ANDROID PROJECT REPORT

On

CIVIC EYE

CIVIC ENGAGEMENT AND PUBLIC SAFETY

Submitted in partial fulfilment of the requirements for the award of degree

Of

Bachelor Of Technology

In

Information Technology

Of

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Department of Information Technology

School Of Engineering

Kochi - 682022

April 2025

ANDROID PROJECT REPORT

CIVIC EYE :

CIVIC ENGAGEMENT AND PUBLIC SAFETY

Submitted by:

SREEHARI S KUMAR

VAISHNAV K.V

SNEHA SIVAKUMAR

SAYANDH PUTHIYA PURAYIL

Under the guidance
Of

Dr. Renumol V.G & Nismi Mol E A

Professor Assistant Professor

DIVISION OF IT

SCHOOL OF ENGINEERING

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

KOCHI - 68022

COCHIN UNIVERSITY OF
SCIENCE AND TECHNOLOGY

ACKNOWLEDGEMENT

We extend our sincere gratitude to all those who have supported and guided us throughout the development of our Android project.

First and foremost, we express our deepest appreciation to Dr. Renumol V.G, whose invaluable insights and encouragement have been instrumental in shaping our project. Her expertise and guidance have greatly contributed to our learning experience.

We are also immensely grateful to Asst. Prof. Nismi Mol E. A for her continuous support and constructive feedback, which have helped us refine our work and overcome challenges.

Additionally, we would like to thank our institution, faculty members, and peers for their encouragement and assistance throughout this project. Their motivation and advice have played a crucial role in our success.

Finally, we extend our gratitude to our friends and family for their unwavering support and patience during this journey

INDEX

Chapter 1: Introduction

Project Overview 1
Market Problem Statement 1
Development Methodology 1
Key Objectives 1

Chapter 2: Literature Review and Problem Statement

2.1 Literature Review 3
2.1.1 Lack of Real-Time Citizen Participation 3
2.1.2 Absence of Structured Evidence Submission 3
2.1.3 Lack of Transparency in Report Tracking 4
2.1.4 Low Motivation for Citizens to Report Violations 4
2.1.5 Data Security and Privacy Concerns 4
2.1.6 Proposed Solution: CIVIC EYE 4
2.2 Problem Statement 4
2.2.1 Lack of Secure and Incentivized Reporting Platforms 5
2.2.2 Limited Public-Law Enforcement Collaboration 5
2.2.3 Inefficient Report Tracking and Categorization 5
2.2.4 Lack of Multimedia Evidence and Metadata Extraction 5
2.2.5 Risk of System Misuse 5
2.2.6 Proposed Solution: CIVIC EYE 6

Chapter 3: Requirement Analysis

3.1 Requirement Gathering and Analysis 7
3.2 Software Requirements Specification (SRS) – IEEE Format 7
3.3 Overall Description 8
3.4 Specific Requirements 9
Functional Requirements (Table 3.1) 9
Non-Functional Requirements (Table 3.2) 10
3.5 Technical Stack 11

Chapter 4: Architecture and Design

4.1 System Architecture Diagram 12
4.2 Use Case Diagram 14
4.3 Entity-Relationship (ER) Diagram 15

Chapter 5: Implementation

5.1 Implementation Overview 18
5.2 System Screenshots 19
5.2.1 User Interface (Fig 5.1, Fig 5.2, Fig 5.3, Fig 5.4) 19
5.2.2 Officer Interface (Fig 5.5, Fig 5.6, Fig 5.7) 21

Chapter 6: Testing

6.1 Testing Methodology 23
6.2 Major Test Cases and Results (Table 6.1) 24
6.2.1 Submit Report Violations Testing 25
6.2.2 Fetch Metadata Testing 25

Chapter 7: Conclusion

7.1 Merits 26
7.2 Demerits 26
7.3 Future Work 27

References

Appwrite Documentation 28
Challan API Guide 28
Flutter Official Documentation 28
EXIF Metadata Extraction Libraries 29

Appendices

Appendix A: User Authentication Code Snippets 30
Appendix B: Incident Reporting Code Snippets 32
Appendix C: UI Components Code Snippets 34

CHAPTER 1

INTRODUCTION

The CIVIC EYE: Civic Engagement and Public Safety is an innovative Android application designed to provide citizens with a **secure, efficient, and rewarding platform for reporting public transportation law violations**. Unlike existing methods that often lack real-time citizen participation and transparency, this application introduces a structured and effective way for the public to contribute to maintaining traffic discipline. By integrating multimedia evidence submission, data input capabilities, and a thorough verification process managed by law enforcement officers, the application ensures that violations are properly documented and addressed.

One of the core objectives of the application is to enhance public participation in traffic law enforcement while maintaining robust security and privacy measures. With an **encrypted, email-based login system**, user data is safeguarded against unauthorized access. The application also facilitates real-time violation reporting by allowing users to submit evidence in the form of images and text. Additionally, it extracts metadata, such as the **date and time from images (EXIF data)**, ensuring that complaints are accurately timestamped.

To streamline the reporting process, the system categorizes reports based on their status—pending, verified, or rejected—allowing users to easily track their submissions. Reports are further classified into specific categories such as road damage, sanitation issues, and public safety concerns, ensuring better organization and more efficient resolution of complaints. The user dashboard provides a structured view of report history, reward earnings, and notifications, allowing citizens to engage effectively with the system.

The **role of law enforcement** is critical in ensuring the success of the application. A dedicated verification system enables police officers to review, verify, and approve or reject submitted reports. Upon approval, valid reports contribute to an incentivized reward

system, where users receive rewards funded by the fines collected from the reported violations.

Furthermore, the application incorporates a **Challan Generation API**, which automates the issuance of digital challans for specific infractions, expediting the process of law enforcement and ensuring that penalties are effectively enforced.

By integrating these features, **CIVIC EYE** serves as a comprehensive solution that promotes accountability, enhances citizen engagement, and improves overall traffic law enforcement. The system not only empowers individuals to take an active role in maintaining public order but also ensures that their contributions are recognized and rewarded.

CHAPTER 2

LITERATURE REVIEW AND PROBLEM STATEMENT

This chapter explores the existing challenges in traffic violation reporting systems and the need for a structured, transparent, and citizen-friendly approach.

2.1 Literature Review

The **Literature Review** analyzes the limitations of traditional reporting methods, including the lack of real-time participation, absence of structured evidence submission, low public engagement, and inadequate data security. Various studies emphasize the importance of digital platforms that enhance user involvement, provide multimedia-based evidence submission, and ensure secure data handling.

2.1.1 Lack of Real-Time Citizen Participation

Traditional reporting methods, such as hotline complaints, written reports, and in-person visits to police stations, are inefficient and discourage citizen participation. Studies indicate that a digital platform with real-time reporting and verification significantly improves public engagement by allowing users to submit complaints instantly.

2.1.2 Absence of Structured Evidence Submission

Most traffic violation reporting systems lack the capability to collect structured multimedia evidence, such as geotagged images and videos. Without proper evidence, law enforcement officers struggle to verify complaints, leading to potential false reports and wasted resources. Research highlights that incorporating automated metadata extraction (time, date, and location) enhances the credibility and reliability of submitted reports.

2.1.3 Lack of Transparency in Report Tracking

Citizens often lose confidence in reporting violations due to the absence of real-time status updates. Studies show that categorizing reports into different statuses (e.g., Pending, Verified, Rejected) builds trust and encourages participation.

2.1.4 Low Motivation for Citizens to Report Violations

Many existing reporting systems lack incentives, leading to low engagement and underreporting of traffic violations. Research suggests that reward-based reporting, where citizens receive incentives for verified reports, increases active participation and accuracy.

2.1.5 Data Security and Privacy Concerns

Existing systems do not provide secure login, encrypted storage, or strict authentication, making personal user data vulnerable to breaches. Studies emphasize the need for encrypted reporting mechanisms and strict moderation policies to ensure user privacy and prevent misuse.

2.1.6 Proposed Solution: CIVIC EYE

To address these challenges, the CIVIC EYE application integrates real-time violation reporting, allowing users to submit complaints instantly. It includes multimedia evidence submission, supporting images and videos for better verification, and automated metadata extraction to ensure accurate timestamping of reports. Categorized report tracking enables users to monitor complaint progress, while a reward-based incentive system motivates citizens to report violations. Enhanced data security is ensured through encrypted storage and user authentication.

2.2 Problem Statement

The **Problem Statement** highlights the inefficiencies in current traffic violation reporting systems, such as the absence of incentivized participation, weak collaboration between citizens and law

enforcement, and ineffective tracking mechanisms. To address these issues, **CIVIC EYE** is proposed as a comprehensive solution, integrating real-time reporting, structured evidence submission, reward-based participation, and automated enforcement to improve public safety and traffic law adherence.

2.2.1 Lack of Secure and Incentivized Reporting Platforms

Many citizens refrain from reporting traffic violations due to concerns about privacy, lack of follow-up, or absence of tangible rewards. This results in underreporting of traffic offenses, reducing the overall effectiveness of traffic law enforcement.

2.2.2 Limited Public-Law Enforcement Collaboration

Existing systems lack a structured mechanism for real-time engagement, causing a disconnect between submitted reports and police action. Reports often get lost, delayed, or ignored due to inefficient communication channels.

2.2.3 Inefficient Report Tracking and Categorization

Once a complaint is submitted, users rarely receive updates about its status. The absence of structured complaint tracking discourages citizens from further participation.

2.2.4 Lack of Multimedia Evidence and Metadata Extraction

Many platforms do not support image/video submission or automatic location tagging, making it difficult for officers to verify complaints. Reports without proper evidence often result in delays or incorrect enforcement actions.

2.2.5 Risk of System Misuse

Without proper user moderation and verification, some individuals submit false or irrelevant complaints, wasting police resources. Research suggests that identity verification and moderation policies help maintain system credibility.

2.2.6 Proposed Solution: CIVIC EYE

To overcome these problems, the CIVIC EYE application provides a secure and incentivized reporting mechanism that encourages citizen participation. It integrates multimedia-based complaint submission, enabling users to provide strong evidence, and real-time tracking to allow users to monitor the progress of their complaints. Automated metadata extraction ensures that submitted reports are accurate, while user moderation and verification prevent false reports and maintain system integrity. Additionally, automated challan generation ensures fast processing and enforcement of penalties.

By implementing these features, CIVIC EYE bridges the gap between citizens and law enforcement, creating a reliable, transparent, and efficient system for traffic violation reporting.

CHAPTER 3
REQUIREMENT ANALYSIS

The Civic Eye project represents a systematic effort to redefine civic engagement in public safety through technology. This chapter delineates the foundational requirements of the system, derived from rigorous stakeholder consultations and technical feasibility studies. By adhering to the IEEE SRS format, it establishes a structured framework for development while addressing the unique challenges of violation reporting, evidence verification, and incentive management. The analysis synthesizes user needs, operational constraints, and regulatory considerations into a cohesive blueprint for implementation.

3.1 Requirement Gathering and Analysis

The requirement elicitation process engaged diverse stakeholder groups, including municipal authorities, traffic police departments, and community representatives, through interviews and focus group discussions. These interactions revealed critical shortcomings in existing reporting mechanisms, such as delayed response times, opaque verification processes, and the absence of citizen motivation structures. Concurrent technical assessments identified key dependencies, including the need for robust metadata handling in multimedia submissions and seamless integration with legacy law enforcement systems. The gathered insights were systematically categorized into functional capabilities, such as real-time status tracking and automated challan generation, and qualitative attributes like data security and interface intuitiveness. This phase culminated in a prioritized feature set that balances innovation with practical deployability.

3.2 Software Requirements Specification (SRS) IEEE Format

The SRS document serves as the contractual and technical anchor for the Civic Eye system development. Its structure aligns with IEEE 830-1998 standards to ensure comprehensive coverage of system capabilities and constraints. The purpose statement articulates the

dual objectives of empowering citizens through participatory safety mechanisms while streamlining enforcement workflows for authorities.

Scope definition carefully bounds the system responsibilities, explicitly including multimedia evidence management and reward distribution while excluding ancillary functions like direct fine collection or judicial appeal processing. This delineation prevents scope creep during development.

References to authoritative sources provide implementation guidance, notably the Appwrite documentation for backend services, Kerala Traffic Penalty Guidelines for challan structuring, and ISO 27001 standards for data security protocols. These references ensure the system meets technical and regulatory expectations.

3.3 Overall Description

The Civic Eye system functions as an integrated platform bridging civilian reporters and enforcement authorities through structured digital workflows. At its core, the system ingests citizen-submitted violation reports enriched with evidentiary media, applies automated metadata extraction for spatiotemporal validation, and routes verified cases to authorized officers for disposition.

Three distinct user classes interact with the system, each with specialized interfaces and permissions. Citizen users leverage mobile interfaces for report submission and status monitoring, while police officers utilize web-based dashboards for evidentiary review and disposition management. System administrators maintain reference data and reward parameters through dedicated configuration modules.

The operational environment demands Android 10+ compatibility for mobile clients, with backend services hosted on Appwrite-managed cloud infrastructure. Critical external dependencies include stable connectivity for real-time operations and availability of government-approved vehicle registration databases for violator identification.

3.4 Specific Requirements

Table 3.1: Functional Requirements

ID	Requirement	Description
FR1	User Registration & Authentication	Users must be able to sign up and log in using an email based system to ensure security.
FR2	Complaint Submission	Users should be able to submit complaints with images, videos, a brief description
FR3	Multimedia Evidence Upload	Users can attach images and videos as evidence to support their complaints
FR4	EXIF Metadata Extraction	The system extracts date and time from images (if available) to ensure complaint accuracy.
FR5	Category-Based Classification	Complaints are categorized into different sections (e.g., road damage, sanitation, public safety) for efficient processing.
FR6	Complaint Status Tracking	Users can check the real-time status of their complaints (e.g., Pending, Under Review, Resolved).
FR7	Reward System	Users receive rewards upon approval of valid complaints, encouraging active participation.
FR8	PoliceDashboard	Police have access to a dashboard where they can review, verify, and manage complaints.
FR9	Automated Challan Generation	When a violation is confirmed, a digital challan is automatically generated and sent to the responsible party.

Table 3.2: Non-Functional Requirements

ID	Requirement	Description
NFR1	Performance	Process API requests within two seconds to ensure quick response times
NFR2	Scalability	Efficiently handle high concurrent user loads to maintain performance and stability under peak usage.
NFR3	Usability	Ensure the UI is user-friendly and visually appealing, with intuitive navigation
NFR4	Security	Implement secure authentication protocols to protect user data and prevent unauthorized access.
NFR5	Data Integrity	Maintain database consistency to prevent data loss or corruption.

The Civic Eye system is engineered to facilitate real-time violation reporting, automate evidence verification through metadata extraction, and streamline law enforcement workflows with digital challan generation. Beyond these core functions, the platform incentivizes civic participation through a transparent reward system while maintaining rigorous data security standards.

To meet operational demands, the system must fulfill critical non-functional requirements. Performance benchmarks include processing report submissions within three seconds under typical loads and supporting concurrent access for up to 5,000 active users during peak incidents. Data integrity is ensured through end-to-end encryption, with compliance to GDPR and local data protection regulations governing user information storage and processing.

The system architecture comprises three primary interfaces: a Flutter-based mobile application for citizen reporters, a responsive web dashboard for police verification, and an administrative portal for system configuration. Hardware compatibility spans Android 10+ and

iOS 13+ devices, with backend services hosted on cloud infrastructure for scalability. External integrations include government vehicle registration APIs for violator identification and payment gateways for reward disbursements. A stable internet connection is mandatory for real-time operations, though offline report queuing is planned for future iterations.

This Software Requirements Specification crystallizes the technical and functional blueprint for the Civic Eye platform. By unifying multimedia evidence collection, automated verification workflows, and incentivized public participation, the project aspires to transform urban safety enforcement through technology. The subsequent chapters will detail the system's architectural implementation, validation methodologies, and deployment strategies to realize this vision.

3.5 Technical Stack

The Civic Eye platform leverages a modern technology stack designed for scalability, security, and cross-platform compatibility. The frontend is developed using Flutter with the Dart programming language.

For backend services, the system utilizes Appwrite as a comprehensive solution, integrating authentication, database management, and file storage into a unified infrastructure. The Appwrite database serves as the primary data repository, offering real-time synchronization and robust querying capabilities to support the application's dynamic reporting and verification workflows.

A custom API handles digital challan generation, ensuring compliance with legal requirements while automating the penalty issuance process. This API interfaces directly with law enforcement systems to validate violations and produce standardized challan documents.

To enhance evidentiary reliability, the platform incorporates an EXIF parsing library for metadata extraction. This library automatically retrieves timestamps and geolocation data from uploaded images, reducing manual entry errors and providing verifiable timestamps for reported violations.

CHAPTER 4

ARCHITECTURE AND DESIGN

This chapter provides an overview of the system architecture and design of the **Civic Eye** application. The system is structured to facilitate efficient civic issue reporting by integrating a mobile application, a backend powered by Appwrite, and an authority dashboard. It ensures secure authentication, structured complaint processing, and a reward-based reporting mechanism. The key design diagrams include the **system architecture diagram, the use case diagram, and the entity-relationship (ER) diagram**, each of which is explained in detail below.

4.1 System Architecture Diagram

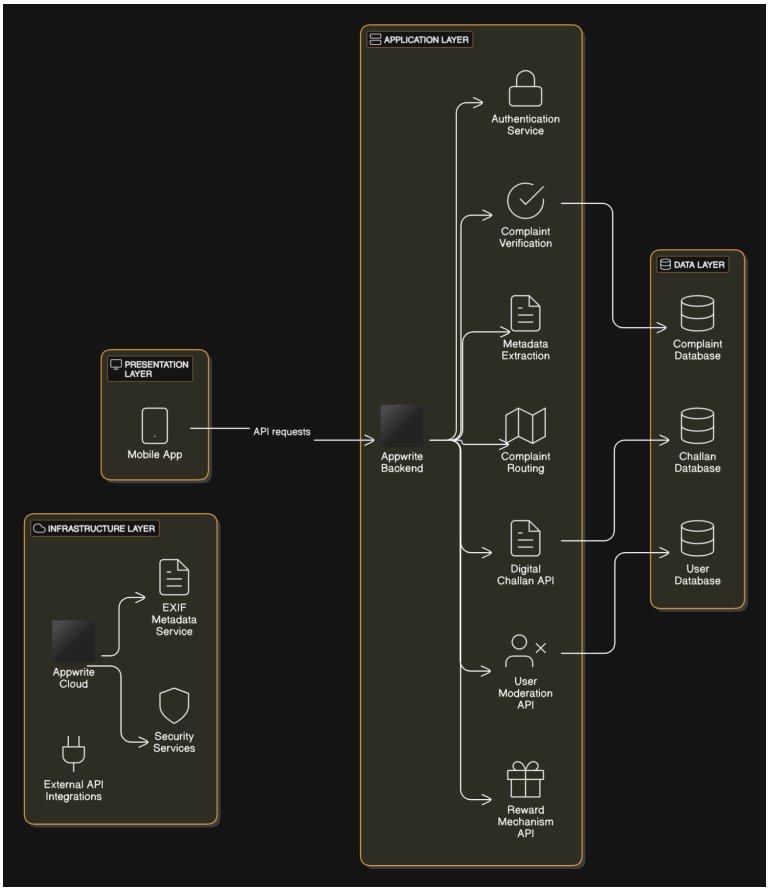

Fig 4.1 System architecture diagram of Civic Eye

The Civic Eye application follows a multi-layered architecture, ensuring a well-structured and scalable design. The system consists of four primary layers: the **Presentation Layer**, the **Application Layer**, the **Data Layer**, and the **Infrastructure Layer**. Each layer plays a crucial role in the overall functionality of the system, ensuring modularity, maintainability, and efficiency in handling user complaints and law enforcement processes.

The **Presentation Layer**, also known as the User Interface Layer, serves as the primary point of interaction between users and the system. It includes the mobile application for users and the authority dashboard for administrators, allowing seamless access to features such as complaint submission, status tracking, and report management. This layer is designed to be intuitive and user-friendly, ensuring accessibility for a wide range of users.

The **Application Layer**, or Business Logic Layer, acts as the core processing unit of the system. It is responsible for handling authentication, complaint management, metadata extraction, complaint routing, digital challan generation, user moderation, reward distribution, and notification services. The backend, powered by Appwrite, processes user inputs, applies validation, and ensures smooth interaction between different system components. This layer also incorporates security measures to protect user data and maintain the integrity of complaint records.

The **Data Layer**, also referred to as the Database Management Layer, is dedicated to storing and managing essential system data. It comprises databases for complaints, users, and challans, ensuring structured and secure data storage. This layer plays a critical role in retrieving and updating information as per system requirements. With optimized database queries and indexing, the system ensures quick access to data, enhancing performance and reliability.

The **Infrastructure Layer**, or Cloud & External Services Layer, provides the necessary backend support for application operations. It includes cloud-based services such as Appwrite Cloud for backend hosting, EXIF metadata extraction for image validation, security services for data protection, external API integrations for law enforcement connectivity, and media storage for handling uploaded evidence. This layer ensures high availability, scalability, and fault

tolerance, allowing the system to handle a growing number of users and complaints effectively.

By organizing the system into these distinct layers, Civic Eye ensures modularity, scalability, and efficient handling of complaints and violations while maintaining security and user engagement. This structured approach not only simplifies system maintenance but also allows for future enhancements without affecting the core functionality.

4.2 Use Case Diagram

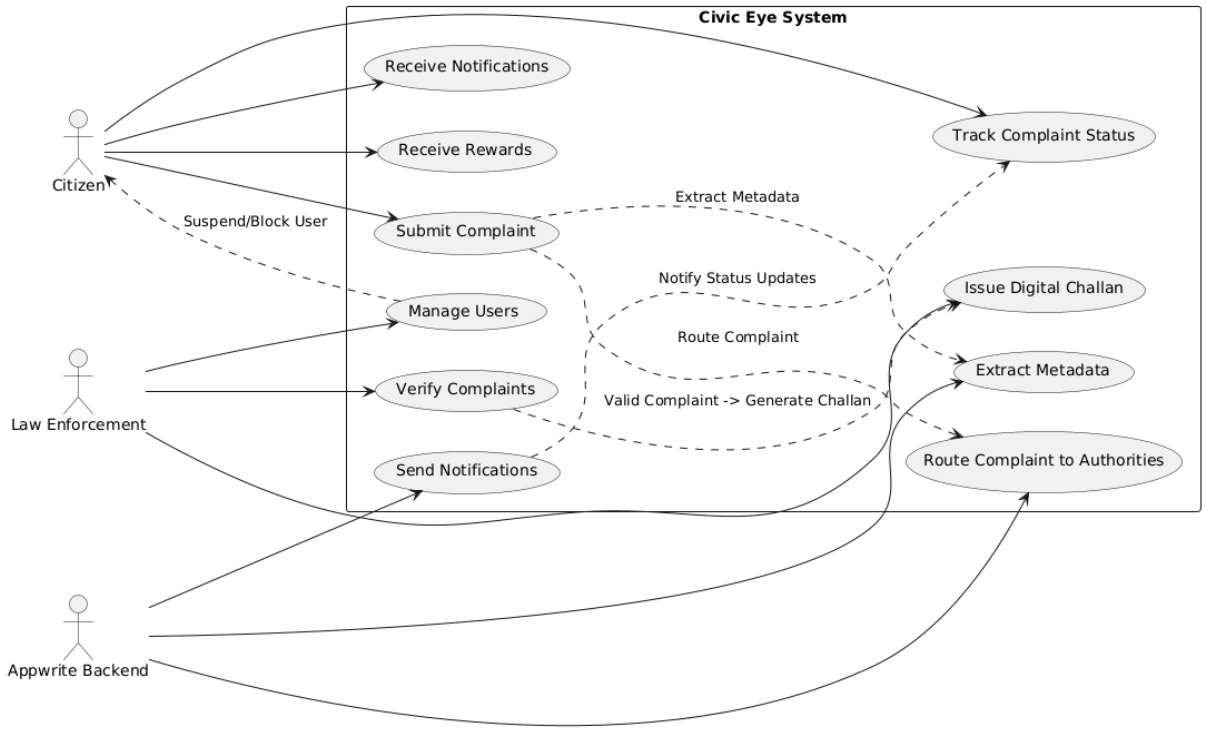

Fig 4.2 Civic Eye use case diagram

The use case diagram for the Civic Eye System provides a structured visualization of how different **users** interact with the system. The diagram consists of three main **actors**: **Citizen, Law Enforcement, and Appwrite Backend**.

The **Citizen** is the primary user who submits complaints and tracks their progress. Citizens can report civic issues by uploading evidence such as images or videos. They can track the progress of their

submitted complaints and receive notifications about status updates. If a complaint is valid and leads to an action, the user may receive a reward.

The **Law Enforcement** officials are responsible for verifying complaints, issuing challans, and managing users. They assess submitted complaints to determine their validity. If a complaint is found to be valid, they issue a **digital challan** to penalize the violator. Authorities also have the ability to manage users, including blocking individuals who misuse the system.

The **Appwrite Backend** plays a crucial role in the system by handling metadata extraction from uploaded complaints, routing complaints to the appropriate authorities, and sending notifications to users. It ensures that complaints are processed efficiently and provides security and authentication services. The backend also supports the reward system by determining eligibility based on complaint verification.

The use case diagram visually represents these interactions, showing how **users** engage with different system functionalities. It highlights dependencies, such as metadata extraction supporting complaint submission and complaint verification leading to challan issuance. This structured approach ensures clear communication between system components, improving efficiency and reliability.

4.3 Entity-Relationship (ER) Diagram

The **ER Diagram** for the **Civic Eye** application represents the core entities involved in the system and their relationships. Each class represents a real-world entity, and the connections between them illustrate how they interact.

The system consists of key entities such as **User, Complaint, Authority, Challan, Reward, Notification, and Metadata**, each playing a crucial role in the workflow. Users submit complaints through the mobile application, which are stored in the system and processed by authorities. Verified complaints may lead to the issuance of **digital challans** or **reward points** based on their validity.

Notifications are used to keep users informed about the status of their complaints, rewards, and penalties. Additionally, metadata extraction helps enhance the accuracy of complaint verification.

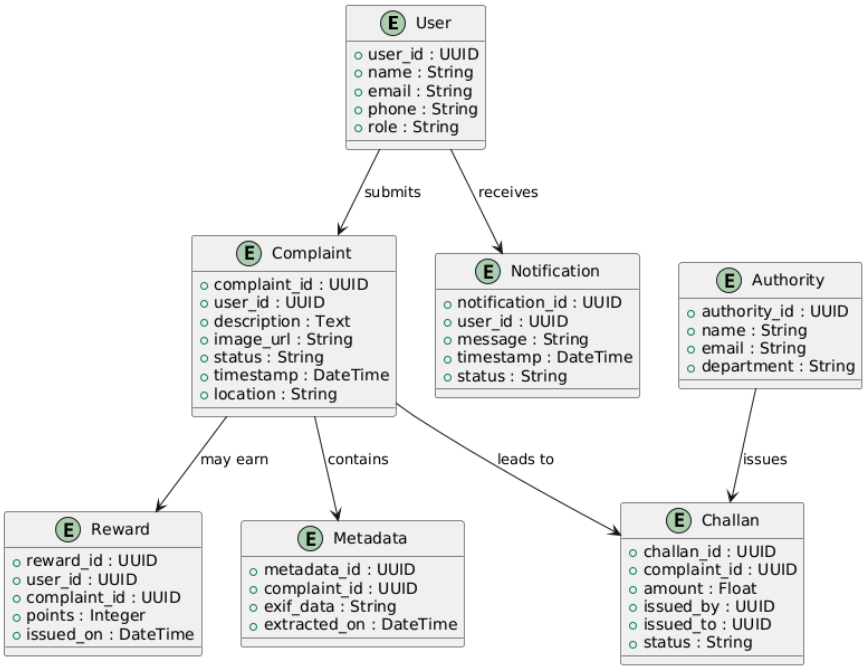

Fig 4.3 ER Diagram of Civic Eye

The **User (Usr)** entity represents individuals who submit complaints. Each user is uniquely identified by a `user_id` and has attributes such as name, email, phone number, and role, which defines whether they are a citizen or an authority.

The **Complaint (Complant)** entity stores user-submitted issues, with each complaint having a unique `complaint_id`. It is associated with a user through `user_id` and includes details like description, image URL, status, timestamp, and location. Complaints may lead to further actions such as verification, penalty issuance, or reward allocation.

The **Authority (Authorlty)** entity represents government officials responsible for handling complaints. Each authority has a unique `authority_id`, along with details like name, email, and department. Authorities verify complaints and take necessary actions, including issuing challans.

The **Challan (Challan)** entity records penalties issued for verified complaints. A challan is uniquely identified by `challan_id` and is linked to a complaint through

`complaint_id`. It includes details such as the amount, the issuing authority, and the recipient.

The **Reward (Reward)** entity manages the reward points allocated to users who submit valid complaints. Each reward is uniquely identified by `reward_id`, linked to a user via `user_id` and a complaint via `complaint_id`, and contains information on points awarded and the date of issuance.

The **Notification (Notifcation)** entity ensures communication with users by storing messages related to complaint status, rewards, or penalties. Each notification has a unique `notification_id`, is linked to a user via `user_id`, and contains a message, timestamp, and status indicating whether it has been read.

The **Metadata (Metadata)** entity stores extracted image metadata, such as EXIF data, to ensure complaint accuracy. Each metadata entry is uniquely identified by `metadata_id`, is linked to a complaint via `complaint_id`, and records the extracted data along with a timestamp.

The relationships among these entities define the system's functionality. Users submit complaints, which may lead to penalties recorded in the challan entity. Verified complaints may earn users rewards. Authorities manage complaints and issue challans as necessary. Notifications are sent to users regarding complaint updates, and metadata is extracted to enhance complaint verification.

CHAPTER 5

IMPLEMENTATION

This chapter discusses the implementation of the **Civic Eye** system, detailing the technological choices, development process, and how the system effectively addresses the problem of reporting and managing civic violations. The implementation phase involved integrating a **user-friendly mobile application**, a **secure backend system**, and a **dashboard for authorities** to monitor and resolve complaints efficiently. The system leverages **Appwrite** for backend services, ensuring seamless authentication, complaint storage, and digital challan management.

The development process followed a **modular approach**, ensuring scalability and maintainability. The frontend was implemented using **Flutter**, providing a cross-platform experience for both Android and iOS users. The backend was built using **Appwrite**, which handles authentication, database management, and API interactions. **Metadata extraction** techniques were incorporated to automatically fetch location, date, and time from images to enhance complaint verification accuracy. Additionally, a **complaint verification system** was implemented to process submitted reports and generate digital challans for detected violations.

One of the major challenges before implementing **Civic Eye** was the lack of an efficient system for citizens to report civic violations with **proper evidence**. Traditional reporting methods were time-consuming, lacked structured data, and were difficult for authorities to manage effectively. The **Civic Eye** application streamlined this process by providing a **structured complaint submission mechanism**, automating complaint verification, and allowing authorities to **track and resolve** violations through a dedicated dashboard.

The system successfully addresses the problem by providing **a structured and transparent platform** for reporting civic issues, reducing manual workload for authorities, and incentivizing users

with a **reward system** for valid reports. **Real-time notifications** ensure that users stay updated on their complaint status, and authorities can efficiently manage multiple cases without delays.

The following sections present major screenshots demonstrating the functionality and interface of the **Civic Eye** system, highlighting its key features, complaint submission process, and the workflow for authorities to manage violations effectively.

5.1 System Screenshots

5.1.1 User

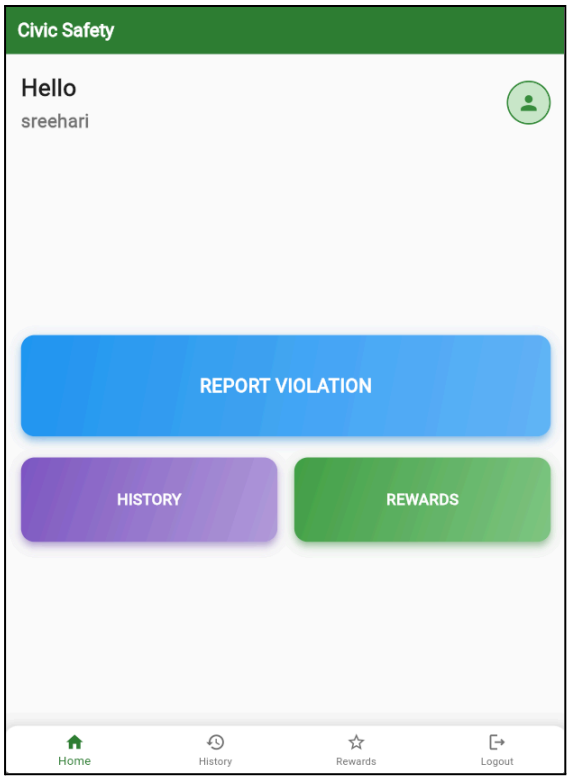

Fig 5.1 User Home Page

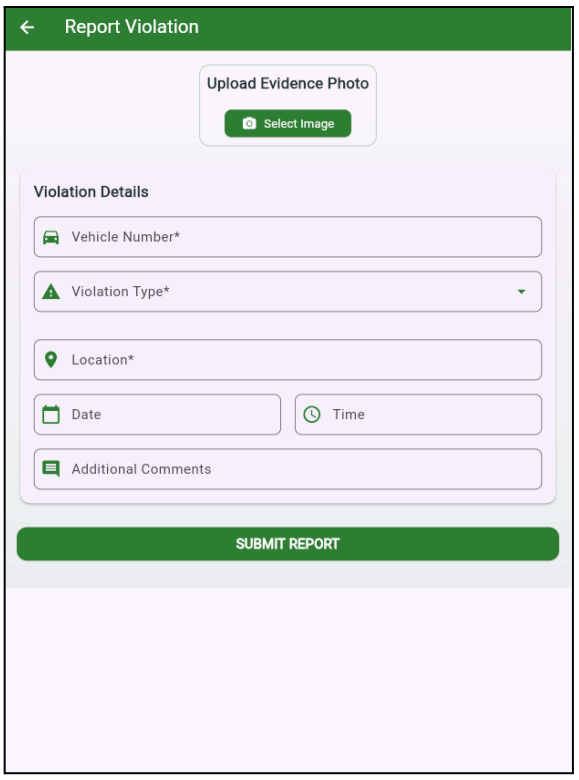

Fig 5.2 User Report Violation

The home page serves as the central hub, providing users quick access to key features such as reporting violations, tracking complaints, and viewing rewards. Implemented using Flutter for a seamless UI, it interacts with the Appwrite backend to fetch real-time data, ensuring users stay updated on their reports and system notifications.

Report violation page allows users to submit complaints by entering details and uploading evidence. The system extracts metadata from images, ensuring accuracy. The backend processes and stores complaints while routing them to the appropriate authorities. Appwrite handles authentication and database management, ensuring data security and integrity.

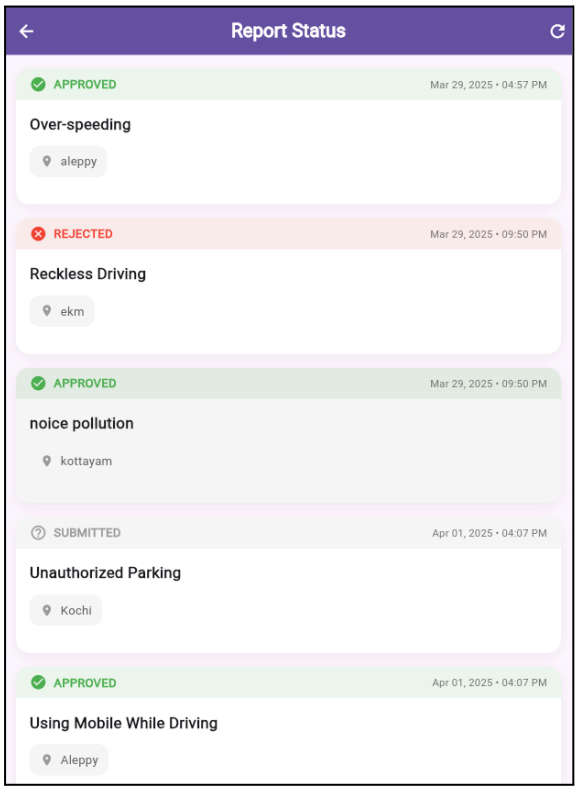

Fig 5.3 User Report Stats View Page

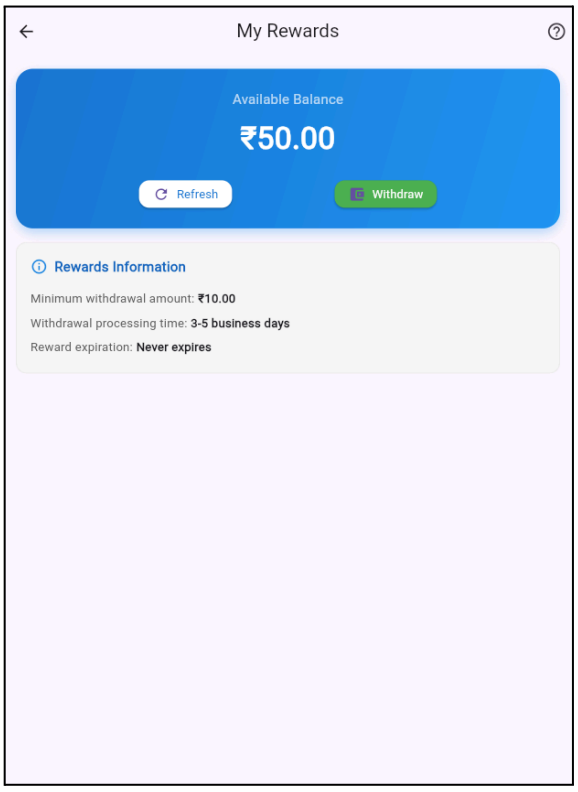

Fig 5.4 User Reward Page

Users can track their complaint progress through Report status page. The system fetches real-time updates from the complaint database, displaying status changes such as pending, under review, or resolved. Notifications are sent to users, keeping them informed throughout the resolution process.

The reward page displays the incentives earned by users for successfully reporting valid violations. Once a complaint is verified and approved by authorities, users receive rewards in the form of points or vouchers. This feature encourages active participation in maintaining civic order and promotes responsible reporting.

5.1.2 Officer

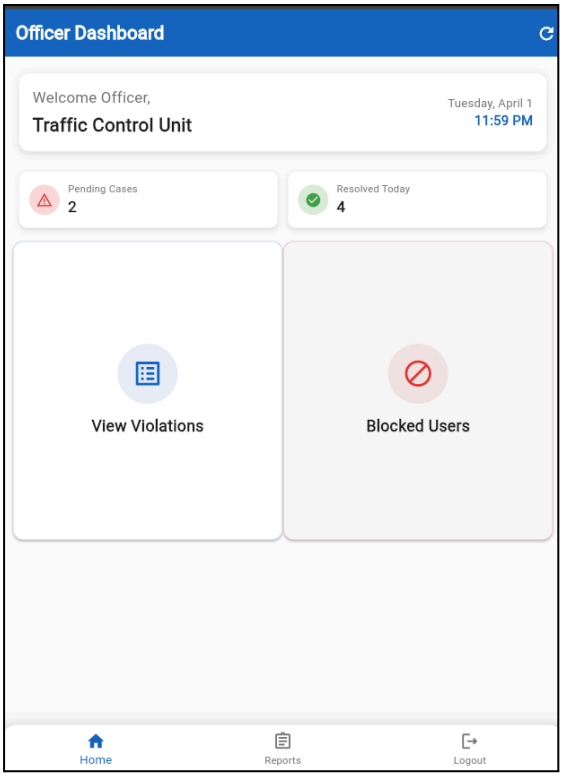

Fig 5.5 Officer Home Page

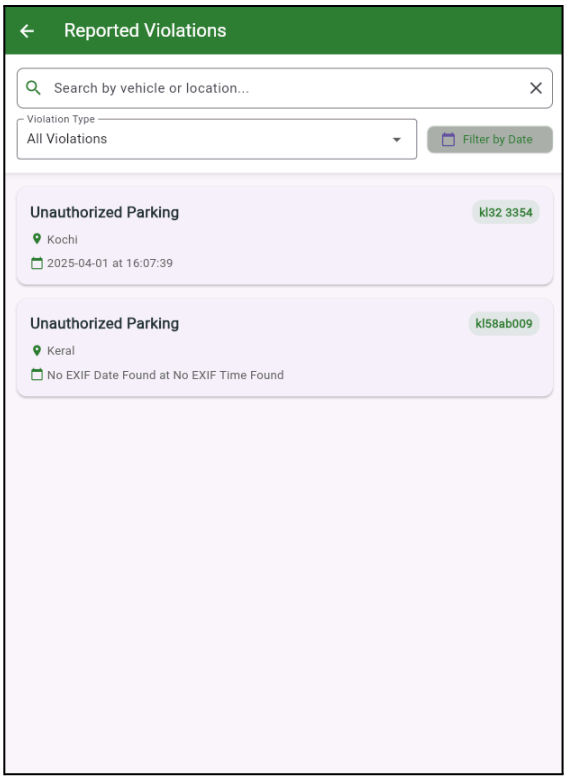

Fig 5.6 View Reported Violations Page

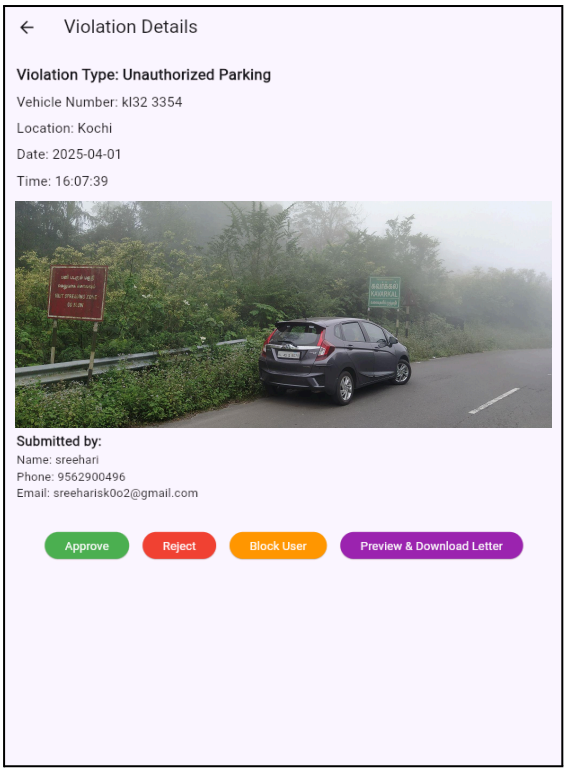

Fig 5.7 Officer Violation Details Page

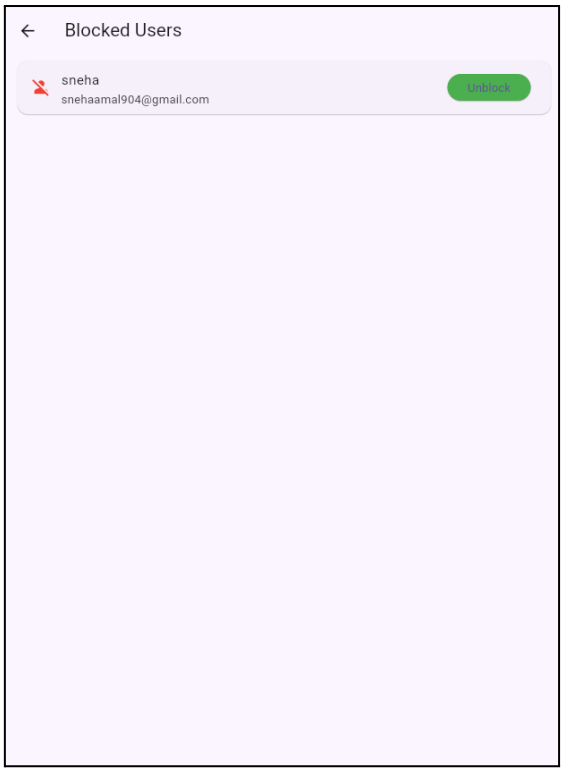

Fig 5.6 Blocked Users Page

The officer home page provides a dashboard displaying an overview of reported violations, pending cases, and system notifications. Implemented with a clean and structured UI, it allows officers to quickly navigate through complaints and take necessary actions. The backend retrieves real-time data, ensuring officers have access to the latest reports and status updates.

Reported violations page enables officers to review reported violations, including submitted evidence and metadata. Complaints are categorized based on priority, and officers can update the status or take action directly from the interface. The system ensures secure data access, allowing only authorized personnel to manage and resolve violations efficiently.

The violation details page provides a comprehensive view of a reported complaint, including user-submitted evidence, location details, timestamps, and system-generated metadata. Officers can review the details, verify authenticity, and take appropriate actions such as issuing digital challans or updating the complaint status. The backend ensures secure retrieval of complaint data while maintaining logs of all actions taken.

The blocked users page allows authorities to manage and monitor users who have violated system policies, such as submitting false reports or abusing the platform. Officers can view a list of blocked users, along with reasons for the block, and have the option to review or reinstate accounts if necessary. The system ensures that only authorized personnel can perform these actions, maintaining platform integrity and preventing misuse.

CHAPTER 6

TESTING

Testing ensures that the system functions as expected, meets requirements, and is free from major defects. The testing process includes unit testing, integration testing, and system testing.

Table 6.1 Major Test Cases and Results

Test ID	**Scenario**	**Expected Result**	**Actual Result**
TC1	Sign up	User successfully created	User successfully created
TC2	Logs in	Successfully authenticated	Successfully authenticated
TC3	Report Violation Page	Opens Report Violation Page	Opens Report Violation Page
TC4	Upload Image	Image uploaded successfully	Image uploaded successfully
TC5	Fetch Metadata	Metadata fetched successfully	Metadata fetched successfully
TC6	Add Text	Text successfully added to the page	Text successfully added
TC7	Submit Report Violation	Report violation uploaded successfully	Report Violation uploaded successfully

TC8	Reward Page	Can view reward Page	Can view reward page
TC9	User Profile	Can view User page	Can view user page
TC10	History	User can retrieve previous violation reports	User can retrieve previous violation reports
TC11	Officer Log In	Successfully Authenticated	Successfully Authenticated
TC12	Approve Violation Report	Violation Report approved successfully	Violation Report approved successfully
TC13	Reject Violation Report	Violation Report rejected successfully	Violation Report rejected successfully
TC14	Block User Profile	Can block user profile	Can block user profile
TC15	Preview&DownloadChallan Document	Can preview& download Challan Document	Can preview& download Challan Document

6.2 Major Test Cases and Results

Testing was conducted to verify the system's core functionalities, including user authentication, sentiment analysis, and stock trend visualization. Each test case was designed to assess accuracy, performance, and reliability. The table below presents key test cases, their expected and actual outcomes, and their status

6.2.1 Submit Report Violations Testing

Submit Report Violations is a core feature of the system, allowing citizens to document public transport violations with evidence. Ensuring a seamless and accurate reporting process enhances user engagement and helps authorities take timely action against offenders.

In the **Submit Report Violations (TC-07)** test, a violation category was selected, and the necessary details were entered. Multimedia evidence (images/videos) was attached, and the **Submit Report** button was clicked. The system successfully processed the report, stored the data securely, and extracted metadata from the images for validation.

The report was verified for completeness and accuracy, ensuring correct classification and timestamp alignment. The system displayed the report status without errors, confirming that the violation reporting feature worked as expected.

6.2.2 Fetch Metadata Testing

Fetch Metadata is a critical feature of the system that ensures accurate timestamping of reported violations by extracting date and time from image metadata (EXIF data). This enhances report credibility and prevents manual entry errors.

In the **Fetch Metadata (TC-05)** test, an image taken from a camera was uploaded as part of a violation report. The system automatically extracted the date and time from the image's EXIF metadata, provided it was captured using a camera and stored in the system.

The system correctly fetched and displayed the metadata without errors, confirming the proper functionality of the EXIF-based timestamp extraction feature.

CHAPTER 7

CONCLUSION

The **CIVIC EYE** app enables citizens to report violations seamlessly, integrating multimedia evidence submission and an incentivized complaint system. With secure data handling, accurate metadata extraction, and real-time tracking, the app ensures efficient issue resolution. By fostering community engagement and streamlining enforcement, it enhances public safety and accountability.

7.1 Merits

The **CIVIC EYE** App empowers citizens by enabling active participation in improving civic infrastructure and safety. Real-time tracking ensures transparency, while multimedia evidence submission strengthens complaint validation. Automated challan generation streamlines fine processing, and encrypted data storage enhances privacy. An efficient authority dashboard helps officials manage reports, while metadata extraction (EXIF) ensures accurate timestamps. Additionally, incentivized reporting encourages valid complaints by rewarding users for verified reports, making the system both effective and engaging.

7.2 Demerits

While our system enhances public participation in reporting violations, it faces some challenges. Verification can be difficult if reports lack clear evidence, and false complaints may arise, requiring strict moderation. The system also relies on active participation from authorities, and a high volume of reports could lead to administrative overload. Despite these challenges, the app remains a powerful tool for improving civic engagement and public safety.

7.3 Future Work

Future enhancements include **AI-powered report verification**, which will utilize image and video analysis to detect fake or manipulated reports. Additionally, **offline complaint submission** will allow users to report issues without an internet connection, ensuring reports are automatically synced once connectivity is restored. These advancements will enhance reliability, security, and accessibility for all users.

REFERENCES

The **Appwrite Documentation** serves as a primary resource for backend integration in the Civic Eye system. It provides comprehensive guides on authentication, database management, and cloud storage. The documentation includes API references and implementation examples that help in securing user data, managing complaints, and ensuring seamless communication between the mobile app and the backend. Appwrite's real-time database and authentication features were crucial in maintaining user sessions, complaint records, and officer management.

The **Challan API Guide** offers detailed instructions on implementing digital challan generation. This guide explains API endpoints, request formats, and fine calculation mechanisms, enabling automatic violation detection and fine issuance. It also includes methods for integrating with government law enforcement databases, ensuring real-time tracking of violations. The guide further details response handling, security measures, and best practices for implementing digital penalty processing in a secure and efficient manner.

The **Flutter Official Documentation** provides extensive coverage of the Flutter framework, which was used to develop the Civic Eye mobile application. It includes resources on widget implementation, navigation management, state handling, and UI/UX optimization. The documentation helped in building a responsive and interactive user interface, supporting cross-platform compatibility for both Android and iOS devices. It also offers best practices on performance optimization, data handling, and accessibility compliance, ensuring a seamless user experience.

The **EXIF Metadata Extraction Libraries** played a key role in validating complaints by retrieving crucial metadata from images, such as timestamps, GPS coordinates, and device information. These libraries ensure the authenticity of the reported violations by verifying whether the captured images were taken at the claimed time and location. The extracted metadata enhances the reliability of complaint processing and helps authorities validate evidence before taking action.

Additional references include **security best practices** for mobile and web applications, which were crucial in safeguarding user data, preventing unauthorized access, and ensuring compliance with cybersecurity guidelines. **API integration guides** were consulted to streamline communication between the mobile application, backend services, and third-party systems, ensuring smooth data exchange and efficient request handling. **Cloud storage solutions** were referenced to optimize media storage and retrieval, ensuring quick access to user-submitted evidence while maintaining data security and privacy.

Together, these resources contributed to the successful development and implementation of the Civic Eye system, ensuring robustness, scalability, and reliability in addressing civic violations efficiently.

APPENDICES

This appendix contains essential code snippets from the Civic Eye app, providing a deeper understanding of its core functionalities. Each section highlights key components, including user authentication, database integration for incident reporting, location tracking, and UI components. These code snippets demonstrate how Appwrite is used for secure authentication and data storage, how the app retrieves and displays user location using the Geolocation API and Google Maps, and how reports are submitted through an intuitive user interface. This structured breakdown serves as a reference for understanding the app's architecture and implementation.

APPENDIX A : USER AUTHENTICATION

User authentication in Civic Eye is managed using Appwrite, providing secure login, registration, and session management.

A.1 Signup and Login with Appwrite

```
import { Client, Account } from "appwrite";

const client = new Client();
client.setEndpoint("https://cloud.appwrite.io/v1").setProject(“project-id”);

const account = new Account(client);

const registerUser = async (email, password) => {
  try {
    await account.create("unique()", email, password);
    console.log("User registered successfully");
```

```
  } catch (error) {
    console.error("Error registering user:", error.message);
  }
};

const loginUser = async (email, password) => {
  try {
    await account.createEmailSession(email, password);
    console.log("User logged in successfully");
  } catch (error) {
    console.error("Login failed:", error.message);
  }
};
```

This snippet handles user authentication using Appwrite. The `Client` instance is configured with the backend API endpoint and project ID, allowing secure communication between the app and the Appwrite service. The `registerUser` function creates a new user account with a unique ID, email, and password, ensuring secure user registration. If an error occurs, it logs the failure message. The `loginUser` function establishes a session for the user using their credentials, allowing access to protected app features. Errors during login are caught and displayed in the console, making debugging easier.

A.2 Logout User

```
export async function logoutUser() {
    try {
        await account.deleteSession('current');
    } catch (error) {
        console.error('Logout Error:', error);
    }
}
```

The logout function makes an API call to Appwrite's `deleteSession` method, specifically targeting the current user session. This ensures that the session token is invalidated, preventing unauthorized access if someone else gains access to the device. Once the session is deleted, the app redirects the user to the login screen, confirming that they have successfully logged out.

APPENDIX B : USER AUTHENTICATION

Incident reporting is a critical feature of the Civic Eye app, allowing users to document and submit details of safety concerns or emergencies directly to the system. This functionality ensures that incidents are recorded systematically, enabling authorities or relevant stakeholders to take appropriate action. The reporting feature captures essential details such as the title, description, user ID, and location, which are stored securely in the Appwrite database.

B.1 Submitting Reports to Appwrite Database

```
import { Databases, ID } from "appwrite";

const databases = new Databases(client, "your-database-id");

const reportIncident = async (title, description, userId, location) => {
  try {
    await databases.createDocument("incident-collection", ID.unique(), {
      title,
      description,
      userId,
      location,
    });
    console.log("Incident reported successfully");
```

```
  } catch (error) {
    console.error("Error reporting incident:", error.message);
  }
};
```

This function enables users to report incidents by storing their details in an Appwrite database. The `Databases` instance is linked to a specific database, allowing structured data storage. The `reportIncident` function creates a new document in the "incident-collection," assigning a unique ID and capturing details like the title, description, user ID, and location. If the process succeeds, it logs a success message; otherwise, an error message is displayed, helping developers troubleshoot any issues.

B.2 Retrieving Reports

```
import export async function fetchReports() {
   try {
const  reports = await databases.listDocuments('db_id','collection_id')
      return reports.documents;
   } catch (error) {
      console.error('Fetch Reports Error:', error);
      throw error;
   }
}
```

The function responsible for retrieving reports interacts with the Appwrite database, specifically querying the "incident-collection" to obtain stored reports. This is achieved through a database `listDocuments` request, which fetches multiple documents from the collection. The function is designed to retrieve all reports or filter them based on specific criteria such as user ID or location, enabling a more personalized and relevant data retrieval process.

APPENDIX C : UI COMPONENTS

The user interface (UI) of the Civic Eye app is designed to be intuitive, accessible, and user-friendly, ensuring a seamless experience for reporting civic issues. The UI components are structured using modern frontend frameworks and libraries, ensuring smooth interactions and responsiveness across various devices.

Each UI element is crafted to enhance usability and engagement. The navigation system allows users to move between different sections effortlessly, while forms and input fields ensure that issue reporting is straightforward and efficient. Buttons, modals, and interactive elements follow a consistent design language, improving visual hierarchy and accessibility.

C.1 Report Submission Form

```
import export default function ReportForm({ onSubmit }) {
  const [description, setDescription] = useState('');

  return (
    <form onSubmit={(e) => { e.preventDefault(); onSubmit(description); }}>
<textarea placeholder=”Describe issue …” onChange={(e) =>
setDescription(e.target.value)} />
      <button type="submit">Submit Report</button>
    </form>
  );
}
```

The report submission form is a crucial component of the Civic Eye app, enabling users to report civic issues efficiently. This form collects essential details such as the type of issue, location, description, and an optional image upload to provide visual evidence. The form is designed with a clean layout, ensuring ease of use while maintaining clarity.

Each input field is validated to prevent incomplete or incorrect submissions. For instance, required fields ensure that users provide necessary information before proceeding, while file uploads are restricted to specific formats and sizes to maintain efficiency. The form also integrates geolocation services, allowing users

to either manually enter their location or use GPS to pinpoint the exact coordinates.

Upon submission, the data is sent to the backend through an API request. The system then processes the report, storing it securely in the database while also sending relevant notifications to administrative personnel for further action. A confirmation message is displayed to the user, ensuring that they receive feedback about their submission status. The overall design and functionality of the report submission form focus on simplicity and effectiveness, ensuring that users can report civic issues quickly and accurately.