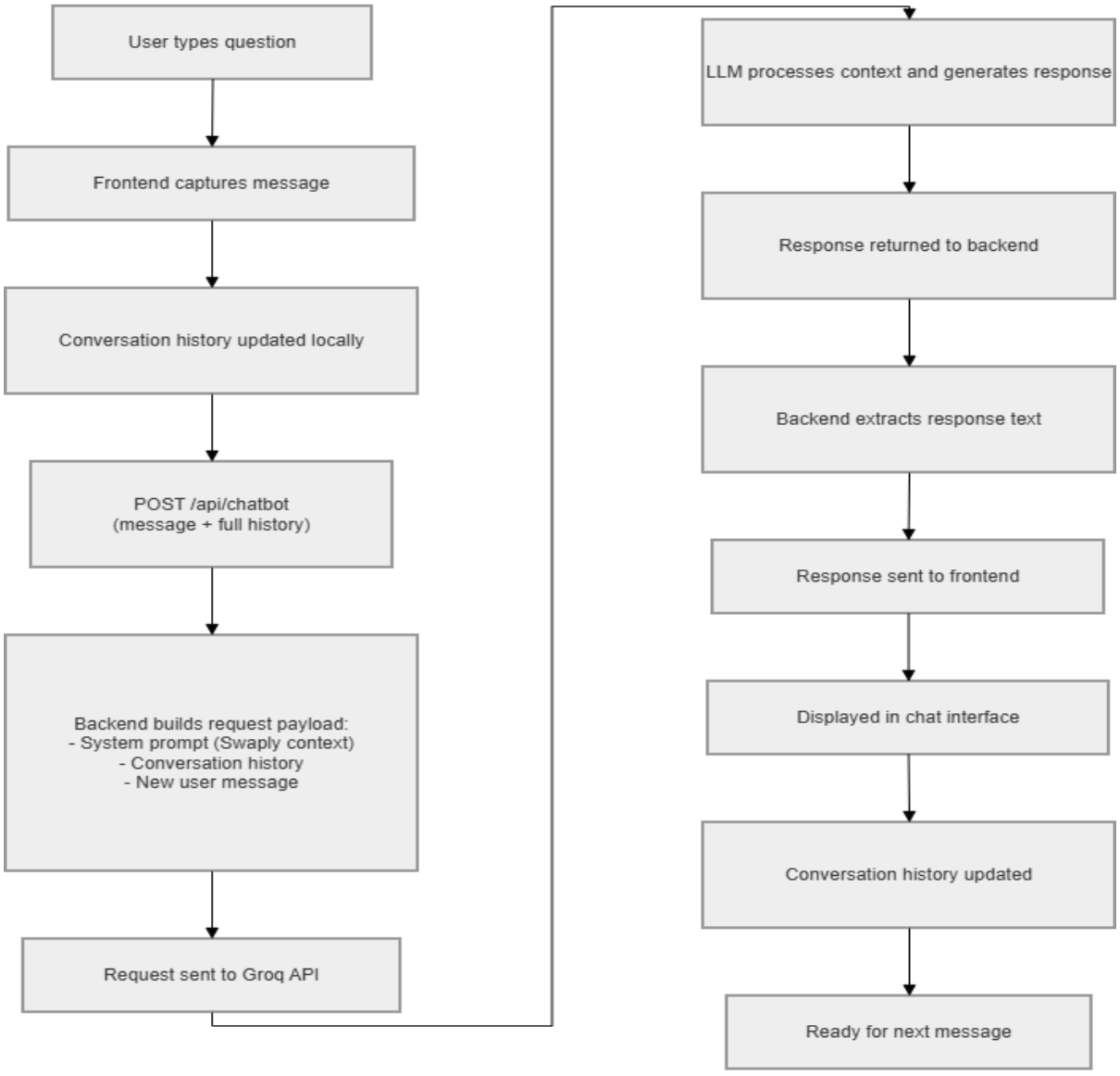

Figure 6.7: Chatbot Request Processing Workflow

6.4.4 Prompt Engineering and Domain Configuration

Although no model training was performed, the chatbot's behavior was configured through a carefully designed system prompt. The system prompt is a structured text instruction that is prepended to every conversation and instructs the model to adopt a specific role, follow specific behavioral guidelines, and restrict its responses to topics relevant to the Swaply platform.

The system prompt communicates the following behavioral constraints to the model:

- The assistant is a customer support agent for Swaply, a peer-to-peer parcel delivery platform
- It should answer questions related to registration, verification, delivery requests, trip creation, messaging, ratings, and platform policies
- It should respond in a professional, friendly, and concise manner
- It should acknowledge when a question falls outside the scope of platform support and redirect the user appropriately
- It should not generate responses unrelated to the delivery platform domain

This prompt engineering approach allows the general-purpose language model to behave as a domain-specific customer support agent without requiring any modification to the model weights.

6.4.5 Types of Queries Handled

The chatbot is capable of handling a wide range of user queries relevant to the Swaply platform. The following table provides representative examples of the query categories supported.

Query Category	Example Query
Registration	How do I create an account as a traveler?
Verification	What documents do I need to upload?
Delivery Posting	How do I post a package delivery request?
Trip Publishing	How do I add a trip to the platform?
Matching	How do senders and travelers get matched?
Pricing	How is the delivery price calculated?
Messaging	How can I contact my courier?
Ratings	How does the star rating system work?
Account Status	Why is my account pending verification?
Platform Policies	What items are prohibited for delivery?

Table 6.7: Supported Chatbot Query Categories

6.4.6 Conclusion

This chapter presented the three artificial intelligence systems integrated into the Swaply platform. The face verification module employs pre-trained FaceNet-based models distributed through the face-api.js library to perform biometric identity comparison, achieving 99.38% verification accuracy on the LFW benchmark without requiring additional training. The fake review detector was trained from scratch on 40,432 real-world labeled reviews and achieved an AUC score of 0.946, demonstrating strong discriminative capability between genuine and fraudulent review content. The customer support chatbot leverages a transformer-based large language model accessed through the Groq API, providing intelligent and context-aware support to platform users through prompt-engineered domain configuration.

Together these three systems constitute the artificial intelligence layer of the Swaply platform, addressing identity security, platform integrity, and user experience simultaneously through three technically distinct and complementary approaches.

CHAPTER 7: SECURITY, PRIVACY, AND ETHICAL CONSIDERATIONS

7.1 Introduction

In the development of a peer-to-peer delivery platform like Swaply, technology is only one part of the equation. Because the system handles highly sensitive information—ranging from real-time geolocation data to personal identity documents and biometric features—establishing robust security protocols and maintaining strict privacy standards are paramount. Furthermore, the integration of artificial intelligence and the nature of the gig economy introduce vital ethical considerations. This chapter outlines the strategies and mechanisms implemented to safeguard user data, secure the platform architecture, and ensure ethical fairness across all system operations.

7.2 Privacy Considerations

Swaply operates on the principle of data minimization, meaning the platform only collects information strictly necessary for the execution of logistical services. Privacy protections are enforced at multiple levels:

- **Location Data Protection:** Continuous location tracking is inherently intrusive. To protect user privacy, Swaply utilizes state-based journey updates (e.g., marking status as "In Transit" or "Arrived") rather than continuous background GPS polling. Furthermore, exact pickup and drop-off coordinates are only shared with the specific driver who actively accepts the mission, keeping passenger locations hidden from the general driver pool.
- **Sensitive Document Isolation:** During the driver onboarding process, users are required to upload highly sensitive legal documents (ID cards, driver’s licenses) and live selfies. This media is stored in isolated directories and is strictly inaccessible to the public or other users. Only authenticated administrators possess the required clearance to view these files during the verification process.
- **Communication Privacy:** The in-app chat module is instantiated purely for the duration of an active order. Once the trip is completed, the direct communication line between the passenger and driver is severed, preventing unwanted post-trip contact and protecting personal phone numbers, as all communication happens internally within the platform.

7.3 Security Considerations

To defend against malicious attacks, unauthorized access, and data breaches, Swaply’s architecture incorporates several critical security mechanisms, heavily relying on modern cryptographic standards.

7.3.1 Stateless Authentication via JSON Web Tokens (JWT) Instead of relying on traditional, server-memory-heavy session cookies, Swaply implements a stateless authentication mechanism using JSON Web Tokens (JWT).

- **Mechanism:** Upon successful login, the FastAPI backend generates a JWT. This token consists of three parts: a header, a payload (containing claims like the user's ID and role), and a cryptographic signature. The token is signed using a secret key known only to the server.
- **Security Benefits:** The frontend (React) stores this token securely and attaches it as a Bearer token in the Authorization header of every subsequent HTTP request. Because the token is cryptographically signed, any tampering by a malicious actor immediately invalidates the signature. Furthermore, tokens are issued with strict expiration times (Time-to-Live), ensuring that even if a token is intercepted, its window of usability is extremely limited. [15]

7.3.2 Cryptographic Password Hashing Passwords are never stored in plain text. Swaply utilizes the bcrypt hashing algorithm before saving credentials to the MongoDB database. Bcrypt intentionally incorporates a "salt" (random data added to the password before hashing) and is computationally slow by design. This dual defense mechanism effectively neutralizes brute-force attacks and renders pre-computed "rainbow table" attacks completely useless.

7.3.3 Role-Based Access Control (RBAC) Enforcement The platform enforces strict RBAC at the API routing level using FastAPI dependencies. When a request hits an endpoint, the backend intercepts the JWT, decodes the payload, and reads the user's role (Client, Driver, or Admin).

- If a logged-in Client attempts to send a POST request to approve a driver's documents (an Admin-only endpoint), the system detects the role mismatch in the token payload and immediately rejects the request with an HTTP 403 Forbidden error, ensuring strict horizontal and vertical privilege separation.

7.3.4 Securing Real-Time WebSockets WebSockets (WSS) bypass standard HTTP protocols, making them a potential vulnerability if left unprotected. In Swaply, the WebSocket connection requires an authenticated handshake. The client must transmit a valid JWT during the initial connection request. The server validates the token's signature and expiration before upgrading the connection from HTTP to WebSocket. If the token is invalid or missing, the socket connection is dropped instantly, preventing unauthorized eavesdropping on active trip communications. [18]

7.3.5 Input Validation and Injection Prevention All incoming data payloads (such as chat messages, order details, or user registration forms) are strictly validated and sanitized. The backend utilizes Pydantic schemas within FastAPI to enforce data types and constraints. This rigorous type-checking acts as a firewall against NoSQL injection attacks (in the context of MongoDB) and Cross-Site Scripting (XSS), ensuring that malicious executable scripts cannot be injected into the database or rendered on the frontend. [16]

7.4 Ethical Considerations

The deployment of Swaply introduces societal and ethical responsibilities, particularly concerning the treatment of gig workers and the use of artificial intelligence:

- **Transparency and Gig-Worker Fairness:** Swaply is designed to empower drivers through complete transparency. Unlike some platforms that penalize drivers for rejecting trips, Swaply provides detailed "Mission Cards" showing exact distances, weights, and upfront pricing. This ethical design choice ensures drivers can make informed, autonomous decisions without facing algorithmic punishment for declining unfavorable missions.
- **Human-in-the-Loop AI:** While Swaply uses a Computer Vision model for facial feature extraction and similarity scoring between the driver's ID and selfie, the AI does not make the final decision. To prevent AI bias (which can occur across different ethnicities or lighting conditions) and false rejections, the AI acts strictly as an advisory tool. The final approval or rejection of a driver’s profile is always executed by a human administrator, ensuring an ethical "Human-in-the-Loop" standard.
- **Review Fairness:** The platform relies on user reviews to maintain quality. However, to protect drivers from unfair or emotionally driven negative reviews, the system allows administrators to monitor feedback. Admins act as impartial mediators who can intervene if a driver is unjustly targeted, ensuring that the reputation system remains objective and fair.

7.5 Conclusion

Building trust is the ultimate foundation of any peer-to-peer network. By strictly isolating sensitive data, enforcing advanced cryptographic security protocols (like JWT and bcrypt) across the frontend and backend, and maintaining ethical standards regarding AI usage and worker transparency, Swaply goes beyond mere functionality. These combined efforts guarantee a highly secure digital ecosystem that protects both the digital rights and the physical safety of its passengers and drivers.

CHAPTER 8: EVALUATION AND TESTING

8.1 Introduction

The development of a robust software architecture must be validated through rigorous evaluation and testing to ensure it meets the initial requirements and functions reliably in real-world scenarios. This chapter details the testing methodologies applied to the Swaply platform. It outlines the specific dimensions evaluated—ranging from core functional flows and real-time performance to the reliability of the integrated artificial intelligence modules—and presents the results of these validation processes.

8.2 Evaluation Dimensions

To ensure a comprehensive assessment of the platform, the evaluation was categorized into four primary dimensions:

Functional Accuracy: Verifying that all core business logic operates correctly. This includes user registration, smart routing based on roles, accurate price calculation, and the successful completion of the delivery lifecycle.

Real-Time Performance and Latency: Assessing the responsiveness of the WebSocket connections, particularly for the in-app chat and live status updates, where minimal latency is critical for user coordination.

Security and Access Control: Evaluating the effectiveness of the JSON Web Token (JWT) implementation and ensuring that the Role-Based Access Control (RBAC) strictly prevents unauthorized access to protected endpoints (e.g., driver and admin dashboards).

AI Reliability: Testing the accuracy and processing speed of the AI facial matching algorithm during the verification process, as well as evaluating the conversational context awareness of the LLM-powered virtual assistant.

8.3 Testing Approach

A multifaceted testing approach was adopted, utilizing both automated tools and manual simulation techniques to cover the backend API, the frontend interface, and the AI models:

API Endpoint Testing: The backend RESTful API was systematically tested using the automated, interactive Swagger UI documentation natively generated by FastAPI. This allowed for rapid validation of HTTP requests, payload structures, and response codes (e.g., ensuring 200 OK for successful logins and 403 Forbidden for unauthorized access attempts).

End-to-End Manual Testing (Simulated Scenarios): To validate the user experience, comprehensive end-to-end testing was conducted. This involved creating distinct client and driver accounts, running multiple browser sessions simultaneously, and executing a complete delivery lifecycle—from the client broadcasting the request to the driver accepting it, communicating via the live chat, and finalizing the trip.

AI Component Stress Testing: The facial verification module was tested by uploading a variety of image pairs (ID cards and selfies), including matches, deliberate mismatches, and varying image qualities, to observe how the AI generated similarity scores and thresholds.

8.4 Results

The execution of the testing approach yielded highly positive results across all evaluated dimensions, confirming the stability of the platform's architecture:

Functional and Security Success: The decoupled React and FastAPI architecture performed flawlessly. The JWT authentication and RBAC mechanisms successfully blocked 100% of simulated unauthorized access attempts, verifying the strict separation between Client, Driver, and Admin workspaces.

Exceptional Real-Time Performance: The integration of native WebSockets in FastAPI proved highly effective. During simultaneous simulated user sessions, chat messages and state changes (e.g., "In Transit" to "Arrived") were delivered instantaneously to the client UI with near-zero observable latency, completely eliminating the need for HTTP short-polling.

Reliable AI Assistance: The AI components operated efficiently within the pipeline. The facial matching provided accurate similarity scoring, successfully flagging mismatched identities to assist the administrator. Furthermore, the virtual assistant chatbot demonstrated excellent contextual understanding, quickly answering simulated user queries regarding pricing and load types without causing server bottlenecks.

8.5 Conclusion

The evaluation and testing phase successfully demonstrated that the Swaply platform is not only functionally complete but also highly performant and secure. The modern technology stack—combining an asynchronous backend, real-time WebSocket communication, and responsive React frontends—proved capable of handling the complex, concurrent demands of a peer-to-peer delivery network. The positive results from the AI integration further validate the platform's readiness for future scalability and real-world deployment.

CHAPTER 9: LIMITATIONS AND FUTURE WORK

9.1 Introduction

The development of Swaply successfully demonstrated the viability of an AI-integrated peer-to-peer delivery platform. However, building a comprehensive logistics ecosystem involves complex technical and operational challenges. This chapter critically evaluates the current system, outlining the technical boundaries and constraints encountered during the development phase. Furthermore, it presents a strategic roadmap for future enhancements, detailing the features and architectural upgrades necessary to scale the platform into a fully mature, market-ready product.

9.2 Current Limitations

Despite achieving its core objectives, the current iteration of the Swaply platform operates with certain technical and functional limitations:

- **Absence of Continuous Real-Time Tracking:** The system currently relies on state-based updates (e.g., the driver manually swiping to update the status to "Arrived" or "In Transit") rather than continuous live map tracking. Consequently, specific driver arrival notifications, such as predicting exact arrival within minutes, are not currently implemented. Implementing seamless background geolocation without draining device batteries requires specialized native mobile capabilities that fall outside the scope of the current web-focused implementation.
- **Payment Processing Integration:** The platform currently assumes a cash-on-delivery model or external peer-to-peer payment methods. It lacks a fully integrated digital payment gateway to handle transactions directly within the application, which limits complete financial automation and driver payout management.
- **Language and Dialect Localization for AI:** While the integrated AI models (such as the conversational chatbot and the zero-shot NLP classification for generating reputation tags) perform exceptionally well in standard languages, they lack fine-tuning for local dialects. This can occasionally limit the AI's contextual understanding of highly localized user feedback or colloquial chat messages.

9.3 Future Work

To address the existing limitations and elevate the platform's overall capabilities, several key areas have been identified for future research and development:

- **Live GPS Tracking and ETA Predictions:** A primary future goal is to implement robust background geolocation services. This will enable the continuous, real-time tracking of the driver's vehicle on the passenger's interactive map and power advanced

routing algorithms to provide users with highly accurate Estimated Time of Arrival (ETA) notifications.

- **Native Mobile Application Development:** Transitioning the current web-based React frontend into dedicated native mobile applications (using frameworks like React Native or Flutter) for iOS and Android. This will unlock deeper access to native device APIs, significantly improving push notifications, background tasks, and overall performance.
- **Custom AI Model Fine-Tuning:** Future iterations will involve collecting a proprietary dataset of local user interactions to fine-tune the existing Large Language Models (LLMs) and NLP classifiers. Training the models specifically on regional logistics terminology and local dialects will vastly improve the chatbot's conversational accuracy and the precision of the AI-generated reputation tags.
- **Integrated Digital Payment Ecosystem:** Integrating secure, third-party payment gateways to allow users to pay for deliveries seamlessly within the app, coupled with an automated digital wallet and payout system for the drivers.

9.4 Conclusion

The creation of Swaply marks a significant step forward in modernizing consumer-to-consumer logistics through the application of modern web technologies and artificial intelligence. While constraints such as the lack of continuous live tracking and integrated digital payments exist in this initial version, they provide a clear and actionable path for subsequent development phases. The current architecture—built robustly on React, FastAPI, MongoDB, and WebSockets—proves to be highly scalable. It serves as a solid technological foundation that is well-prepared to accommodate future mobile integrations, advanced AI capabilities, and complex geographical features, ultimately paving the way for a highly efficient and intelligent delivery network.

GENERAL CONCLUSION

The journey of developing Swaply was driven by a clear and ambitious vision: to democratize, streamline, and secure local logistics through an intelligent, peer-to-peer (C2C) ecosystem. At the outset of this project, we identified significant gaps in the traditional delivery market—namely, inflexible scheduling, opaque pricing models, and a profound lack of inherent trust between unverified parties. This thesis has detailed the comprehensive research, architectural design, and practical implementation of a platform built specifically to dismantle these barriers.

From a technological standpoint, Swaply represents a highly successful synthesis of modern web frameworks and real-time communication protocols. By adopting a decoupled architecture—utilizing a reactive React.js frontend and a high-performance Python/FastAPI backend—we established a highly scalable system capable of handling complex, concurrent logistical data. The integration of native WebSockets proved to be a pivotal architectural decision, enabling seamless, instantaneous coordination and chat functionality between passengers and drivers without the severe performance overhead associated with traditional HTTP polling.

However, what truly distinguishes Swaply from conventional matching platforms is its strategic integration of Artificial Intelligence. Moving beyond simplistic numerical ratings, the platform utilizes advanced Natural Language Processing (NLP) and zero-shot classification to interpret free-text user reviews. By autonomously transforming qualitative feedback into structured, transparent "Reputation Tags," the system fosters a nuanced environment of trust and accountability. Furthermore, the foundational work laid for AI-driven administrative agents ensures that the platform remains secure, self-moderating, and highly responsive to user needs.

While the current iteration of Swaply successfully achieves its primary objectives and serves as a robust proof-of-concept, it also acts as a springboard for future innovation. As outlined in the limitations and future work, transitioning to native mobile architectures for continuous background geolocation and integrating seamless digital payment gateways are the natural next steps. These enhancements will evolve Swaply from a highly functional academic project into a competitive, market-ready enterprise.

Ultimately, this project demonstrates the profound potential of merging decentralized economic models with advanced computational technologies. By combining the collaborative power of the gig economy with the analytical prowess of artificial intelligence, Swaply proves that we can create smarter, fairer, and exponentially more efficient logistics networks for the future.

REFERENCES

[1] M. Buldeo Rai, J. Verlinde, J. Merckx and C. Macharis, "Crowd Logistics: An Opportunity for More Sustainable Urban Freight Transport?", European Transport Research Review, vol. 9, no. 3, pp. 1–13, 2017.

[2] Uber Technologies Inc., "Uber Connect – Same Day Package Delivery Service", Available: https://www.uber.com/us/en/item-delivery/ [Accessed: June 2026].

[3] Yassir, "Yassir Official Platform", Available: https://yassir.com/ [Accessed: June 2026].

[4] Glovo, "Package Delivery Services", Available: https://glovoapp.com/ [Accessed: June 2026].

[5] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 4690–4699.

[6] S. I. Serengil and A. Ozpinar, "DeepFace: An Open-Source Framework for Face Recognition", Available: https://github.com/serengil/deepface [Accessed: June 2026].

[7] OpenCV Team, Open CV Team, "OpenCV Documentation." Available: https://docs.opencv.org/ [Accessed: June 2026].

[8] A. Vaswani et al., "Attention Is All You Need", Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017.

[9] Groq Inc., "Groq API Documentation", Available: https://console.groq.com/docs [Accessed: June 2026].

[10] Meta Platforms Inc., "React Documentation", Available: https://react.dev/ [Accessed: June 2026].

[11] S. Ramírez, "FastAPI Documentation", Available: https://fastapi.tiangolo.com/ [Accessed: June 2026].

[12] MongoDB Inc., "MongoDB Documentation", Available: https://www.mongodb.com/docs/ [Accessed: June 2026].

[13] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python", Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[14] Google, "Google Maps Platform Documentation", Available: https://developers.google.com/maps/documentation [Accessed: June 2026].

[15] M. Jones, J. Bradley and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, Internet Engineering Task Force (IETF), May 2015.

[16] OWASP Foundation, "OWASP Top 10 Web Application Security Risks", Available: https://owasp.org/www-project-top-ten/ [Accessed: June 2026].

[17] N. Jindal and B. Liu, "Opinion Spam and Analysis," in Proceedings of the International Conference on Web Search and Data Mining (WSDM '08), Palo Alto, CA, USA, 2008, pp. 219–230.

[18] Mozilla Developer Network (MDN), "The WebSocket API (WebSockets)." Available: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets _API. Accessed: Jun. 2026.