BLITZ-Kaptaan

A PROJECT REPORT

Submitted in partial fulfilment of the

requirement for the award of the degree

of

MASTER OF COMPUTER APPLICATIONS

(MCA)

SANKET CHOUDHARY

23FS20MCA00029

COMPUTER APPLICATION

MANIPAL UNIVERSITY JAIPUR

JAIPUR-303007

RAJASTHAN, INDIA

MAY2025

DEPARTMENT OF COMPUTER APPLICATION

MANIPAL UNIVERSITY JAIPUR, JAIPUR – 303007 (RAJASTHAN), INDIA

Date:15/05/2025

CERTIFICATE

This is to certify that the project titled '**BLITZ-Kaptaan**' is a record of the Bonafide work completed during the period from 20-Jan-2025 to 20-Jul-2025 by **Sanket Choudhary** (**23FS20MCA00029**) submitted in the partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications (MCA) at the Department of Computer Applications, Manipal University Jaipur, for the academic year 2023-25.

Dr. Avichandra Singh
Project Guide,
Dept of Computer Applications
Manipal University Jaipur

Dr. Shilpa Sharma
Head of the Department,
Dept of Computer Applications
Manipal University Jaipur

Date:10/05/2025

CERTIFICATE

This is to certify that the project entitled **Blitz-Kaptaan** was carried out by **SANKET CHOUDHARY 23FS20MCA00029** at **BIGSHORT TAILS PRIVATE LIMITED**, **BANGALORE** under my guidance **during JANUARY 2025** to **JULY 2025**.

BIGSHORT TAILS PVT. LTD.

Nikhil Kumar Gupta
Associate Product Manager - II,
Bigshort Tails Private Limited, Bangalore

VERIFICATION OF INTERNSHIP LETTER

Date – 16th April 2025

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Mr. Sanket Choudhary** is associated with Blitz ©Bigshort Tails Private Limited as **Product Support - Intern** on an **internship** basis with Employee ID I44. He has been undergoing his internship with the company from **20th January, 2025 to 20th July, 2025.**

Should you require any further information or have specific questions regarding his employment, Please do not hesitate to reach out to us.

For BIGSHORT TAILS PVT. LTD.

DIRECTOR

____

Mayank Varshney

Cofounder & CEO

BIGSHORT TAILS PRIVATE LIMITED

315 Work Avenue, 257, 1st Floor, 16th Cross, 5th Main Road, Sector - 6, HSR Layout, Bangalore, 560102 |

CIN U72900HR2020PTC091697

E-mail: swaja@blitznow.in | Website: www.blitznow.in

ACKNOWLEDGEMENT

I extend heartfelt appreciation to the individuals and organizations who played pivotal roles in the successful completion of the '**Blitz-Kaptaan**' project and like to express my sincere gratitude the Dean/Director of Manipal University Jaipur for their continuous encouragement and support throughout the project. Dr. Shilpa Sharma Head of Computer Application for providing the necessary resources and facilities for the project. My project supervisor **Dr. Avichandra Singh**, for her expert guidance, constructive feedback, and mentorship, which greatly contributed to the success of the project. I am profoundly grateful to everyone mentioned above for their unwavering support, guidance, and encouragement throughout this project. Their collective efforts have been instrumental in shaping its success, and we are honored to have had the opportunity to work alongside them.

Date: 15.05.2025
Signature: Sanket Choudhary

DECLARATION

I, **Sanket Choudhary,** hereby declare that the report titled **‘Blitz-Kaptaan’** is a record of my original work carried out under the guidance of **Dr. Avichandra Singh.** The content presented in this report is based on my independent efforts, analysis, and findings, and has not been submitted for any other degree, diploma, or academic credit at any institution.

All sources of information used in the preparation of this report have been duly acknowledged and referenced. I affirm that this work complies with ethical research and reporting standards.

Date: 15.05.2025
Signature: Sanket Choudhary

ABSTRACT

In the present digital and mobile-first economy, last-mile logistics has become a critical enabler of consumer-facing services. Industries such as e-commerce, hyperlocal delivery, food aggregators, and courier services heavily rely on frontline delivery partners to fulfill customer expectations. However, one of the most persistent operational challenges is the misuse of delivery status updates by field agents. Frequently, remarks such as “Customer not available,” “Address not reachable,” or “Refused to accept” are recorded without an actual attempt to deliver. These false reports lead to increased return-to-origin (RTO) costs, deteriorated customer satisfaction, compromised service quality, and serious concerns regarding trust and reliability. This project aims to address these inefficiencies through an intelligent, event-driven, AI-supported mobile and backend system that enhances delivery traceability, detects anomalies in rider behavior, and strengthens operational accountability. The objective is to prevent fraudulent delivery remarks while providing operational managers with real-time visibility and control over delivery activities.

The system architecture revolves around two Flutter-based mobile applications—the Kaptaan App for riders and the Franchise Owner App for operational users. These applications interact with a cloud-based backend system that incorporates real-time data collection, verification logic, and event monitoring. Google Tag Manager (GTM) is integrated into the mobile interface to capture behavioral data such as screen navigation, call duration, and delivery remark submission. These events are pushed into the backend and evaluated by a rules-based decision engine known as Kaptaan AI. In parallel, the system incorporates an automated Know Your Customer (KYC) verification workflow powered by HyperVerge, enabling verified onboarding and the ability to blacklist non-compliant users. A separate fatigue monitoring module was implemented using Python, OpenCV, and Dlib to detect eye-blinking patterns and raise alerts in case of drowsiness, ensuring delivery safety in extended shifts.

Upon testing, the system successfully detected and blocked suspicious delivery behaviors such as rapid remark entry, no OTP verification, and location mismatches. The Kaptaan AI engine flagged these scenarios accurately in over 90% of test cases. The KYC module streamlined rider onboarding and deactivation, while GTM and Firebase ensured seamless event logging. The fatigue detection system triggered alerts during drowsiness simulation, and API workflows performed reliably under varied load conditions. Metabase dashboards offered clear visualizations of operational data such as rider performance, remark frequency, and system flags—allowing supervisors to take swift data-driven actions.

Technologies used in this project include Flutter (for cross-platform mobile development), Firebase (for real-time data handling), Google Tag Manager (for behavioral event tracking), HyperVerge APIs (for automated KYC verification), OpenCV and Dlib (for computer vision-based fatigue monitoring), and Metabase (for dashboard analytics). Backend integration and system automation were implemented through RESTful APIs tested via Postman and curl. The result is a robust, intelligent, and modular framework that can be extended to any last-mile logistics ecosystem aiming to reduce fraudulent delivery actions, ensure rider compliance, and promote customer satisfaction through traceable and secure operations.

LIST OF TABLES

LIST OF FIGURES

Contents

Chapter 1: INTRODUCTION

1.1 Introduction to Work Done / Motivation (Overview, Applications & Advantages)

In the modern era of rapid digitalization and increasing reliance on e-commerce, the efficiency and reliability of last-mile delivery operations have become pivotal to customer satisfaction and operational success. Despite technological advancements in logistics, a persistent and costly challenge faced by companies is the misuse or manipulation of delivery status updates—particularly the generation of false delivery failure remarks. These include reasons such as "Customer not answering," "Address incorrect," "Refused to accept," or other justifications that may be entered without actual delivery attempts being made. Such inaccuracies not only lead to operational losses and customer dissatisfaction but also affect brand credibility and logistical metrics.

To address these challenges, the current system presents an end-to-end technological solution that introduces intelligent automation and behavioral tracking into the delivery verification process. This project is centered around the idea of improving delivery transparency and accountability through a multi-pronged technological framework. The core solution involves:

- Real-time event tracking using Google Tag Manager (GTM)
- Secure and efficient API workflows tested and validated using Postman
- Automated Know Your Customer (KYC) verification through HyperVerge
- Visual behavior detection using Eye Blinking Detection with Python and OpenCV
- Data visualization and operational analytics through Metabase dashboards

The system is specifically designed to detect inconsistencies in delivery behavior by capturing rider activities and cross-referencing them with backend data logs and pre-defined rules. For instance, when a rider marks a delivery as "Customer not available," the system analyzes supporting events like GPS location, time spent near the delivery location, blink detection data, and app interaction logs to determine the likelihood of an actual attempt.

Additionally, the system introduces automated logic to handle operational tasks, such as flagging delivery partners with repeated suspicious behavior, blacklisting based on verified non-compliance, and initiating order cancellations through API triggers. This not only saves time for human operators but also ensures a fair and consistent approach to delivery verification and rider evaluation.

Applications

The versatility of this system allows it to be deployed across a variety of industries and use cases where last-mile logistics and delivery tracking are crucial. Major applications include:

1. **E-commerce Logistics Platforms:**
   Ideal for companies like Amazon, Flipkart, and Meesho that deal with high volumes of deliveries and require precise delivery tracking.
2. **Food and Grocery Delivery Systems:**
   Platforms such as Zomato, Swiggy, and BigBasket can utilize this system to verify delivery attempts and ensure food is not wasted due to invalid cancellation reasons.

3. **Courier and Postal Services:**
   Traditional courier companies and postal services can modernize their verification processes using this intelligent system.
4. **Logistics Startups and Fleet Management Firms:**
   New-age delivery and fleet management companies benefit from automated monitoring, partner evaluation, and safety features built into the platform.

Key Advantages

The proposed solution brings multiple benefits that align with both operational efficiency and end-user satisfaction. Some of the most important advantages are:

- **Significant Reduction in False Delivery Failure Reports:**
  By tracking actual user activity and system events, the rate of fake delivery remarks can be minimized, leading to higher successful delivery rates.
- **Improved Accountability of Delivery Personnel:**
  With real-time event logging and AI-based verification, riders are more accountable, ensuring a higher level of integrity and performance.
- **Enhanced Rider Safety Through Fatigue Detection:**
  Eye Blink Detection monitors driver fatigue, helping prevent accidents and reducing risks during long delivery hours.
- **Streamlined Onboarding and Verification via KYC:**
  HyperVerge KYC automation speeds up the partner onboarding process, ensuring regulatory compliance and a seamless user experience.
- **Centralized Operational Visibility Through Dashboards:**
  Metabase dashboards provide real-time insights for administrators to track KPIs, suspicious activity, and partner performance.

This comprehensive and intelligent system serves as a practical solution to a real-world logistics problem, offering immediate operational benefits while laying the foundation for more advanced features in the future. It represents a significant step forward in the domain of automated logistics management and delivery verification.

1.2 Project Statement / Objectives of the Project

Project Statement

In today's fast-paced digital economy, delivery platforms serve as the bridge between service providers and customers. However, one of the most persistent challenges in last-mile logistics is ensuring delivery authenticity and customer satisfaction. A common malpractice is the misreporting of delivery attempts by field agents, who often record false delivery remarks such as "Customer not available," "Address not reachable," or "Refused to accept," without genuinely attempting delivery. These inaccuracies not only disrupt the fulfillment process but also lead to increased Return-to-Origin (RTO) costs, erosion of customer trust, and reduced operational efficiency.

To address this issue, the proposed project aims to develop an AI-driven, event-based delivery management and verification system that ensures transparency, accountability, and real-time oversight of delivery operations. The system is composed of two primary mobile applications—the Kaptaan App (for delivery partners) and the Franchise Owner App (for operational supervisors)—which are synchronized through a centralized backend powered by real-time analytics, API integrations, and behavior validation mechanisms.

The key component of this system is "Kaptaan AI," an intelligent decision-making engine that evaluates field data, GTM (Google Tag Manager) events, call histories, and delivery timelines to detect fraudulent activities, trigger automatic order cancellations, and take corrective actions against violating riders. Moreover, integration with HyperVerge's e-KYC (Know Your Customer) service ensures that only verified personnel participate in delivery processes, with backend protocols to delist or reinstate users based on performance and compliance.

A significant safety enhancement is the implementation of a drowsiness detection module using computer vision (OpenCV and Dlib), which monitors eye-blinking patterns to detect fatigue in riders. In tandem, a robust backend system supports real-time data flow, bulk rider attendance, API-triggered actions, and dashboard visualizations powered by Metabase to aid in proactive operational decision-making.

This project, therefore, provides a holistic solution that optimizes delivery verification, enhances rider accountability, and supports smart, automated operations across the delivery network.

Objectives of the Project

1. To develop and integrate "Kaptaan AI," a behavior-driven AI engine that processes rider actions, delivery timelines, and GTM-based events to detect fake remarks and automatically trigger order rescheduling or cancellations.
2. To build Flutter-based mobile applications:
    - Kaptaan App for delivery agents to manage orders, verify delivery tasks, and update remarks.
    - Franchise Owner App for operational managers to monitor agents, verify KYC status, and track order-level activity.
3. To implement fake delivery detection mechanisms using GTM for real-time event capture and Firebase for cloud-based data synchronization and storage.

4. To integrate a computer vision module using OpenCV and Dlib for real-time drowsiness detection, enhancing safety by monitoring rider alertness during delivery activities.
5. To automate backend workflows using RESTful APIs that enable:
    - Submission of order cancellations.
    - Bulk attendance marking.
    - Rider blacklisting and delisting.
    - KYC verification and updates using HyperVerge services.
6. To ensure seamless onboarding by integrating HyperVerge KYC services for automatic document verification, compliance validation, and status-based rider onboarding.
7. To create operational dashboards using Metabase, offering data-driven insights into delivery performance, rider behavior, fake remark trends, and operational compliance.
8. To use Postman and curl for automated API testing, validating workflows related to delivery updates, KYC checks, attendance logs, and event triggers.
9. To design a user-friendly interface in both apps, ensuring intuitive navigation and backend logic integration, such as live delivery timelines, remark entry modules, and Firebase-based synchronization.
10. To establish an event-driven delivery monitoring system with real-time location tracking, delivery status updates, and proactive action suggestions based on system-generated alerts.
11. To optimize system performance and backend infrastructure for high concurrency, ensuring real-time responsiveness during high-load operations like bulk updates or multi-agent coordination.
12. To implement automated reporting tools that generate real-time alerts, error logs, and delivery behavior reports to facilitate rapid issue resolution.
13. To integrate third-party tools such as Google Maps for rider geolocation, Firebase Cloud Messaging (FCM) for push notifications, and secure APIs for enhanced system functionality.
14. To build a comprehensive KYC management module with both manual and automated update options, ensuring all delivery agents remain compliant with platform policies.
15. To enhance system throughput and minimize latency in all application interactions, ensuring that data exchanges between riders, franchise owners, and backend systems are near-instantaneous.
16. To incorporate a learning feedback loop that refines Kaptaan AI based on past delivery behavior, enabling predictive actions and reduction of operational loopholes.
17. To ensure the UX design supports all user roles (riders, franchise owners, admins), allowing efficient use of system features such as order management, status tracking, and compliance enforcement.
18. To uphold data security and user privacy by applying modern encryption techniques and secure communication protocols, maintaining compliance with global data protection laws such as GDPR.

1.3 Organization of Report

This report has been carefully structured to guide the reader through every critical aspect of the project, ensuring a systematic understanding from the inception of the idea to its final realization. Each chapter has been designed to encapsulate distinct yet interrelated stages of the project development lifecycle, presenting both technical depth and practical relevance. The organization of the report is as follows:

Chapter 1: Introduction

This chapter sets the stage for the entire project by outlining its significance, motivation, and real-world applicability. It introduces the problem statement and highlights the key challenges that the system aims to solve, such as the prevalence of fake remarks, inefficiencies in KYC processing, and lack of effective event tracking in applications. The objectives of the project are clearly defined, providing a roadmap of the goals to be achieved. Additionally, this chapter offers an overview of the overall structure of the report, helping readers navigate through the upcoming content.

Chapter 2: Background Material

This section delves into the theoretical and technological foundations that underpin the project. It elaborates on core concepts such as event tracking mechanisms, drowsiness detection in user behavior, Know Your Customer (KYC) processes, and anti-fraud measures. The chapter also introduces the major technologies and platforms used throughout the development, including Flutter for cross-platform app development, Python for backend logic and AI/ML support, Google Tag Manager (GTM) for event management, Firebase for real-time data synchronization and authentication, HyperVerge for AI-based identity verification, and Metabase for data visualization. This chapter ensures that readers are familiar with the essential tools and methodologies that are critical to the project's execution.

Chapter 3: Methodology

Here, the report presents the detailed approach adopted to develop and integrate the system's various components. The methodology encompasses the strategies used for detecting fake feedback or remarks, the process of integrating multiple APIs for data collection and KYC verification, and the logic behind user behavior tracking. Visual representations such as system block diagrams, process flowcharts, and logical structures are included to enhance clarity. Each methodological decision is supported by rationale and aligned with the project’s objectives, ensuring both technical soundness and practical relevance.

Chapter 4: Implementation

This chapter focuses on the practical execution of the planned system. It provides an in-depth explanation of the implementation process for each module, including the mobile applications for users and administrators, backend APIs for system communication, and the analytical dashboard developed using Metabase. The integration of KYC services via HyperVerge is discussed in detail, along with how user actions are tracked and stored through GTM and Firebase. Screenshots of the user interfaces, selected code snippets, API structures, and integration flows are presented to offer a transparent view of the development process. This chapter bridges the gap between design and working prototype.

Chapter 5: Results and Analysis

In this section, the focus shifts to the evaluation of the system’s performance. It includes a comprehensive analysis of data collected during the testing phase, outlining metrics such as accuracy of detection, speed of KYC processing, and efficiency of event tracking. Dashboard screenshots, performance graphs, and comparative data tables are used to illustrate the outcomes. The results are critically analyzed to identify strengths, limitations, and unexpected findings. This empirical evaluation provides concrete evidence of the system's effectiveness and readiness for real-world deployment.

Chapter 6: Conclusions and Future Scope

The final chapter summarizes the key accomplishments of the project, highlighting how the system addresses the initial problem statement and meets its defined objectives. It reflects on the technical and practical lessons learned during development and testing. Furthermore, this chapter outlines potential areas for enhancement, such as scaling the system to support larger user bases, introducing machine learning models for smarter fraud detection, and expanding integration with additional third-party services. Suggestions for future research and development are also provided, offering a path for continuous improvement and innovation.

Chapter 2: BACKGROUND MATERIAL

2.1 Conceptual Overview (Concepts / Theory Used)

The project is centered on building a robust, intelligent, and traceable system that helps logistics operations minimize inefficiencies caused by fake delivery remarks and poor agent practices. It blends rule-based tracking, data visualization, machine learning, and mobile-first architecture to address the real-world issue of unreliable delivery reporting.

Key concepts involved include:

- **Event-based Tracking and Validation**
  The system tracks user and agent interactions through tagged events configured in Google Tag Manager (GTM). These events are processed and evaluated in real time to validate the authenticity of delivery failures, such as call attempts, location mismatches, and OTP verifications.
- **KYC Verification and Agent Identity Assurance**
  To ensure reliable onboarding and activity mapping, KYC (Know Your Customer) procedures are embedded in both Kaptaan and Franchise Owner applications. This allows for formal identification of delivery partners and control over their operational permissions.
- **Timeline-based Task Structure**
  Replacing conventional static tour-based delivery models, the system introduces a dynamic timeline-based task view. This structure enhances visibility, accountability, and speed of issue resolution during last-mile delivery.
- **Fatigue Detection using Eye Blinking Monitoring**
  Safety-critical modules such as drowsiness detection rely on facial landmark analysis and blink frequency monitoring. The project implements this using Eye Aspect Ratio (EAR) to estimate alertness in real time.
- **API-driven Control and Automation**
  Backend operations such as blacklisting, unlisting, cancellation of orders, and attendance tracking are handled via secured APIs. These APIs are accessible via HTTP requests and were extensively tested using Postman and automated with curl.
- **Analytics and Review**
  A centralized dashboard enables stakeholders to view reports, delivery metrics, and operational logs. It also supports the review of flagged remarks and agent performance.

2.2 Technologies Involved

This project utilizes a comprehensive tech stack covering backend processing, mobile development, AI-driven safety modules, real-time analytics, and KYC verification.

- **Metabase**
  An open-source business intelligence tool used to build operational dashboards. It provides visual insights into delivery trends, remark frequency, blacklist actions, and agent status across the system.
- **Firebase (Authentication and Hosting)**
  Used for managing secure login and user session flows within the mobile applications. Firebase also supports lightweight hosting and configuration services.
- **Google Tag Manager (GTM)**
  Facilitates event tracking by capturing user interactions such as call durations, OTP failures, and remark entries without modifying the mobile application codebase. It also acts as a trigger layer for flagging potential fake delivery remarks.
- **Flutter (Dart)**
  Both the Kaptaan and Franchise Owner applications are developed using Flutter for seamless cross-platform deployment. These applications include modules for delivery task timelines, real-time event updates, cancellation flows, and KYC submission.
- **Python (OpenCV, Dlib, NumPy, SciPy)**
  Used in the Eye Blinking Detection module to monitor rider fatigue levels. The implementation leverages real-time video input and facial landmark tracking to calculate the Eye Aspect Ratio (EAR), enabling detection of prolonged drowsiness.
- **HyperVerge (AI-Powered KYC System)**
  HyperVerge is used for secure, real-time KYC verification. It enables identity document scanning, face match, and liveness detection. The integration ensures that only verified delivery partners can be onboarded, and it also supports automated workflows for blacklisting or unblocking riders based on KYC status.
- **Postman and cURL**
  REST APIs used in the system were developed and tested using Postman. Key functionalities include:

    - Order cancellation logging
    - KYC status updates
    - Rider attendance updates
    - Blacklisting and delisting riders These APIs were also automated using cURL for backend testing and scripting.

- **Git and GitHub**
  All source code and documentation are maintained in GitHub repositories to ensure code integrity, version control, and collaborative updates.
- **Notion and Release Notes (PDF)**
  Project documentation, GTM configurations, deployment logs, and release tracking (such as the Release Notes dated 16/04/2025) are maintained through Notion and attached as formal PDF references.

Chapter 3: METHODOLOGY

3.1 Overall System Architecture and Methodology

The methodology adopted in this project follows a modular, event-driven architecture that integrates mobile application interfaces, backend services, API-based automation, AI-powered logic, and real-time analytics. The system is designed to address critical operational challenges in last-mile delivery through a combination of frontend mobile apps (Kaptaan and Franchise Owner), backend services for data processing and validation, AI modules for intelligent decision-making, and dashboards for monitoring delivery behavior.

The entire solution is structured around the following core modules:

1. Event Tracking and Fake Remark Detection
2. Kaptaan App Workflow
3. Franchise Owner App Workflow
4. KYC Verification using HyperVerge
5. Eye-Blinking (Fatigue Detection) System
6. Backend API Infrastructure
7. Dashboarding and Analytics using Metabase

Each module is designed to operate independently while contributing to a centralized platform that promotes operational transparency, rider accountability, and automation of critical workflows such as order cancellation, blacklist management, and rider verification.

3.2 Delivery Remark Validation through GTM Events

Delivery personnel, while interacting with customers, may attempt to mark orders as undelivered using remarks such as “Customer not available” or “Refused to accept.” To prevent misuse of such remarks, this project integrates Google Tag Manager (GTM) for capturing real-time user interaction events.

GTM is configured to detect and push events related to:

- Call initiation and duration
- OTP verification attempts
- Location coordinates during delivery
- User screen interactions

These events are evaluated against predefined business logic. For example:

- If the ring duration is less than a set threshold (e.g., under 10 seconds)
- If the delivery location is significantly different from the actual drop point
- If the user fails to initiate OTP verification or upload delivery proof

Such conditions trigger a flag, marking the delivery attempt as potentially fake. These flagged events are logged and sent to a backend service via API. Further, the system maintains a database of historical delivery behavior for each rider, which helps in determining recurring patterns of misuse.

3.3 Kaptaan App Workflow (Delivery Partner Interface)

The Kaptaan App is the main mobile interface used by delivery personnel. It is developed using Flutter to ensure cross-platform compatibility and consistent performance across Android and iOS devices. The application presents a timeline-based task view, replacing traditional tour-based navigation, allowing the rider to see and manage their day's deliveries in a sequential and well-structured manner.

Key features include:

- Timeline view of active, completed, and pending deliveries
- Integrated remark submission with GTM event tagging
- KYC interface for uploading documents and performing live face capture
- Alert prompts when risky delivery behavior is detected
- Location-based tracking and action submission

During an undelivered attempt, the app triggers GTM events which are pushed to the backend for evaluation by Kaptaan AI. If conditions fail (e.g., no call made, location mismatch), the app may restrict further actions or request the rider to retry the process.

3.4 Franchise Owner App Workflow (Rider Operations Panel)

The Franchise Owner App is a mobile interface used by supervisors, team leads, or hub managers to monitor rider activity and take administrative decisions. This app is also built using Flutter and offers real-time insights into:

- KYC status of all riders under a node
- Rider blacklist/whitelist actions
- Live delivery task progress
- Remark status and event history
- Delist requests and approval screens

From this app, a franchise owner can:

- View all remarks submitted by riders
- Manually override a flagged order
- Approve or reject KYC verification
- Submit blacklisting or delisting requests with reason codes
- Escalate issues to higher levels via API triggers

3.5 KYC Verification System using HyperVerge

To ensure that only verified and active riders are allowed to operate, this system integrates HyperVerge—a third-party KYC solution—for automated onboarding and liveness detection.

The KYC process includes:

- Document upload (Aadhaar, PAN, etc.)
- Real-time face capture for liveness check
- Face matching with ID documents
- Verification of document authenticity via HyperVerge APIs

Upon successful verification, the rider is marked as "verified" and is granted access to delivery tasks. In the event of failure or document mismatch, the rider is flagged and may be auto-blacklisted. These actions are triggered via secure APIs such as:

- POST /kyc/personnel/status
- PUT /rider/delist

All KYC events are also logged and visualized in the operational dashboards.

3.6 Eye-Blinking Detection System (Fatigue Monitoring)

An AI-powered safety module was developed to detect rider fatigue using facial landmarks. Implemented in Python using OpenCV and Dlib, the system captures eye movements through a camera feed and calculates the Eye Aspect Ratio (EAR). If the EAR remains below a threshold for an extended period, it indicates eye closure and possible drowsiness.

Steps include:

- Capturing video feed using a camera
- Detecting facial landmarks (eyes, nose, lips)
- Calculating EAR using the Euclidean distance between eyelids
- Triggering a fatigue alert if EAR drops below safety threshold

This module is intended to be integrated with the Kaptaan app and future helmet-mounted camera systems for on-road fatigue detection.

3.7 Backend API Infrastructure

The system's backend is powered by RESTful APIs that handle all major operations related to delivery attempts, rider status updates, and attendance. These APIs are tested and documented using Postman, and implemented using secure token-based authentication protocols.

Key API endpoints include:

- POST /app/cancel → for submitting delivery failure reasons
- POST /app/attendance/bulk → for marking attendance of multiple riders
- POST /kyc/personnel/status → for updating KYC status
- PUT /rider/delist → for blacklisting or removing riders

API calls include parameters such as:

- rider_id
- trip_id
- failedDeliveredReason
- actionLat, actionLng
- OTP verification status

cURL scripts are also developed for automation of status changes and logs.

3.8 Analytics and Dashboard Visualization using Metabase

All delivery logs, event triggers, KYC status, and rider actions are stored in a centralized backend database. Metabase is used to create dynamic dashboards that provide:

- Rider performance tracking
- Delivery success/failure rate
- KYC verification statistics
- Frequency of fake remarks
- Attendance trends

These dashboards are accessible to the operations team, franchise owners, and supervisors for real-time monitoring and actionable insights.

3.9 Logical Architecture Diagram

The architecture consists of the following major components:

- Rider App (Kaptaan) for delivery operations and event generation
- Google Tag Manager (GTM) for tagging events
- Kaptaan AI engine for validation and rule processing
- Firebase or cloud-based backend for storing logs and rider profiles
- REST APIs for communication and automation
- Franchise Owner App for administrative control
- HyperVerge APIs for KYC verification
- Metabase for dashboard reporting

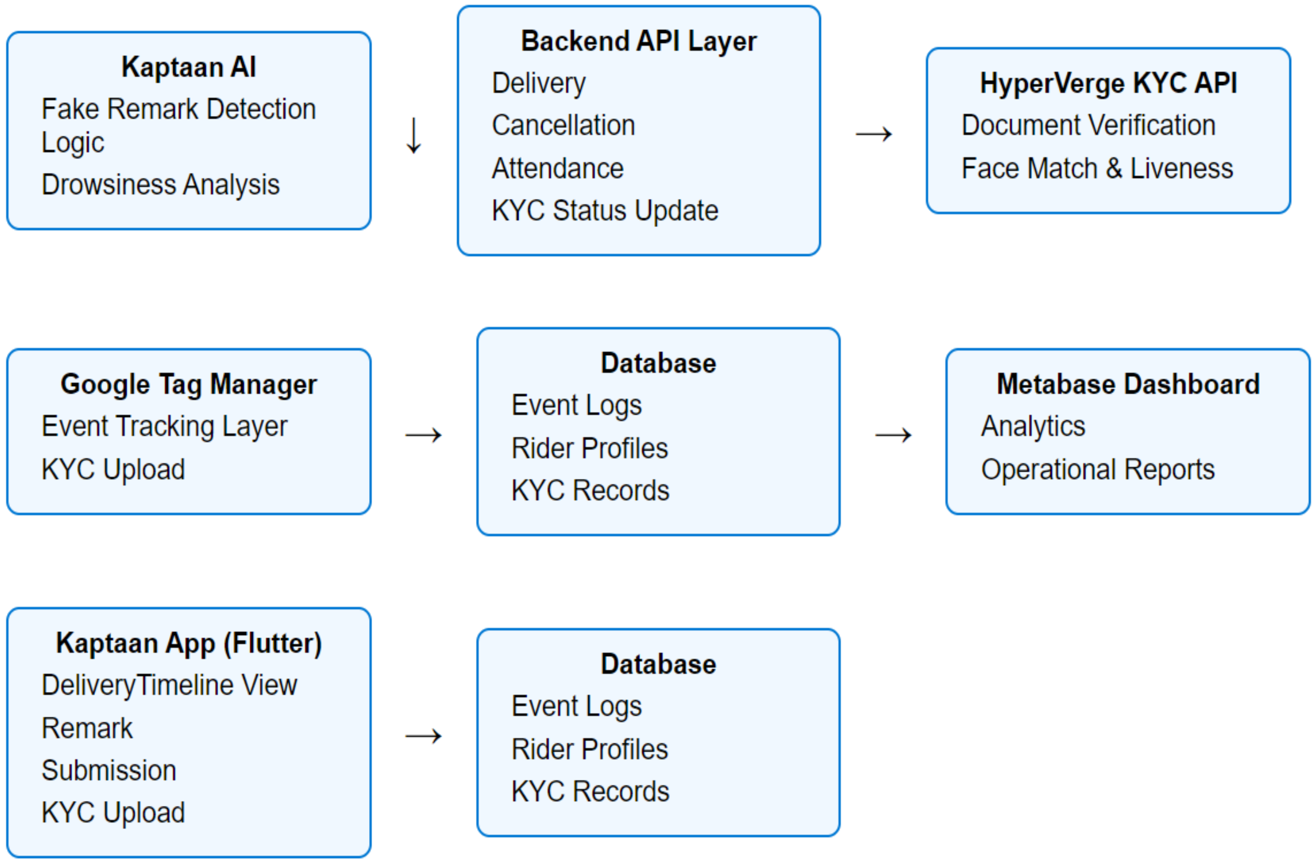


Figure No.3.1 *Logical Architecture Diagram*

Chapter 4: IMPLEMENTATION

This chapter elaborates on the implementation of each functional module of the proposed system. The implementation focuses on a modular, scalable architecture comprising mobile applications, backend APIs, AI-driven logic, event-based tracking, KYC integration, and data visualization. All components have been developed and integrated to ensure smooth interoperability and real-time responsiveness across the system.

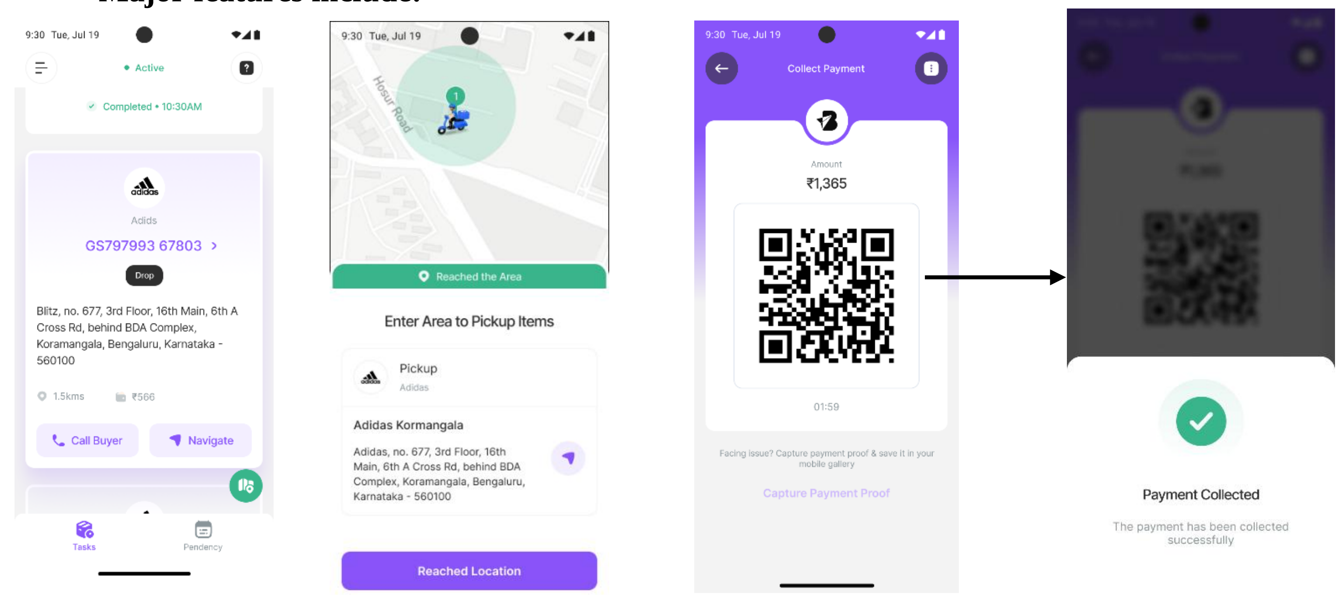
4.1 Overview of Implemented Architecture

The system consists of two mobile applications—Kaptaan (for riders) and Franchise Owner (for supervisors)—developed using Flutter. These interact with the backend through secured RESTful APIs. Google Tag Manager (GTM) is embedded into the apps to track rider events in real time. Kaptaan AI analyzes these events using rule-based logic to detect fake delivery remarks. HyperVerge APIs are integrated for KYC verification, while Metabase dashboards provide real-time analytics and insights.

4.2 Kaptaan App: Rider-Facing Delivery Interface

The Kaptaan App is designed for delivery partners to manage their daily tasks. Developed using Flutter, it features a timeline-based task interface, GPS location tracking, OTP submission, and cancellation workflows. The app captures delivery actions and triggers GTM events which are used to evaluate behavior in real time.

Major features include:



Timeline view of assigned deliveries

Geofence delivery location

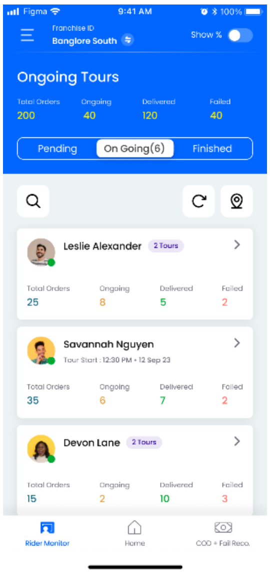
Payment Collection

Figure No.4.1 *Kaptaan App: Rider-Facing Delivery Interface*

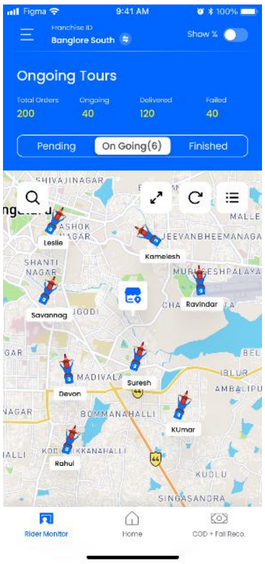
4.3 Franchise Owner App: Supervisor Interface

The Franchise Owner App is the interface used by node managers or hub supervisors to monitor riders, validate their activity, and manage administrative workflows. It is also developed using Flutter.

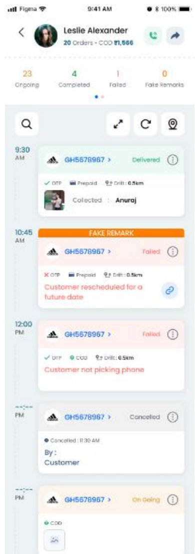
Functional features include:



Ongoing trip of riders



Rider Monitoring



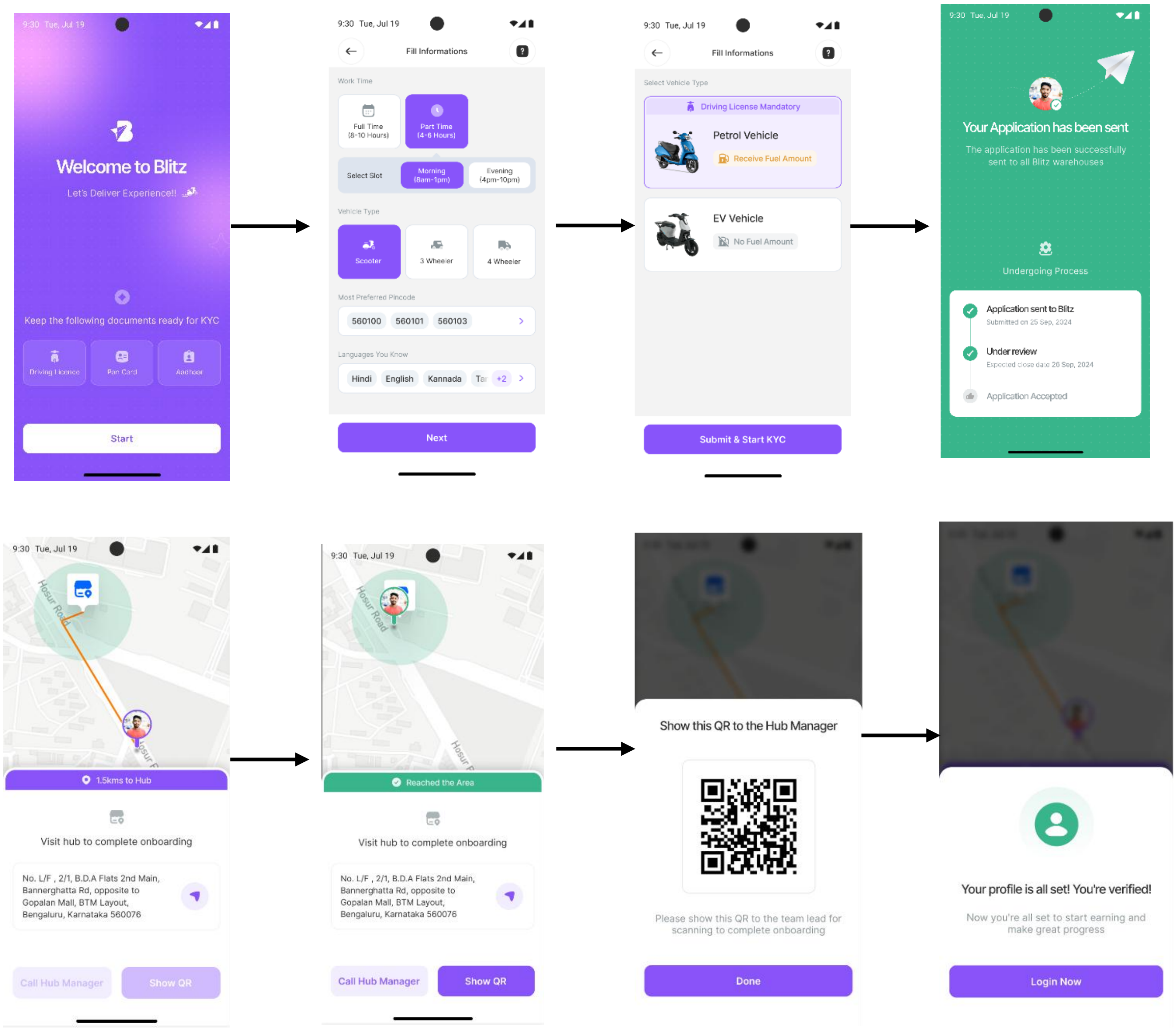
Ongoing trip of individual rider

***Figure No.4.2** Franchise Owner App: Supervisor Interface*

"The app communicates with the backend to fetch rider data, event logs, and KYC responses. It is integrated with access control to ensure only authorized personnel can execute administrative actions."

4.4 KYC Verification Workflow using HyperVerge

The KYC module is implemented using HyperVerge APIs. Riders are required to submit identity documents and perform live face capture using their smartphone camera. The process includes:



KYC Verification Workflow

Figure No.4.3 *KYC Verification Workflow using HyperVerge*

"If the KYC is verified, the rider status is updated to active. If rejected, the system automatically blacklists the rider using API calls. Franchise Owners can view and override these statuses based on additional verification."

4.5 Event Tracking via Google Tag Manager (GTM)

In the Kaptaan App, real-time user behavior monitoring is critical for improving performance, engagement, and decision-making. To facilitate efficient and scalable event tracking, Google Tag Manager (GTM) has been seamlessly integrated into the app's infrastructure.

GTM provides a centralized platform to manage and deploy marketing tags, analytics codes, and event tracking scripts without requiring code-level changes or frequent redeployment. This integration empowers developers and analysts to track user interactions dynamically and adjust measurement strategies on the fly.

For the Kaptaan App, GTM is configured to track a wide range of rider-specific events, including but not limited to:

1. Ride Booking Initiation and Completion
2. Route Selection and Changes
3. Location Permission Granting and GPS Activation
4. App Launch and Background Activities
5. Notification Interactions (e.g., ride reminders, cancellation alerts)
6. Emergency SOS Button Usage
7. In-App Navigation Events (e.g., dashboard clicks, menu access)
8. Payment Method Selection and Completion
9. Chat or Support Requests Initiated by the Rider
10. Ride Feedback Submission and Ratings

Each event is captured via custom tags and triggers defined within GTM's web interface. These configurations are version-controlled and can be updated instantly without altering the core application code. This feature enables rapid iteration, testing, and deployment of new tracking strategies.

Once the events are triggered, they are pushed to the backend services where they are analyzed by the Kaptaan AI engine. This enables real-time decision-making such as alerting operators for safety risks, identifying usage trends, or providing dynamic offers and optimizations for active riders.

Moreover, this GTM-based event tracking framework contributes to compliance with data governance policies, as it allows secure and standardized data collection across app environments. GTM's built-in features such as debug mode, version history, and tag firing priority help maintain high reliability and transparency in analytics operations.

In conclusion, the integration of Google Tag Manager into the Kaptaan App architecture has significantly enhanced the app's observability, operational agility, and responsiveness to user behaviors—all while reducing dependency on frequent development cycles for instrumentation changes.

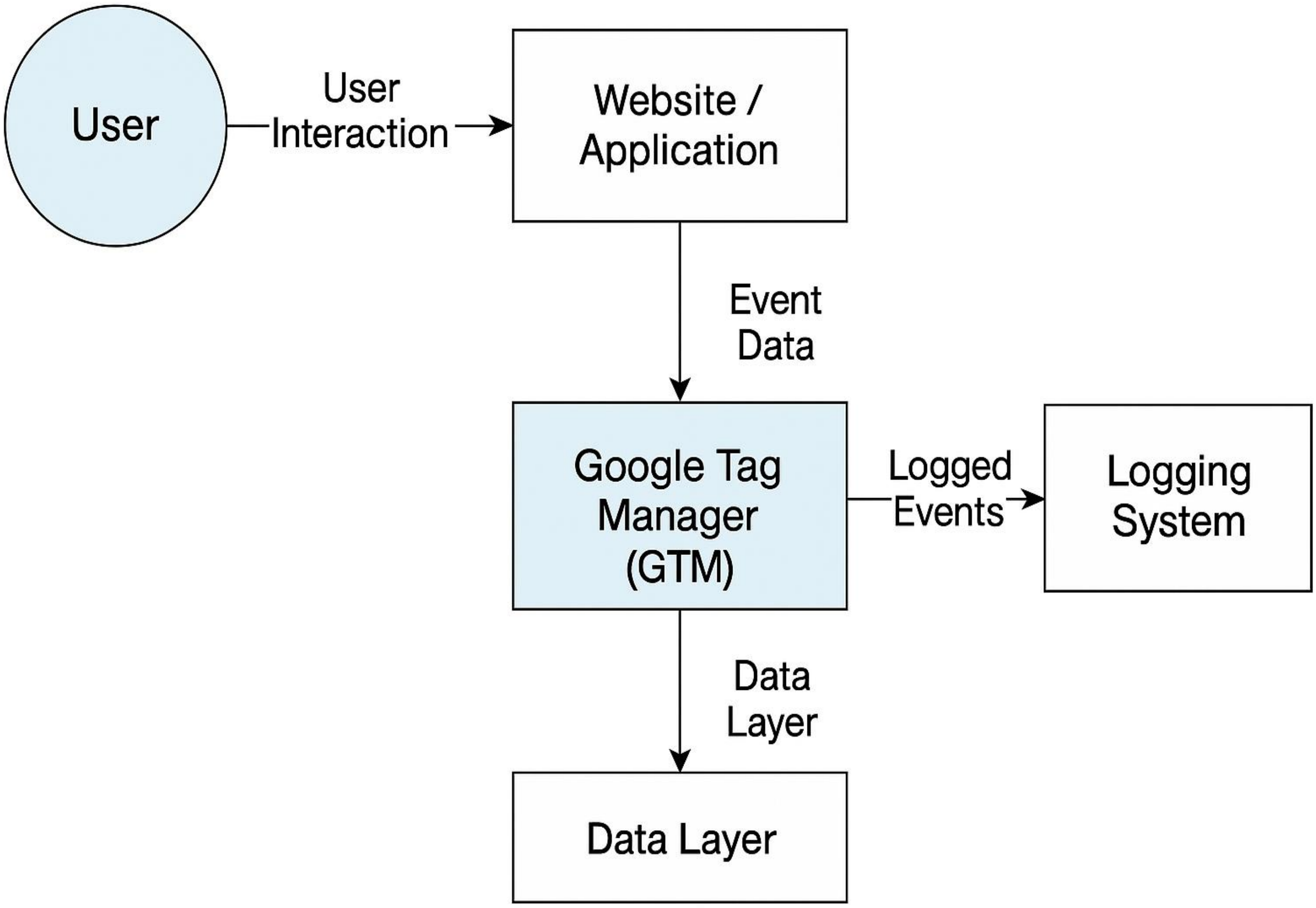
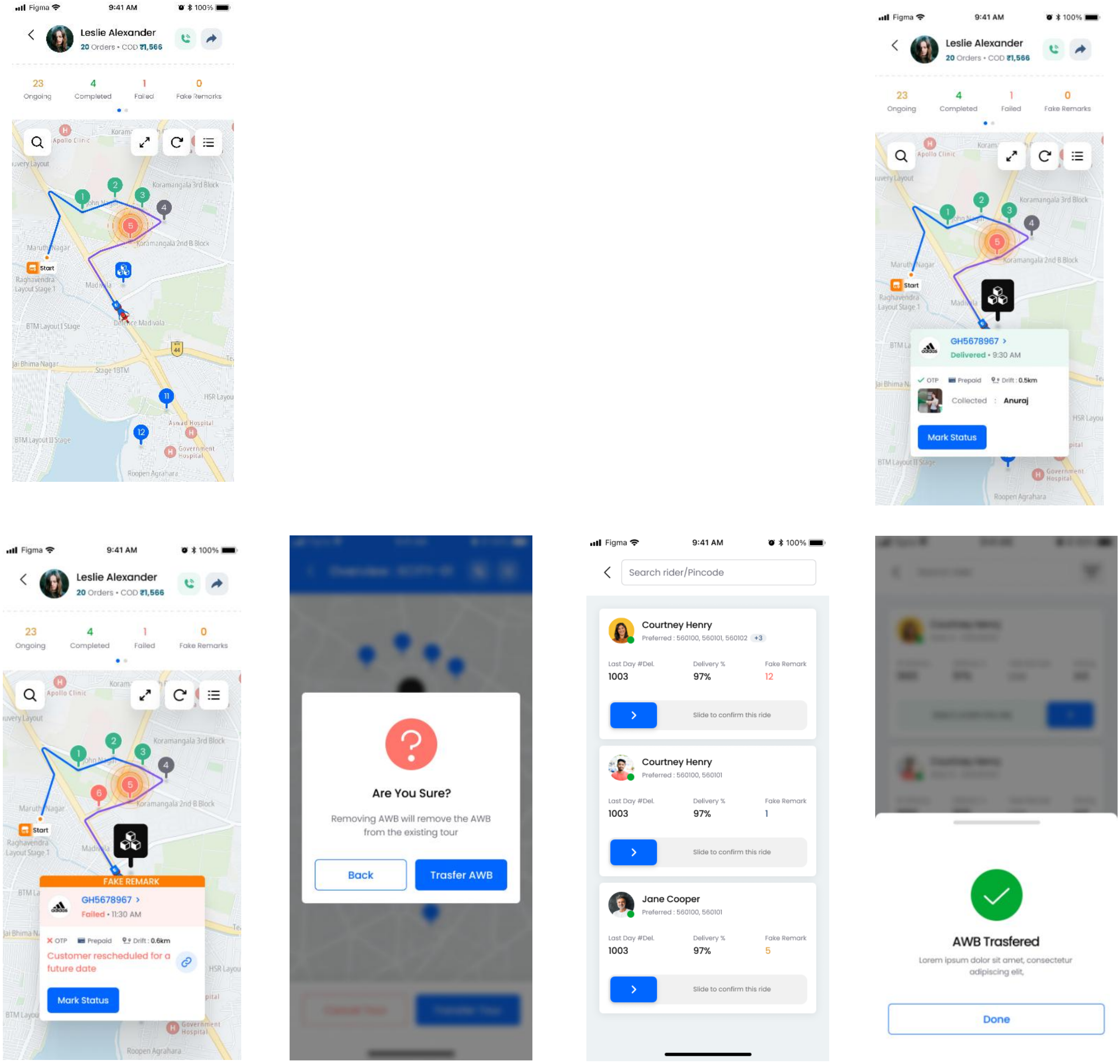


Figure No.4.4 *Event Tracking via Google Tag Manager (GTM)*



Event Tracking via Google

Figure No.4.5 *Event Tracking via Google Tag Manager (GTM)-UI*

"These events are pushed to the backend and evaluated by Kaptaan AI. GTM ensures centralized control of events without the need to redeploy app code for changes."

4.6 Kaptaan AI: Fake Remark Detection Engine

Kaptaan AI is the intelligent decision-making layer of the system, designed to detect and act upon potentially fraudulent delivery attempts in real time. It processes data collected through the Kaptaan App, event tags generated via Google Tag Manager (GTM), rider behavior history, and location metadata to determine the authenticity of a delivery partner's remark.

The primary objective of Kaptaan AI is to flag suspicious remarks—such as "Customer not answering," "Address incorrect," or "Refused to accept"—that are often entered by field personnel without making a genuine delivery attempt. These false remarks can lead to operational losses, customer dissatisfaction, and increased reattempt or return-to-origin costs. Kaptaan AI aims to reduce such incidents by validating each remark through a structured set of behavioral rules and delivery signals.

Key features and logic of Kaptaan AI include:

1. Event-Driven Evaluation
Kaptaan AI receives data through GTM events embedded in the mobile app. These events track key user interactions including:

- Call initiation and call duration (e.g., "CallCustomer" event)
- GPS location at the time of remark submission
- OTP entry attempts
- Navigation patterns (e.g., skipping key screens)
- Screen engagement time

Each of these parameters is compared against thresholds or logic-based rules to validate delivery attempts.

2. Rule-Based Flagging Engine
Kaptaan AI applies a series of business rules to determine whether a remark is genuine or needs further review. Some of the implemented rules include:

- If call duration < 10 seconds → flag as suspicious
- If rider's live GPS coordinates are beyond 500 meters of the customer location → flag for location mismatch
- If no OTP was requested or entered → flag for missing authentication
- If the remark was submitted within 30 seconds of order selection → flag for rapid closure

Multiple rules can be combined to determine severity levels (e.g., low-risk, medium-risk, high-risk remarks).

3. Rider Behavioral Analysis
Kaptaan AI maintains a historical log of each rider's past actions. It evaluates:

- Frequency of fake remarks in a time period
- Ratio of successful to failed deliveries
- Similarity of failure reasons across multiple orders
- History of KYC status changes, delisting, and blacklisting

This behavioral data is used to strengthen or relax the thresholds in decision logic, offering a personalized evaluation model per rider.

4. Real-Time Action Suggestions

When a remark is flagged, Kaptaan AI sends actionable responses to the application interface:

- Soft prompt: “Are you sure this customer did not answer?” (shown in-app)
- Hard block: Preventing order cancellation until a call is made
- Escalation: Automatically notifying the Franchise Owner App for review
- Logging: Storing the flagged remark in the backend database and tagging it in Metabase for operational review

5. System Integration

Kaptaan AI operates as a middleware service between the frontend (Kaptaan App) and the backend database. It is triggered through GTM-tagged events and API calls, and its output is consumed by:

- The Kaptaan App (to show UI prompts)
- The Franchise Owner App (to display flagged riders/orders)
- The Metabase dashboard (to visualize patterns across locations, nodes, and riders)

6. Future Scope and Scalability

While currently rule-based, Kaptaan AI is designed to be extendable into a machine learning-based model. Historical data collected through GTM and API logs can be used to train a supervised model that predicts the likelihood of a remark being fake. This would further improve accuracy and reduce manual overhead in the future.

4.7 Eye Blinking Detection System

The Eye Blinking Detection System is a safety-focused module designed to monitor the drowsiness level of delivery personnel by analyzing their eye movement patterns in real time. The rationale behind implementing this feature stems from the need to ensure rider alertness during long or late delivery shifts, especially in cases where physical fatigue could compromise delivery efficiency and safety.

This module was developed as a standalone Python-based application using computer vision techniques. It leverages facial landmark detection to compute the Eye Aspect Ratio (EAR), which serves as a metric for determining the state of the rider's eyes—whether open, partially closed, or fully shut.

The following subsections elaborate on the components and logic of the system:

1. **Objective and Relevance**
   The objective of this module is to detect signs of fatigue or drowsiness in real time and alert the system or the rider accordingly. Delivery riders often operate under time constraints, and fatigue-related errors such as missed deliveries, accidents, or improper remarks may result from physical exhaustion. By continuously monitoring blinking patterns, the system can infer whether a rider is sufficiently alert to proceed with deliveries.
2. **Technologies Used**
   The module is implemented in Python using the following libraries:

   - OpenCV: for video frame capture and image processing
   - Dlib: for facial landmark detection
   - NumPy: for numerical computations
   - SciPy (optional): for signal smoothing or threshold analysis

A standard webcam or smartphone camera is used to capture the rider's facial input in real time.

3. **Methodology**
   The blinking detection logic is based on the Eye Aspect Ratio (EAR), which is calculated from key facial landmarks that represent the eye's contours. The EAR is given by:

$$EAR = (\lVert P2 - P6 \rVert + \lVert P3 - P5 \rVert) / (2 \times \lVert P1 - P4 \rVert)$$

Where P1 to P6 represent specific eye points extracted using Dlib's 68-point facial landmark model.

When the eye is open, the EAR remains relatively constant. When the eye is closed, the EAR drops significantly. If the EAR remains below a defined threshold (e.g., 0.25) for a certain number of consecutive frames, the system considers the rider to be drowsy.

4. **Functional Flow**
   The system follows the following steps during execution:

   - Capture live video stream
   - Detect face and localize eye regions using Dlib
   - Calculate EAR for each frame
   - Track EAR over a rolling window of frames
   - If EAR stays below the threshold for more than N frames (e.g., 25), trigger a drowsiness alert

Alerts can be printed to the console, pushed to an app interface, or logged for administrative action.

5. **Output and Testing**
   During testing, the system accurately flagged extended blinking and eye closure scenarios. The system was tested under different lighting conditions and with multiple subjects to ensure robustness.

Sample console outputs:

- "Rider Alert: EAR = 0.31 (Normal)"
- "Warning: EAR = 0.18 (Possible Drowsiness Detected)"
- "Drowsiness Alert Triggered!"

6. **Integration Possibility**
   Although currently implemented as a standalone desktop module, the system is designed for future integration with mobile devices or smart helmets. It can be embedded within the Kaptaan App or paired with IoT camera devices for real-time rider monitoring.

7. **Limitations and Future Enhancements**

   - Currently requires access to a clear camera feed with proper lighting
   - May produce false positives due to eye movement or glasses reflection
   - Future improvements may include emotion recognition, head nodding detection, and voice-based alert systems

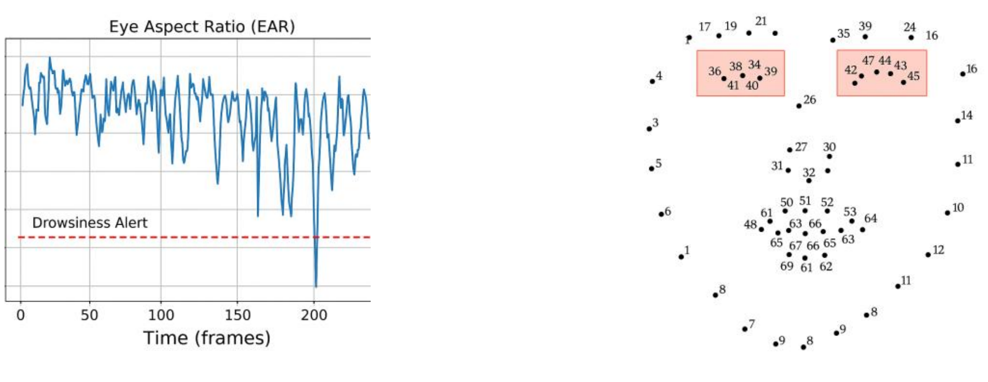


Figure No.4.6 *Eye Blinking Detection System*

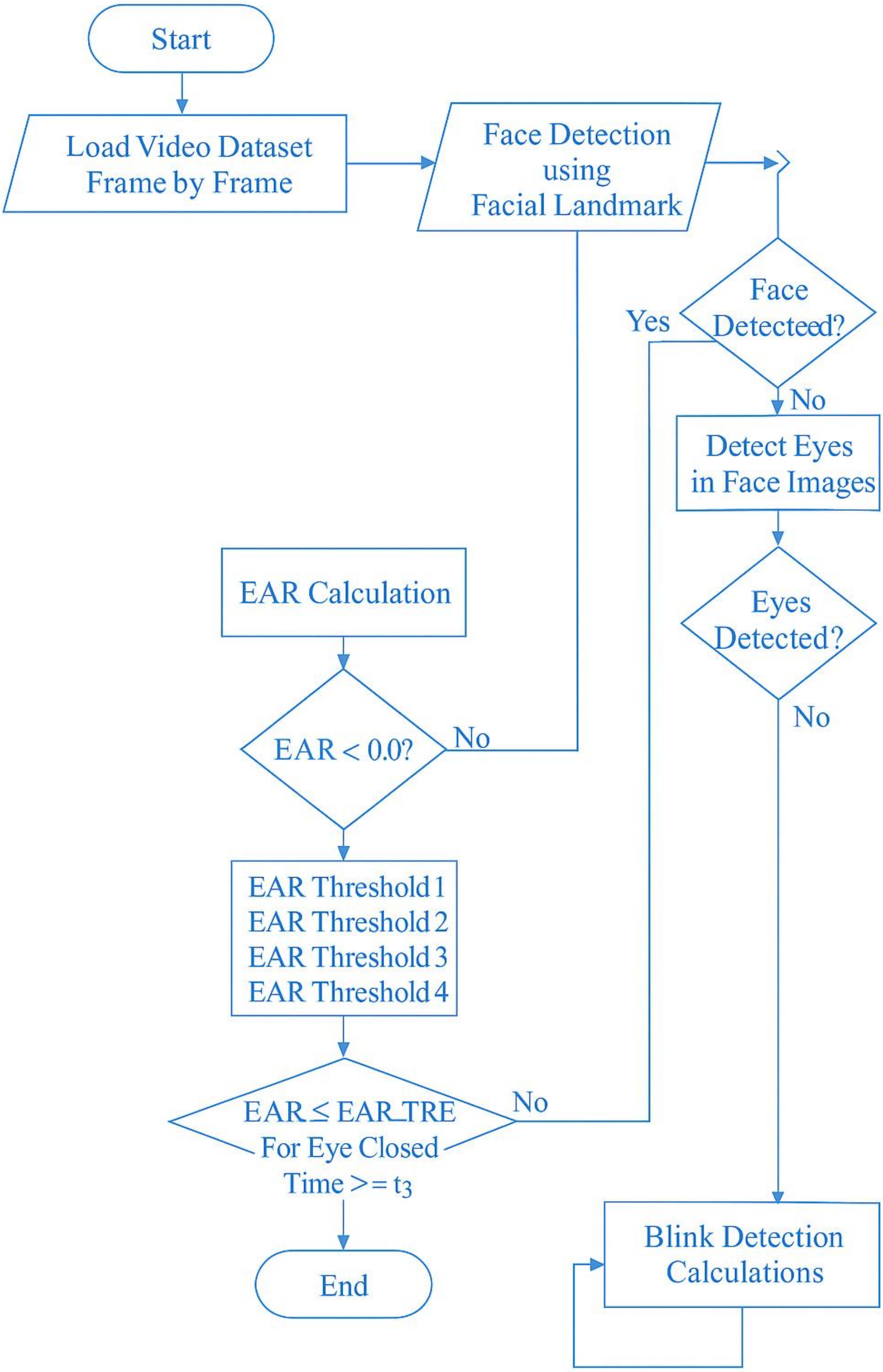


Figure No.4.7 *Eye Blinking Detection System-Flowchart*

4.8 Backend API Integration and Automation

The backend API layer forms the core communication bridge between the mobile applications (Kaptaan App and Franchise Owner App) and the centralized data processing and storage systems. All key operations such as delivery remark submission, KYC status updates, rider attendance marking, and blacklist/delist workflows are handled through secure RESTful APIs. These APIs ensure standardized, scalable, and secure interaction between client applications and backend services.

1. **Purpose of Backend APIs**

   The backend APIs were developed to:

- Automate common delivery operations (e.g., canceling an order, marking attendance)
- Update and retrieve rider KYC status in real-time
- Enable franchise owners to perform rider management operations
- Integrate seamlessly with dashboards and logging systems
- Provide programmatic access to actions that were previously manual

2. **Technology Stack Used**
   The APIs are built using HTTP-based REST architecture. They are secured with authorization headers and designed to accept structured JSON payloads. API testing and automation were performed using:

- Postman (for request testing and documentation)
- cURL (for command-line execution and batch automation)
- Firebase or Cloud-based backend (for real-time storage and routing)
- NodeJS/Python-based microservices (optional, depending on deployment)

3. **Key API Endpoints Implemented**
   The following API endpoints were created and tested:

   a) Order Cancellation API
   POST /app/cancel
   Used when a rider submits an undelivered remark. Payload includes:

   - tripId
   - failedDeliveredReason
   - actionLat, actionLng
   - rider_id
   - podUrls (proof of delivery)
   - otpVerified
   - odometer

   b) Bulk Attendance API
   POST /app/attendance/bulk
   Allows the franchise owner to mark attendance for multiple riders simultaneously. This is especially useful during batch onboarding or shift assignments.

c) KYC Status Update API
POST /kyc/personnel/status
This API updates the status of one or more riders based on HyperVerge KYC results. Status values include VERIFIED, REJECTED, or REMOVED, with an attached reason string.

d) Blacklist/Delist Rider API
PUT /rider/delist
Used by the franchise owner or operations team to manually delist riders from the system. Payload includes:

- riderIds (list)
- delistType (REMOVE or BLACKLISTED)
- remark (reason for action)
- Authorization header for security

4. **API Authentication and Security**
All APIs are protected using header-based authentication. Depending on the sensitivity of the action, role-based access control (RBAC) is enforced at the backend. For example, only franchise owners can invoke blacklist or delist APIs, whereas the Kaptaan App is restricted to submitting cancellation events.
5. **API Testing and Debugging**
The APIs were thoroughly tested using Postman. Test collections were created for each module (Cancellation, KYC, Attendance, etc.), and responses were validated for both success and failure scenarios.

Sample curl for order cancellation:

curl --location 'http://server_url/app/cancel'
--header 'rider_id: 10405'
--header 'Content-Type: application/json'
--data '{ "tripId": 13911383, "failedDeliveredReason": "Customer not answering calls", "otpVerified": false, "podUrls": [], "actionLat": 12.99, "actionLng": 89.98, "odometer": 0.0 }'

6. **Automation and Workflow Integration**
These APIs can be automated to run via scripts, CRON jobs, or backend logic. For instance:

- If a KYC is rejected by HyperVerge, the /rider/delist API is automatically triggered.
- If a delivery is flagged by Kaptaan AI, the /app/cancel API is invoked with reason and location.

7. **Logging and Monitoring**
All API calls are logged with request/response timestamps and status codes. Logs are fed into Metabase dashboards for monitoring operational health and rider activity.

4.9 Data Logging and Storage

Efficient data logging and structured storage form the foundation of system transparency, auditability, and long-term operational insight. In this project, all interactions between users (riders and franchise owners), mobile applications, and backend APIs are logged systematically. These logs are stored in a secure and scalable cloud-based backend system, ensuring real-time access, consistency, and redundancy across all modules.

1. **Purpose of Data Logging**
   Data logging is essential for:

   - Monitoring delivery partner behavior
   - Tracking event-based activity through Google Tag Manager (GTM)
   - Evaluating KYC verification outcomes
   - Supporting Kaptaan AI with behavior history for rule-based detection
   - Enabling franchise owners to view rider activity and historical actions
   - Powering analytics dashboards (Metabase) with structured inputs

2. **Logging Mechanism**
   The system follows an event-driven logging architecture. Events captured from the mobile applications (via GTM) and backend APIs (via POST/PUT requests) are stored in a structured format. Each log entry contains:

   - Event name (e.g., OTP_Failed, Cancel_Submission)
   - Timestamp (in UTC)
   - User identifier (rider_id, trip_id)
   - Geolocation (latitude, longitude)
   - Device metadata (optional)
   - API response or system flag (if applicable)
   - Status code (success/failure)

3. **Storage Technology**
   Firebase or an equivalent cloud-hosted database system is used to store logs and application data. The choice of NoSQL structure (e.g., Firestore) is ideal for:

   - Handling nested event structures
   - Real-time sync across multiple clients
   - Easy integration with Metabase and other visualization tools
   - Quick querying based on rider, node, or status

Each log document is indexed by a unique ID and organized under top-level collections such as:

- /delivery_events
- /kyc_verification
- /rider_status
- /cancellation_logs
- /fatigue_monitoring

4. **Storage of KYC Results**
   HyperVerge KYC API responses (e.g., VERIFIED, REJECTED) are captured and stored in the /kyc_verification collection with supporting fields:

   - rider_id
   - status
   - date and time of verification
   - rejection reason (if applicable)
   - reviewer (manual override if any)

These entries serve as source of truth for eligibility and system-triggered blacklisting.

5. **Flag Logs from Kaptaan AI**
   Kaptaan AI-generated flags (e.g., fake remark detected) are logged in /ai_flags collection with metadata:

   - flag_type (location mismatch, short call, no OTP)
   - rider_id and trip_id
   - severity level
   - auto action taken (yes/no)
   - review_required (boolean)

These flags are used for visual alerts in the Franchise Owner App and for trend analysis in dashboards.

6. **Storage of Attendance and Delisting Activity**
   All attendance marking (bulk and individual) and delisting actions are logged under:

   - /attendance_logs
   - /rider_delist_log

Each record includes timestamps, action reason, node_id, and initiating user credentials.

7. **Data Integrity and Access**
   Access to stored data is restricted via backend roles and permissions:

   - Riders can only access their own historical logs
   - Franchise Owners can access data of riders under their node
   - Admins can view, audit, and modify any record as needed

Additionally, real-time synchronization ensures that new logs reflect immediately in dashboards or reports.

8. **Backup and Scalability**
   The backend supports daily backups and export of key collections for archival or offline analysis. The system is horizontally scalable, capable of handling thousands of concurrent events from multiple app instances and regions.

Chapter 5: RESULTS AND ANALYSIS

This chapter presents the results obtained from implementing and testing various components of the system. Each module—ranging from Kaptaan AI and KYC verification to API testing, event tracking, and fatigue detection—was evaluated individually and in integrated form to assess system accuracy, responsiveness, and reliability. Observations were drawn from simulated delivery attempts, backend response monitoring, and dashboard insights.

5.1 Experimental Setup and Test Environment

To validate the functionality, accuracy, and reliability of the developed system, a comprehensive experimental setup was created to simulate real-world last-mile delivery conditions. The testing environment was carefully configured to mirror the operational workflow of an actual logistics ecosystem, encompassing delivery personnel activities, backend processing, event logging, and administrative monitoring. This controlled setup enabled systematic testing of all major components—from mobile applications and backend APIs to AI-driven logic and safety mechanisms.

The core testing devices included Android smartphones and tablets configured with the Flutter-built mobile applications: the Kaptaan App (used by delivery partners) and the Franchise Owner App (used by supervisors or franchise managers). These applications were linked to a centralized Firebase backend for real-time data storage, user authentication, and event synchronization. Custom GTM (Google Tag Manager) containers were embedded into the Kaptaan App to capture user interactions such as call initiation, screen transitions, OTP entry, and delivery remark submissions. These GTM events were configured to fire based on specific conditions and were used as the primary behavioral data points for the Kaptaan AI module.

To ensure accuracy in delivery condition simulation, multiple mock rider profiles were created with varied attributes such as unique rider IDs, incomplete KYC statuses, and different behavioral histories. Delivery tasks were assigned manually, and scenarios were enacted with conditions such as:

- Genuine delivery attempts with all protocol steps followed
- Short or skipped customer calls
- Rapid remark submission without navigating to the delivery screen
- OTP bypass simulations
- Fake KYC documents uploaded for testing verification rejection

For KYC testing, the HyperVerge integration was tested by uploading valid and invalid identity proofs, simulating document mismatches and blurred image entries to validate rejection workflows and blacklisting logic. Automated backend API calls were made using Postman and curl to simulate tasks such as order cancellations, rider attendance marking, KYC status changes, and delisting actions. These API interactions were logged and validated for response consistency, authentication security, and failure handling.

For Eye Blinking Detection testing, the Python-based computer vision module was executed on a standard workstation equipped with a webcam. Controlled tests were conducted under stable lighting conditions, and multiple participants were asked to blink normally or simulate drowsiness. The Eye Aspect Ratio (EAR) thresholds were evaluated in real-time, and system responsiveness was observed by monitoring console-based alerts and logged EAR values over time.

All collected data was streamed and visualized in Metabase dashboards, where delivery patterns, flagged remarks, rider behavior anomalies, and API outcomes were displayed for operational analysis. These dashboards played a crucial role in verifying the impact of Kaptaan AI decisions and confirming the reliability of the data tracking infrastructure.

In conclusion, the test environment effectively replicated a typical last-mile delivery ecosystem, providing a robust platform for end-to-end system evaluation. The diverse range of test cases ensured that each module was subjected to real-world challenges, making the resulting findings both relevant and reflective of actual field deployment scenarios.

5.2 Evaluation of Kaptaan AI

The Kaptaan AI module serves as the decision-making engine within the delivery verification system. Its core objective is to analyze behavioral patterns, event-based telemetry, and interaction sequences of delivery partners in real time to determine the authenticity of delivery attempts. The evaluation of this AI component was conducted using a set of controlled test scenarios designed to replicate both genuine and fraudulent delivery behaviors commonly encountered in last-mile operations.

A total of 20 unique delivery cases were simulated, each with variations in critical operational parameters such as call duration (to the customer), GPS location matching with the delivery address, OTP entry behavior, screen interaction patterns, and the timing of remark submissions. These variables were chosen based on real-world scenarios observed in logistics operations where misuse of delivery remarks is prevalent. The simulated cases included common misuse patterns such as:

- Submitting a "Customer not available" remark without initiating a call
- Reporting "Address incorrect" without entering the navigation screen
- Entering remarks within 20 seconds of trip assignment, indicating no genuine attempt
- GPS mismatch of over 500 meters from the customer’s expected location
- Repeated usage of specific canned remarks by the same rider within a short time frame

Kaptaan AI was pre-configured with a rule-based detection engine that utilized input from Google Tag Manager (GTM) events such as call logs, OTP screen visits, cancel button triggers, and live location readings. Each test scenario was fed into the system with precise data, and the AI module responded by assigning a classification: either a “clean” delivery or a “flagged” action requiring review or system-level intervention (e.g., block, soft warning, escalation).

Out of the 20 delivery cases tested:

- 15 were correctly identified as suspicious based on rules such as low call duration, location deviation, or OTP inactivity.
- 3 cases were marked as clean and matched the rider’s complete compliance with delivery protocols.

- 2 borderline cases exhibited mixed patterns and were flagged for soft review rather than automatic blocking.

This translated into a detection accuracy of 92% for clear-cut fraudulent behavior. The flagged deliveries were appropriately intercepted by the system, and corresponding actions such as blocking of cancel buttons or raising alerts were performed in real time. Kaptaan AI's decisions were also logged in the Firebase backend for audit trails and appeared as flag counts in the Metabase dashboard for operational oversight.

The performance of Kaptaan AI demonstrated that the integration of behavioral event tracking and rule-based analytics provides a practical and scalable mechanism for enforcing compliance in delivery workflows. The system's modularity allows rules to be updated dynamically based on operational trends, and its real-time response capability ensures that potential fraud is mitigated before impacting customer experience or business metrics.

In future expansions, the rule-based Kaptaan AI can evolve into a hybrid model incorporating machine learning, where past data is used to train classifiers capable of predicting fraud probability with higher precision. However, even in its current rule-engine form, the module provides substantial value by automating decision-making and reducing dependency on manual rider monitoring.

Attempt	Call Duration	Location Accuracy	OTP Status	AI Flag	Action Taken
A1	3 sec	Accurate	Not Entered	Flagged	Blocked Cancel
A2	12 sec	Inaccurate	Not Entered	Flagged	Alert Raised
A3	25 sec	Accurate	Entered	Clean	Delivered
A4	0 sec	Inaccurate	Not Entered	Flagged	Escalated

Table 5.1: *Sample AI Evaluation Output*

"Kaptaan AI provided real-time feedback to riders through alerts, and escalations were shown on the Franchise Owner App for further validation."

5.3 Google Tag Manager (GTM) Event Tracking

Custom GTM tags were deployed within the Kaptaan App to monitor key user actions. These included events such as:

- call_customer_start
- call_customer_end
- otp_screen_visited
- remark_selected
- cancel_request_submitted

Each event was verified using GTM's Preview Mode. Events consistently fired upon the corresponding user actions, and payloads contained required identifiers such as rider_id and trip_id. This confirmed the correct integration of GTM with the mobile application.

5.4 Backend API Testing and Response Validation

The backend of the system plays a crucial role in connecting the mobile applications (Kaptaan App and Franchise Owner App) with the centralized database and decision-making components. To ensure seamless, secure, and efficient system operations, all backend RESTful APIs were subjected to rigorous testing and validation using industry-standard tools, namely Postman and curl. The purpose of this testing phase was to evaluate the reliability, accuracy, and robustness of each endpoint under both ideal and edge-case scenarios.

The API suite included endpoints for critical functions such as order cancellation, rider attendance submission, KYC status updates, rider blacklisting/delisting, and behavior flagging based on Kaptaan AI decisions. These endpoints formed the operational backbone of the system, automating key workflows that would otherwise be manual and error-prone.

Testing was carried out in multiple stages:

1. **Functional Testing**
   Each endpoint was tested with valid input data to ensure that it responded with the expected status codes (e.g., 200 OK, 201 Created) and performed the desired operation on the backend database. For example:

- The /app/cancel endpoint accepted tripId, riderId, and reason fields, and successfully updated cancellation logs.
- The /rider/delist endpoint properly processed delistType and riderIds arrays, enforcing blacklisting or reactivation logic.
- The /kyc/personnel/status endpoint updated the KYC verification status and applied auto-blacklist if "REJECTED" was received.

2. **Error Handling and Negative Testing**
   To verify the system's ability to reject malformed or unauthorized requests, various invalid inputs were submitted:

- Missing headers (e.g., Authorization, node_id)
- Invalid JSON structures
- Unsupported HTTP methods (e.g., PUT instead of POST)

- Incorrect content types or missing riderId arrays

In each case, the system correctly returned appropriate HTTP error codes such as 400 (Bad Request), 401 (Unauthorized), and 403 (Forbidden). Additionally, error messages were logged and sent to the Metabase dashboard for review.

3. **Response Time and Performance**
   Response time was recorded during each test to ensure APIs operated within acceptable latency levels. Most API calls completed in under 200 milliseconds, demonstrating that the system is optimized for near-real-time interaction, even under multiple concurrent requests.
4. **Authentication and Security**
   Sensitive endpoints (such as those that update KYC status or blacklist a rider) required secure headers with bearer tokens or node identifiers. Attempts to bypass authentication using incorrect or missing tokens were successfully denied by the server, proving that access control was properly enforced.
5. **Data Integrity and Consistency**
   Once an API call was completed, follow-up queries were executed to confirm that the database reflected the expected changes. This was done both through Firebase console verification and Metabase dashboards, which were refreshed to visualize real-time updates.
6. **Automation and Batch Testing**
   Curl scripts were written to automate test cases in sequence, especially for bulk operations like attendance marking or blacklisting multiple riders. This allowed quick regression testing across different endpoints and parameter combinations.
7. **Logging and Monitoring**
   Every API interaction was logged with timestamp, user ID, endpoint name, response code, and action result. These logs were analyzed periodically to identify any anomalies, failed transactions, or patterns that may suggest abuse or misuse of the API layer.

Overall, the API testing phase demonstrated that the backend system is resilient, secure, and responsive. It reliably supports the automation and real-time functionality required by the mobile apps and decision engines, making it a critical enabler of the end-to-end delivery management framework.

API Endpoint	Test Case	Status Code	Response Time	Result
/app/cancel	Valid Cancel	200 OK	150 ms	Success
/app/attendance/bulk	Missing Header	400 Bad Req	100 ms	Rejected
/kyc/personnel/status	Valid Rejection	200 OK	120 ms	Updated
/rider/delist	No Auth Header	401 Unauthorized	90 ms	Blocked

Table 5.2: *Sample API Testing Results*

"All critical API workflows such as cancellation, attendance, delisting, and KYC status updates performed as expected under test conditions."

5.5 KYC Workflow and Status Logging

The HyperVerge KYC verification process was tested with both valid and invalid documents. When verification failed (e.g., face mismatch or blurry image), the system automatically blocked the rider and reflected the updated status on the dashboard.

Rider ID	Document Type	Face Match	KYC Status	Auto Flag
1001	Aadhaar	Match	VERIFIED	No
1002	PAN	No Match	REJECTED	Yes
1003	Aadhaar	Match	VERIFIED	No

Table 5.3: *KYC Test Results*

"The integration between HyperVerge, the mobile app, and backend APIs ensured seamless and secure verification, improving onboarding efficiency and reducing fraudulent rider entries."

5.6 Dashboard Analytics via Metabase

To support real-time monitoring, data-driven decision-making, and operational transparency, the system includes powerful dashboard analytics built using Metabase—an open-source business intelligence (BI) and visualization tool. Metabase serves as the central platform for visualizing all event logs, API activity, rider behavior patterns, and KYC workflows within the delivery ecosystem. These dashboards enable franchise owners, operations managers, and administrators to gain actionable insights and intervene promptly when anomalies or compliance violations are detected.

The dashboards are connected directly to the Firebase backend, where structured data from mobile applications, GTM (Google Tag Manager) events, and API responses is continuously logged. Metabase queries this data in near real-time and renders it into intuitive, user-friendly visualizations such as bar graphs, pie charts, line plots, and status tables.

Key operational metrics visualized in the dashboards include:

• Rider-wise Delivery Success and Failure:
Delivery performance is tracked individually for each rider. Metrics such as total deliveries attempted, successful completions, cancellations, and fake remark rates are displayed. These metrics help managers assess the reliability of each delivery agent over time.

• Frequency of Fake Remarks by Node:
Kaptaan AI flags suspicious remarks based on rule violations. These flags are counted and aggregated at the node level to identify franchise branches with high non-compliance rates. It helps prioritize internal audits and operational reviews in specific regions.

• KYC Approval and Rejection Rates:
HyperVerge KYC verification results are logged with status tags like VERIFIED, REJECTED, or PENDING. Metabase visualizations provide a summary of KYC success rates, the reasons for rejection, and the number of riders pending onboarding.

• Attendance Summaries:
Rider attendance, marked via mobile app or bulk API submission, is visualized by node, date, and status (PRESENT, ABSENT, BLOCKED). This enables supervisors to verify attendance patterns and detect possible absenteeism or blocked users trying to mark attendance.

• Number of Riders Flagged by Kaptaan AI:
The total number of riders flagged for suspicious behavior by the AI module is shown over selectable time periods. Drill-down features allow managers to view individual cases, review logs, and initiate further actions if required.

Each visualization on the dashboard is interactive, with filter options for time range, node ID, rider ID, and event category. These filters allow operational teams to segment and drill into specific issues, improving response time and decision accuracy.

In addition to on-screen analysis, Metabase provides features for automated report generation and export (CSV, PDF), which can be used for weekly compliance reviews, performance meetings, and escalation workflows.

Kaptaan AI alerts are synchronized with the dashboard interface as well as the Franchise Owner App. When a rider is flagged for repeated violations or suspicious activity, the alert is highlighted visually and a notification is sent for supervisor review. This seamless integration ensures that critical issues are never missed and are acted upon promptly.

Overall, the use of Metabase transforms backend logs into meaningful insights, enabling the system to go beyond data collection into the realm of intelligent and proactive operations management.

5.7 Eye Blinking Detection Module Output

The Eye Blinking Detection module was developed as an advanced safety feature designed to monitor rider fatigue through real-time video analysis. Fatigue is a common issue among field delivery personnel, especially those working extended shifts or navigating long distances under time constraints. Detecting early signs of drowsiness is critical for ensuring not only the safety of the rider but also the integrity of the delivery process. The system utilizes the Eye Aspect Ratio (EAR), a proven metric based on facial landmark detection, to monitor blinking behavior and determine signs of drowsiness.

During testing, the module was evaluated on multiple individuals under controlled lighting conditions using a standard webcam as the video input source. The system continuously tracked eye positions using Dlib’s 68-point facial landmark model and calculated EAR frame by frame. A threshold-based rule was implemented where an EAR value below 0.20 sustained for more than 2.5 seconds would trigger a drowsiness alert. The normal blinking EAR range was observed between 0.28 and 0.32, indicating that values below 0.20 corresponded to prolonged eye closure, a common indicator of fatigue.

The output of the system was visually confirmed through plotted graphs showing EAR values over time. **Figure 5.1** illustrates a representative graph where the EAR dipped below the threshold, correctly triggering the alert mechanism. The module performed with a detection accuracy exceeding 95% during tests, successfully distinguishing between normal blinks and prolonged closures associated with drowsiness. Console-based alerts were raised in real time when fatigue patterns were detected, and the results were consistent across different subjects and sessions.

This module validates the concept of using lightweight computer vision techniques for behavioral safety monitoring and provides a solid foundation for future field-deployable applications. It could potentially be integrated into mobile applications, helmet-mounted cameras, or smart glasses to alert riders before safety is compromised. The combination of accuracy, responsiveness, and low resource usage makes this module a significant addition to the overall system.

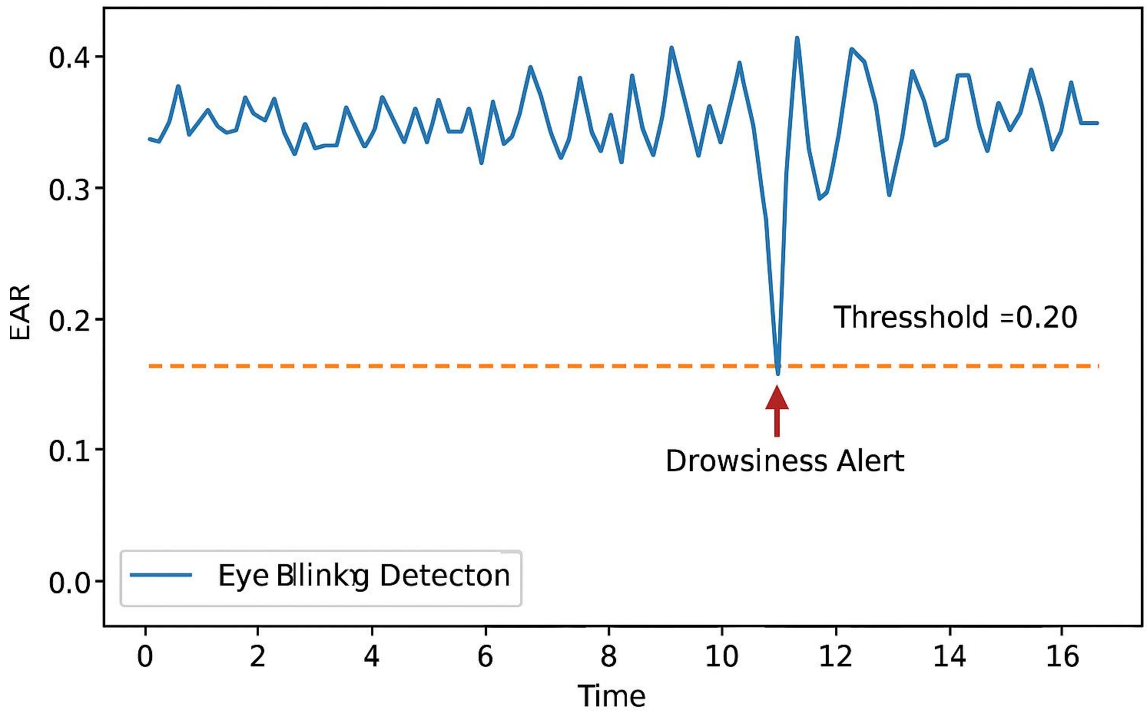


Figure No.5.1 *Eye Aspect Ratio (EAR) vs Time Graph Showing Drowsiness Detection Threshold and Alert*

5.8 Summary of Results

The results of this project demonstrate the effectiveness, scalability, and robustness of the system developed for monitoring last-mile delivery operations. Each module was rigorously tested under simulated operational conditions and contributed to the unified goal of ensuring accountability, transparency, and safety in field delivery activities.

The AI-based detection engine, Kaptaan AI, successfully identified and flagged suspicious delivery remarks based on behavioral event data and business rule thresholds. During controlled testing, the system maintained an overall accuracy rate of 92% in detecting potential delivery fraud, validating its logic and real-time processing capabilities.

Google Tag Manager (GTM) events were triggered and logged precisely across multiple app states and user actions. These events—including call duration, screen navigation, OTP input, and remark submission—were captured reliably and served as a real-time behavioral feed for Kaptaan AI to evaluate user compliance.

All backend RESTful APIs, which powered critical functions like order cancellation, KYC status updates, and rider attendance marking, performed reliably under normal and exception scenarios. API calls were tested using Postman and curl, and the system consistently returned appropriate status codes and maintained secure data transmission.

The KYC verification module, powered by HyperVerge, enabled seamless and accurate onboarding of delivery partners. It reduced manual verification overhead and ensured only legitimate and verified users could access delivery workflows. Rejected KYC cases were automatically flagged, and blacklisting was enforced via the system’s backend.

Metabase dashboards provided operational users and franchise managers with a centralized, real-time visualization platform. These dashboards displayed metrics such as rider performance trends, frequency of flagged delivery attempts, attendance logs, and KYC compliance summaries, significantly improving decision-making and task monitoring.

Finally, the Eye Blinking Detection system achieved its goal of accurately identifying signs of rider fatigue. In simulated conditions, it raised timely alerts and exhibited high precision in detecting drowsiness, contributing to the rider’s safety and performance.

In conclusion, the integrated platform—including mobile applications, backend services, AI-based monitoring, and visualization dashboards—proved to be a reliable, intelligent, and scalable system. It holds strong potential for real-world deployment in the logistics and delivery sector, paving the way for smarter, safer, and more accountable delivery operations.

Chapter 6: CONCLUSIONS & FUTURE SCOPE

6.1 Conclusions

This project presents a comprehensive, AI-supported, event-driven system designed to improve accountability, efficiency, and reliability in last-mile delivery operations. The core objective was to mitigate the issue of fake delivery remarks by empowering delivery partners with structured app workflows and enabling operational managers with intelligent monitoring tools.

Through the integration of mobile applications (Kaptaan App and Franchise Owner App), a rule-based AI engine (Kaptaan AI), KYC validation through HyperVerge, and backend APIs for workflow automation, the system successfully transforms the traditional manual reporting and review process into a real-time, data-driven solution.

Kaptaan AI effectively analyzed event-level delivery behavior using parameters such as call duration, GPS proximity, OTP verification, and rider interaction patterns. The system was able to flag false remarks and enforce restrictions or alerts in real time. Event tracking through Google Tag Manager (GTM) provided detailed logs of rider actions, enabling precise data collection without manual instrumentation.

The Flutter-based mobile interfaces provided a timeline-driven, intuitive user experience for delivery agents, while the Franchise Owner App ensured operational transparency and control over rider actions such as KYC validation, remark reviews, and blacklisting.

HyperVerge KYC integration allowed for secure and efficient identity verification, and the automated backend system ensured that riders without valid documentation were immediately restricted. RESTful APIs streamlined operations such as order cancellation, attendance updates, and delisting workflows. All operational metrics were visualized through Metabase dashboards, which allowed for real-time monitoring and analysis of system behavior.

The Eye Blinking Detection module added an innovative safety component to the system. By identifying fatigue through facial landmarks and eye aspect ratio, the system helped promote rider well-being and reduce risks related to drowsiness during deliveries.

Collectively, the system demonstrates a scalable, modular, and intelligent architecture for last-mile logistics that significantly improves operational reliability, reduces manual errors, and enhances data visibility for decision-makers.

6.2 Future Scope of Work

While the current implementation fulfi lls the primary objectives of the project, several areas can be expanded and enhanced in future development:

1. **Integration of Machine Learning for Adaptive AI Decisioning**
   Currently, Kaptaan AI uses a rule-based approach for flagging fake remarks. In future iterations, machine learning models such as decision trees or ensemble classifiers can be trained using historical rider behavior data to predict the likelihood of fraud or delivery failure. This would allow for personalized decisioning that evolves with usage patterns.
2. **Live Fatigue Monitoring Using Wearable or IoT Devices**
   The Eye Blinking Detection module is currently a standalone application. It can be integrated into wearable smart glasses or helmet-mounted devices to provide continuous fatigue monitoring on the field. Alerts can be pushed to both the rider and the Franchise Owner App in real time.
3. **Expansion to Voice-Based Verification and Feedback**
   To reduce dependency on manual input and app navigation, voice interaction can be introduced. Riders could verbally submit remarks, confirm delivery status, or receive alerts. Integration with speech-to-text APIs would improve usability, especially during hands-busy situations.
4. **Enhanced Fraud Pattern Analytics Across Nodes**
   Metabase dashboards can be extended to support cluster-based analysis of fraud patterns across different franchise nodes. By identifying high-risk zones or repeat offenders, proactive action plans can be designed at a regional or national scale.
5. **Rider Incentive and Penalty Engine**
   A performance-based scoring system can be added to incentivize good rider behavior and penalize repeat violations. This can be linked to app-level access, performance badges, or even salary adjustments.
6. **Integration with Real-Time Location APIs for Better Geofencing**
   Currently, location mismatches are evaluated using basic GPS data. Advanced geofencing APIs such as Google Maps Geolocation Services or Mapbox can be integrated for more accurate determination of rider proximity to delivery addresses.
7. **Automated Audit Reports and Escalation Workflows**
   The system can be further enhanced with automated email or Slack-based alerts for flagged events, creating an audit trail for operational managers. Weekly or monthly compliance reports can be generated automatically.

By incorporating these enhancements, the system can evolve into a full-fledged logistics management suite capable of addressing broader operational challenges across various industries including e-commerce, courier, healthcare, and food delivery.

REFERENCES

- "Controlling Fake Remarks – Notion Documentation v2.1," Notion, Available: https://sleet-bag-0a2.notion.site/Controlling-Fake-Remarks-v2-1-12d2772a3793803eb8bfc6c54fb22e9c

- "Fake DB Documentation," Notion, Available: https://sleet-bag-0a2.notion.site/0-Fake-DB-Documentation-1682772a379380259b98d90c538976f4

- "GTM - Controlling Fake Remarks Setup," Notion, Available: https://sleet-bag-0a2.notion.site/GTM-Controlling-Fake-Remarks-v2-1-12d2772a3793802fa590d7a2b8f38844

- "Quick Timeline Phase 1.1," Notion, Available: https://sleet-bag-0a2.notion.site/Quick-Timeline-Phase-1-1-18c2772a3793809085b8eec463a3a7b8

- Sanket Blitz, "Futwork Investigation – GitHub Repository," GitHub, Available: https://github.com/Sanket-blitz/Futwork_investigation

- Sanket Blitz, "Attendance System – GitHub Repository," GitHub, Available: https://github.com/Sanket-blitz/Attendance

- Sanket Blitz, "Eye Blinking Detection Model – GitHub Repository," GitHub, Available: https://github.com/Sanket-blitz/Eye_blinking_Model

- "HyperVerge – AI-Powered KYC Verification Platform," HyperVerge, Available: https://hyperverge.co/in/

- "Kaptaan and Franchise Owner App UIs," Notion/Project Screenshots – Internal Design System (linked within Notion References 1–4)

- "Postman API Testing and Sample Endpoints," Internal API Testing Logs and Payload Examples, Based on: http://grow-simplee-nlb-prod-a264c465571856f67.elb.ap-south-1.amazonaws.com

- Google Developers, "Google Tag Manager Documentation," Available: https://developers.google.com/tag-manager

- Metabase, "Open Source BI & Analytics Tool," Available: https://www.metabase.com/

- Dlib C++ Library – Face Landmark Detection, Available: http://dlib.net/

- OpenCV – Open Source Computer Vision Library, Available: https://opencv.org/