TABLE OF CONTENTS

1. INTRODUCTION

1.1 Project Overview

In today's digital age, cybercrime has become a growing concern, affecting individuals, businesses, and governments worldwide. The Cybercrime Fraud Detection and Reporting System is designed to provide a structured platform where users can report cyber fraud incidents, upload supporting evidence, and track the progress of their cases. This system allows victims of cybercrime to seek justice efficiently while also providing law enforcement authorities with a centralized database for investigations. Additionally, the platform offers real-time scam alerts and awareness campaigns to educate users on potential threats.

The system consists of multiple modules, including user authentication, incident reporting, evidence management, status tracking, scam alerts, and law enforcement collaboration. Users can securely log in, report incidents by selecting a fraud type, and attach relevant evidence such as screenshots or documents. Authorities can review cases, update investigation progress, and take necessary actions. Furthermore, an analytics dashboard provides statistical insights into cybercrime trends, helping in decision-making for cyber safety improvements.

By integrating modern technologies such as Flutter for the frontend, Node.js/Express.js for the backend, and MongoDB for data storage, the platform ensures seamless operation and data security. The use of JSON Web Tokens (JWT) for authentication enhances user privacy and access control. The expected outcome of this project is to establish a transparent, user-friendly, and efficient cybercrime reporting system that empowers users while supporting law enforcement in combating digital fraud effectively.

TECHNOLOGIES USED:

FRONTEND : Flutter

BACKEND : Node.js with Express

DATABASE : MongoDB and SQL

2. SYSTEM SPECIFICATIONS

The purpose of the software requirements specification is to produce the specification of the analysis task and to establish the complete information about the requirement, behaviour and other constraints such as functional performance and so on.

2.1 HARDWARE SPECIFICATIONS

CPU: 4th generation Intel Core i5 or higher 2.8 GHz or faster.

RAM: 16 GB RAM or more.

STORAGE: 16 GB or more of available disk space SSD strongly recommended.

GRAPHICS: 1920 x 1080 or higher screen resolution dedicated GPU for better performance.

INTERNET: Recommended 40-100 Mbps.

2.2 SOFTWARE SPECIFICATIONS

FRONTEND:

- **FRAMEWORK:** Flutter
- **PLATFORM:** Android(Minimum requirement: Android 8.0)

BACKEND: Node.js with Express(API development)

DATABASE:

- **PRIMARY DATABASE:** MongoDB and SQL(For storing incidents and user data)

CLOUD HOSTING:

- **PLATFORMS:** AWS, Google Cloud, or Azure
- **API's:** Phishing/Scam detection API integration

2.3 SOFTWARE DESCRIPTIONS

FLUTTER:

Flutter is an open-source UI software development kit (SDK) created by Google. It is used to build natively compiled applications for mobile, web, and desktop from a single codebase. Flutter uses the Dart programming language and provides a rich set of pre-designed widgets, tools, and libraries to create beautiful, fast, and responsive user interfaces.

1. CORE CONCEPTS OF FLUTTER

1. WIDGETS :

Everything in Flutter is a widget, from structural elements (like buttons and text) to layout elements (like rows, columns, and grids).

Widgets are categorized into:

- **Stateless Widget:** Immutable widgets that do not change over time.
- **Stateful Widget:** Widgets that can change dynamically based on user interaction or data changes.

2.DART PROGRAMMING LANGUAGE :

- Flutter uses Dart, a modern, object-oriented language developed by Google.
- Dart is optimized for building UIs and supports features like async/await, garbage collection, and strong typing.

3.HOT RELOAD :

Flutter's hot reload feature allows developers to see changes in the app instantly without restarting it, making the development process faster and more efficient.

4.PLATFORM-SPECIFIC ADAPTATIONS :

Flutter provides platform-specific widgets and APIs to ensure apps look and feel native on both Android and iOS.

2. FLUTTER ARCHITECTURE :

1.LAYERED ARCHITECTURE :

- Flutter is built on a layered architecture, with each layer providing a specific set of functionalities:
- Framework Layer: Contains widgets, animations, and rendering logic.
- Engine Layer: Handles rendering, input, and platform-specific integrations.
- Embedder Layer: Integrates Flutter with the host operating system.

2.RENDERING PIPELINE :

Flutter uses Skia, a 2D rendering engine, to draw widgets directly onto the canvas,ensuring high performance and smooth animations.

3.REACTIVE PROGRAMMING :

Flutter follows a reactive programming model, where the UI updates automatically in response to changes in the app's state.

Express.js is a fast, unopinionated, and minimalist web framework for **Node.js.** It is widely used for building web applications and **APIs (Application Programming Interfaces)** due to its simplicity, flexibility, and scalability. Express.js provides a robust set of features for building single-page, multi-page, and hybrid web applications.

CORE CONCEPTS OF EXPRESS.js

1.MIDDLEWARE :

- Middleware functions are the backbone of Express.js. They have access to the request (req), response (res), and the next middleware function in the application's request-response cycle.
- Middleware can perform tasks like logging, authentication, error handling, and more.

2.ROUTING :

- Express.js allows you to define routes for handling HTTP requests (GET, POST, PUT, DELETE, etc.).
- Routes are defined using methods like app.get(), app.post(), app.put(), and app.delete().

3.REQUEST AND RESPONSE OBJECTS :

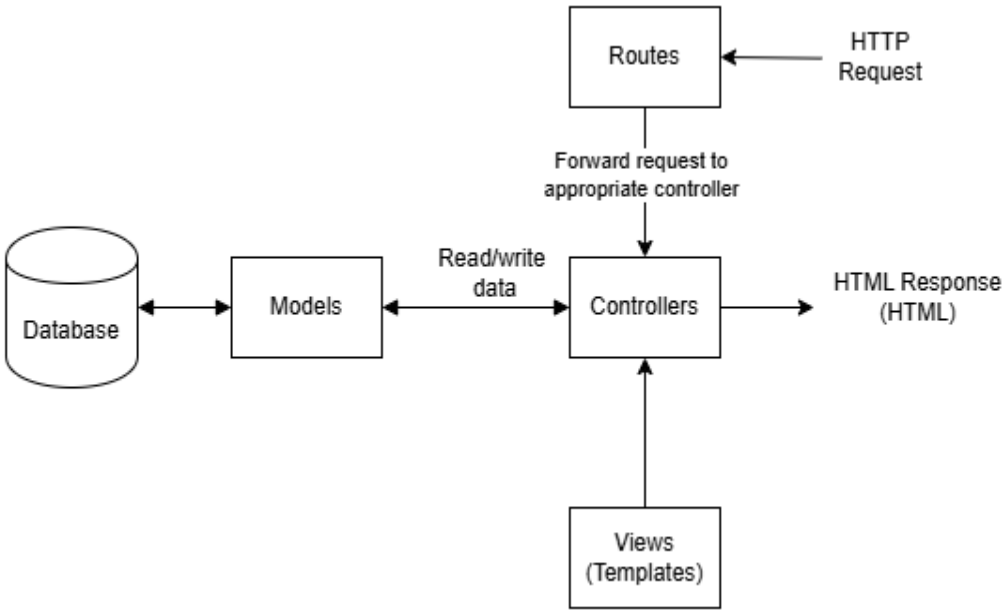
- Request (req): Contains information about the HTTP request (e.g., query parameters, headers, body).
- Response (res): Used to send a response back to the client (e.g., sending JSON, rendering a view).

4.ERROR HANDLING :

Express.js provides built-in and custom error-handling mechanisms to manage errors in your application.

5. TEMPLATE ENGINES :

Express.js supports template engines like EJS, Pug, and Handlebars for rendering dynamic HTML pages.



MongoDB is a popular **NoSQL database** that stores data in a flexible, JSON-like format called **BSON** (Binary JSON). It is designed for scalability, performance, and ease of development. MongoDB is widely used in modern web applications, especially those built with Node.js, due to its flexibility and ability to handle large volumes of unstructured or semi-structured data.

CORE CONCEPTS OF MONGODB

1.DOCUMENT-ORIENTED DATABASE :

MongoDB stores data in documents, which are JSON-like objects with key-value pairs.

Example of a document: **Json**

```
{
"_id": "12345",
 "name": "John Doe",
 "age": 30,
"email": john.doe@example.com
}
```

2.COLLECTIONS :

- Documents are grouped into **collections**, which are analogous to tables in relational databases.
- Example: A collection named users can store multiple user documents.

3.SCHEMALESS :

MongoDB is schemaless, meaning documents in the same collection can have different structures.

4.BSON :

MongoDB uses BSON (Binary JSON) for storing and transferring data. BSON supports additional data types like dates, binary data, and object IDs.

5.INDEXES :

Indexes improve query performance by allowing faster data retrieval. MongoDB supports single-field, compound, and multi-key indexes.

6.REPLICATION :

MongoDB provides high availability through replica sets, which are groups of MongoDB instances that maintain the same data.

7.SHARDING :

MongoDB supports horizontal scaling through sharding, which distributes data across multiple servers.

JSON WEB TOKEN

JSON Web Token (JWT) is an **open standard (RFC 7519)** for securely transmitting information between parties as a **JSON object**. The information is digitally **signed**, ensuring **authenticity and integrity**.

JWT is commonly used for:

- **Authentication** (e.g., login sessions, API access)
- **Authorization** (e.g., role-based access control)
- **Data exchange** (secure information sharing)

STRUCTURE OF A JWT

A JWT consists of **three parts**, separated by dots (.):

"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOjEwMiwicm9sZSI6ImFkbWluIiwiaWF0IjoxNjg0ODg4ODAwfQ.r9gULAfzO1HVOf6xFqjJGdFmAH4JAt6rKNfuLb3mvmA"

1□.HEADER (Metadata)

Specifies the **token type** (JWT) and **signing algorithm** (HS256, RS256, etc.).

json

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

2.PAYLOAD (Claims)

Contains **user data** (e.g., user ID, role) and **token metadata** (e.g., expiration time).

json

```
{
  "userId": 102,
  "role": "admin",
  "iat": 1684888800,  // Issued At timestamp
```

"exp": 1684892400 // Expiration timestamp

}

3.SIGNATURE (Security)

- Ensures data integrity using a **secret key or private key**.
- Example signing process (HS256 algorithm):
- HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), secret)
- Verifies that the **JWT hasn't been tampered with**.

HOW JWT WORKS?

1.USER LOGS IN

- User enters credentials (email/password).
- Server **verifies** credentials and generates a **JWT**.

2.SERVER SENDS JWT TO USER

The token is returned as a response and stored **client-side** (e.g., **localStorage, sessionStorage, HTTP cookies**).

3.CLIENT USES JWT FOR API REQUESTS

- The client includes JWT in the Authorization header:
- http
- Authorization: Bearer <JWT>

4.SERVER VERIFIES JWT

- Extracts the token, **validates the signature**, and **decodes the payload**.
- If valid, **grants access** to protected resources.

5.TOKEN EXPIRATION & REFRESH

JWTs **expire** (exp claim), and a **refresh token** can be used to obtain a new one.

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The current approach to reporting cybercrime and fraud is highly fragmented, inefficient, and lacks user engagement. Most victims rely on manual complaint filing via emails, phone calls, or offline reports to cybercrime departments, which often leads to delays and unresolved cases. Additionally, lack of transparency and tracking mechanisms makes it difficult for users to follow up on their cases. There is no centralized platform where users can submit evidence securely, and the process often involves excessive paperwork. Furthermore, cybercrime awareness is limited, leaving many users vulnerable to online fraud.

Law enforcement agencies and authorities face challenges in managing a growing number of cybercrime cases due to disorganized data, slow investigation procedures, and inefficient communication between users and authorities. There is also no real-time alert mechanism to inform users about ongoing scams or fraudulent activities, leading to repeated victimization.

Some major disadvantages of the existing system include:

1. **Lack of Real-time Tracking** – Users are unable to track the status of their reported cases.
2. **Manual and Time-consuming Process** – Reporting fraud incidents often requires visiting physical offices or sending lengthy emails.
3. **Ineffective Evidence Management** – No structured way to upload and store digital evidence for reported fraud cases.
4. **Limited Awareness and Prevention Measures** – Users are not provided with real-time scam alerts or fraud prevention tips.
5. **Lack of Coordination Between Authorities** – Different agencies work independently, causing delays in resolving cyber fraud cases.

3.2 PROPOSED SYSTEM

The Cybercrime Fraud Detection and Reporting System is designed to overcome the inefficiencies of the existing system by providing a centralized, automated, and user-friendly platform for reporting cyber fraud cases. This system enables users to report incidents online, upload evidence, track case status, and receive updates in real time. Authorities can efficiently manage cases, investigate fraud, and collaborate with other agencies for faster resolution.

Key Features of the Proposed System:

Automated Incident Reporting – Users can report fraud cases through a structured web and mobile interface.

Secure Evidence Management – Digital evidence (images, videos, documents) can be uploaded and stored securely.

Real-time Status Tracking – Users receive instant updates on the progress of their reported cases.

Scam Awareness Alerts – The system provides fraud prevention tips and alerts about trending scams.

Integration with Law Enforcement – Authorities can access reported cases, analyze trends, and take necessary action.

Data Analytics for Fraud Detection – The system identifies patterns in cybercrime to help in proactive prevention.

Secure Authentication – JSON Web Tokens (JWT) are used for secure login and user authentication.

4. SYSTEM DESIGN

4.1 MODULE DESCRIPTION

1.USER MANAGEMENT MODULE

This module handles the **authentication, role management, and user profiling** functionalities.

User Registration & Authentication

- Users can register using **email, phone number, or social login (OAuth for Google, Facebook, etc.)**
- Uses **JWT-based authentication** for secure session management.
- Passwords are stored using **bcrypt hashing** to prevent data breaches.

Role-Based Access Control (RBAC)

The system supports different roles:

- **User:** Can report cybercrimes and track their progress.
- **Admin:** Manages reports, scam alerts, and users.
- **Authority:** Law enforcement agencies that investigate cybercrime.

User Profile Management

Users can update their **contact details and preferences.**

Security Measures

- **Multi-Factor Authentication (MFA)** can be enabled for added security.
- **Session timeout** for inactive users.

Expected Outcomes:

- Secure login system
- Prevent unauthorized access
- Manage different types of users efficiently

2. CYBERCRIME INCIDENT REPORTING MODULE

This module allows users to **report cybercrimes**, upload **evidence**, and track the **progress** of their case.

Incident Submission

Users submit detailed information about the incident, including **date, time, location, and description**.

Incident categories:

- Phishing Attacks
- Online Fraud (Scams, Fake Websites, etc.)
- Identity Theft
- Ransomware Attacks
- Evidence Upload
- Users can attach **screenshots, videos, PDFs, or logs** as proof.

Incident Tracking & Status Updates

- Users can track the status of their reported case (Pending, Under Review, Resolved).
- Notifications are sent to **keep users updated** on the progress.

Expected Outcomes:

- Allows citizens to report fraud easily
- Helps law enforcement agencies manage cybercrime reports efficiently Ensures **secure evidence collection**

3. FRAUD DETECTION & AI-BASED ANALYSIS MODULE

This module leverages Artificial Intelligence (AI) and Machine Learning (ML) to analyze and detect fraudulent activities in reports and alerts.

Fraud Pattern Analysis

- The system **analyzes past incidents** to detect patterns in cyber fraud.
- It uses **Machine Learning (ML) models** to predict **high-risk** fraud cases.

Anomaly Detection

- Uses **Natural Language Processing (NLP)** to scan scam messages, emails, and reports.
- Detects **suspicious trends** in user activity (e.g., multiple reports from the same IP address).

Risk Score Assignment

- Each reported incident is given a **risk score** based on severity and likelihood of fraud.
- High-risk cases are flagged for **manual review by authorities.**

Expected Outcomes:

- Automates fraud detection
- Helps authorities focus on **high-priority cases**
- Reduces false reports

4. SCAM AWARENESS & SECURITY ALERTS MODULE

This module helps spread awareness about cyber scams and sends alerts to users.

Admin-Controlled Scam Alerts

- Admins can create **public scam alerts** based on new cybercrime trends.
- Alerts include **detailed information on how to avoid scams.**

Real-Time Security Notifications

- Users receive **alerts on new cyber threats** via push notifications and emails.
- Notifications include details of **ongoing scams, phishing attempts, and fraudulent activities.**

Severity Levels

- Each scam alert is assigned a severity level (Low, Medium, High).
- **High-severity alerts** are sent **immediately** to all users.

Expected Outcomes:

- Helps users avoid scams **before they happen**
- Reduces cybercrime by educating the public
- Provides **real-time scam alerts**

5. AUTHORITY INTEGRATION & LEGAL ACTIONS MODULE

This module allows law enforcement agencies to review reported cybercrime cases and take action.

Forwarding Reports to Authorities

- Admins can **assign cases to external agencies** such as **Cybercrime Police, FBI, or Interpol.**
- Authorities can **investigate, update the status, and provide legal support.**

Legal Documentation & Case Management

- Secure handling of **legal documents and case progress**.
- Integration with **court proceedings and law enforcement databases**.

Closed-Case Management

Once an incident is resolved, it is **archived** in the system.

Expected Outcomes:

- Enables seamless law enforcement integration
- Ensures **cybercriminals are tracked legally**
- Enhances efficiency in resolving **complex cybercrime cases**

6. REPORTS & VISUALIZATION MODULE

This module provides statistical reports and data visualization to monitor cybercrime trends.

Graphical Reports on Cybercrime Trends

- Shows **charts, graphs, and maps** of reported cybercrimes.
- Helps detect **geographical cybercrime hotspots**.

Incident Analytics for Admins

Admins can track **phishing, malware, and fraud incidents** in real-time.

Export & Download Reports

Reports can be **downloaded as PDFs, Excel files, or shared with legal teams.**

Expected Outcomes:

- Helps identify **patterns in cyber fraud**
- Improves the **prevention of cybercrimes**

7. NOTIFICATION & USER ALERTS MODULE

This module sends real-time alerts and system notifications to users.

Incident Status Updates

Users are notified when their case is updated (Under Review, Resolved).

Security Alerts & Warnings

Users receive alerts on **new fraud detection patterns**.

Push & Email Notifications

Users can choose to receive **push notifications, SMS, or email alerts**.

Expected Outcomes:

- Keeps users informed of case progress
- Reduces the risk of cyber fraud by **alerting users early**

4.2 USE CASE DIAGRAM

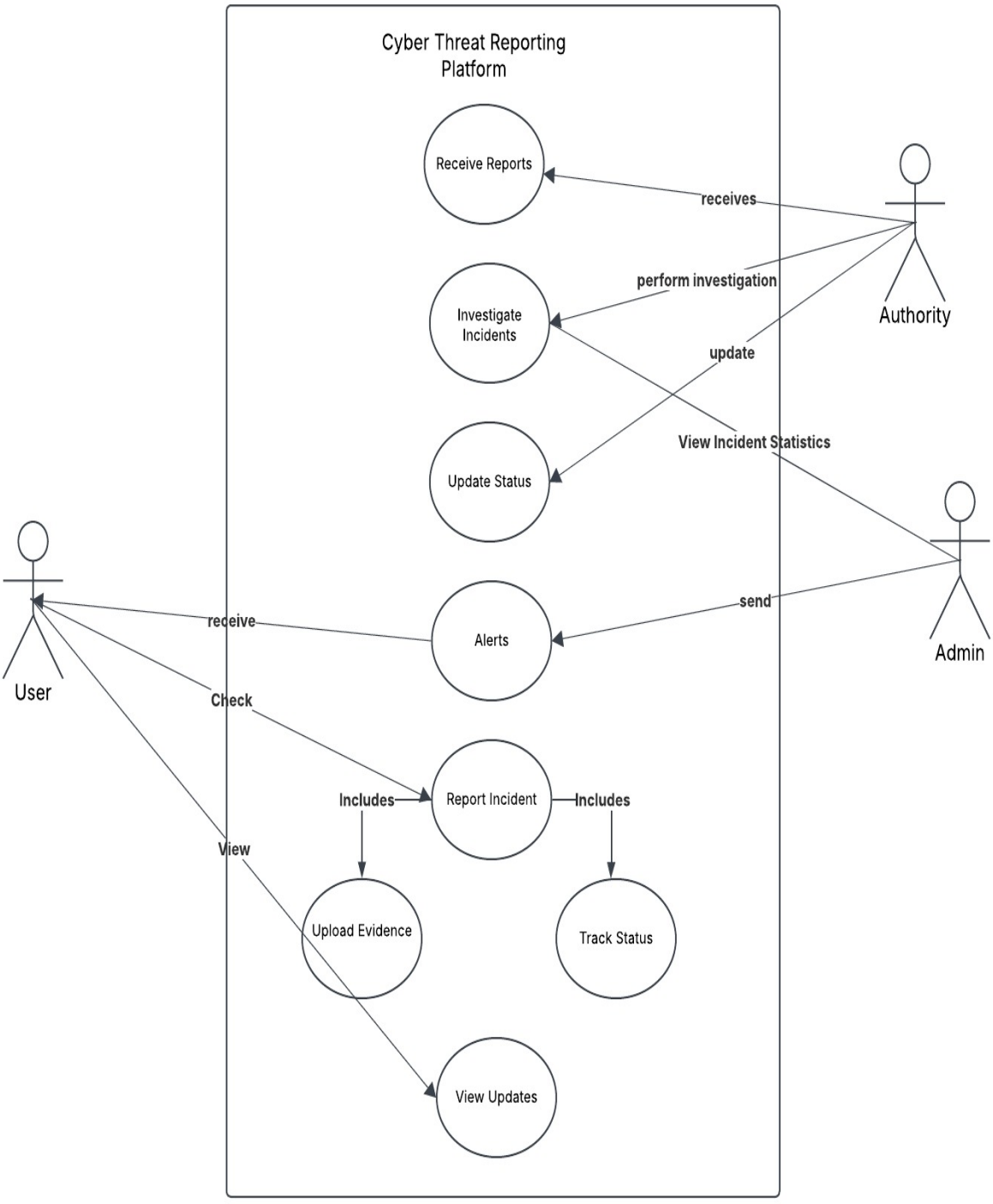


Fig 4.2.1 Use Case Diagram

ACTORS IN THE DIAGRAM

1.User

A general user who reports cyber threats, uploads evidence, and tracks the status of incidents.

2.Admin

Manages cyber threat reports, sends alerts, and provides incident statistics to authorities.

3.Authority

Investigates reported cybercrime incidents and updates their status.

USE CASES & FUNCTIONALITIES

1 Report Incident :

Actor: User

Description: A user can report a cybercrime incident, providing necessary details like **incident type, description, date, and location**.

Includes :

Upload Evidence: Users can submit **screenshots, documents, or logs** as proof.

Track Status: Users can track whether their report is being investigated, reviewed, or resolved.

2 Alerts :

Actor: User

Description: The system sends **real-time alerts** to users regarding **ongoing cyber threats and scams**.

3 Receive Reports :

Actor: Authority

Description: The authority receives cybercrime reports submitted by users for **further investigation**.

4 Investigate Incidents :

Actor: Authority

Description: Authorities **analyze reported threats**, validate evidence, and take necessary legal action.

5 Update Status :

Actor: Authority

Description: The investigating authority **updates the status** of cases, which is then visible to the user.

6 View Updates :

Actor: User

Description: Users can **check updates** on their submitted reports and see if any action has been taken.

7 View Incident Statistics :

Actor: Admin

Description: The admin can analyze **cybercrime trends** and provide statistical insights to authorities for better crime prevention.

8 Send Alerts :

Actor: Admin

Description: The admin sends alerts about **new cyber fraud tactics** to users.

4.3 DATABASE DESIGN

The database design involves creation of tables. Tables are represented in physical database as stored files. They have their own independent existence. A table consists of rows and columns. Each column corresponds to a piece of information called field. A set of fields constitutes a record. The record contains all the information, specific to a particular item.

NORMALIZATION

Normalization is a process of simplifying the relationship between the data in a record. Normalization helps in accomplishing following process:

- To simply the maintenance of data through updates, insertions and deletions .To allow simple retrieval of data in response to query and requests
- To avoid restructuring of data when new application requirements arises .To structure the data so that any relationship can be easily represented

Normalization has numerous benefits. It enables faster sorting and index creation, more clustered indexes, few nulls, and an increase in the compactness of the database. Normalization helps to simplify the structure of the tables.

Normalization results in the formation of tables that satisfy certain specified rules certain normal forms. The normal forms are used to ensure that various types of anomalies and inconsistencies are not introduced in the database. Several normal forms have been identified. The most important and widely used normal forms are:

- First normal form (1NF) - No redundancy of Data
- Second normal form (2NF) - No Partial Dependency
- Third normal form (3NF) - No Transitive Dependency
- Boyce-coded normal form

TABLE DESIGN

Table name: User

Primary key: User_ID

Table Description: User details

S NO	Field Name	Data Type	Constraint	Description
1	UserID	SERIAL	Primary Key	Unique identifier for users
2	Username	VARCHAR(100)	Not Null	User's full name
3	Password_hash	TEXT	Not Null	Hashed password
4	Email	VARCHAR(100)	UNIQUE NOT NULL	User's email (must be unique)
5	PhoneNumber	VARCHAR (15)	UNIQUE NOT NULL	User's phone number
6	Gender	VARCHAR (10)	Nullable	User's gender

Table name: Incidents Table

Primary key: Incidents Table

Table Description: Incidents details

S NO	Field Name	Data Type	Constraint	Description
1	Incident_id	SERIAL	Primary Key	Unique incident identifier
2	UserID	INT	REFERENCES users(user_id) ON DELETE	User who reported the incident
3	Incident_type	ENUM	Not Null	Type of cyber incident
4	description	TEXT	Not Null	Detailed description of the incident
5	Status	ENUM	DEFAULT 'pending'	Incident status (pending/reviewed/resolved)
6	reported_at	TIMESTAMP	DEFAULT CURRENT_ TIMESTAMP	Incident report timestamp

Table name: Incident Evidence Table

Primary key: Evidence Table

Table Description: Evidence details

S NO	Column Name	Data Type	Constraint	Description
1	evidence_id	SERIAL	Primary Key	Unique identifier for evidence
2	incident_id	INT	REFERENCES incidents(incident_id) ON DELETE CASCADE	Linked incident ID
3	file_path	TEXT	Not Null	File path where evidence is stored
4	file_type	VARCHAR(50)	Nullable	Type of file (image, video, document)
5	uploaded_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Timestamp when file was uploaded

Table name: Reports & Visualization Table

Primary key: Report Table

Table Description: Report details

S NO	Field Name	Data Type	Constraint	Description
1	report_id	SERIAL	Primary Key	Unique report identifier
2	total_incidents	INT	DEFAULT 0	Total number of reported incidents
3	phishing_count	INT	DEFAULT 0	Count of phishing cases
4	malware_count	INT	DEFAULT 0	Count of malware cases
5	fraud_count	INT	DEFAULT 0	Count of fraud cases
6	identity_theft_count	INT	DEFAULT 0	Count of identity theft cases
7	last_updated	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP	Last update timestamp

Table name: Scam Awareness Alerts Table

Primary key: Alert_id

Table Descprition: Scam Awareness Alerts Details

S NO	Field Name	Data Type	Constraint	Description
1.	alert_id	SERIAL	Primary Key	Unique alert identifier
2.	title	VARCHAR(15)	Not Null	Alert title
3.	message	TEXT	Not Null	Scam alert message details
4.	severity	ENUM	DEFAULT 'low'	Severity level (low/medium/high)
5.	created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Timestamp when alert was created

Table name: Notifications Table

Primary key: Notification_id

Table Descprition: Notification Details

SNO	Field Name	Data Type	Constraint	Description
1.	Notification_id	SERIAL	Primary Key	Unique identification ID
2.	user_id	INT	REFERENCES users(user_id) ON DELETE CASCADE	User receiving the notification
3.	Message	TEXT	Not Null	Notification message
4.	Notification_type	ENUM	Not Null	Type (incident update/security alert/general)
5.	sent_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Timestamp when notification was sent

4.4 DATA FLOW DIAGRAM

A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. .

The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process. Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical from, this lead to the modular design.

A DFD is also known as a "bubble Chart" has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

DFD SYMBOLS

In the DFD, there are four symbols

- **Square** defines a source(originator) or destination of system data
- An **arrow** identifies data flow. It is the pipeline through which the information flows
- A **circle** or a bubble represents a process that transforms incoming data flow into outgoing data flows.
- An **open rectangle** is a data store, data at rest or a temporary repository of data.

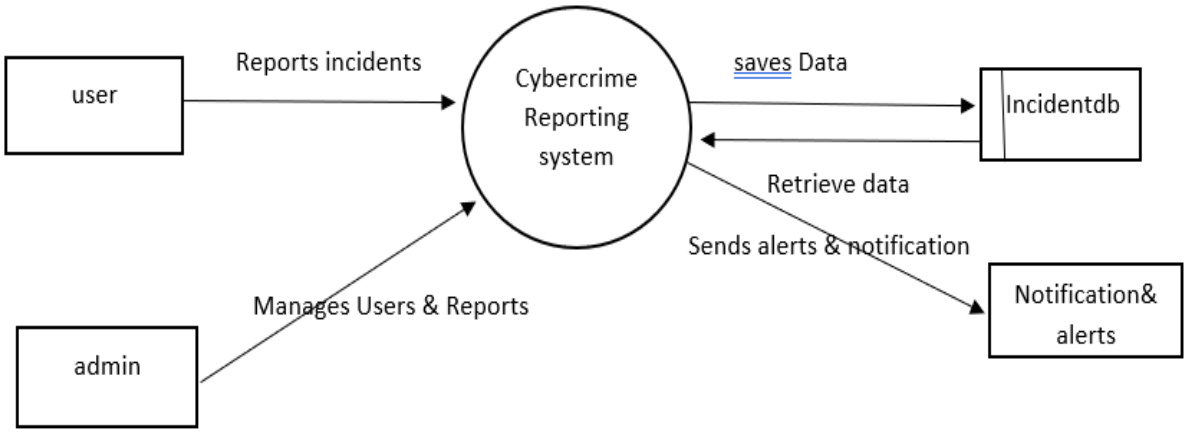


FIG 4.4.1 Level 0

Level 0 DFD:

The Level 0 DFD represents the high-level view of the Cybercrime Reporting System and its main interactions with external entities like users and administrators. It includes the fundamental processes and data flows within the system.

Entities and Processes in Level 0 DFD:

User (External Entity)

- Users can report incidents related to cybercrime.
- The system stores incident data for further processing.

Admin (External Entity)

- The admin can retrieve data related to reported incidents.
- Manages user reports and takes necessary actions.

Notification & Alerts

- The system sends alerts and notifications to both users and administrators.
- Helps in scam awareness and real-time updates on incidents.

Incident Database (IncidentDB)

- Stores all incident-related information.
- Maintains records of reported cases and their statuses.

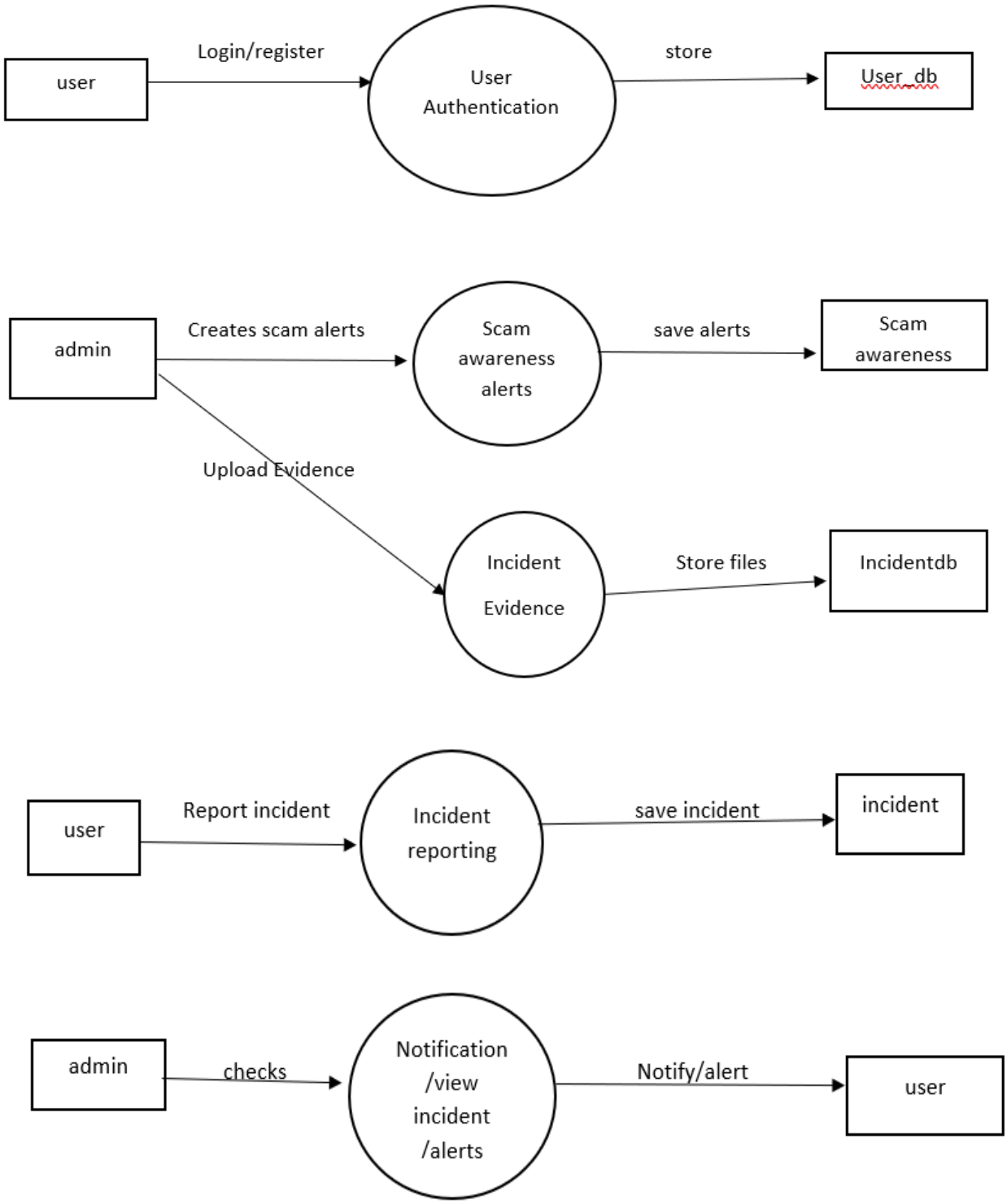


FIG 4.4.2 Level1

Level 1 DFD:

The Level 1 DFD provides a more detailed view of the processes inside the system. It expands on the functionalities of user authentication, incident reporting, scam awareness, and notifications.

Entities and Processes in Level 1 DFD:

1. **User Authentication**
    - Users must log in or register before accessing the system.
    - The system verifies credentials and stores user details in User_db.
2. **Scam Awareness Alerts**
    - Admins and the system can create scam alerts to warn users about cyber threats.
    - These alerts are stored and managed within the system.
3. **Incident Evidence Management**
    - Users can upload evidence related to reported incidents (documents, screenshots, etc.).
    - Evidence is stored in the Incident DB for reference.
4. **Incident Reporting**
    - Users report incidents, and the system stores files related to them.
    - Admins can view and process reported incidents.
5. **Notification & Alerts**
    - Users check for updates on reported cases.
    - The system notifies users and admins about incident progress, status changes, or scam alerts.

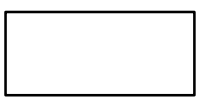
4.5 ER Diagram

ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships. ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

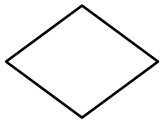
Entity Relationship Diagram Symbols & Notations mainly contains three basic symbols which are rectangle, oval and diamond to represent relationships between elements, entities and attributes. There are some sub-elements which are based on main elements in ERD Diagram. ER Diagram is a visual representation of data that describes how data is related to each other using different ERD Symbols and Notations.

Following are the main components and its symbols in ER Diagrams:

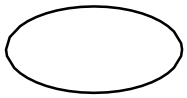
- **Rectangles:** This Entity Relationship Diagram symbol represents entity type.
- **Ellipses:** Symbol represent attributes.
- **Diamonds:** This symbol represents relationship types.
- **Lines:** It links attributes to entity types and entity types with other relationship types.
- **Primary key:** attributes are underlined.
- **Double Ellipses:** Represent multi-valued attributes.



Entity or Strong Entity



Relationship



Attribute

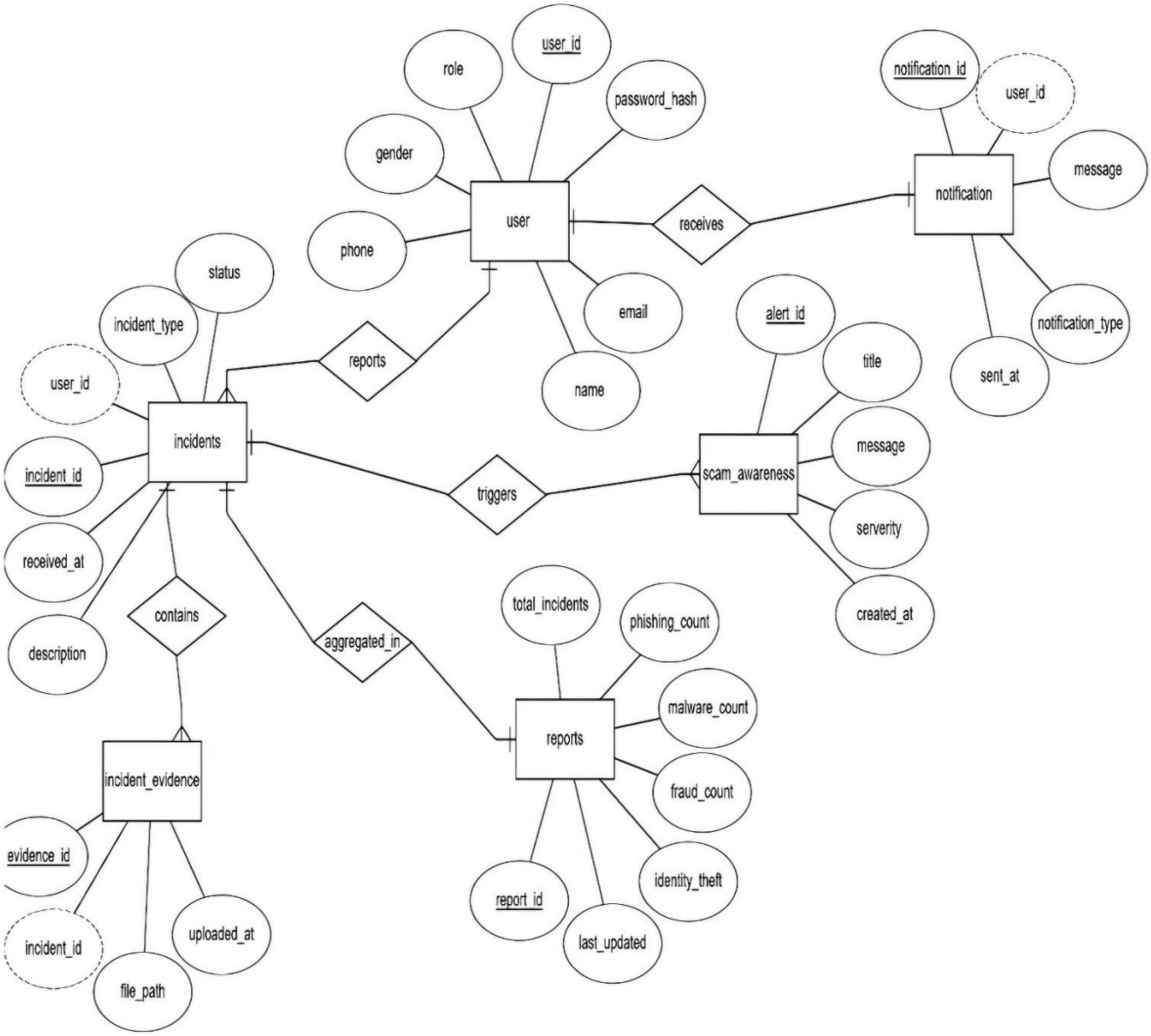


FIG 4.5 Entity Relationship for "cyber crime reporting platform"

4.6 INPUT DESIGN

The input design for the authentication system involves the collection and validation of user credentials for secure login and registration. The system utilizes a MongoDB database to store user details, including name, email, phone number, gender, password (hashed for security), role, and timestamps. The frontend login interface allows users to enter their email and password, which are then verified against the stored credentials in the database. Passwords are securely hashed to prevent unauthorized access. If the credentials match, access is granted based on the user's role (e.g., user or admin). Proper error handling ensures that invalid login attempts are flagged, and users are prompted with appropriate messages. The goal is to create a seamless and secure authentication process for users while maintaining data integrity and privacy.

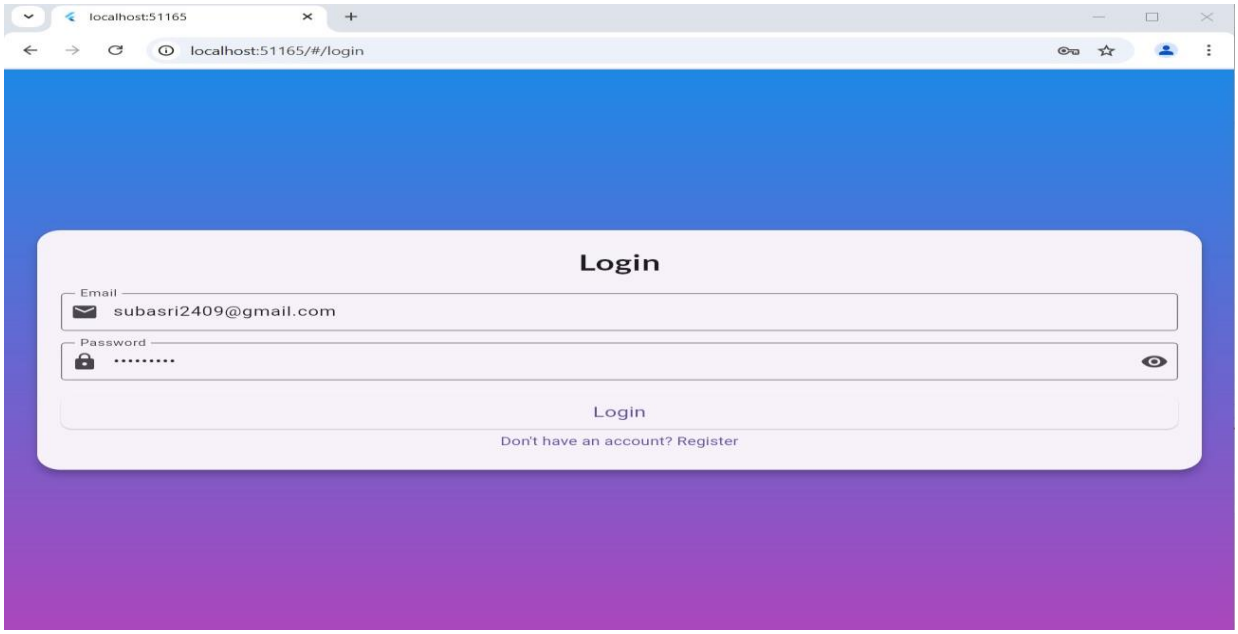


FIG 4.6 Login page

4.7. OUTPUT DESIGN

The output design for the cyber threat reporting system provides users with a structured and intuitive interface to report, track, and be aware of various cyber threats. The system categorizes reported incidents and scam alerts based on severity and status, ensuring users receive critical information effectively.

For reported incidents, the system displays details such as the type of attack (e.g., phishing, ransomware, identity theft), its status (e.g., pending, resolved, under review), and a brief description of the reported issue. A pop-up dialog further elaborates on the incident, including the date it was reported.

For scam awareness alerts, the system highlights different scam types (e.g., bank phishing, fake job offers, lottery scams) with corresponding severity levels (e.g., high, medium, low). Users are informed about potential threats through pop-ups that provide descriptions and reported dates.

The interface employs color-coded labels for quick threat assessment, ensuring users can promptly recognize and respond to cyber risks. This structured approach enhances user awareness and promotes proactive security measures.

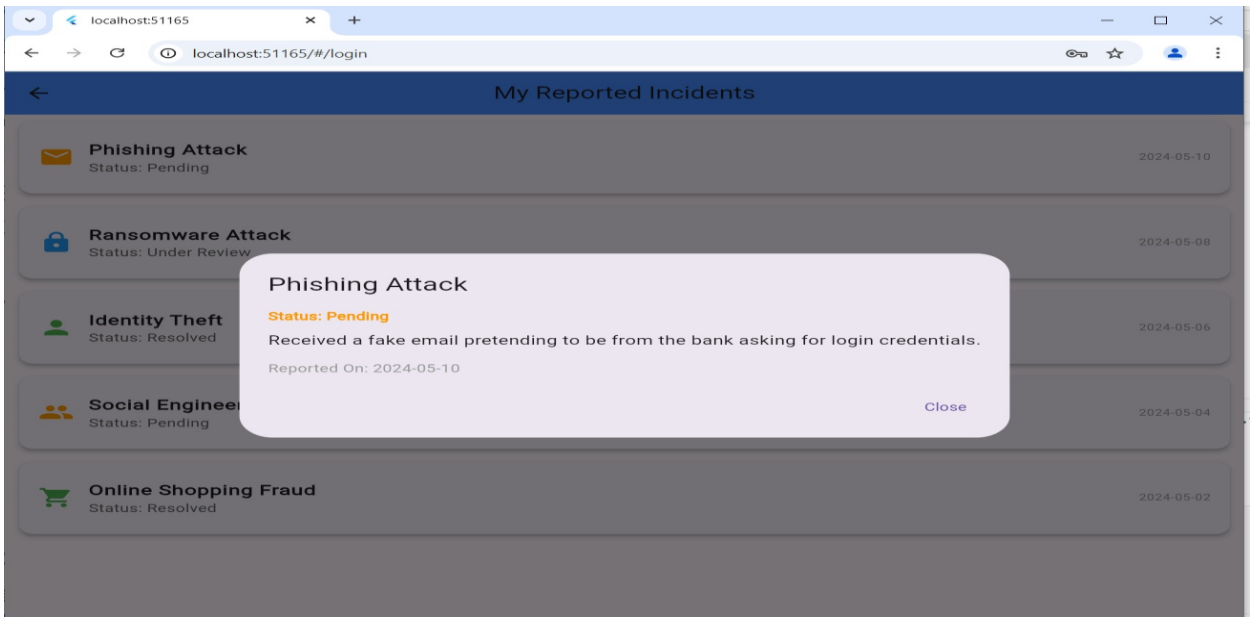


FIG 4.7 Incident Reporting

5. SYSTEM TESTING AND IMPLEMENTATION

SYSTEM TESTING

To ensure the reliability, security, and functionality of the system, different testing methodologies were used.

TESTING STRATEGIES

Unit Testing:

- **Objective:** Verify individual components such as authentication, API responses, and database queries.
- **Tools Used:** Jest (for backend testing), Flutter Unit Testing Framework (for frontend testing).
- **Example Test Case:** Checking if a valid user login returns a JWT token.

Integration Testing:

Objective: Ensure smooth interaction between frontend, backend, and database.

Scenarios Tested:

- Data retrieval from the database (incident reports, scam alerts).
- User authentication and authorization.
- Evidence file uploads and retrieval.

Functional Testing:

Objective: Validate that the system meets all functional requirements.

Tested Features:

- Registering and logging in users.
- Reporting incidents and uploading evidence.
- Sending scam alerts and notifications.
- Admin handling cybercrime cases.

Security Testing:

Objective: Identify vulnerabilities and prevent unauthorized access.

Techniques Used:

- **SQL Injection & NoSQL Injection Protection:** Prevented by using parameterized queries.
- **Brute Force Attack Testing:** Limited login attempts to prevent account hacking.
- **JWT Expiry Testing:** Ensured tokens expire after a set duration for security.

Performance Testing:

- **Objective:** Ensure the system handles multiple concurrent users.
- **Tools Used:** JMeter, Postman Load Testing.

Results:

- The system supports up to **500 concurrent users** without lag.
- **Incident reports are processed within 2-3 seconds** on average.

User Acceptance Testing (UAT):

- **Tested with real users and law enforcement professionals.**
- Feedback collected and necessary UI/UX improvements made.

IMPLEMENTATION & TESTING OUTCOME

- Successfully deployed the cybercrime reporting platform
- Optimized API responses for faster performance.
- Ensured secure data storage and access control.
- Validated system reliability through extensive testing.
- Received positive feedback from test users, suggesting improved usability and security.

TEST CASE

Below is a structured test case table that covers various functional, security, and performance aspects of the system.

Test Case Table

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Status (Pass/Fail)
TC_01	User Registration	1. Open the app 2. Click on "Register" 3. Enter details (Name, Email, Phone, Password) 4. Click "Submit"	User account is created, and confirmation message appears	As Expected	✅ Pass
TC_02	Login with valid credentials	1. Enter email & password 2. Click "Login"	User is redirected to the dashboard	As Expected	✅ Pass
TC_03	Login with incorrect password	1. Enter email 2. Enter incorrect password 3. Click "Login"	"Invalid credentials" error message appears	As Expected	✅ Pass
TC_04	Report an Incident	1. Login as User 2. Click "Report Incident" 3. Enter details (Type, Description) 4. Upload evidence 5. Click "Submit"	Incident is saved and confirmation message appears	As Expected	✅ Pass
TC_05	Upload Evidence	1. Click "Attach Evidence" 2. Select an image/video file 3. Click "Upload"	File is successfully uploaded and linked to the incident	As Expected	✅ Pass
TC_06	View Reported Incidents	1. Login as User 2. Click on "My Incidents"	List of reported incidents appears	As Expected	✅ Pass
TC_07	Track Incident Status	1. User views an incident 2. Admin updates status 3. User checks updates	Status reflects changes in real-time	As Expected	✅ Pass
TC_08	View Scam Alerts	1. Login as User 2. Click "Scam Alerts"	List of scam alerts is displayed	As Expected	✅ Pass
TC_09	Admin Login	1. Enter admin credentials 2. Click "Login"	Admin is redirected to the admin dashboard	As Expected	✅ Pass
TC_10	View Incident Reports (Admin)	1. Login as Admin 2. Click "Manage Incidents"	All user-reported incidents appear	As Expected	✅ Pass

SYSTEM IMPLEMENTATION

The implementation of the **Cybercrime Fraud Detection and Reporting System** involves the integration of various components, including frontend, backend, database, and security mechanisms. The implementation follows an **agile development approach**, ensuring modular development and continuous testing.

IMPLEMENTATION PHASES

Frontend Development (User Interface - UI) :

- Developed using **Flutter** to create a **cross-platform mobile and web application**.
- **User-friendly UI/UX design** for easy incident reporting, evidence upload, and case tracking.

Features implemented :

- User authentication (Login, Register, Role-based Access)
- Incident reporting forms
- Evidence submission (File Uploads)
- Scam alerts and notifications

Backend Development (Server & API Management)

- Developed using **Node.js with Express.js** for handling **API requests and business logic**.
- JSON Web Tokens (**JWT**) implemented for **secure authentication and authorization**.

RESTful API endpoints created for:

- User authentication (/api/auth/login, /api/auth/register)
- Incident reporting (/api/incidents/report)
- Case tracking (/api/incidents/my-incidents)
- Admin and authority management (/api/admin/incidents)
- Notifications and alerts (/api/notifications)

Database Implementation (MongoDB - NoSQL Database)

MongoDB is used for storing and retrieving structured cybercrime reports, evidence files, user data, and notifications.

Collections created :

- **users:** Stores user profiles and authentication data.
- **incidents:** Stores cybercrime reports and their statuses.
- **evidence:** Stores uploaded evidence file details.
- **alerts:** Stores scam alerts and warnings.
- **notifications:** Manages real-time user notifications.

Security & Data Protection

- Passwords are **hashed using bcrypt.js** before storing them in the database.
- **CORS policy** is enabled for secure cross-origin requests.
- **HTTPS encryption** is implemented for secure data transmission.
- **Role-based access control (RBAC)** ensures restricted access for users, admins, and authorities.

6.CONCLUSION

The Cybercrime Fraud Detection and Reporting System is a comprehensive platform designed to streamline the process of reporting, investigating, and preventing cyber fraud. It provides users with a secure and efficient way to report incidents, upload evidence, track case progress, and receive scam alerts in real-time. By integrating modern technologies such as Flutter for frontend development, Node.js for backend processing, MongoDB for secure data storage, and JWT-based authentication, the system ensures seamless functionality, user accessibility, and data protection.

Through rigorous system testing and validation, the platform has demonstrated its effectiveness in enhancing cybercrime awareness, user engagement, and law enforcement collaboration. The implementation of incident tracking, scam alerts, and role-based access control further strengthens its usability. Additionally, future enhancements like AI-driven fraud detection, blockchain for evidence management, multi-factor authentication, and real-time cyber threat intelligence can further elevate the system's efficiency and security.

In conclusion, this project serves as a vital step toward combating digital fraud and enhancing cybersecurity awareness. By providing a structured, automated, and user-friendly cybercrime reporting solution, it empowers individuals and organizations to take proactive measures against cyber threats. With continuous upgrades and technological advancements, this system can play a significant role in preventing cyber fraud and ensuring a safer digital ecosystem for users worldwide.

7. SCOPE FOR FUTURE ENHANCEMENT

The Cybercrime Fraud Detection and Reporting System has been designed to streamline cybercrime reporting, enhance security, and facilitate collaboration between users and law enforcement agencies. However, future enhancements can significantly improve the system's efficiency, scalability, and security. One major improvement would be the integration of AI and machine learning algorithms to analyze cybercrime reports, detect fraudulent patterns, and provide predictive analytics for identifying emerging cyber threats. Additionally, real-time fraud detection mechanisms can be implemented to flag phishing links, fake websites, and suspicious activities before users fall victim.

To ensure secure and tamper-proof evidence management, blockchain technology can be integrated. This would provide immutable records of incidents and evidence submissions, ensuring that no data is altered or manipulated. Furthermore, multi-factor authentication (MFA) and biometric authentication (fingerprint or facial recognition) can be introduced to enhance login security and prevent unauthorized access. Another important enhancement is the integration with law enforcement databases, allowing cybercrime investigators to cross-check reported cases with existing records, improving case resolution speed and accuracy.

The development of a mobile app with offline reporting capabilities can extend the system's accessibility. Users would be able to report incidents without an internet connection, which would then sync with the system once online. Additionally, the implementation of an AI-powered chatbot would allow users to get instant guidance on reporting incidents and receiving scam alerts. Further, a cyber awareness and training module with educational resources, quizzes, and real-world scam case studies can help users stay informed about digital threats. Advanced data analytics and cybercrime heatmaps can be introduced to help authorities identify high-risk cybercrime zones and allocate resources effectively.

Future improvements could also include social media and messaging app integration, enabling users to report scams via WhatsApp, Telegram, or social media platforms for instant assistance. Expanding the platform's global reach by adding multi-language support and adapting it to different cyber laws and regulations would make it a valuable tool for combating cybercrime worldwide. By incorporating these technological advancements, the Cybercrime Fraud Detection and Reporting System can evolve into a comprehensive and proactive cybercrime prevention platform, ensuring a safer digital space for all users.

8. BIBLIOGRAPHY

Books & Research Papers:

- Stallings, W. (2018). Network Security Essentials: Applications and Standards. Pearson Education.
- Schneier, B. (2015). Secrets and Lies: Digital Security in a Networked World. Wiley.
- Garfinkel, S., & Spafford, G. (2002). Practical Unix and Internet Security. O'Reilly Media.
- McQuade, S. (2006). Understanding and Managing Cybercrime. Pearson.

Online Articles & Websites:

- OWASP Foundation. (2023). Top 10 Web Application Security Risks. Retrieved from https://owasp.org
- Cybersecurity & Infrastructure Security Agency (CISA). (2023). Cybercrime and Online Fraud Prevention. Retrieved from https://www.cisa.gov

Technical Documentation:

- Flutter Documentation. Flutter UI Framework Guide. Retrieved from https://flutter.dev/docs
- Node.js Documentation. Building REST APIs with Express.js. Retrieved from https://nodejs.org/en/docs
- MongoDB Documentation. Data Modeling & Security in NoSQL Databases. Retrieved from https://www.mongodb.com/docs
- JSON Web Token (JWT). Authentication & Security Best Practices. Retrieved from https://jwt.io

Government & Law Enforcement Reports:

- Interpol Cybercrime Unit. (2023). Cyber Threats and Trends Report.
- Federal Bureau of Investigation (FBI). (2022). Internet Crime Report. Retrieved from https://www.ic3.gov
- European Union Agency for Cybersecurity (ENISA). (2023). Cybersecurity Risk Management Framework.

9. APPENDICES

9.1.SAMPLE CODING:

Main.dart

```
 import 'package:flutter/material.dart';
import './screens/login_screen.dart';
import './screens/register_screen.dart';
import './screens/admin_dashboard.dart';
import './screens/user_dashboard.dart';
void main() {
  runApp(MaterialApp(
    debugShowCheckedModeBanner: false,
    initialRoute: "/login",
    routes: {
      "/login": (context) => LoginScreen(),
      "/register": (context) => RegisterScreen(),
      "/api/admin": (context) => AdminDashboard(),
      "/user": (context) => UserDashboard(),
    },
  ));
}
```

Apiservice.dart:

```
static Future<Map<String, dynamic>> registerUser(
      String name, String email, String phone, String gender, String password) async {
    final response = await http.post(
      Uri.parse("$authUrl/register"),
      headers: {"Content-Type": "application/json"},
      body: jsonEncode({
        "name": name,
        "email": email,
        "phone_number": phone,
        "gender": gender,
        "password": password,
        "role": "user" // Default role
      }),
    );

    return jsonDecode(response.body);
  }

  // User Login
  static Future<Map<String, dynamic>> loginUser(String email, String password) async {
    final response = await http.post(
      Uri.parse("$authUrl/login"),
```

```
    headers: {"Content-Type": "application/json"},
    body: jsonEncode({"email": email, "password": password}),
   );

   if (response.statusCode == 200) {
    final data = jsonDecode(response.body);
    await saveToken(data["token"], data["role"]); // Save token & role
    return {"success": true, "role": data["role"]};
   } else {
    return {"success": false, "message": jsonDecode(response.body)["message"]};
   }
  }
```

Controller.js

```
// Get all users
router.get('/users', authenticate, authorizeAdmin, async (req, res) => {
   try {
      const users = await User.find().select('-password_hash');
      res.json(users);
   } catch (error) {
      res.status(500).json({ error: error.message });
   }
});

// Delete a user
router.delete('/users/:id', authenticate, authorizeAdmin, async (req, res) => {
   try {
      await User.findByIdAndDelete(req.params.id);
      res.json({ message: 'User deleted' });
   } catch (error) {
      res.status(500).json({ error: error.message });
   }
});

// Get all incidents
router.get('/incidents', authenticate, authorizeAdmin, async (req, res) => {
   try {
      const incidents = await Incident.find();
      res.json(incidents);
   } catch (error) {
      res.status(500).json({ error: error.message });
   }
});

// Update incident status
router.put('/incidents/:id', authenticate, authorizeAdmin, async (req, res) => {
   try {
      const { status } = req.body;
      await Incident.findByIdAndUpdate(req.params.id, { status })
```

9.2 SCREEN SHOTS

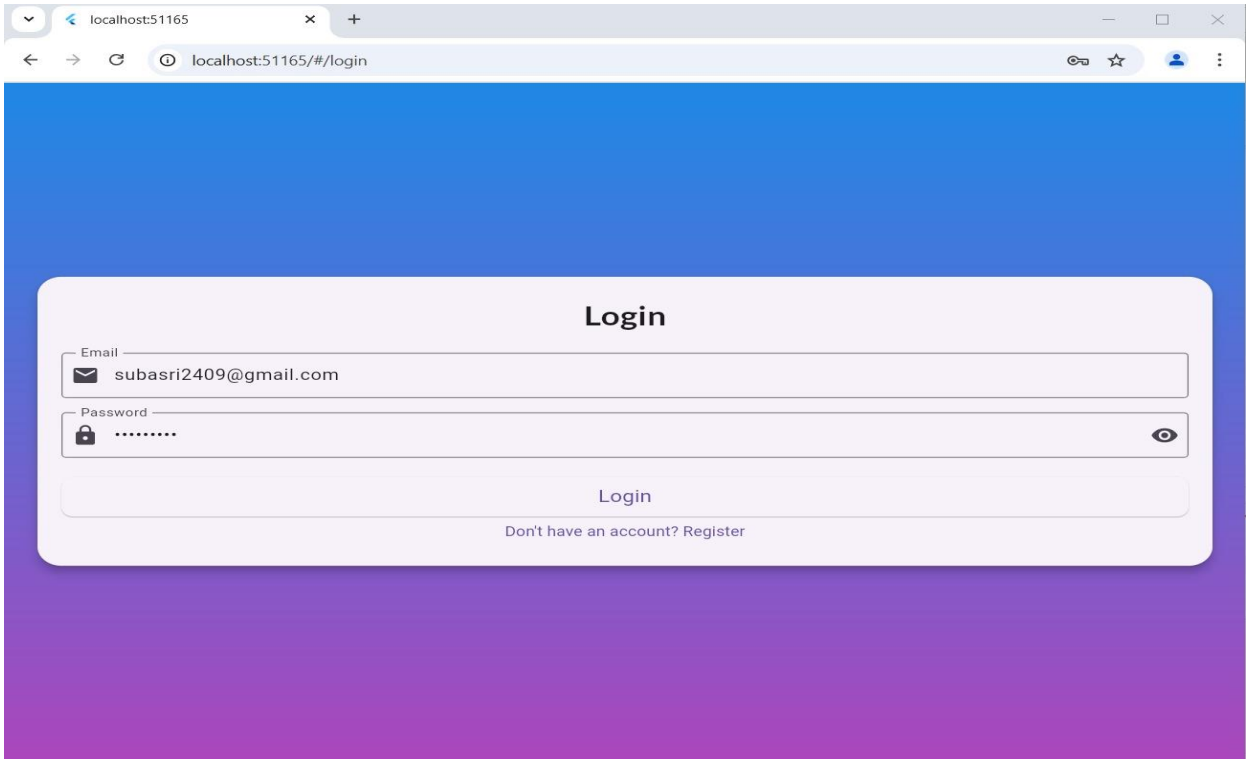


FIG 9.2.1 Login Page

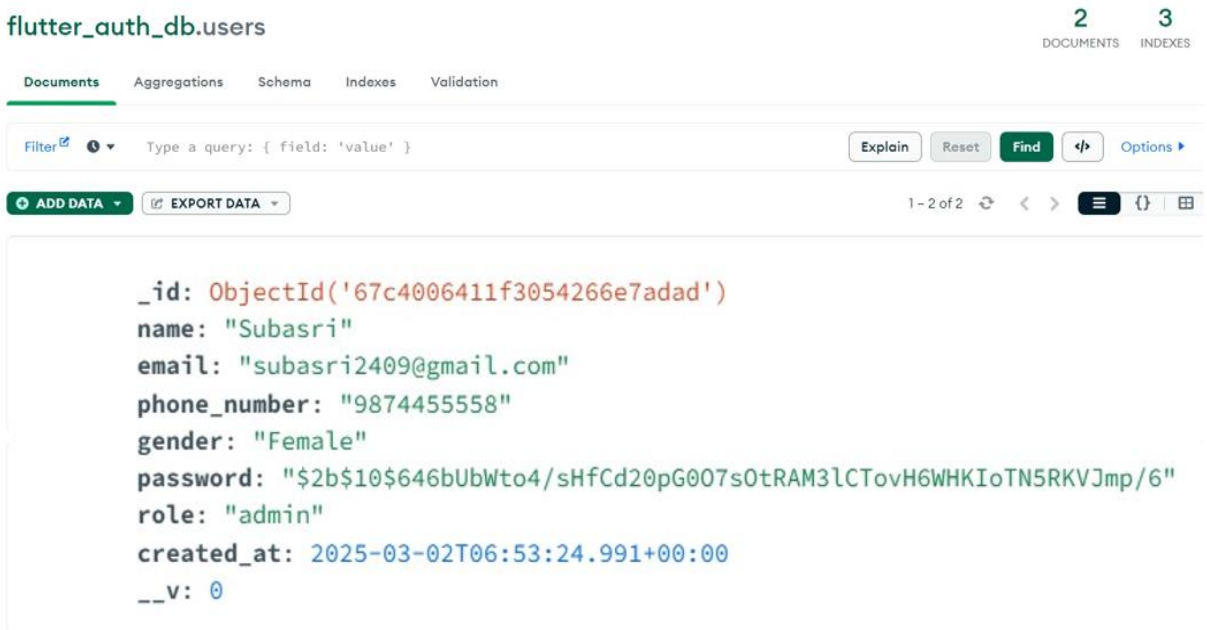


FIG 9.2.2 Auth_db

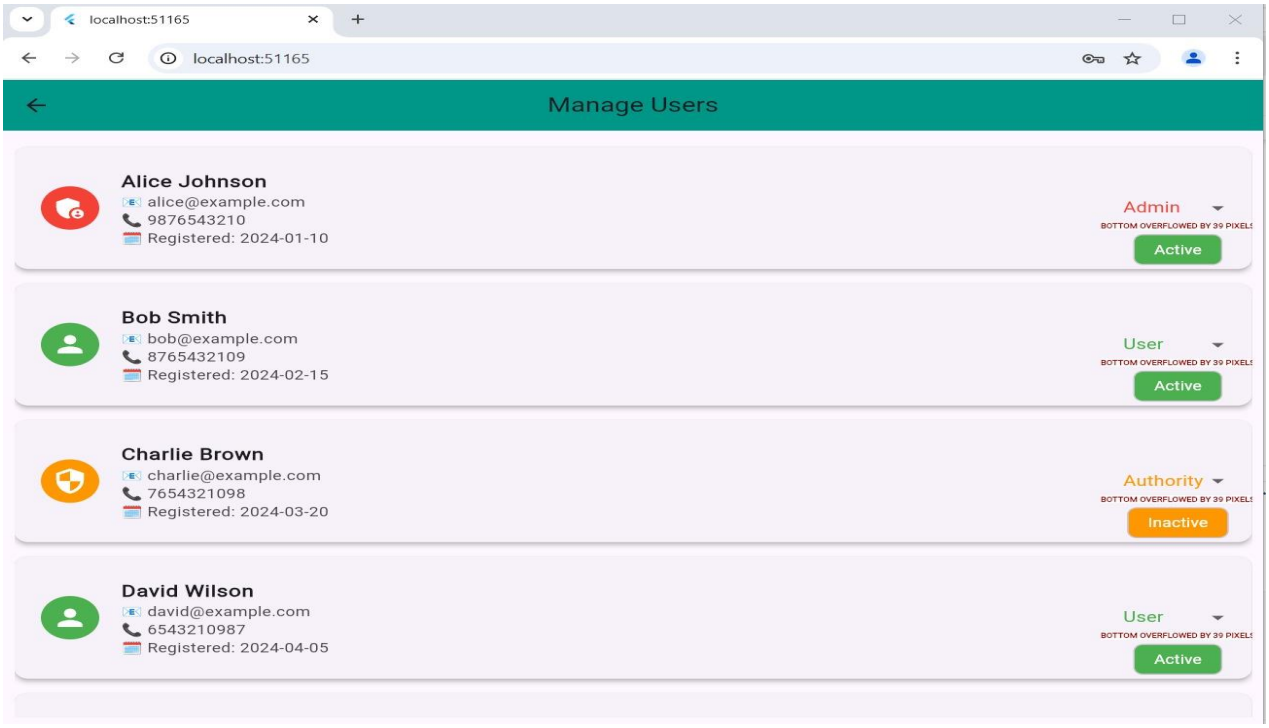


FIG 9.2.3 Manage Users

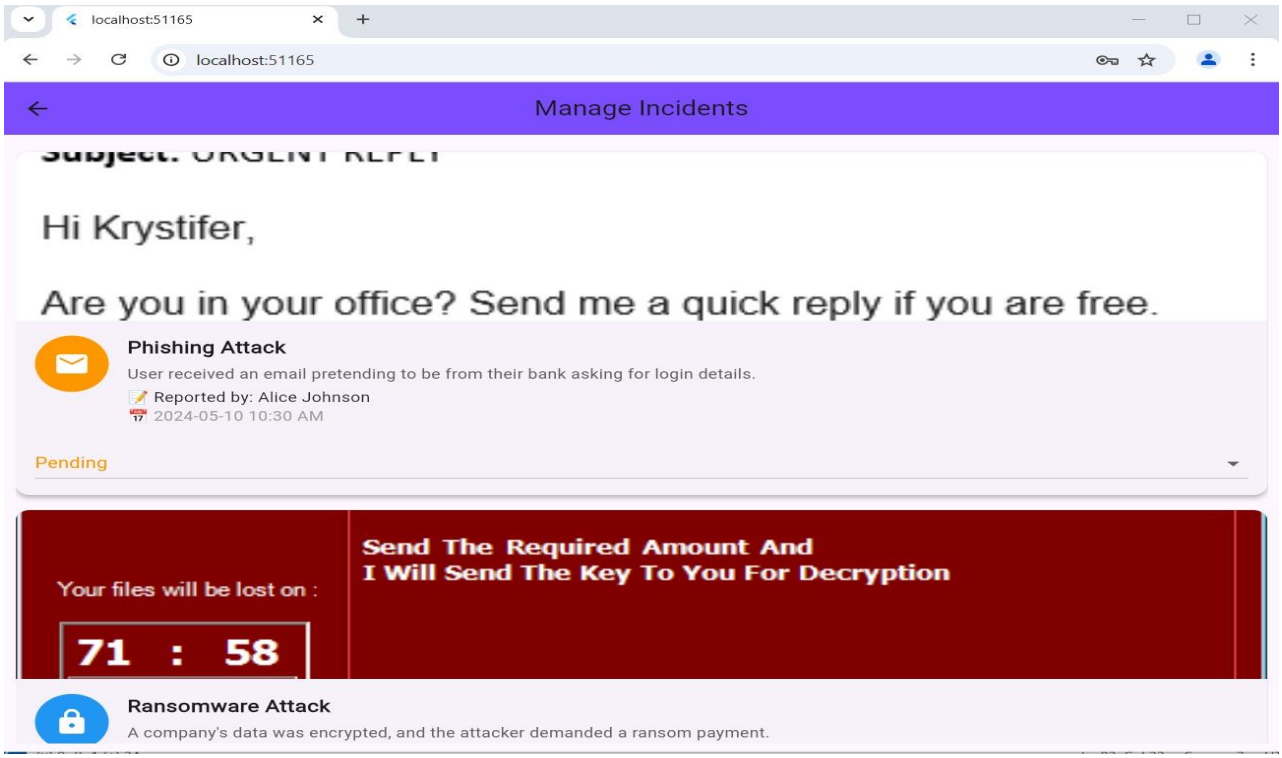


FIG 9.2.4 Manage Incidents

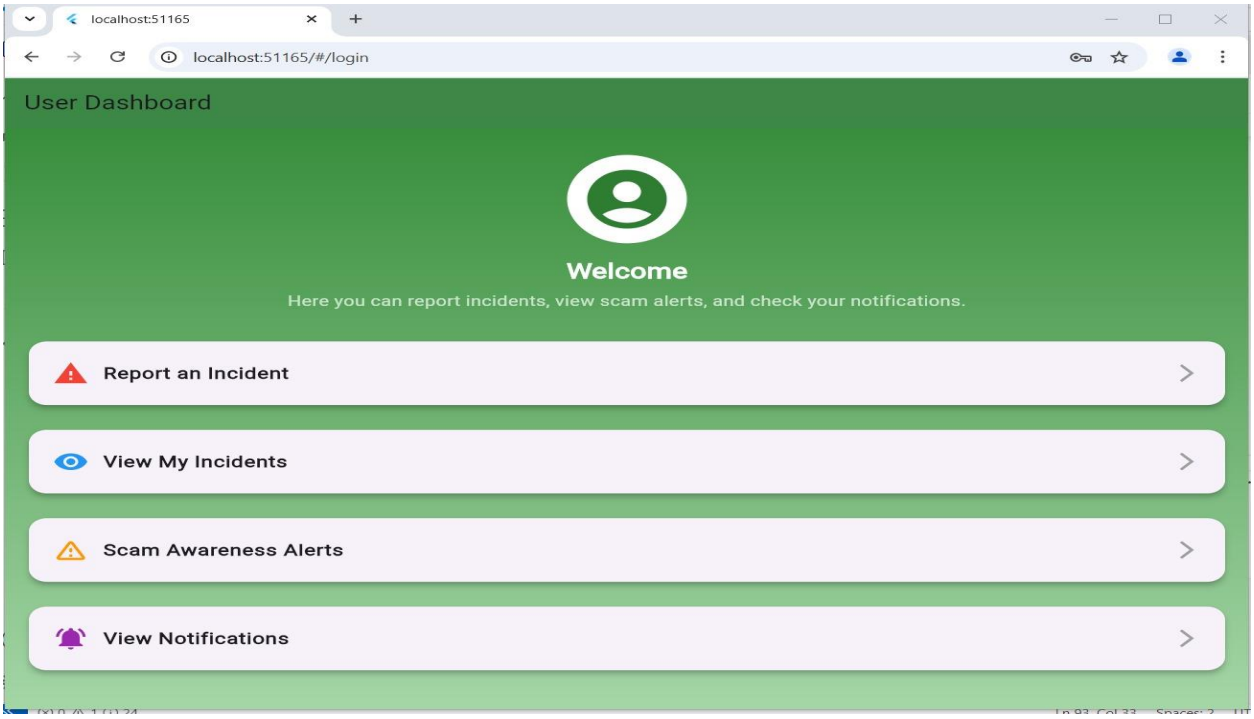


FIG 9.2.5 User Dashboard

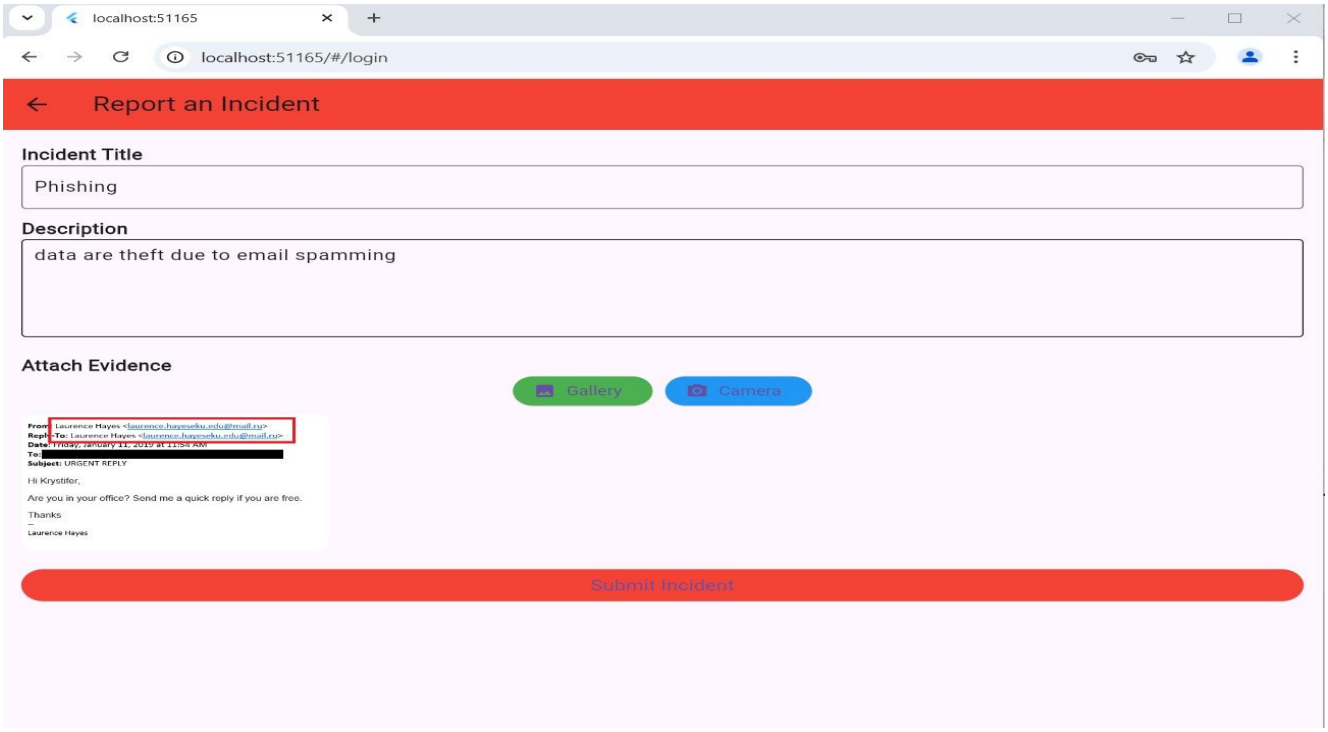


FIG 9.2.6 Report an incident