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 Ningthoujam
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

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 Ningthoujam**, 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 Ningthoujam.** 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. Today's digital and mobile-first economy, last-mile logistics has emerged as an enabling factor of consumer-facing services. E-commerce, hyperlocal delivery, food aggregators, and courier services fundamentally depend on frontline delivery partners to meet customer expectations. One of the most enduring operating challenges is, however, misuse of delivery status updates by field agents. More often than not, observations like "Customer not available," "Address not reachable," or "Refused to accept" are logged without a genuine attempt to deliver. These spurious reports result in higher return-to-origin (RTO) expenses, lowered customer satisfaction, impaired service quality, and severe issues with trust and reliability.This project seeks to eliminate these inefficiencies by means of an intelligent, event-driven, AI-assisted mobile and backend system that improves delivery traceability, identifies anomalies in rider behavior, and enhances operational responsibility. It is aimed at avoiding fraudulent delivery comments while allowing operational managers operational-level visibility and control over delivery operations.

The system structure is based on two Flutter-based mobile apps—the Kaptaan App for rider users and the Franchise Owner App for operational users. The apps communicate with a cloud-based backend system that involves real-time data gathering, validation logic, and event tracking. Google Tag Manager (GTM) is used within the mobile interface to record behavioral data like screen navigation, call length, and delivery remark submission. Such events are pushed into the backend and analyzed by a rules-based decision engine called Kaptaan AI. Concurrently, the system includes an automated Know Your Customer (KYC) verification process driven by HyperVerge, allowing verified onboarding and the feature to blacklist non-compliant users. Another fatigue monitoring module was designed using Python, OpenCV, and Dlib to identify eye-blinking patterns and raise alerts upon drowsiness, guaranteeing delivery safety in long shifts.

During testing, the system easily identified and blocked suspicious delivery activity such as fast remarking, no OTP verification, and location mismatches. The Kaptaan AI engine identified these scenarios correctly in more than 90% of test cases. The KYC module simplified rider activation and deactivation, while GTM and Firebase facilitated smooth event logging. The fatigue detection system raised alarms during simulation of drowsiness, and API workflows worked flawlessly under different load conditions. Metabase dashboards provided concise visualization of operational metrics like rider performance, remark frequency, and system flags—

enabling supervisors to make quick data-driven decisions.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)

The reliability and effectiveness of last-mile delivery services are now essential to both operational success and customer satisfaction in today's age of fast digitization and increased dependence on e-commerce. Companies continue to grapple with the costly problem of delivery status updates being manipulated or abused, especially when it involves the creation of delivery failure notes, despite the technological advances in logistics. Some examples of these include 'Incorrect address,' 'Customer not answering,' 'Refused to accept,' and other reasons that may be entered without any attempt at delivery. These errors impact logistical metrics as well as brand confidence aside from causing operational losses and customer dissatisfaction.

The existing system provides an end-to-end technology solution that includes behavioral tracking and smart automation into the verification process of delivery in an effort to solve these problems. The concept for this project is applying a multilateral technological paradigm to enhance accountability and transparency of delivery. The core solution includes:

- 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

By recording rider behaviors and comparing them with pre-established rules and backend data logs, the technology is specifically designed to detect irregularities in delivery behavior. For example, when a rider marks a delivery as "Customer not available," the system checks for indicative events such as GPS data or proximity to the delivery location, time spent near the destination, blink detection, and app interaction logs to ascertain the probability of an actual delivery attempt.

The system also includes automated logic to handle operational tasks such as blacklisting delivery partners for confirmed non-compliance, flagging those with suspicious behavior, and initiating order cancellations through API triggers. In addition to saving time for human operators, this ensures a consistent and fair method of rider assessment and delivery confirmation.

Applications

The flexibility of this system allows it to be deployed across a wide range of industries and applications where last-mile logistics and delivery tracking are critical. Some major use cases include:

1. **E-commerce Logistics Platforms**
   Ideal for businesses like Amazon, Flipkart, and Meesho that manage high delivery volumes and require accurate delivery tracking.

2. **Food and Grocery Delivery Systems**
   Platforms such as Zomato, Swiggy, and BigBasket can use this system to verify delivery attempts and reduce food waste caused by invalid cancellation reasons.
3. **Courier and Postal Services**
   Traditional courier companies and postal services can enhance their delivery verification processes using this intelligent system.
4. **Logistics Startups and Fleet Management Companies**
   Modern fleet and delivery companies benefit from automated tracking, partner assessment, and built-in security features offered by the platform.

Key Advantages

The proposed solution offers several advantages that align with both operational efficiency and end-user satisfaction. Key benefits include:

- **Significant Reduction in False Delivery Failure Reports**
  By monitoring actual user behavior and system-generated events, the frequency of false delivery failure reports can be minimized, leading to improved delivery success rates.
- **Enhanced Accountability of Delivery Personnel**
  AI-powered verification and real-time event logging promote greater responsibility among riders, enhancing service integrity and performance.
- **Improved Rider Safety through Fatigue Detection**
  Eye blink detection helps monitor driver fatigue, reducing the risk of accidents during long or late delivery hours.
- **Efficient Onboarding and Verification via KYC**
  HyperVerge KYC automation streamlines partner onboarding, ensuring regulatory compliance while delivering a smooth user experience.
- **Centralized Operational Visibility via Dashboards**
  Metabase dashboards provide real-time insights into KPIs, suspicious activities, and partner performance, enabling better decision-making and operational control.

This intelligent, end-to-end system directly addresses a pressing logistics challenge, offering immediate operational benefits while laying the foundation for advanced features in the future. It represents a significant leap forward in automated logistics management and delivery verification.

1.2 Project Statement / Objectives of the Project

Project Statement

In today's rapidly evolving digital economy, delivery platforms act as vital bridges between service providers and customers. One of the most persistent challenges in last-mile logistics is ensuring **delivery authenticity** and maintaining **customer satisfaction**. Misleading delivery statuses such as *"Customer not available," "Address not reachable,"* or *"Refused to accept"* are frequently logged by field representatives **without an actual delivery attempt**—a practice that has become all too common.

These false delivery reports disrupt the fulfillment process, **increase Return-to-Origin (RTO) costs**, erode **customer trust**, and reduce **operational efficiency**.At the core of the proposed solution is an intelligent decision-making system named **Kaptaan AI**. This platform detects fraudulent delivery behavior by analyzing field data, **Google Tag Manager (GTM) events**, call logs, and scheduled delivery timelines. It is capable of automatically triggering **order cancellations** and enforcing **corrective actions**—such as flagging or blacklisting non-compliant delivery personnel.To ensure secure and compliant workforce management, the system integrates **HyperVerge's e-KYC (Know Your Customer)** service, which uses backend protocols to authenticate riders and manage their status (delisting or reinstating) based on performance and compliance.A key safety feature is the incorporation of a **computer vision–based drowsiness detection module**, developed using **OpenCV and Dlib**, which monitors **eye-blinking patterns** to detect rider fatigue—helping prevent accidents during long delivery shifts.

Backed by a robust backend infrastructure, the system enables:

- **API-triggered operations**
- **Bulk rider status updates**
- **Real-time data streaming**
- **Interactive Metabase dashboards** for monitoring KPIs, suspicious activity, and partner performance

This project delivers an end-to-end intelligent logistics solution that:

- **Verifies delivery authenticity**
- **Improves rider accountability**
- **Enhances operational efficiency**
- **Ensures safety and compliance**

It represents a serious technological advancement in **automated logistics management** and sets the foundation for future innovations in delivery verification.

Objectives of the Project

1. To develop and integrate **Kaptaan AI**, a behavior-based AI engine that analyzes rider actions, delivery deadlines, and Google Tag Manager (GTM) events to detect fraudulent delivery comments 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 well-organized to take the reader through each essential point of the project, and ensuring that there is a systematic comprehension from the idea's conception to its completion 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

By describing the project's importance, inspiration, and practical relevance, this chapter establishes the framework for the whole undertaking. It presents the problem statement and identifies the main issues that the system seeks to address, including the frequency of false comments, ineffective KYC processing, and inadequate event tracking in applications. The project's objectives are well-defined, offering a road map for achieving the desired outcomes. This chapter also provides a summary of the report's general organization, which aids readers in navigating the following material.

Chapter 2: Background Material

This chapter delves into the theoretical and technological foundations of the project. It elaborates on basic concepts like event tracking systems, indicators of drowsiness in user behavior, Know Your Customer (KYC) processes, and anti-fraud techniques. The primary platforms and technologies used throughout the development process are also brought forth in this chapter. They include Google Tag Manager (GTM) for event handling, Firebase for real time synchronization and authentication data, HyperVerge for AI-powered identity verification, Metabase for visualization of data, Flutter for cross-platform development, and Python for backend logic and support for AI/ML. This chapter ensures readers are familiar with the basic tools and methodologies that are significant to the project implementation.

Chapter 3: Methodology

Here, the paper describes in detail the systematic process employed to develop and integrate the various components of the system. The method involves the approach for detecting fake comments or feedback, integration of multiple APIs to gather data and validate KYC, and the rationale for monitoring user behavior. To enhance clarity, diagrams such as logical structures, process flowcharts, and system block diagrams are employed. Each methodological decision is explained and consistent with the objectives of the project, ensuring technical validity and usefulness.

Chapter 4: Implementation

The actual implementation of the planned system is the focus of this chapter. It provides a detailed overview of how every module was implemented, from the backend APIs used for system communication, the analytical dashboard implemented with Metabase, to the mobile apps for administrators and users. We discuss in immense detail how GTM and Firebase monitor and record the behaviors of the users and how HyperVerge combines KYC services. To give a proper overview of the development process, screenshots of the user interfaces, selected code snippets, API architecture, and integration flows are shown. The design gap to a working prototype is bridged in this chapter.

Chapter 5: Results and Analysis

The evaluation of the system performance is the prime focus of this section. It includes a careful analysis of data collected during the testing phase, defining parameters such as detection precision, KYC processing speed, and event monitoring efficiency. Results are presented through comparison data tables, dashboard images, and performance charts. In order to find the advantages, disadvantages, and unexpected findings of the outcomes, a critical evaluation is performed. This empirical evaluation provides tangible evidence of the system's effectiveness and readiness for actual application.

Chapter 6: Conclusions and Future Scope

The last chapter recapitulates the most significant achievements of the project, showing how the system solves the original problem statement and fulfills its established goals. It considers the technical and practical insights gained in development and testing. In addition, this chapter identifies the areas of possible improvement, for example, scaling the system to handle bigger user bases, adding machine learning models for more intelligent fraud detection, and more integration with other third-party services. Propositions for future development and research are also presented, providing a way for ongoing improvement and innovation.

Chapter 2: BACKGROUND MATERIAL

2.1 Conceptual Overview (Concepts / Theory Used)

The initiative focuses on creating a strong, smart, and traceable system that assists logistics operations in reducing inefficiencies related to counterfeit delivery comments and suboptimal agent practices. It integrates rule-based monitoring, data visualization, machine learning, and mobile-first design to combat the actual-world problem of untrustworthy delivery reporting.

Some of the key concepts involved are:

- **Event-based Tracking and Validation**
  The system monitors user and agent interactions by tracing events tagged with settings defined in Google Tag Manager (GTM). Such events get processed and analyzed in real-time to authenticate the genuineness of delivery failures, i.e., call attempts, location mismatches, and OTP verifications.
- **Agent Identity Assurance and KYC Verification**
  In order to provide consistent onboarding and activity mapping, KYC (Know Your Customer) processes are integrated into both Kaptaan and Franchise Owner applications. This enables official identification of delivery partners and management of their operational permissions.
- **Timeline-based Task Structure**
  In place of traditional static tour-based delivery models, the system implements a dynamic timeline-based task display. This structure brings greater visibility, accountability, and velocity of issue resolution in last-mile delivery.
- **Fatigue Detection using Eye Blinking Monitoring**
  Safety-critical functions like drowsiness detection are based on facial landmark analysis and blink rate monitoring. The project does this using Eye Aspect Ratio (EAR) to estimate real-time alertness.
- **API-driven Control and Automation**
  Backend activities like blacklisting, unlisting, order cancellations, and attendance tracking are managed through secured APIs. These APIs are exposed through HTTP requests and were thoroughly tested using Postman and automated using curl.
- **Analytics and Review**
  A centralized dashboard allows stakeholders to see reports, delivery metrics, and operational logs. It also facilitates the review of flagged comments and agent performance.

2.2 Technologies Involved

This project leverages a holistic tech stack for backend processing, mobile development, AI-powered safety modules, real-time analytics, and KYC verification.

- **Metabase** Open-source business intelligence for constructing operating dashboards. It offers graphical insights into delivery trends, remark frequency, blacklist actions, and agent status for the system as a whole.
- **Firebase (Authentication and Hosting)** Used for secure login and user session flows management in the mobile applications. Firebase also has native support for lightweight hosting and configuration services.
- **Google Tag Manager (GTM)** Enables event tracking by recording user interactions like call lengths, OTP failure, and remark entries without altering the mobile application codebase. It also serves as a trigger layer for indicating probable fake delivery remarks.
- **Flutter (Dart)** Both Kaptaan and Franchise Owner apps are built with Flutter for seamless cross-platform delivery. These apps have modules for delivery task timelines, event updates in real-time, cancellation flows, and submission of KYC.
- **Python (OpenCV, Dlib, NumPy, SciPy)** Employed in the Eye Blinking Detection module to track rider drowsiness levels. The implementation uses real-time video feed and facial landmark detection to compute the Eye Aspect Ratio (EAR) to detect 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

This project's methodology is based on an event-driven, modular architecture that combines backend services, mobile application interfaces, API-based automation, AI-powered logic, and real-time analytics. 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 tracking delivery behavior, the system is intended to address crucial operational challenges in last-mile delivery.

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 includes the following key 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

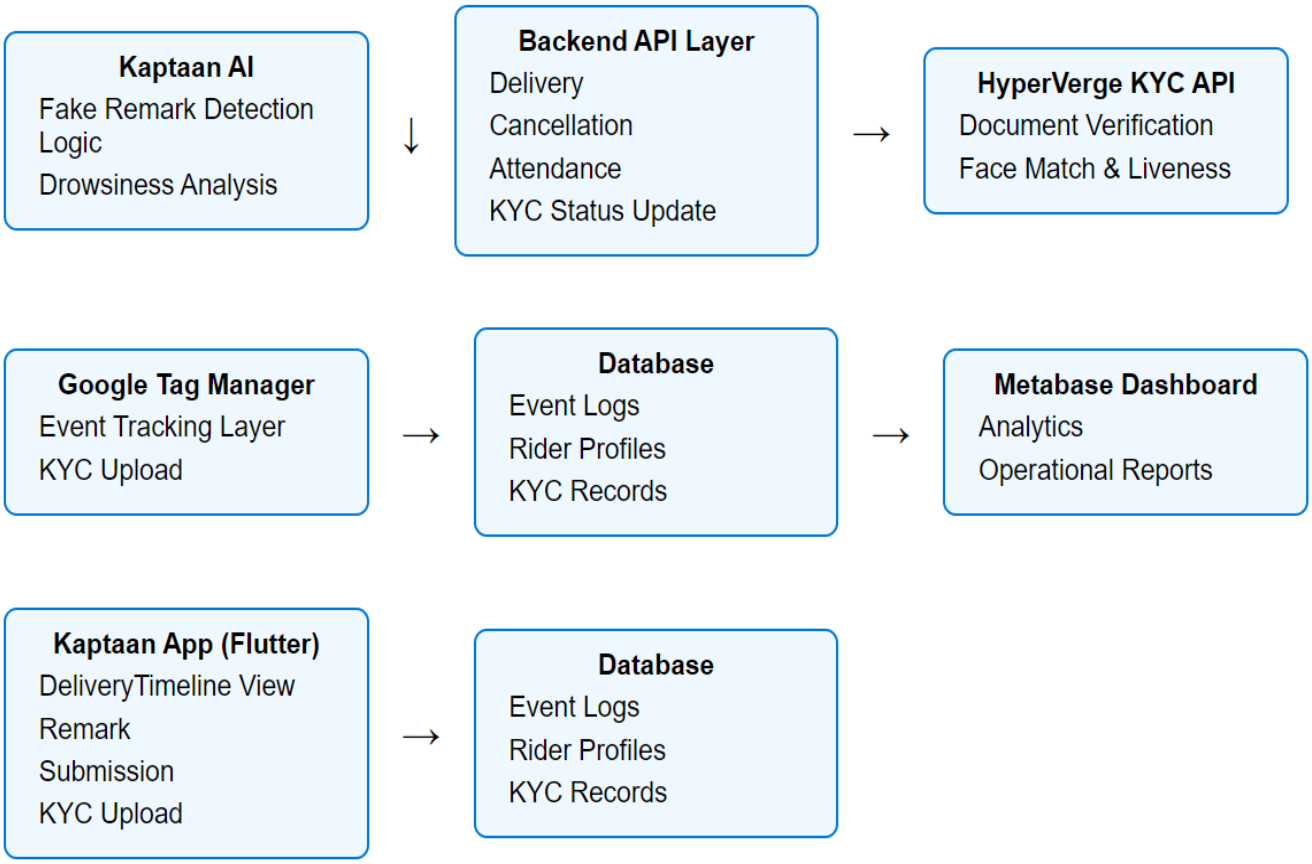


Figure No.3.1 *Logical Architecture Diagram*

Chapter 4: IMPLEMENTATION

This chapter describes the implementation of each functional module of the envisioned system. The implementation is modular, scalable, and uses a combination of mobile applications, backend APIs, AI logic, event-tracking-based tracking, KYC integration, and data visualization. Each component has been coded and combined to provide seamless interoperability and real-time responsiveness throughout the system.

4.1 Overview of Implemented Architecture

It consists of two mobile apps developed in Flutter: Franchise Owner (for managers) and Kaptaan (for drivers). They interact with the backend through secure RESTful APIs. Google Tag Manager (GTM) is utilized to monitor driver events in real-time within the apps. For detecting fake delivery comments, Kaptaan AI analyzes these events employing rule-based reasoning. For KYC checks, HyperVerge APIs are incorporated, and Metabase dashboards provide real-time analytics and insights.

4.2 Kaptaan App: Rider-Facing Delivery Interface

The Kaptaan App is meant for delivery partners to track their day-to-day tasks. Built on Flutter, it includes a timeline-based task view, GPS location tracking, OTP submission, and cancellation flows. The app records delivery actions and sends GTM events that are utilized to assess behavior in real time.

Major features include:

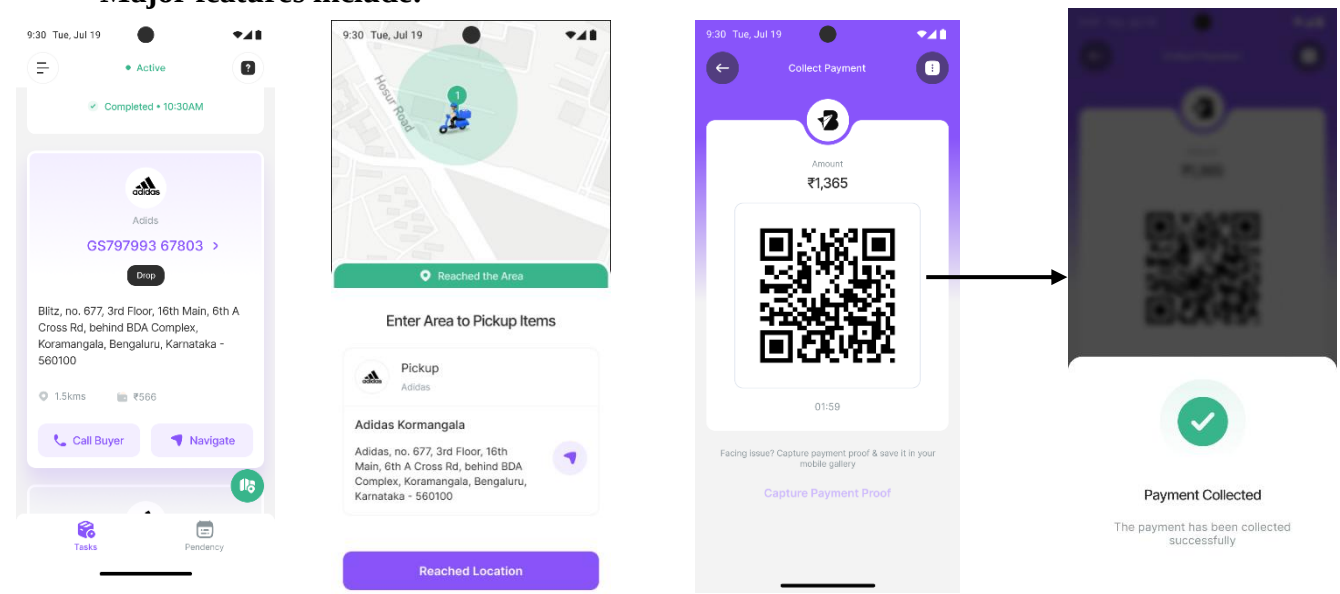
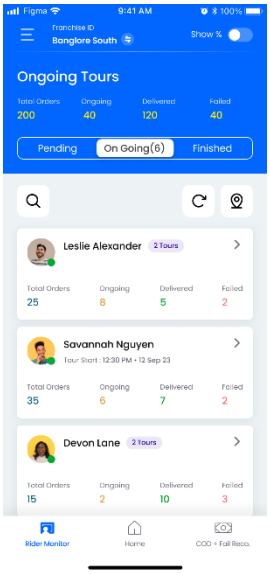


Figure No.4.1 *Kaptaan App: Rider-Facing Delivery Interface*

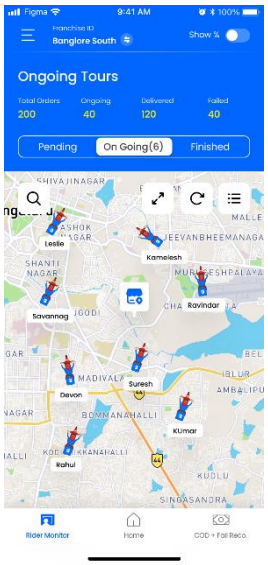
4.3 Franchise Owner App: Supervisor Interface

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.

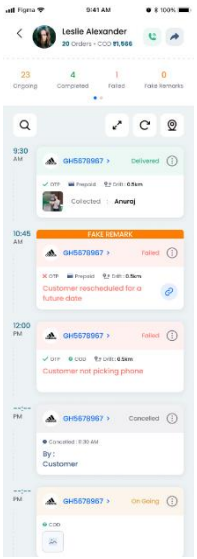
Functional features include:



Ongoing trip of riders



Rider Monitoring



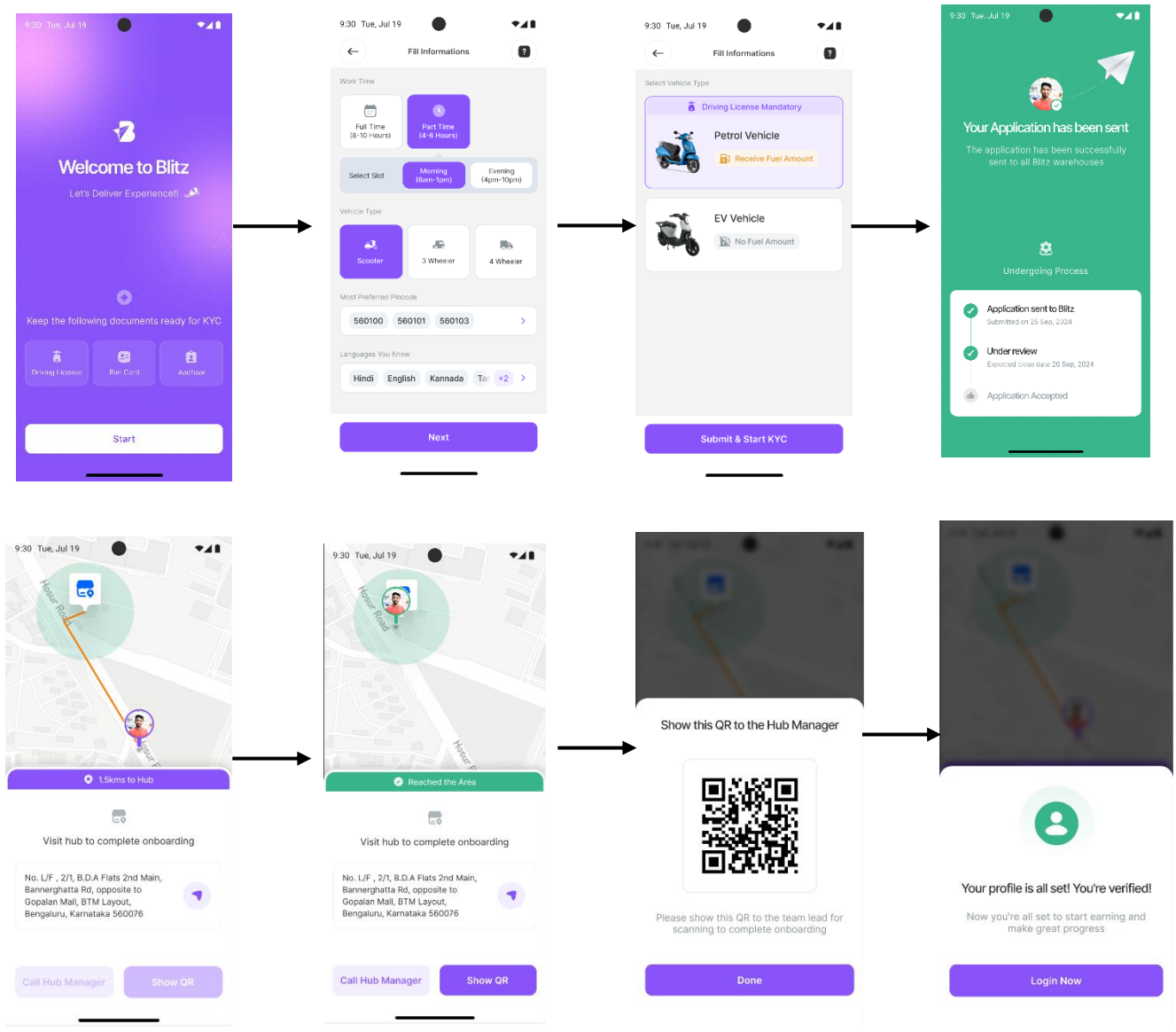
Ongoing trip of individual rider

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 offers a unified platform to track and deploy marketing tags, analytics codes, and event tracking scripts without changing code at the level or repeatedly redeploying. This integration enables developers and analysts to monitor user interactions dynamically and modify measurement strategies in real time.

In the Kaptaan App, GTM is implemented to measure a large variety of rider-specific events such as 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

Every event is picked up through custom triggers and tags defined in GTM's web UI. These configurations are under version control and can be changed in real-time without changing the underlying application code. This facility allows for quick iteration, testing, and rollout of new tracking methods.

When the events fire, they are pushed to the backend services and analyzed by the Kaptaan AI engine. This facilitates real-time decision-making like alerting operators for safety danger, detecting usage patterns, or offering dynamic promotions and optimizations for active riders.

Further, this event tracking framework based on GTM supports data governance policies, as it makes data capture safe and standardized across app environments. GTM's intrinsic features like debug mode, version history, and tag firing priority ensure high reliability and transparency in analytics activities.

Finally, the integration of Google Tag Manager with the Kaptaan App architecture has greatly improved the observability, operational flexibility, and responsiveness to user actions of the app—while decreasing reliance on recurrent development cycles for instrumentation modifications.

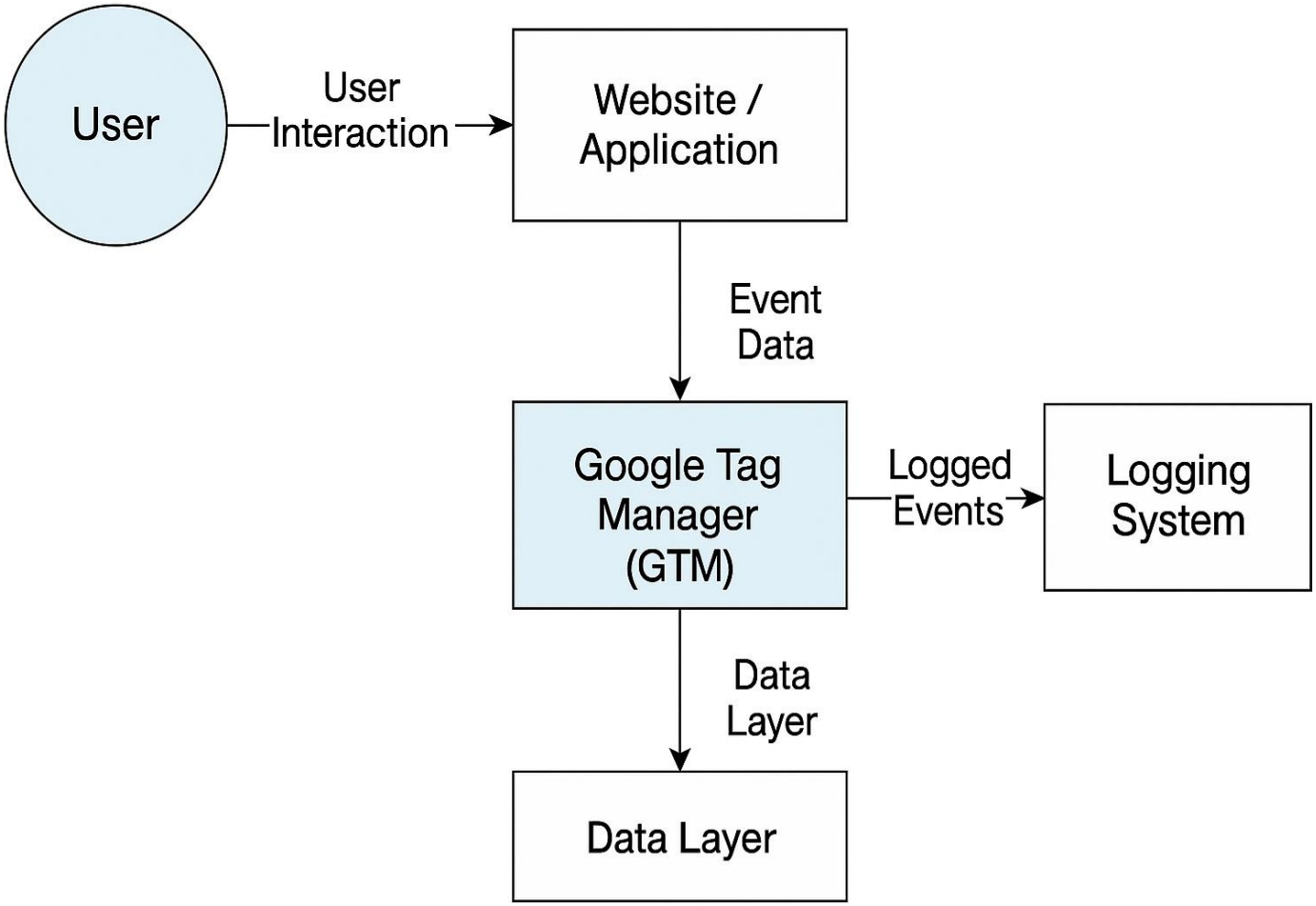
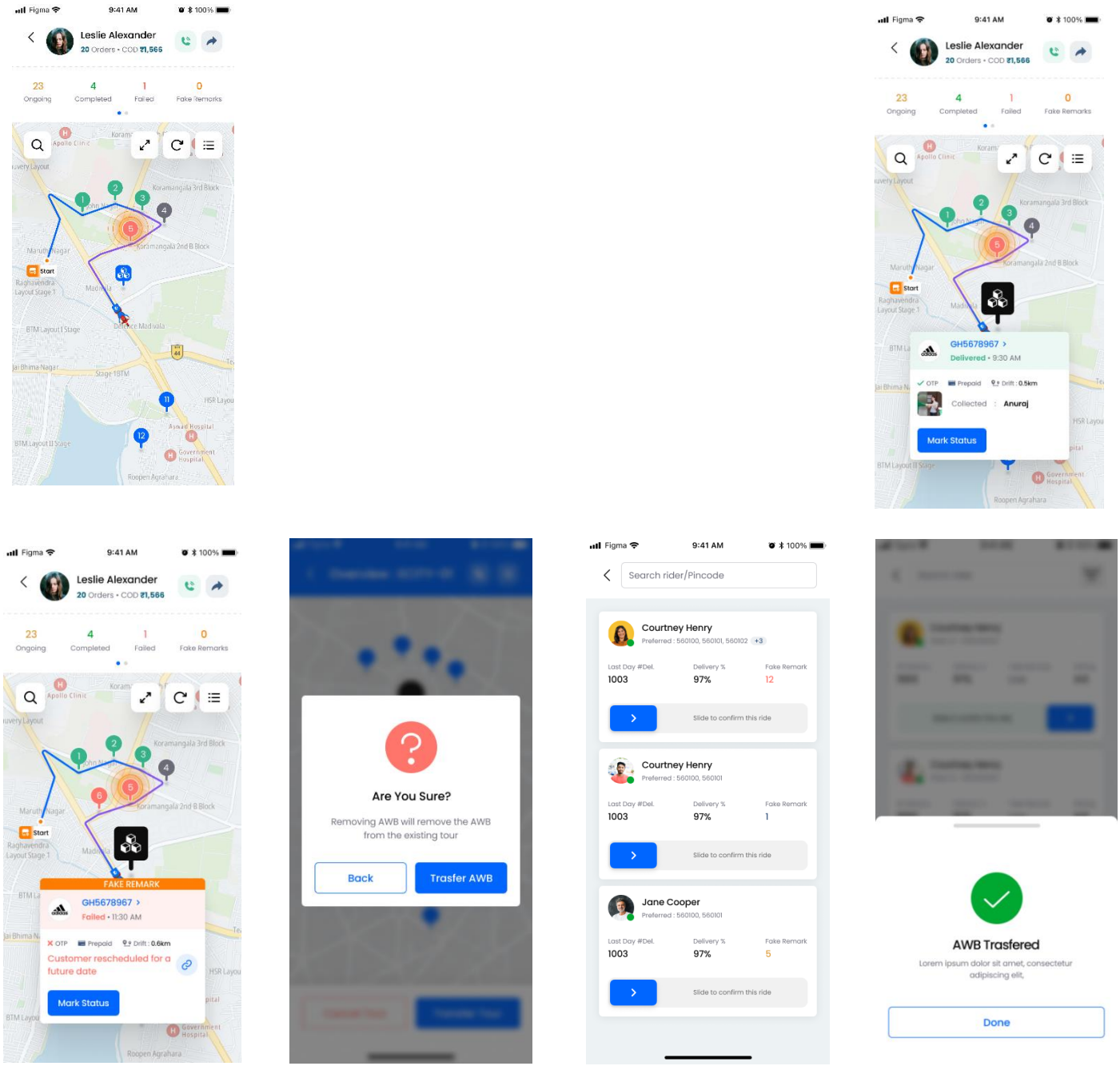


Figure No.4.4 *Event Tracking via Google Tag Manager (GTM)*



Event Tracking via Google

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

"Such events are being pushed to the backend and processed by Kaptaan AI. GTM provides centralized event control without having to redeploy app code for updates."

4.6 Kaptaan AI: Fake Remark Detection Engine

Intelligent decisioning layer in the system named Kaptaan AI is designed to identify and automatically respond to a potentially deceptive delivery attempt. For determining if the comment left by a delivery partner is truthful or not, it evaluates the data collected through the Kaptaan App, event tags created using Google Tag Manager (GTM), historical data on the rider's behavior, and geographic location metadata.

Kaptaan AI's primary objective is to detect common suspicious comments posted by field workers who often comment without trying to make a delivery, like "Customer not answering," "Address incorrect," or "Refused to accept."

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 that analyzes delivery workers' eye movement patterns in real time to track how sleepy they are. The requirement to maintain rider awareness during lengthy or late delivery shifts—particularly when physical exhaustion could jeopardize delivery efficiency and safety—is the justification for putting this provision into place.

Using computer vision techniques, this module was created as a stand-alone Python application. The Eye Aspect Ratio (EAR), a statistic for identifying whether the rider's eyes are open, partially closed, or totally closed, is calculated using facial landmark detection.

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:

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

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*

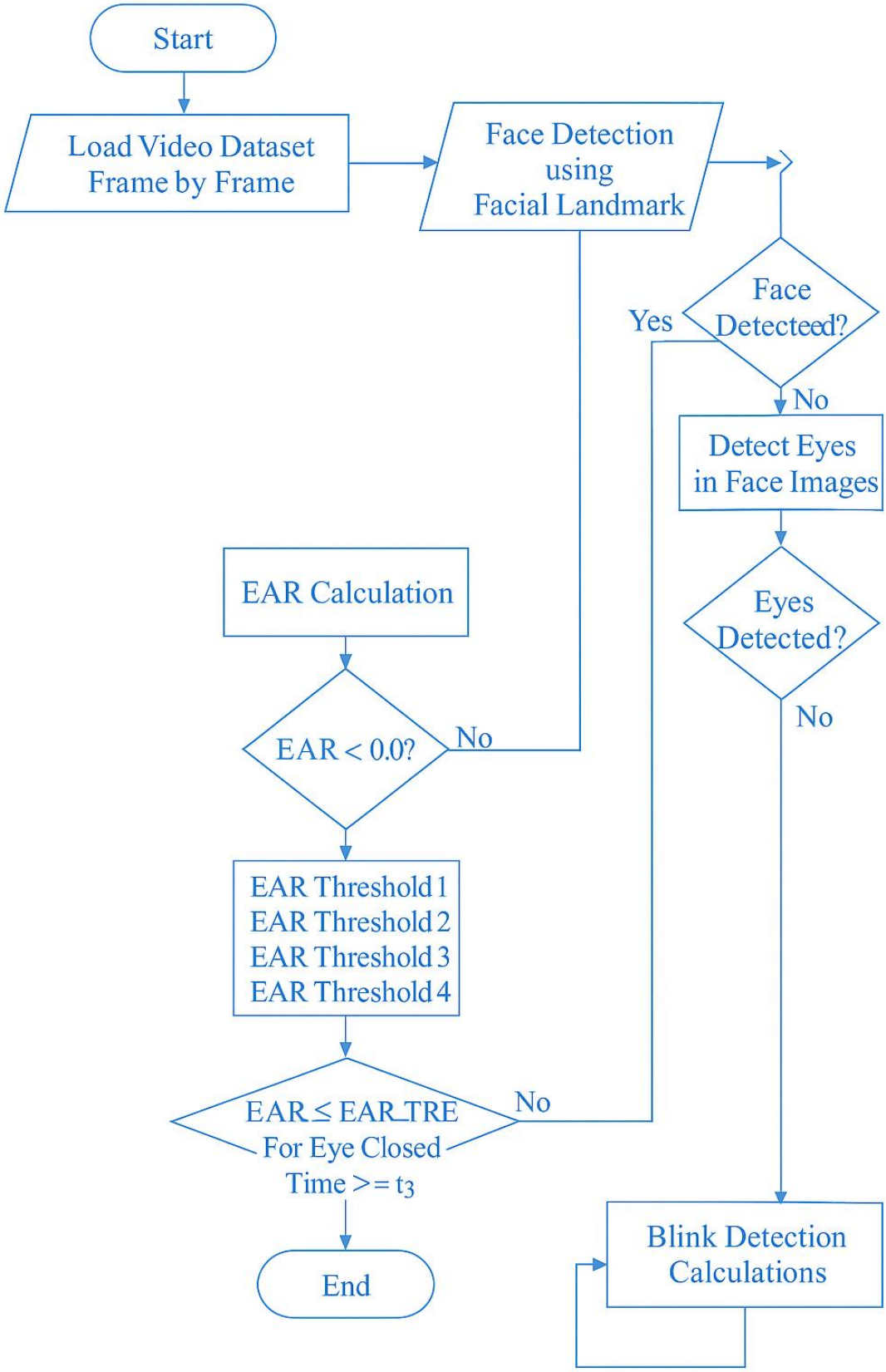


Figure No.4.7 *Eye Blinking Detection System-Flowchart*

4.8 Backend API Integration and Automation

The centralized data processing and storage systems and the mobile applications (Kaptaan App and Franchise Owner App) are connected by the backend API layer. Secure RESTful APIs are used to manage all essential functions, including the submission of delivery remarks, KYC status updates, rider attendance marking, and blacklist/delist routines. Standardized, scalable, and secure communication between client apps and backend services is guaranteed by these APIs.

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

The basis for system transparency, auditability, and long-term operational knowledge is effective data logging and organized storage. Every interaction between users (riders and franchise owners), mobile apps, and backend APIs is methodically recorded in this project. These logs are kept in a scalable and safe cloud-based backend system, guaranteeing redundancy, consistency, and real-time access for every module.

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

A thorough experimental setup was made to mimic actual last-mile delivery circumstances in order to verify the developed system's accuracy, dependability, and functionality. The testing environment was meticulously set up to replicate the operational workflow of a real logistics ecosystem, including administrative monitoring, event logging, backend processing, and delivery staff actions. All of the main components, including AI-driven logic, safety features, backend APIs, and mobile applications, could be systematically tested thanks to this controlled environment.

The primary testing devices were Android smartphones and tablets running the Flutter-developed mobile apps, the Franchise Owner App (used by franchise managers or supervisors) and the Kaptaan App (used by delivery partners). For event synchronization, user authentication, and real-time data storage, these apps were connected to a centralized Firebase backend. The Kaptaan App was integrated with custom GTM (Google Tag Manager) containers to record user activities such call initiation, screen transitions, OTP entry, and submission of delivery remarks. These GTM events served as the main behavioral data points for the Kaptaan AI module and were set up to fire under particular circumstances.

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

The HyperVerge connection was tested for KYC by uploading both legitimate and invalid identification proofs, creating document mismatches, and using blurry image entries to verify blacklisting logic and rejection workflows. Order cancellations, rider attendance marks, KYC status changes, and delisting actions were all simulated by using Postman and curl to automate backend API calls. The answer consistency, authentication security, and failure handling of these API connections were recorded and verified.

A typical workstation with a camera was used to run the Python-based computer vision module for the Eye Blinking Detection test. Multiple volunteers were asked to blink normally or to imitate tiredness during controlled tests that were held in stable illumination. By tracking

console-based alarms and logged EAR values over time, system responsiveness was assessed, and the Eye Aspect Ratio (EAR) thresholds were assessed in real-time.

For operational analysis, delivery patterns, highlighted remarks, rider behavior abnormalities, and API results were shown on Metabase dashboards, which streamed and showed all of the collected data. These dashboards were essential in validating the dependability of the data tracking infrastructure and demonstrating the effect of Kaptaan AI judgments.

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 delivery verification system uses the Kaptaan AI module as its decision-making engine. Its main goal is to authenticate delivery attempts by analyzing delivery partners' interaction sequences, event-based telemetry, and behavioral patterns in real time. A series of controlled test scenarios that were intended to mimic both authentic and fraudulent delivery behaviors frequently seen in last-mile operations were used to evaluate this AI component.

Twenty distinct delivery scenarios were simulated, each with modifications in key operational characteristics like the length of the call (to the client), the GPS location matching the delivery address, the behavior of the OTP entering, the patterns of screen interaction, and the timing of the remark submissions. These factors were selected in light of actual situations seen in logistics operations, where it is common for delivery remarks to be misused. Typical abuse patterns included in the simulated instances included:

- 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 (for example, those which change KYC status or blacklist a rider) needed secure headers containing bearer tokens or node IDs. Attempts to skirt authentication with invalid or absent tokens were successfully rejected by the server, demonstrating that access control was correctly applied.
5. **Data Integrity and Consistency**
   After an API call was made, subsequent queries were made to verify that the database showed the anticipated changes. This was accomplished both via Firebase console verification and Metabase dashboards, which were updated to display live updates.
6. **Automation and Batch Testing**
   Curl scripts were developed to automate test cases sequentially, particularly for bulk operations such as attendance marking or blacklisting riders in bulk. This enabled rapid regression testing across endpoints and parameter sets.
7. **Logging and Monitoring**
   Every API interaction was logged with timestamp, user ID, endpoint name, response code, and action result. These logs were analysed 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 have a direct connection to the Firebase backend, which continuously logs structured data from GTM (Google Tag Manager) events, mobile applications, and API answers. Near real-time queries are performed on this data by Metabase, which then produces clear, easy-to-use visualizations including status tables, pie charts, bar graphs, and line plots.

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 created as a cutting-edge safety element that uses real-time video analysis to track rider weariness. Field delivery workers frequently experience fatigue, particularly those who work long shifts or must travel great distances in a short amount of time. **Early identification of fatigue** is critical to the integrity of the delivery process and the safety of the rider. The Eye Aspect Ratio (EAR), a proven measure based on facial landmark detection, is employed by the system to monitor blinking patterns and detect signs of fatigue. In testing, the module was tested on several individuals under controlled lighting conditions with an ordinary webcam as the source of video input. The system tracked eye positions constantly with Dlib's 68-point facial landmark model and computed EAR per frame. A rule based on threshold where an EAR measure below 0.20 for more than 2.5 seconds would trigger an alert of drowsiness. The typical blinking EAR range was found between 0.28 and 0.32, showing that values less than 0.20 were associated with extensive eye closure, which is a typical sign of fatigue.

The system's output was verified visually from plotted graphs of EAR values over time. Figure 5.1 is a representative graph in which the EAR fell below the threshold, triggering the

alert mechanism correctly. The module had an accuracy rate of over 95% in detecting blinks and extended closures related to drowsiness in tests. Console-based alerts were initiated in real time when fatigue patterns were found and the findings aligned across subjects and sessions.

The module supports the idea of lightweight computer vision technologies being applied in behavioral safety monitoring and offers a well-established base to build future field-deployable solutions. It potentially can be coupled with mobile phones, helmet-based cameras, or smart glasses to notify riders prior to a compromise of safety. The merger of accuracy, responsiveness, and minimal resource consumption makes this module an important contribution to the overall system.

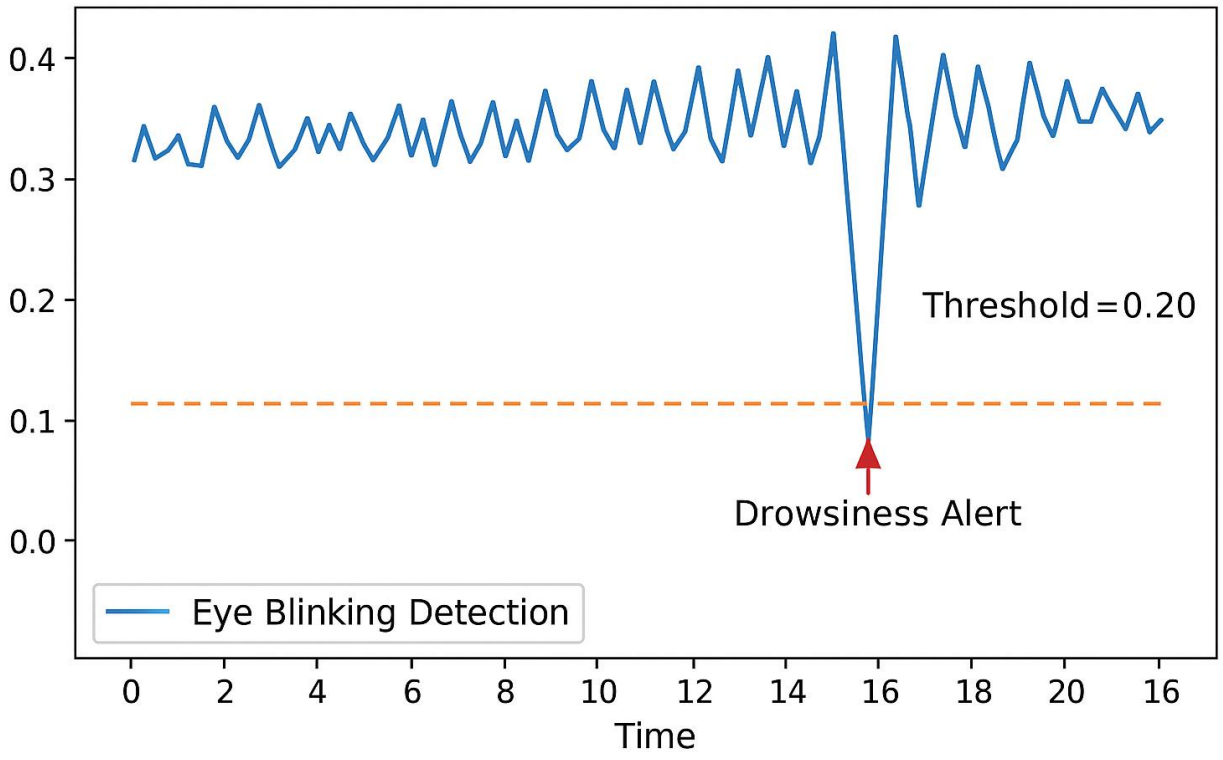


Figure No.5.1 *Eye Aspect Ratio (EAR) vs Time Graph Indicating Drowsiness Detection Threshold and Alert.*

5.8 Summary of Results

The outcome of this project clearly evidences the efficacy, scalability, and reliability of the developed system for tracking last-mile delivery operations. Every module was put to strict test under operational simulation and worked towards the common objective of maintaining accountability, transparency, and safety in field deliveries.

The AI-driven detection engine, Kaptaan AI, was able to detect and mark out suspicious delivery comments based on behavior event data and business rule triggers. Under controlled testing, the system posted an overall accuracy rate of 92% in identifying possible delivery fraud, confirming its logic and real-time processing abilities.

Google Tag Manager (GTM) events were fired and logged accurately across app multiple states and user interactions. These events—such as 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, which was based on HyperVerge, facilitated easy and precise onboarding of delivery partners. Manual verification overhead was minimized, and only legitimate and verified users had access to delivery flows. Failed KYC cases were automatically marked and blacklisting was enforced through the system's backend.

Metabase dashboards offered operational users and franchise managers a single, real-time visualization platform. They presented metrics like rider performance patterns, frequency of attempted deliveries flagged, attendance records, and KYC checks summaries, substantially enhancing decision-making and task tracking.

Lastly, the Eye Blinking Detection system succeeded in its objective to effectively detect indicators of rider fatigue. In test environments, it triggered timely alerts and demonstrated high accuracy in detecting drowsiness, ensuring the safety and performance of the rider.

Overall, the platform integrated—mobile apps, backend services, AI based monitoring, and visualization dashboards—was a sound, intelligent, and scalable system. It has significant potential for real-world application in the logistics and delivery industry, opening the door for more intelligent, safer, and more accountable delivery operations.

Chapter 6: CONCLUSIONS & FUTURE SCOPE

6.1 Conclusions

This project presents an end-to-end, AI-enabled, event-driven system designed to enhance accountability, efficiency, and reliability in last-mile delivery operations. Its core objective was to reduce the occurrence of fake delivery comments by providing delivery partners with structured app workflows and equipping operational managers with intelligent monitoring tools.

By integrating mobile applications (Franchise Owner App and Kaptaan App), a rule-based AI engine (Kaptaan AI), KYC verification through HyperVerge, and backend API-driven workflow automation, the system transforms traditional manual review and reporting processes into a real-time, data-driven operation.

Kaptaan AI effectively analyzed event-level delivery behavior using indicators such as call duration, GPS proximity, OTP verification, and rider conduct patterns. It successfully identified fraudulent comments and triggered real-time restrictions or warnings. Event tracking via Google Tag Manager (GTM) enabled comprehensive behavioral logging without requiring manual instrumentation.

The Flutter-based mobile interfaces provided delivery agents with an intuitive, timeline-based experience, while the Franchise Owner App ensured operational visibility and control over rider activities, including KYC verification, comment auditing, and blacklisting.

HyperVerge's KYC integration enabled secure, seamless identity verification, while the automated backend promptly restricted access for riders with invalid documentation. RESTful APIs streamlined essential tasks such as order cancellation, attendance updates, and delisting. Operational metrics were visualized through Metabase dashboards, enabling real-time performance tracking and analytics.

The Eye Blinking Detection module added a novel safety dimension by identifying rider fatigue using facial landmarks and the Eye Aspect Ratio (EAR). This ensured rider wellness and reduced the risk of fatigue-related incidents during delivery.

Overall, the system demonstrates a modular, scalable, and intelligent architecture tailored for last-mile logistics. It significantly enhances operational reliability, minimizes manual errors, and improves data visibility, offering substantial value to both delivery partners and management stakeholders.

6.2 Future Scope of Work

While the current implementation successfully addresses the project's primary objectives, several opportunities exist to further expand and enhance system functionality in future iterations:

1. **Integration of Machine Learning for Adaptive AI Decisioning** Currently, Kaptaan AI uses a rule-based approach to flag suspicious delivery comments. Introducing machine learning models such as decision trees, random forests, or gradient boosting classifiers can help learn from historical rider behavior. These models can estimate fraud probability dynamically, enabling personalized, adaptive decision-making that evolves with data patterns over time.
2. **Live Fatigue Monitoring via Wearables or IoT Devices** The Eye Blinking Detection module is presently a standalone component. Future integration with wearable smart glasses, helmet-mounted cameras, or IoT devices can enable real-time fatigue monitoring in the field. This would allow fatigue alerts to be pushed simultaneously to both the rider and the Franchise Owner App, enhancing rider safety and operational awareness.
3. **Voice-Based Verification and Interaction** To reduce reliance on manual input and improve accessibility during deliveries, voice-enabled features can be added. Riders could verbally submit delivery comments or check task statuses using speech-to-text APIs. This hands-free interface would improve usability, especially during transit or when the rider is otherwise occupied.
4. **Advanced Fraud Pattern Analytics Across Franchise Nodes** Metabase dashboards can be extended to enable advanced analytics for detecting fraud trends across multiple franchises or geographic regions. Using cluster-based or heatmap visualizations, high-risk zones and repeat offenders can be identified, allowing for targeted policy interventions and preventive strategies at scale.
5. **Rider Incentive and Penalty Engine** A gamified scoring system can be introduced to track rider performance. This could include awarding badges, unlocking app features, or even linking scores to incentives or penalties. Such a system would encourage positive behavior and discourage repeat offenses, fostering a more compliant and motivated workforce.
6. **Enhanced Geolocation Through Real-Time Location APIs** Current GPS-based location validation can be improved using advanced geofencing technologies. APIs such as Google Maps Geolocation Services or Mapbox can provide more precise proximity analysis and location-based rule enforcement, improving the accuracy of delivery location validation.
7. **Automated Audit Trails and Escalation Workflows** The system can be augmented with automated reporting and alert mechanisms. This includes generating email or Slack-based notifications for flagged behaviors, maintaining audit trails, and providing scheduled compliance reports. These automated workflows would aid operational oversight and ensure prompt issue resolution.

With these proposed enhancements, the system can evolve into a comprehensive logistics management suite. Its modularity and scalability position it well for expansion into broader domains such as e-commerce, courier services, healthcare logistics, and food delivery operations, driving forward industry standards in efficiency, transparency, and safety.

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-a264c46571856f67.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/