TRACEBACK

A COMMUNITY-DRIVEN MISSING PERSON NETWORK

BITS ZC4999T: Capstone Project

by

RAPOLU TARUN SAI

TRACEBACK

A COMMUNITY-DRIVEN MISSING PERSON NETWORK

ACKNOWLEDGEMENT

Without the assistance of few people, this endeavour would not have been feasible.
Only when the goal of an activity is achieved can it be considered an accomplishment. Its success requires constant, unwavering work, inspiration, and assistance from its mentors. I would want to use this time to express my gratitude to my friends Jahnavi, Priyatham, and Naveen, who have continuously motivated me to take on new challenges and who have given me the necessary technological knowledge and advice on how to put together this final report. I gained a better understanding of the fundamental use of different technologies and tools by carrying out different tasks and participating in the project's quality assurance.

Contents

Table of Contents

BITS ZC4999T: Capstone Project

ABSTRACT

NAME OF THE STUDENT : Rapolu Tarun Sai

ORGANIZATION & LOCATION

CAPSTONE PROJECT TITLE : TraceBack: A Community-Driven Missing Person Network

ABSTRACT :

The TraceBack project is a web-based platform designed to help locate and track missing individuals. Built with Spring Boot backend and React frontend, it provides a robust authentication system with location-aware features. Users can register and log in securely, with their locations automatically detected or manually entered using geocoding services.

The system employs modern security practices including password encryption and secure session management. The frontend delivers an intuitive user interface with real-time city suggestions and location validation. The backend REST API handles user management, data persistence, and authentication flows.

The project uses Gradle for build automation, incorporates email services, and leverages external mapping APIs for enhanced location accuracy. This solution aims to create a reliable platform for managing and sharing information about missing persons while maintaining data security and user privacy.

The architecture follows best practices with clear separation of concerns between frontend components and backend controllers, making it scalable and maintainable.

Keywords:

- Missing Persons Tracking
- Geolocation Services
- Authentication System
- Web Application
- Spring Boot
- React.js
- RESTful API
- User Management
- Location-based Services
- Real-time City Detection
- Secure Password Management
- Session Handling
- Geocoding Integration
- Email Services
- Database Management

Broad Academic Area of Work:

Software Engineering, Human-Computer Interaction, Geographic Information Systems, Database Systems, Information Security

Chapter 1 - Introduction

In this project is a modern web-based solution designed to address the critical need for an efficient missing persons tracking system. It combines location-aware technology with secure user management to create a platform where users can report and track missing individuals effectively.

To make this application more effectively an email service has been developed so if any match has been found it will trigger a mail on missing person reported user a mail consists of original image, compared image and the match face(if the compared face has more human faces it will crop exactly the matched face)

The steps that can follow to use this application:

1. User Registration
    - Visit the signup page
    - Enter personal details (name, email)
    - Set location (automatic detection or manual entry)
    - Create secure password
    - Complete registration
2. Authentication
    - Login using registered email and password
    - System verifies credentials
    - Creates secure session
3. Main Application Features
    - Report missing persons
    - Track cases
    - Update information
    - Community Support
    - Receive notifications through mail
    - Share profiles via Social Media
    - Manage location settings

1.1 Goal

The MissingPersonsGradleProject establishes a comprehensive centralized platform dedicated to managing and tracking missing persons information efficiently. By leveraging real-time location-based tracking capabilities, the system provides accurate and up-to-date geographical data crucial for locating individuals. The platform prioritizes data security and privacy through robust authentication mechanisms and encrypted storage, ensuring sensitive information remains protected. The system facilitates seamless communication between users, enabling quick and effective information sharing among concerned parties. The implementation focuses on a user-friendly interface that makes navigation and interaction intuitive for all users, regardless of their technical expertise. This integrated approach creates a powerful tool that combines technological innovation with practical utility in the critical domain of missing people's tracking and management.

1.2 Existing Method

Traditional methods for handling missing persons cases primarily rely on two main approaches. The conventional police station filing system involves extensive manual paperwork, where reports are physically filed and stored, leading to geographical limitations and delayed information sharing across jurisdictions. This method suffers from time-consuming processes, limited accessibility, and challenges in maintaining real-time updates. The second common approach involves using social media platforms to share information about missing persons. While this method offers wider reach, it presents challenges such as unstructured information scattered across multiple platforms, lack of verification systems, limited search capabilities, and significant privacy concerns. These existing methods, while serving their basic purpose, fall short in providing organized, secure, and efficient means of tracking and updating missing persons cases. The limitations of these traditional approaches highlight the need for a modern, centralized system that can address these challenges through technology-driven solutions, proper data management, and secure information sharing capabilities.

Chapter 2 - Objectives

The TraceBack encompasses comprehensive technical objectives focused on building a robust authentication system with secure data storage capabilities. The platform leverages a responsive user interface integrated with advanced location services, ensuring a scalable architecture that delivers real-time updates across all devices. The system's foundation is built on providing cross-platform accessibility while maintaining the highest standards of data security and user privacy.

From a user-centric perspective, the objectives prioritize simplifying the reporting process and reducing response time through enhanced location awareness and community participation features. The system emphasizes maintaining data integrity through high availability and support for multiple concurrent users, enabling quick searches and providing powerful analytical capabilities. The security framework implements stringent authentication protocols, protecting user data while ensuring privacy compliance and preventing unauthorized access. The combination of these technical, user-focused, and security objectives creates a powerful platform that effectively serves the critical need for missing persons tracking while maintaining the highest standards of data protection and user experience.

Chapter 3 - Literature Review

3.1 Overview of TraceBack Web Application

TraceBack is a powerful web-based platform built using Spring Boot and React that revolutionizes missing persons tracking and management. The system features a robust authentication system with secure user registration and login functionality, leveraging modern security practices including password encryption and session management.

The platform effectively combines technical innovation with practical utility, creating a centralized system for managing missing persons information while maintaining high standards of data security and user privacy. Its modern architecture ensures scalability and maintainability, making it a valuable tool for communities and organizations involved in missing persons cases.

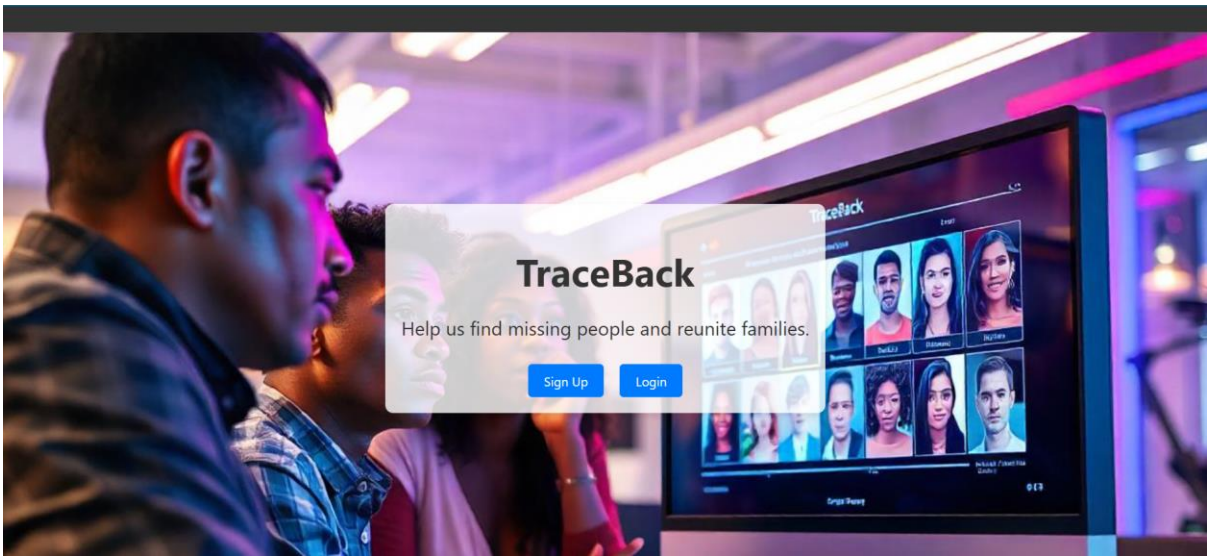

Fig.1 Overview

3.2 Existing Missing Person Tracking Systems

The landscape of missing persons tracking systems has evolved significantly, with various approaches serving different needs. The AMBER Alert System, operated by government agencies, focuses primarily on child abduction cases through broadcast-based alerts. While effective for its specific purpose, it lacks real-time tracking capabilities and interactive user engagement. The National Missing and Unidentified Persons System (NamUs) serves as a comprehensive database but remains limited to official entries with basic search functionality.
Social media platforms have emerged as informal tracking systems, offering wide reach but suffering from unstructured information and verification challenges.

Traditional police database systems, though official, are constrained by jurisdictional boundaries and paper-based processes that delay critical updates. Missing Children Organizations operate specialized systems with focused objectives but limited geographical coverage and basic tracking features.

3.3 Comparison of Different Missing Person Systems

The evolution of Missing Person Tracking systems showcases diverse approaches in addressing this critical need. The AMBER Alert System, a government-operated platform, excels in rapid broadcast alerts for child abduction cases through emergency response networks. Its integration with law enforcement provides official credibility, though it operates within specific case criteria and lacks real-time tracking capabilities.
The National Missing and Unidentified Persons System (NamUs) serves as a centralized database with comprehensive case management features. Its strength lies in maintaining detailed records and facilitating cross-jurisdictional searches, yet it primarily caters to law enforcement and official agencies, limiting public interaction and real-time updates.

3.4 Identification of Gaps in Current Missing Person Tracking Systems

Current missing person tracking systems exhibit several significant gaps that impact their effectiveness. Traditional systems like AMBER Alert and police databases operate in isolated environments, creating information silos that hinder cross-jurisdictional collaboration. The lack of real-time location tracking and update capabilities leads to delayed responses and reduced chances of successful outcomes.

Existing platforms often struggle with data accuracy and verification. Social media-based tracking, while offering wide reach, suffers from unstructured information and reliability issues. The absence of centralized verification mechanisms makes it difficult to distinguish authentic reports from false leads, potentially misdirecting valuable resources.

Technical limitations in current systems restrict their functionality. Many platforms lack modern features like geolocation services, mobile accessibility, and real-time notifications. The user interface of traditional systems often presents barriers to quick information input and retrieval, crucial factors in time-sensitive situations.

Chapter 4 – Potential Challenges and Risks

4.1 Potential Challenges and Risks

The TraceBack faces several significant technical challenges in implementation. The real-time location tracking system requires precise integration of geolocation services while maintaining accuracy across different devices and browsers. The current codebase shows implementation of location detection in Signup.js using browser geolocation and geocoding APIs, which needs careful handling of edge cases and API limitations.

The challenges across each requirement are:

1. Data Security

Data security presents critical challenges given the sensitive nature of missing persons information. The authentication system shown in AuthController.java implements password encryption and session management but requires continuous enhancement to protect against evolving security threats. The system must maintain robust data encryption, secure session handling, and protection against unauthorized access while ensuring quick access for legitimate users.

2. Scalability

Scalability challenges emerge as the system grows. The database architecture needs to handle increasing user loads, multiple concurrent sessions, and large volumes of location data without compromising performance. The React frontend components must maintain responsiveness while processing real-time updates and location data, as seen in the current implementation of user interfaces.

3. Resource Management

- Server infrastructure optimization
- Database performance tuning
- API response time optimization
- Cross-browser support
- Real-time data synchronization

Chapter 5 - Methodology and System Design

5.1 Project Methodology

The Agile Software Development method is selected to develop this project out of all the existing software development methodologies that excel in various perspectives. This decision was supported by factors such as time restraints, and project size and my regular participation in this project.

The phases of the agile methodology are as follows:

1.Planning Stage

The project begins with comprehensive planning, establishing project scope and objectives. The system architecture combines Spring Boot backend with React frontend, as evidenced in the AuthController.java and Signup.js implementations. Resource allocation includes development environments, API keys for geocoding services, and database infrastructure. The planning phase establishes timelines, milestones, and delivery schedules while identifying key stakeholders and their requirements. The detailed planning stage is explained in Outline /Abstract process meeting

2. The Analysis and Design Stage

This phase focuses on detailed system analysis and architectural design. The backend structure is designed around RESTful APIs with secure authentication flows. The frontend design emphasizes user experience with intuitive interfaces for registration, login, and location management. Database schema design accommodates user data, location information, and case management requirements. The system's security architecture includes password encryption, session management, and data protection measures.

3. Implementation Stage

The TraceBack project implementation phase demonstrates a comprehensive development approach, starting with the robust authentication system built using Spring Boot. The AuthController.java handles secure user registration and login processes, while the React-based frontend components, including Signup.js, deliver an intuitive user interface with integrated location services and real-time tracking capabilities.

The system's core functionality extends beyond basic authentication to include sophisticated case management features, allowing users to create, track, and update missing person cases efficiently. The implementation incorporates advanced location services with geocoding integration, real-time tracking, and interactive mapping features, all working seamlessly with the secure backend infrastructure.

The platform's communication system enables real-time messaging and notifications, keeping users informed of case updates and relevant information. The data management layer ensures secure storage and efficient retrieval of case information, while maintaining strict privacy controls and data protection measures. This comprehensive implementation creates a powerful tool for missing persons tracking while maintaining high standards of security and user experience.

4. Testing Phase:

Comprehensive testing includes(Uploaded all these scenarios in the Detailed of Work Done):

- Unit testing of individual components (Using Postman)
- Integration testing of frontend-backend communication
- Security testing of authentication systems
- Performance testing under various load conditions
- User acceptance testing of interface functionality
- Compared the Missing Persons for variable scenarios
- Email Notification sent of various scenarios
- Location service testing across different scenarios

5.2 Technologies Involved

The TraceBack project leverages a robust stack of modern technologies:

Backend Technologies:

- Spring Boot framework for server-side development
- Java for core business logic
- Gradle for build automation and dependency management
- Spring Security for authentication and authorization
- RESTful API architecture
- JPA/Hibernate for database operations
- Email service integration

Frontend Technologies:

- React.js for dynamic user interface
- Axios for HTTP requests
- CSS Modules for styling
- HTML5 Geolocation API
- Using Filters for knowing the Missing Person Profiles (Ex: Recently Missing, Recently Reported etc.,)

Database and Storage:

- Oracle for data persistence
- File storage for images and all other data

External Services:

- Geocoding API for location services
- News API for Missing person related News
- Maps integration for visualization
- Email service providers
- Python Script to compare the images

Development Tools:

- IntelliJ IDEA for development
- Postman for API testing
- npm/yarn for package management

5.3 System Architecture

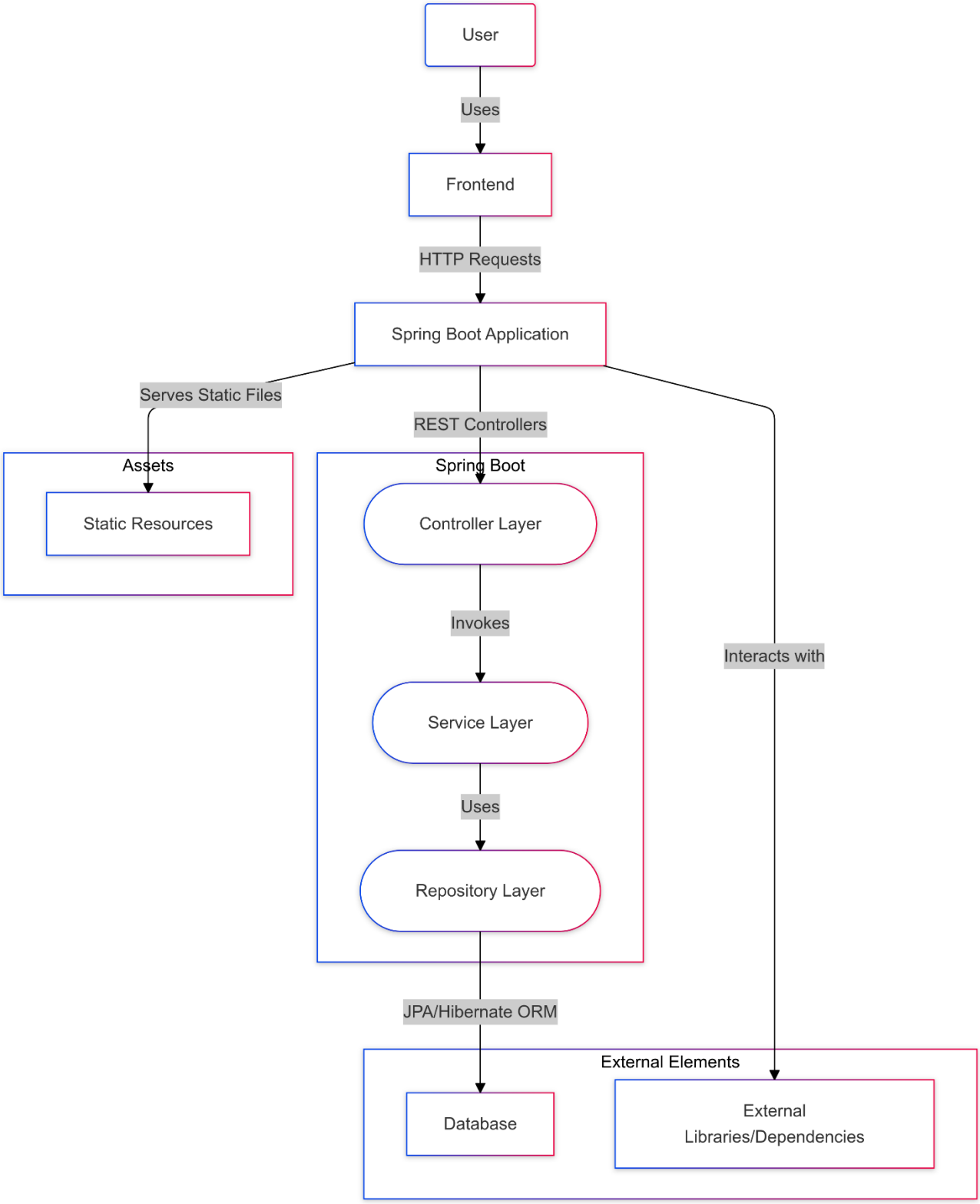

Fig.2 System Architecture

Above is the high level architecture diagram for the project and the detailed explanation is :

1.Frontend

- **Interaction with Backend**: The frontend sends HTTP requests to the server and receives responses. Users interact with a web interface, which could be developed using technologies like HTML, CSS, and JavaScript.
- **Data Presentation**: Renders data received from the backend in a user-friendly manner. It may involve using frameworks like React or Angular, based on the dependencies found in your Node.js modules.

2.Spring Boot Application

- **REST Controllers**: Act as endpoints in the server that process incoming HTTP requests and determine what response to send back. They translate HTTP requests to service method calls.
- **Service Layer**: Contains business logic. This layer processes data applies necessary rules, and coordinates between the controllers and the repositories.
- **Repository Layer**: Manages data access. This layer interacts with the database using ORM (Object Relational Mapping) tools like Hibernate in Spring Boot. It provides CRUD operations (Create, Read, Update, and Delete) to the application.

3.Database

- **Data Persistence**: The database is where the application's data is stored. This project likely uses a relational database, and Java applications commonly utilize JPA (Java Persistence API) along with Hibernate for database interaction.
- **Schema Management**: Defines tables, relationships, and constraints in the database to ensure data integrity and efficient retrieval.

4.Static Resources

- **Assets Delivery**: Contains all necessary frontend assets like HTML templates, CSS for styling, and JavaScript scripts. These resources are served by the Spring Boot application and do not change during runtime.
- **Resource Handling**: Handled through the Spring Boot's built-in resource handlers, which efficiently manage and cache static resources.

5.External Libraries/Dependencies

- **Third-party Libraries**: External libraries used by the application can enhance functionality without having to implement it from scratch. This includes libraries for UI components (like Leaflet for maps) or utility functions.
- **Dependency Management**: Managed by tools like Gradle (for Java) and npm (for Node.js modules), ensuring that the correct versions of dependencies are used and resolving any potential conflicts between them.
- **Python Libraries** : Used External Python libraries for the comparison of the images for the accurate comparison used models like FaceNet etc.,

Integration and Execution

- The application integrates the Java backend and the Node.js-based frontend. The Gradle configuration is set up to build both Java and Node.js assets and package them appropriately.

All these components work together to form a cohesive, well-functioning software system capable of responding to user requests by processing data and interfacing with necessary databases and delivering a smooth user experience.

5.4 System Flowchart

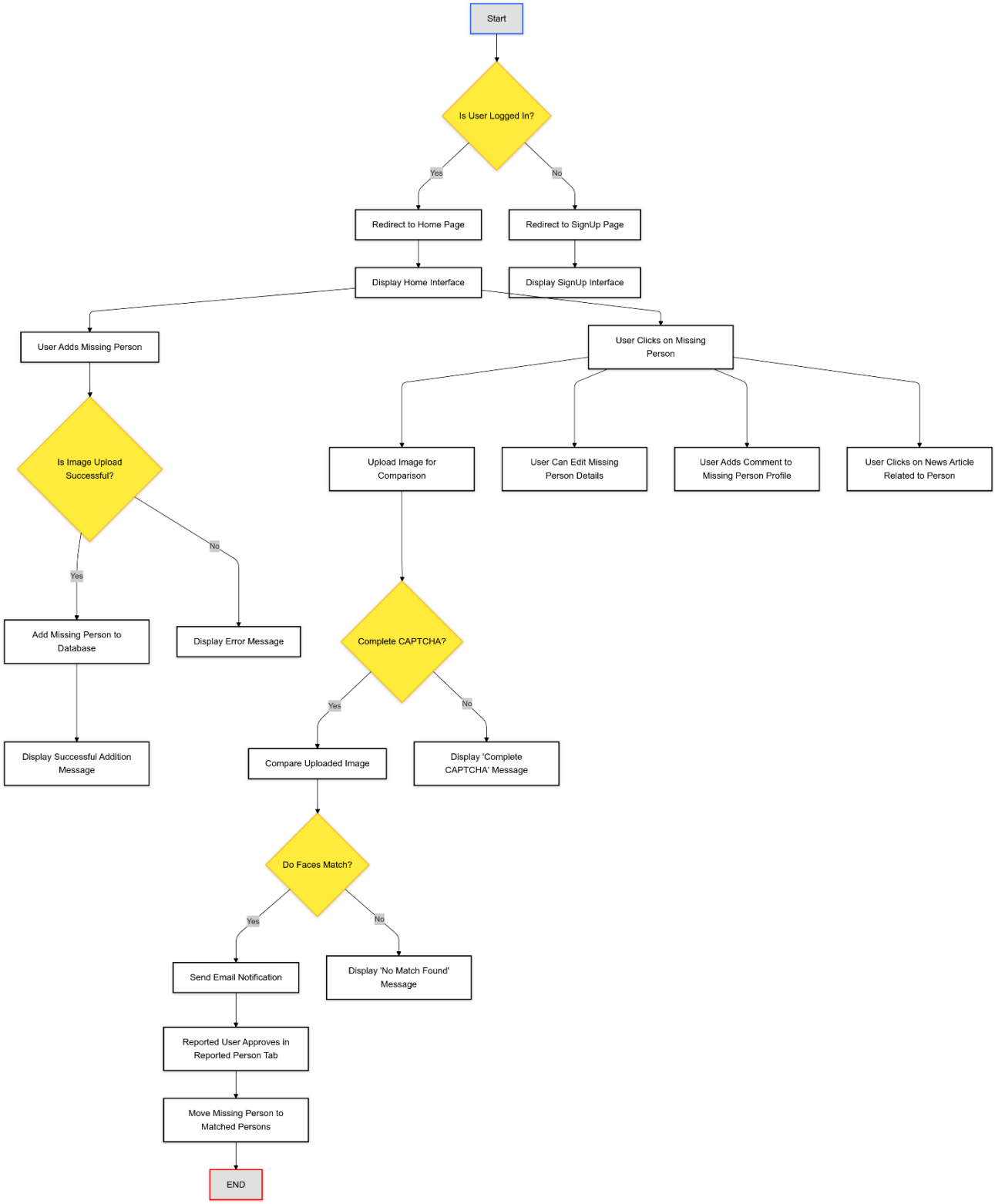

Fig.3 Flow Diagram

5.5 Use Case Diagram and the use cases

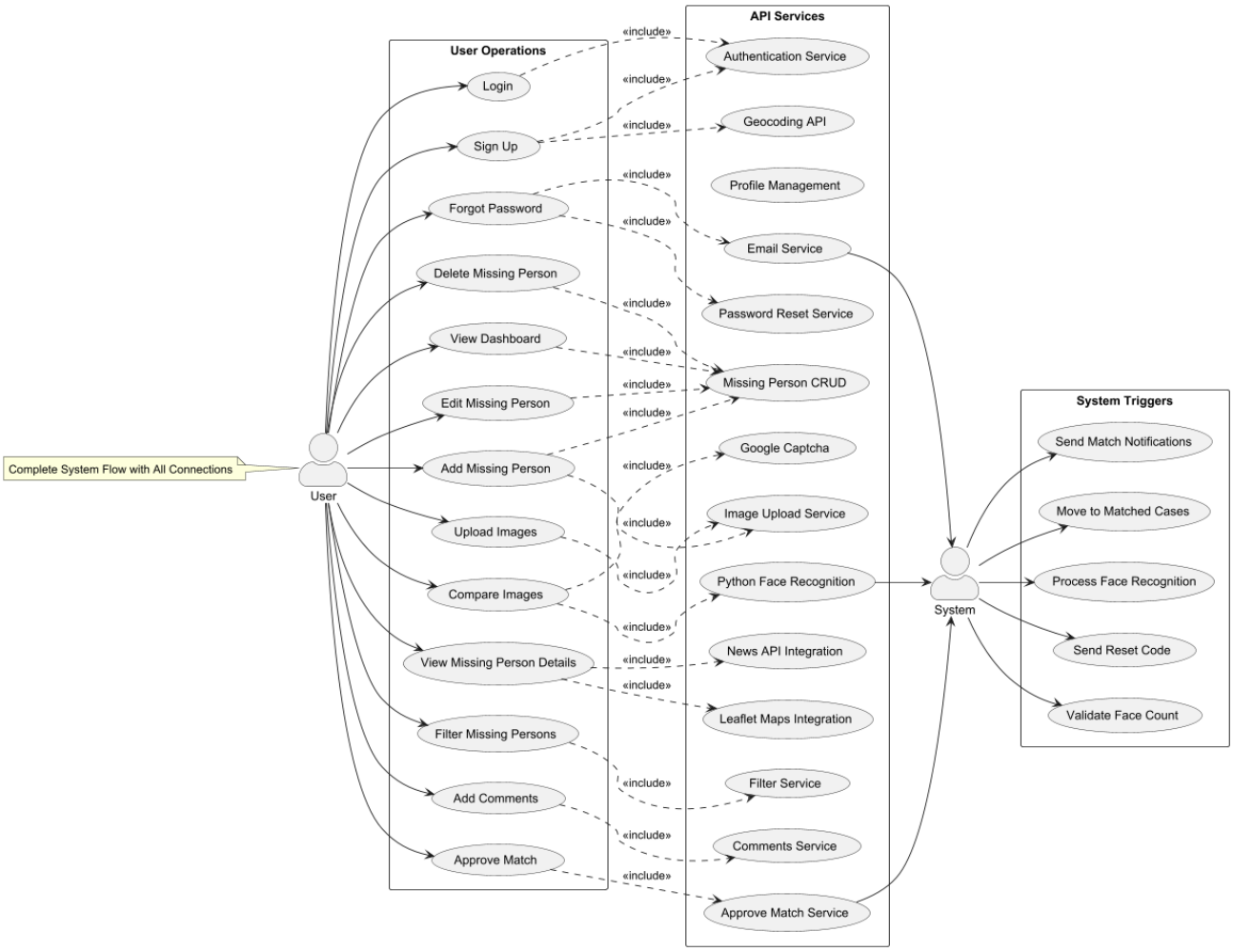

Fig.4 Use Case Diagram

Use Cases (Also Functional Requirements):

1. **Authentication Operations:**

1.1 Use Case Name: Sign Up
Actor: User

Details: New user registration in the application

Steps:

1. User clicks "Sign Up" button
2. System displays registration form
3. User enters email, password, name, location
4. System validates email format
5. System uses Geocoding API to validate/detect location
6. System checks if email already exists
7. System creates new user account
8. System redirects to login page

1.2 Use Case Name: Login
Actor: User

Details: User authentication to access the application

Steps:

1. User clicks "Login" button
2. System displays login form
3. User enters email and password
4. System validates credentials
5. System redirects to Home page
6. System loads user profile data

1.3 Use Case Name: Forgot Password
Actor: User

Details: Password recovery process

Steps:

1. User clicks "Forgot Password" link
2. System displays email input form
3. User enters registered email
4. System validates email existence
5. System generates reset code

6. System sends reset code via email

7. User gives the reset code

8. System displays password reset form

9. User enters new password

10. System updates password

2. Core Operations:

2.1 Use Case Name: View Missing Persons

Actor: User

Details: Access home page with missing person's overview

Steps:

1. User logs into system

2. System displays home page

3. System loads recent missing persons

4. System displays user's reported cases

5. System shows nearby cases based on location

6. System presents filtering options

7. System has the Navbar consists of Reports Persons, Add Missing Person, Dashboard, Profile and Logout

2.2 Use Case Name: Add Missing Person

Actor: User

Details: Report new missing person case

Steps:

1. User clicks "Report Missing Person"

2. System displays report form

3. User enters person details (name, age, location, date missing)

4. User uploads images

5. User adds additional information in description

6. System saves report after validating the face detection(single face requirement and human face should be there)

7. System notifies the mail to user of successful submission

2.3 Use Case Name: Compare Images

Actor: User

Details: Compare uploaded images with database

Steps:

1. User selects "Compare Images" option in the Missing Person profile page
2. System displays image upload interface
3. User uploads image
4. System validates Google Captcha (To avoid the spam “I am not a robot”)
5. System runs Python face recognition with the missing person image and User uploaded image
6. System gives similarity and if isMatch is true from Python generate a mail to the Reported User
7. Reported User reviews matches

2.4 Use Case Name: Update Profile

Actor: User

Details: Modify user profile information

Steps:

1. User clicks "Profile" section
2. System displays profile settings
3. User can update:
   - Contact Details
   - Location Settings
4. System validates changes
5. System uses Geocoding API for location updates
6. System saves profile changes
7. System confirms successful update

3.Management Operations:

3.1 Use Case Name: Edit Missing Person

Actor: User

Details: Modify existing missing person report

Steps:

1. User selects case to edit in Reported Person tab in Navbar
2. System displays edit form
3. User modifies information
4. System validates changes
5. System updates record

3.2 Use Case Name: Delete Missing Person

Actor: User

Details: Remove missing person report

Steps:

1. User selects case to delete
2. System displays confirmation dialog
3. User confirms deletion
4. System removes record
5. System updates the homepage

3.3 Use Case Name: Add Comments

Actor: User

Details: Add comments to missing person case

Steps:

1. User selects a missing person profile
2. System displays Community Support form
3. User enters comment
4. System saves comment
5. System updates case thread

3.4 Use Case Name: Approve Match

Actor: User

Details: Confirm potential match for missing person

Steps:

1. User reviews match details
2. User clicks "Approve Match"
3. System requests confirmation
4. User confirms match
5. System updates case status
6. System moves to matched cases
7. System updates statistics

3.5 Use Case Name: Social Media Sharing

Actor: User

Details: Share missing person cases across social platforms

Steps:

1. User clicks share button on missing person case at down of the page
2. System displays social sharing options:
   - Twitter Share
   - WhatsApp Share
   - Facebook Share
3. User selects platform
4. System generates sharing link with:
   - Case title
   - Case URL
   - Basic details
5. System opens platform-specific share dialog
6. User can customize share message

4.Viewing Operations:

4.1 Use Case Name: View Missing Person Details

Actor: User

Details: Access detailed information about specific case

Steps:

1. User selects Missing Person Profile from the Home Page
2. System loads case details
3. System loads News API results related to this person
4. System displays location on map
5. System displays comments

4.2 Use Case Name: Filter Missing Persons

Actor: User

Details: Filter and search through cases

Steps:

1. User accesses filter options in the homepage
2. System displays filter criteria
3. User selects filters (location, recently missed, recently reported etc.,)
4. System applies filters
5. System displays filtered results
6. User can sort results
7. System updates view

4.3 Use Case Name: View Matched Cases

Actor: User

Details: Access and manage matched missing person cases

Steps:

1. User clicks "Matched Cases" tab
2. System displays list of matched cases
3. User can:
   - View match details
   - Access original case data
   - View matching case data
4. System shows match timestamp
5. System displays verification status

4.3 Use Case Name: View Dashboard Analytics

Actor: User

Details: Access comprehensive dashboard with statistical charts and analytics

Steps:

1. User accesses dashboard in the Navbar after the login
2. System loads DashboardCharts with:
   - Cases by Location (Bar Chart)
   - Match Success Rate (Pie Chart)
   - Cases Timeline (Line Chart)
3. User can:
   - View total cases count
   - See matched vs unmatched ratios
   - Analyze geographical distribution
   - Track case trends over time
4. System updates charts in real-time

5. Security Operations

5.1 Use Case Name: View Shared Missing Person Profile

Actor: Public User/ Non-Logged In User

Details: Access and view shared missing person case details through social media links

Steps:

1. User clicks shared link from social media platforms:
   - Twitter link
   - WhatsApp link
   - Facebook link
2. System displays public view of case with:
   - Missing person's name and photo
   - Last seen location with map
   - Case description
   - Related news articles
   - Reporter contact details

3. If user attempts restricted actions:
   - Compare Images
   - Add Comments
4. System:
   - Stores current URL in localStorage
   - Redirects to login page
   - After login returns user to original case and can perform the operations
5. User can view but not interact until authenticated

Restrictions:

- No image comparison allowed
- No commenting permitted

Purpose: Enable public viewing of critical case information while maintaining system security through authentication requirements for interactive features

5.6 Non-Functional Requirements:

1. **Performance**
   - Quick search results
   - Responsive UI
   - Efficient database queries
2. **Security**
   - Password encryption
   - CAPTCHA verification
   - Secure API endpoints
   - Protected user data
3. **Scalability**
   - Handle multiple concurrent users
   - Process multiple image comparisons
   - Store large numbers of cases
4. **Reliability**
   - Accurate face recognition
   - Consistent email delivery
   - Stable map integration
   - Reliable news feeds

5. **Usability**
   - Intuitive interface
   - Clear navigation
   - Accessible design
6. **Integration**
   - Maps API integration
   - News API connection
   - Email service integration
   - Social media sharing

These requirements ensure a robust, secure, and user-friendly missing persons management system.

5.7 Database Design and ER Diagram

1. USERS Table:
- Primary purpose: Stores user account information
- Fields:
  - id: Unique identifier using USER_SEQ sequence
  - name: User's full name
  - email: Unique email address for login
  - location: User's geographical location
  - password: Encrypted password
  - verificationCode: For email verification process

2. MISSING_PERSONS Table:
- Primary purpose: Records details of missing individuals
- Fields:
  - id: Unique identifier using MISSING_PERSON_SEQ
  - name: Missing person's name
  - age: Age of missing person
  - gender: Gender identification
  - lastSeenLocation: Last known location
  - description: Detailed description
  - reported_by_id: Links to USERS table (who reported)
  - imageData: Stores photo of missing person
  - reportedAt: Timestamp of report
  - status: Current status of case(Missing or Found)

3. MATCHED_PERSONS Table:

- Primary purpose: Tracks potential matches of missing persons
- Fields:
    - id: Unique identifier using MATCHED_PERSON_SEQ
    - missing_person_id: References original missing person record
    - reported_by_id: User who reported the match
    - found_by_id: User who found the person
    - matchedAt: Timestamp of match
    - approved: Match verification status
    - matchedImageData: Photo evidence of match

4. COMMENTS Table:

- Primary purpose: Enables user discussions on missing person cases
- Fields:
    - id: Unique identifier using COMMENT_SEQ
    - content: Comment text
    - user_id: Links to comment author
    - missing_person_id: Links to relevant missing person case
    - createdAt: Timestamp of comment creation

Each table uses sequence generators for ID management and maintains referential integrity through foreign key relationships, creating a robust system for tracking missing persons cases and related interactions.

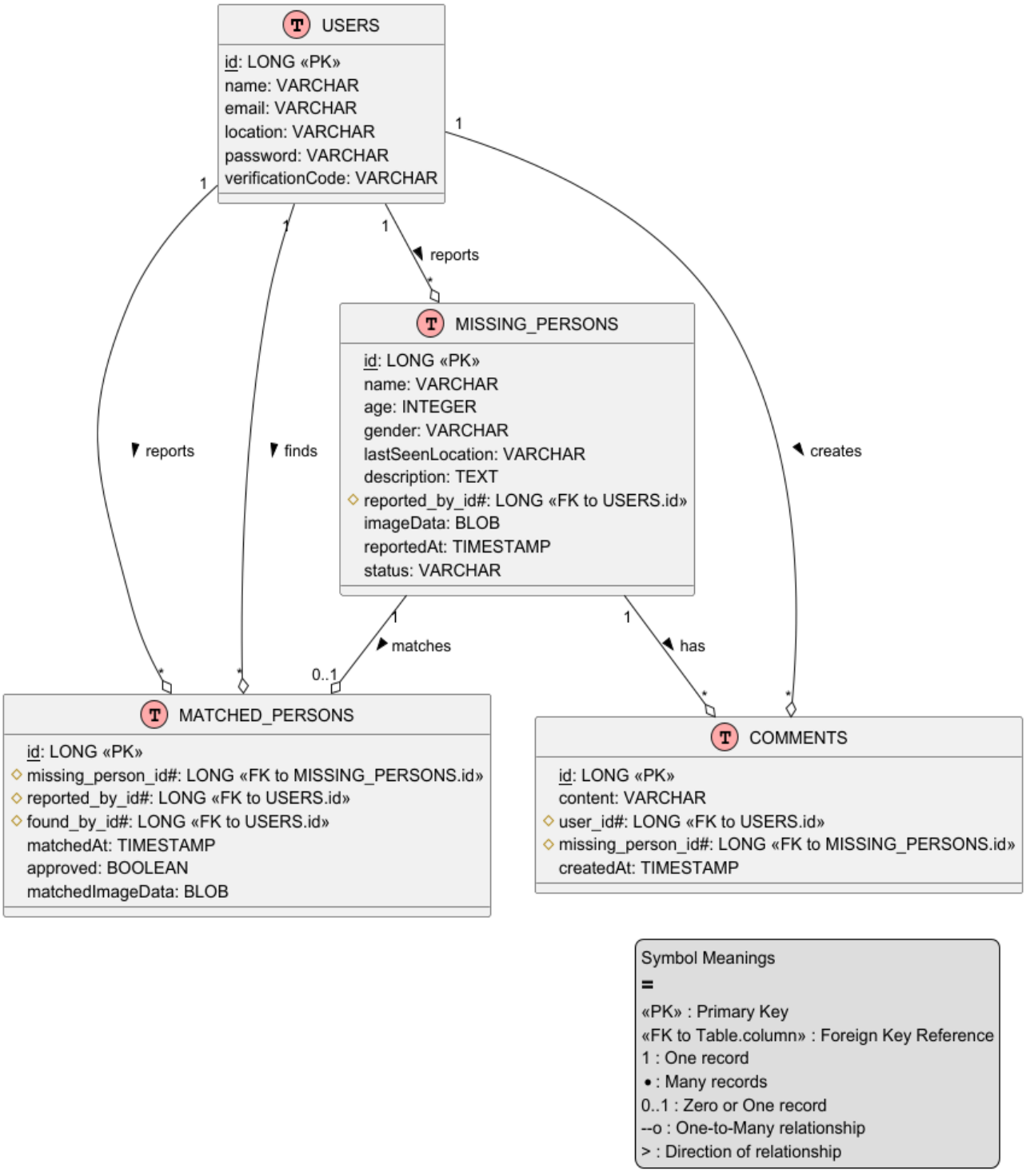

Fig.5 ER Diagram

5.8 Application Module Diagram

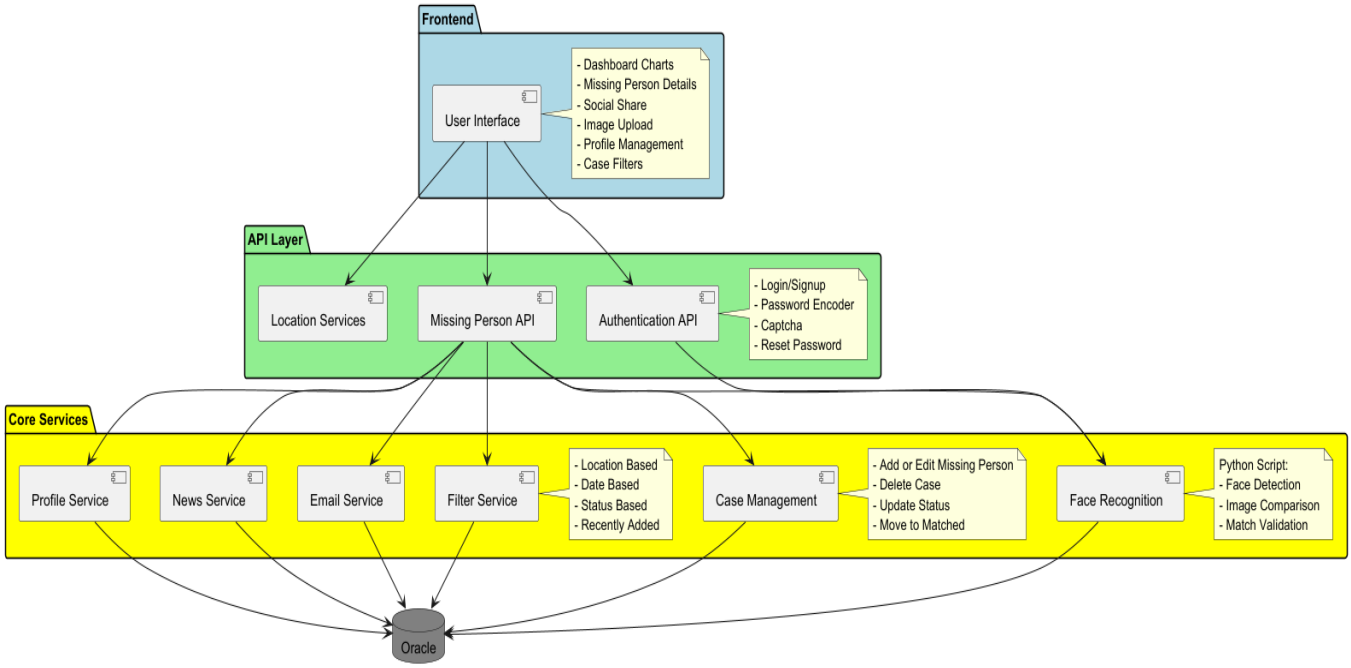

Fig.6 Application Modules

Chapter 6 – Project Resources

6.1 Software Used

IntelliJ is a powerful and flexible editor Provides powerful code editing with syntax highlighting for **Java, Python, and React JS**. Offers smart code completion and real-time error detection and it integrates seamlessly with **Gradle** for build management and dependency resolution

Oracle is a robust and enterprise-grade relational database management system (RDBMS) that provides comprehensive data management solutions. It excels in handling large-scale applications with features like efficient transaction processing, high availability, and advanced security mechanisms.

6.2 Hardware Used

The TraceBack Project is a comprehensive full-stack application designed to help track and match missing persons cases efficiently. The system leverages Spring Boot's robust backend architecture with JPA repositories managing complex database relationships across users, missing persons, matched cases, and comments. The React frontend provides an intuitive user interface while Python handles sophisticated face recognition capabilities for matching missing persons.

This concise yet powerful system is built using modern technologies and runs smoothly on the **Asus TUF Gaming F15 laptop**, utilizing its processing power for face recognition tasks and development tools. The combination of Oracle database for data persistence, IntelliJ IDEA for development, and Git for version control creates a professional-grade solution for this project.

6.3 Resources needed for this Project

1. **Development Tools**
   - IntelliJ IDEA Ultimate Edition
   - Git for version control
   - Oracle Database
   - Gradle 7.6 build tool
2. **Core Technologies**
   - Java JDK 17
   - Spring Boot 2.7.7
   - React JS 18.2.0
   - Python 3.10 for face recognition
3. **Network Requirements**
   - Stable internet connection
   - Local development server
   - Database server connectivity
4. **Storage Requirements**
   - Space for image data storage
   - Source code repository
5. **Documentation Tools**
   - PlantUML for diagrams
   - Project documentation
   - API documentation

These resources work together to create an efficient development environment for building and maintaining this missing person tracking system.

Chapter -7 Comprehensive Details of the Project Work Done with Test Samples

The Respective Systems has implemented in the project with all the configurations :

Landing Page of TraceBack Application :

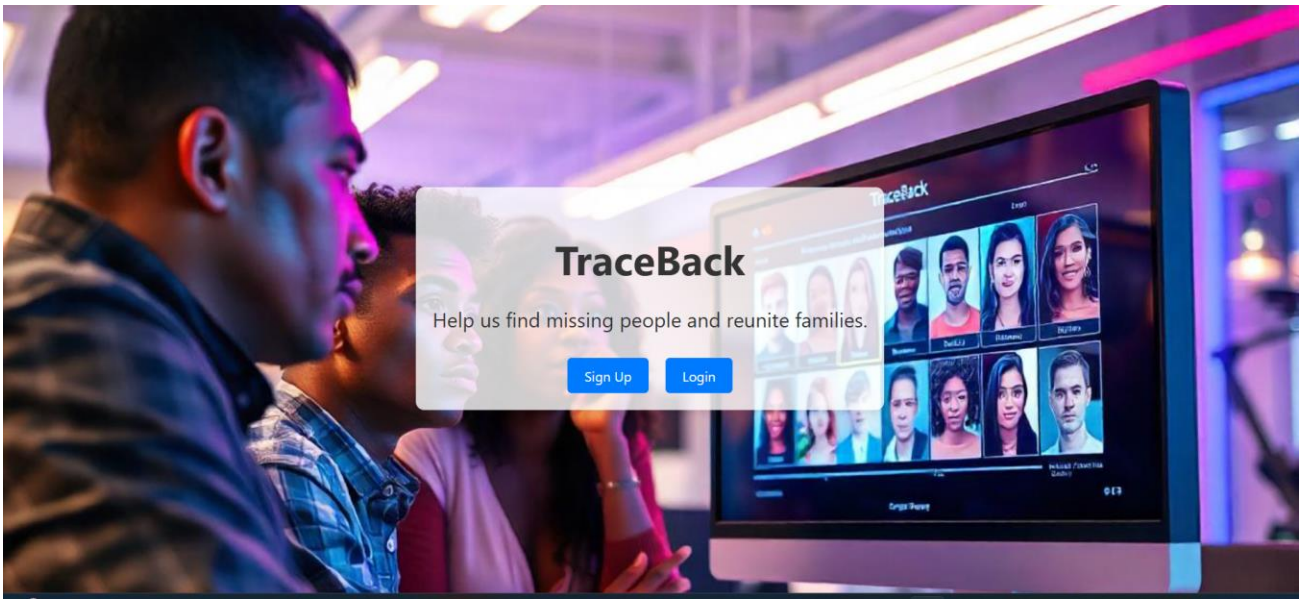

Fig.7 Landing Page of application

7.1 User Authentication System

1. Login Implementation:

- Frontend: Login component with form validation and error handling
- Backend: AuthController handles /login endpoint
- Successful login redirects to Home Page with user-specific data

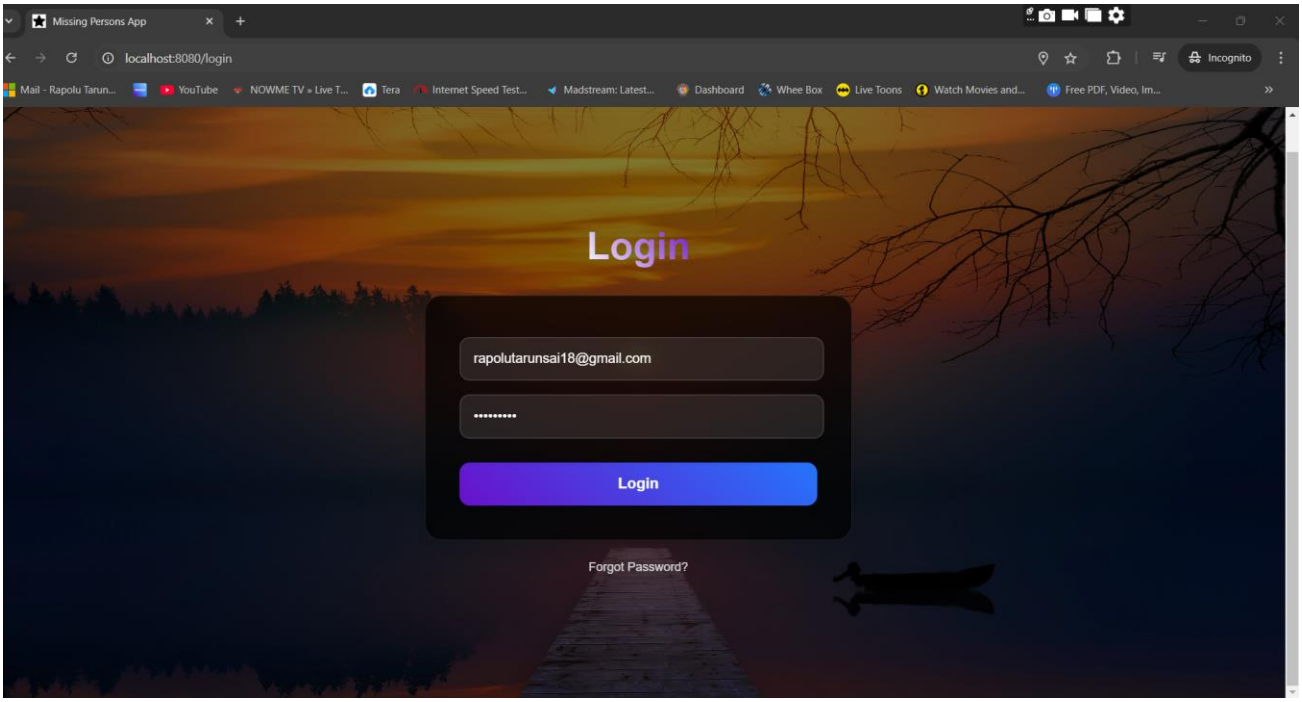

Fig.8 Login Page

2. Registration Process:

- Frontend: Signup component with field validation with email has unique and Location can be updated manually or can give location access to take the location.
- Backend: /signup endpoint and password encryption
- Success creates user profile and sends verification email

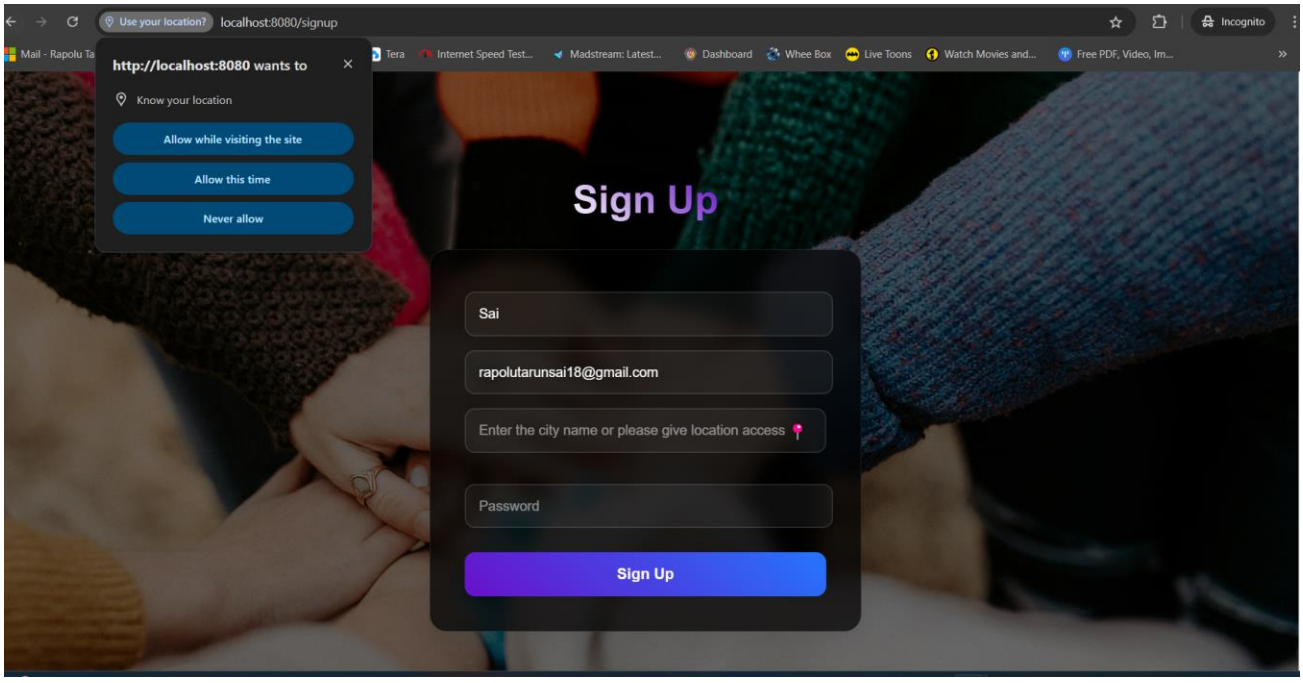

Fig.9 SignUp Page (Allowing Location Access)

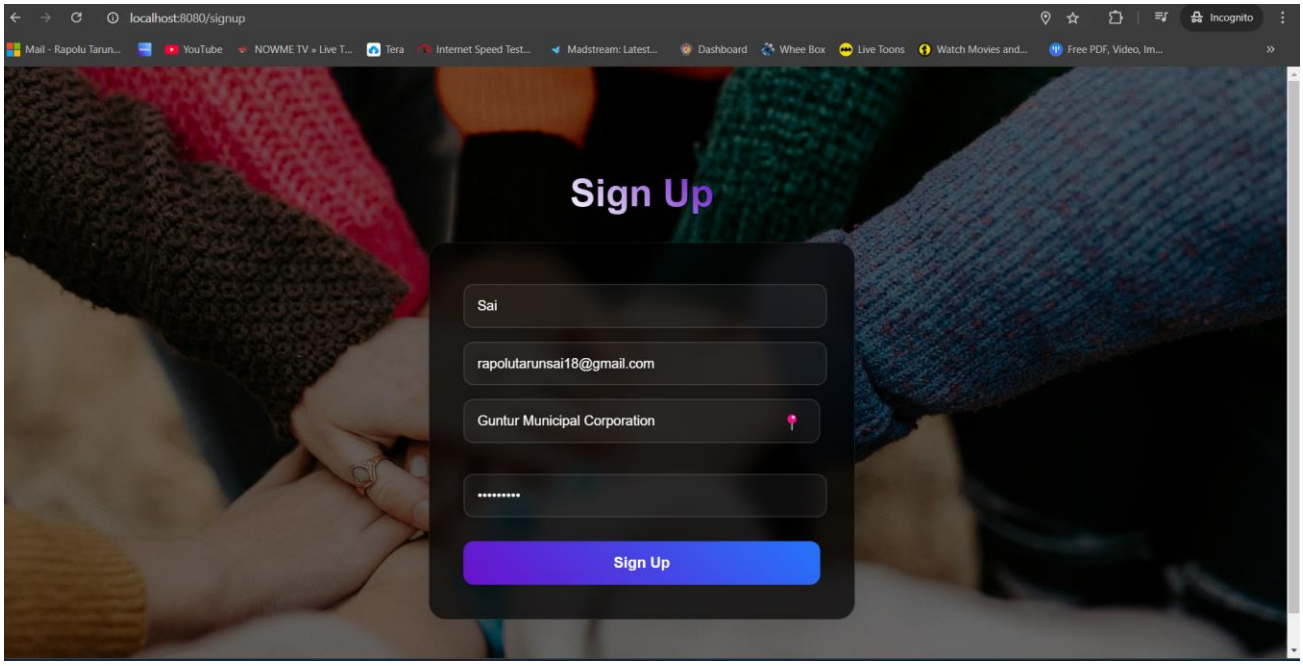

Fig.10 Current Location after allowing the Location Access

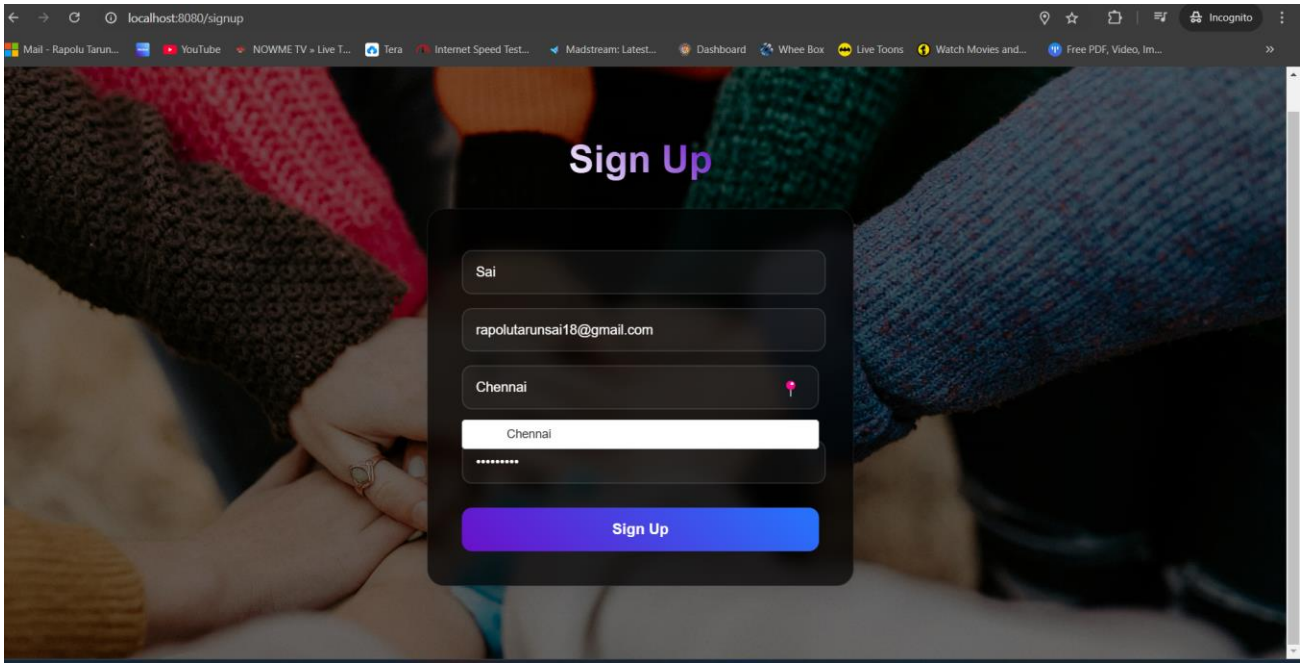

Fig.11 Manually Entering the location and getting recommendations

3. Forgot Password Flow:

- Frontend: Password reset form with email verification
- Backend: /forgot-password generates reset token and sends email
- /reset-password endpoint validates token and updates password

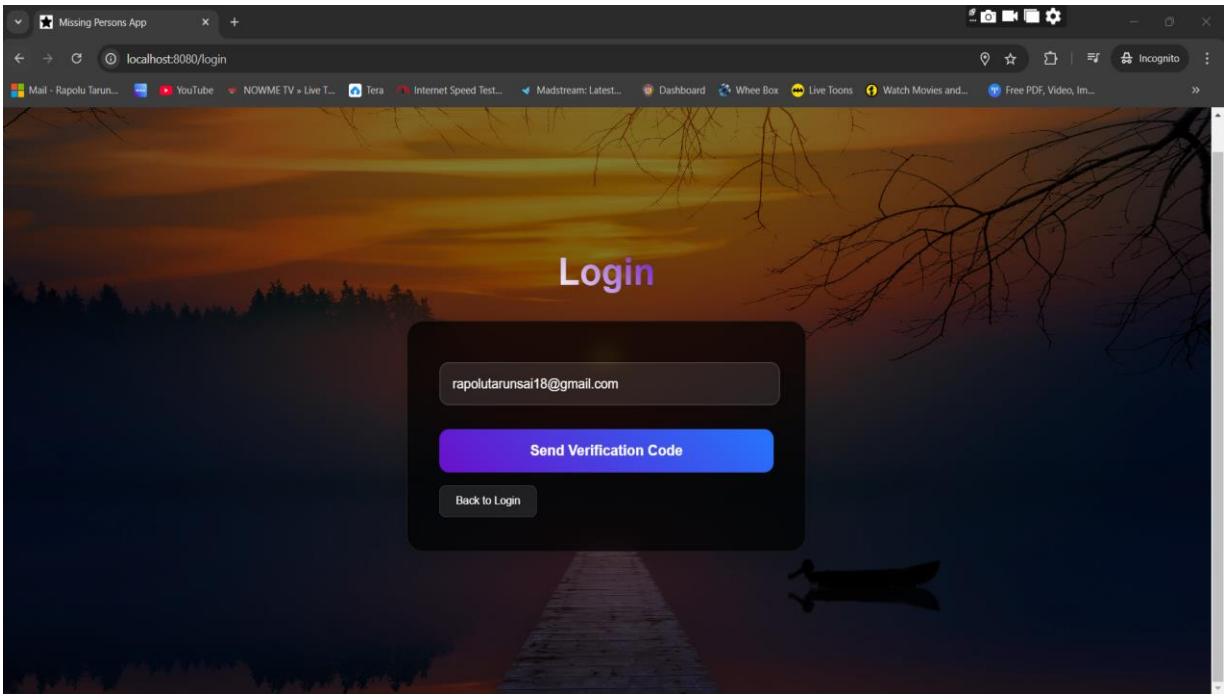

Fig.12 Forgot Password

Fig.13 Reset Verification Code

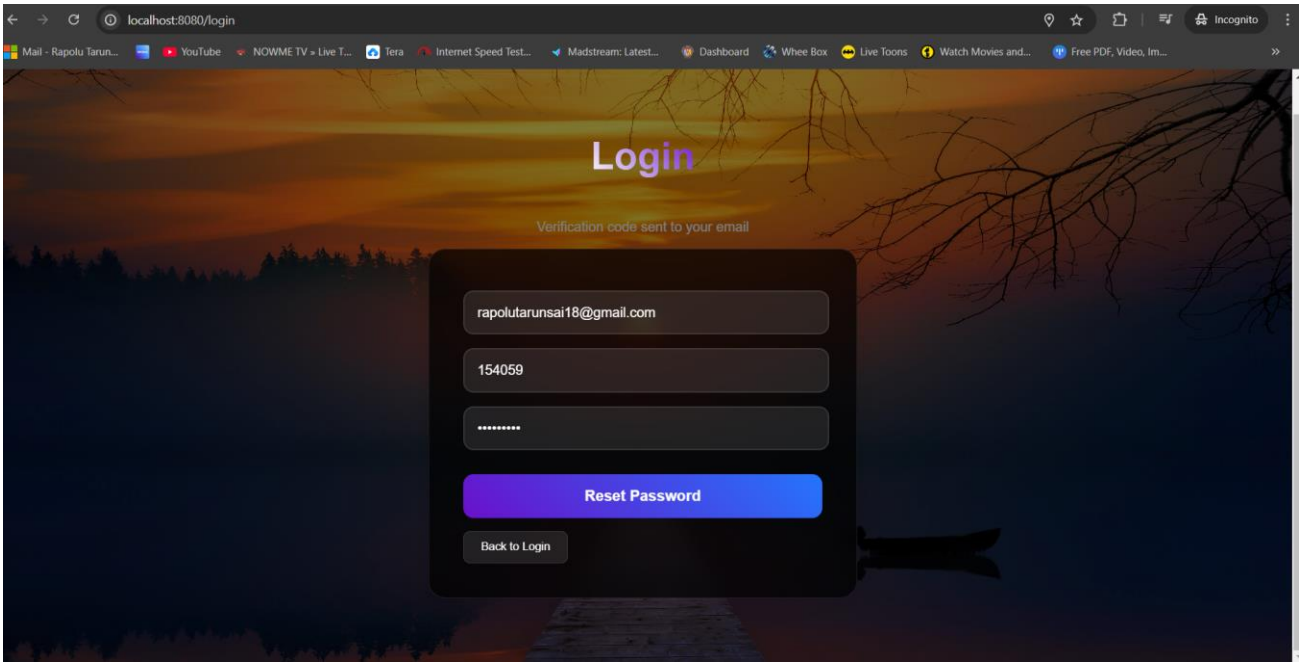

Fig.14 Resetting the Password

7.2 Missing Persons Management:

1. Create Missing Person:

- Frontend: Form with Image upload, preview and the respective details
- Backend: POST /add processes images through face recognition validates if there is any human face and the human faces should be only 1
- Stores case data with the image and the location details in the DB

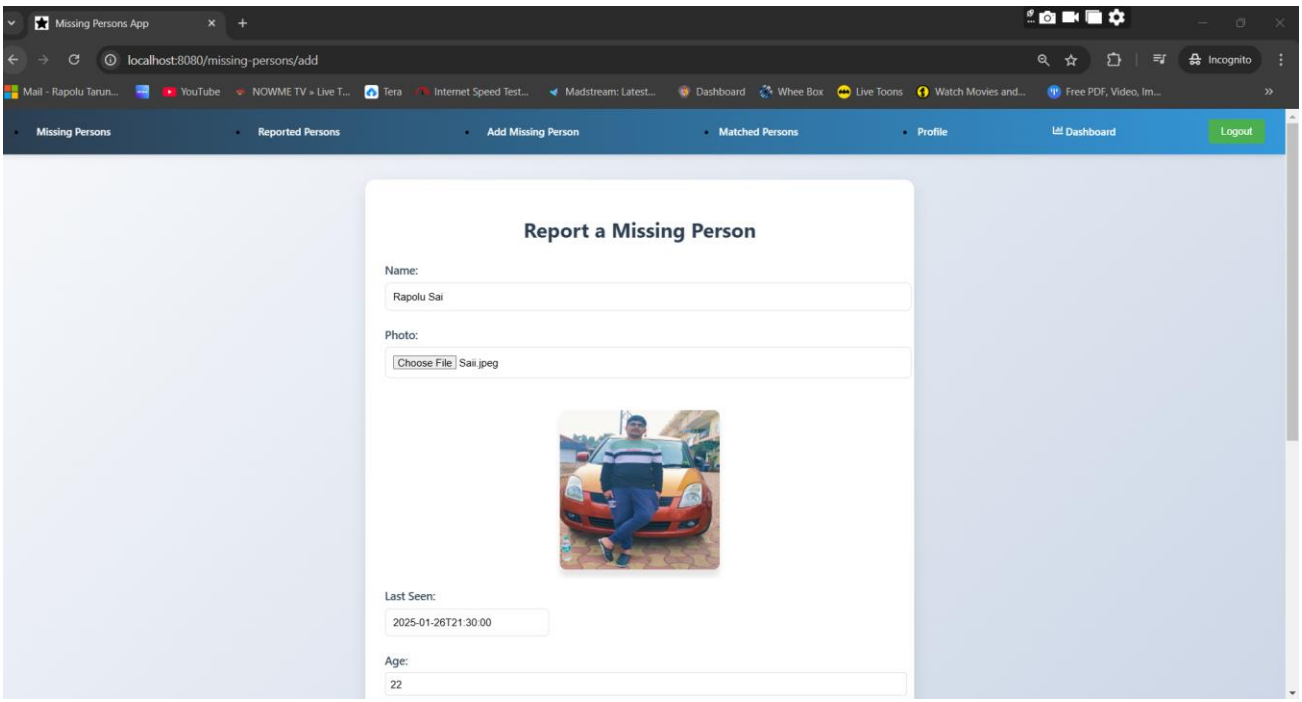

Fig.15 Reporting a Missing Person

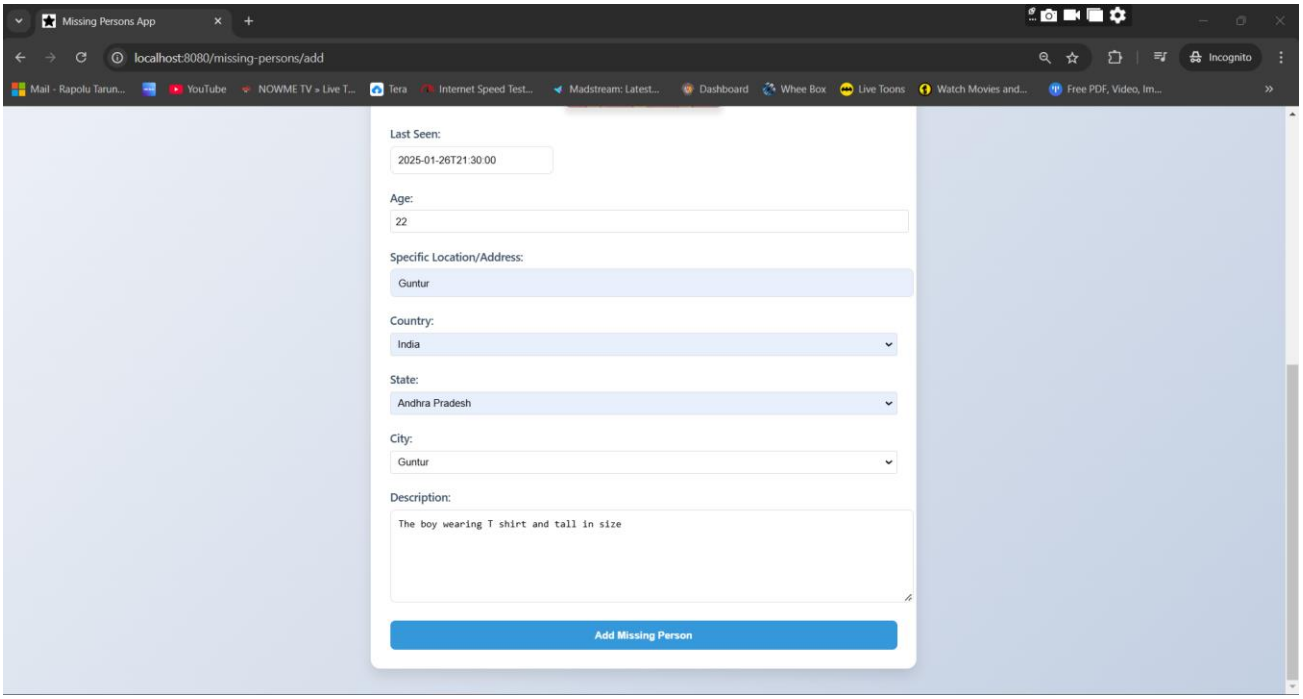

Fig.16 Adding the person

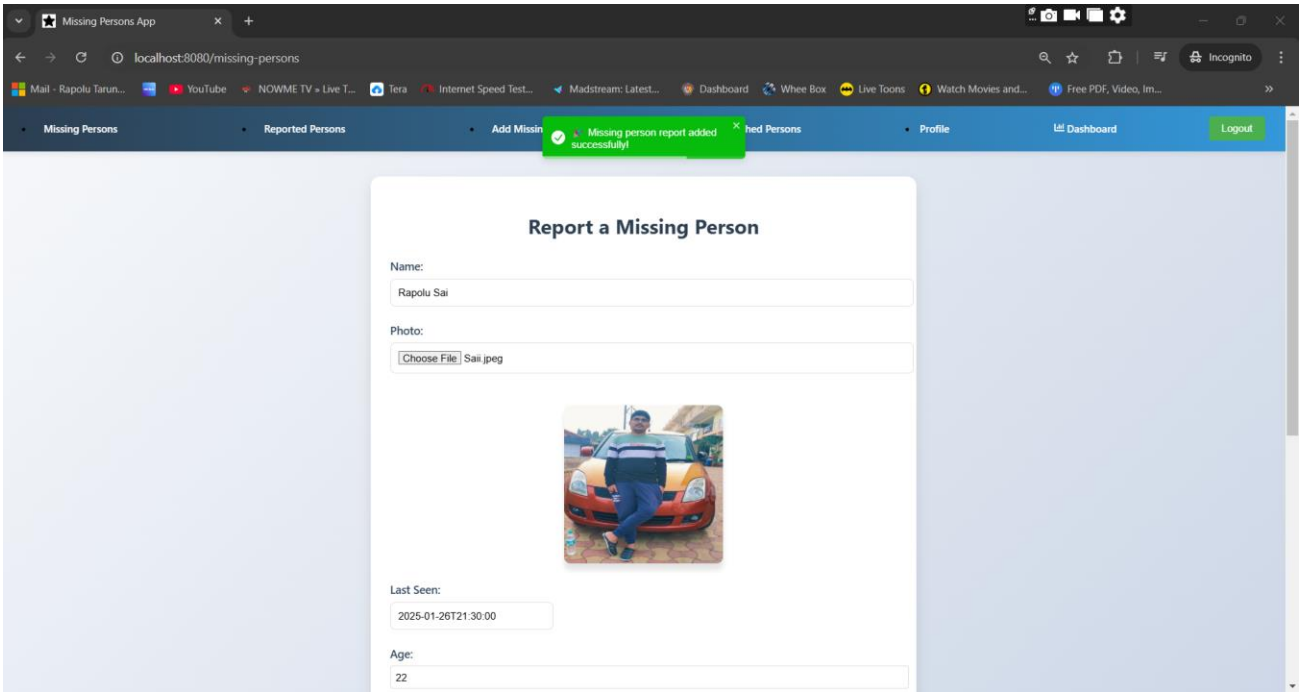

Fig.17 Missing Person Toastify Message

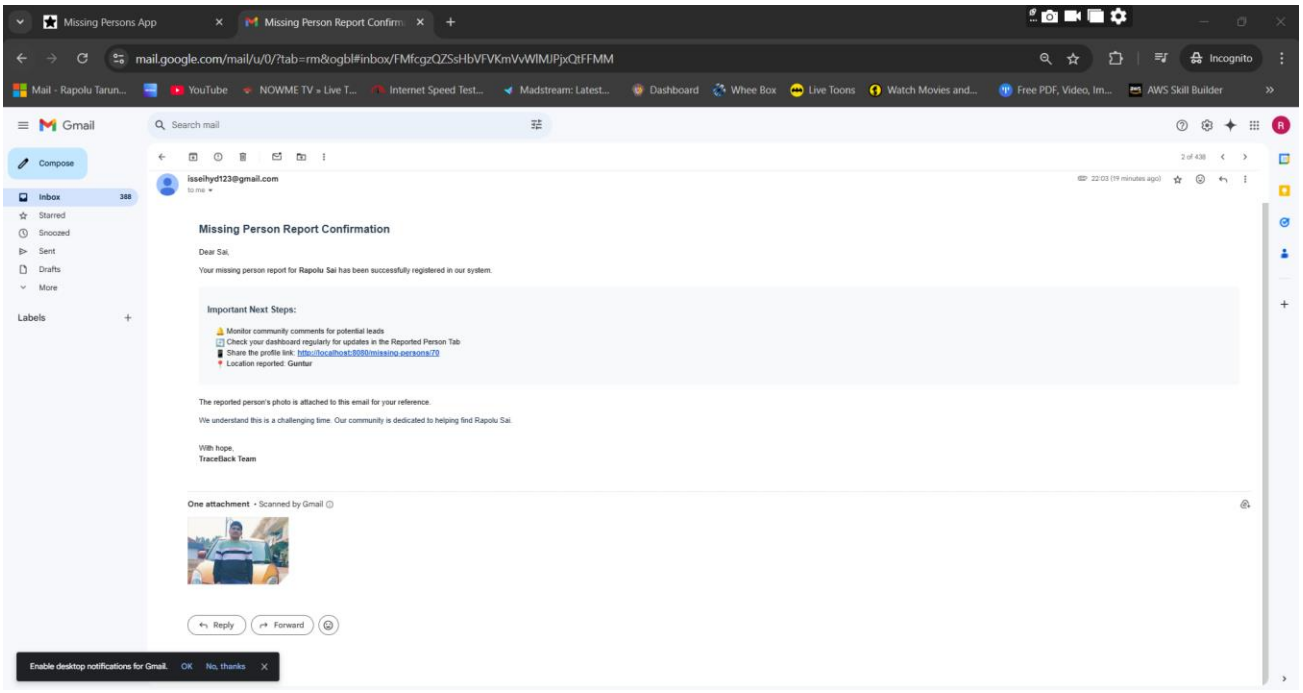

Fig.18 Email Confirmation after adding the report

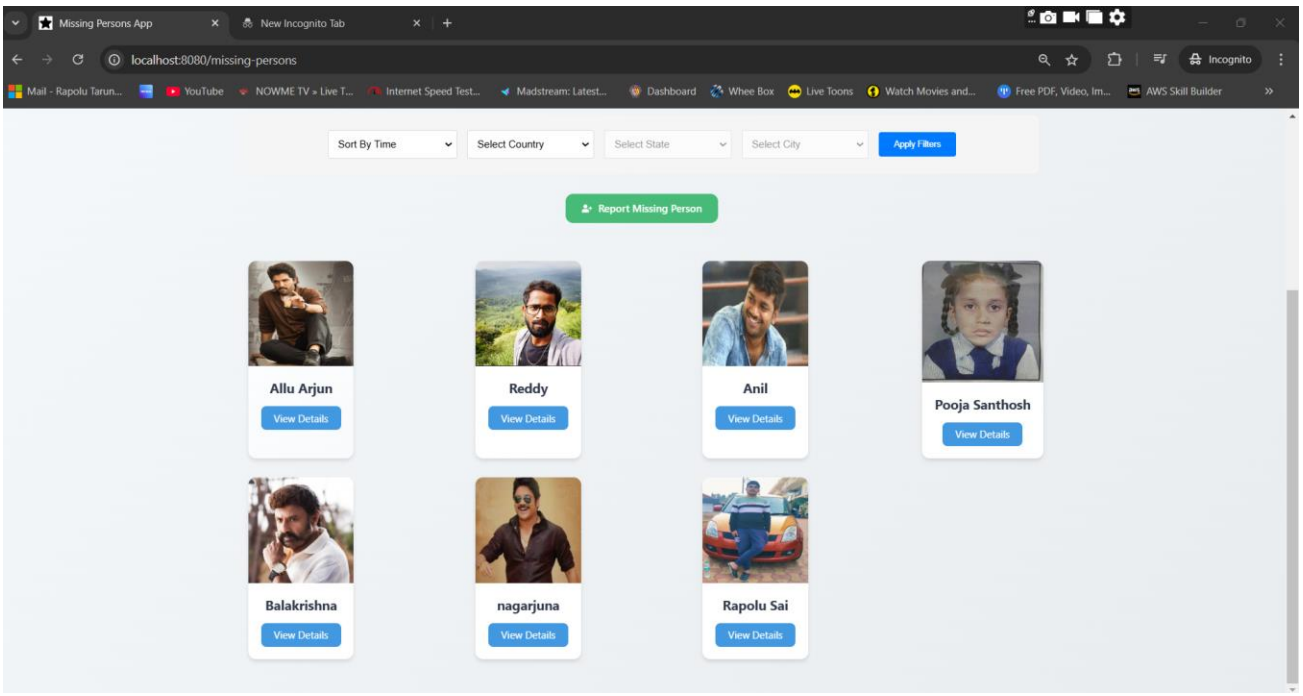

Fig.19 Missing Person Added(Rapolu Sai)

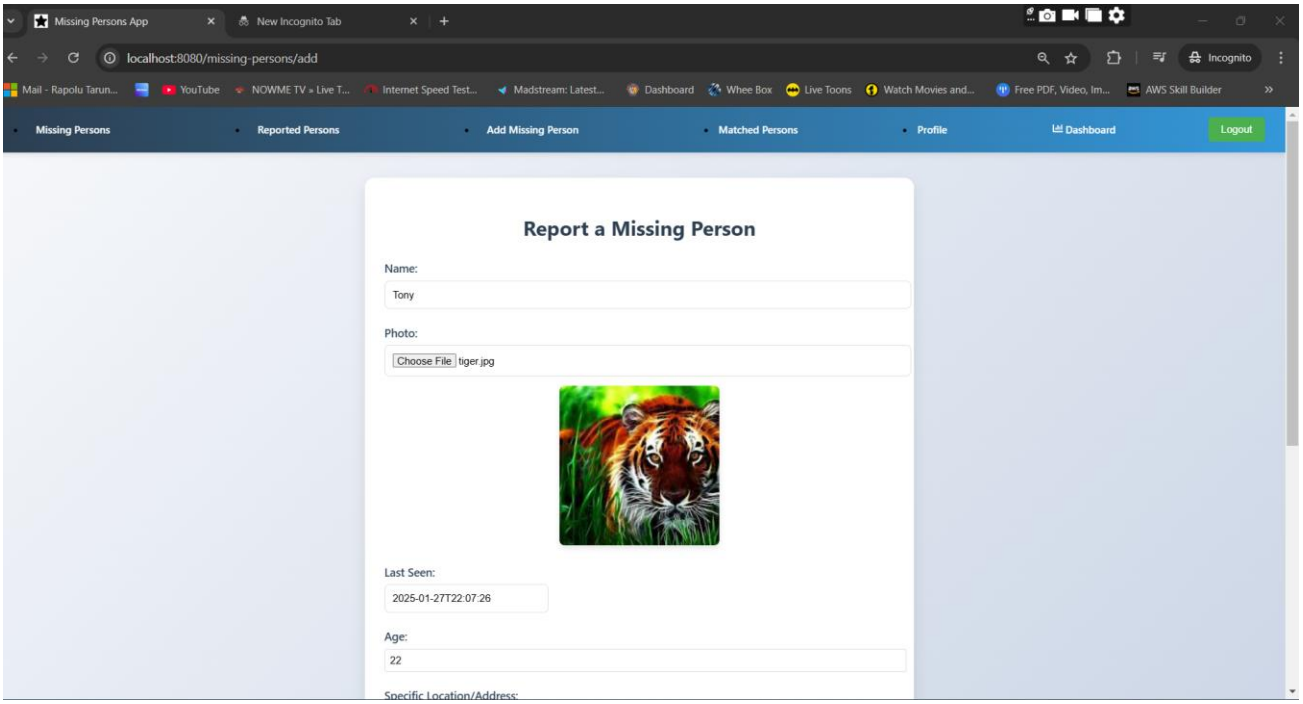

Fig.20 Adding an faceless photo

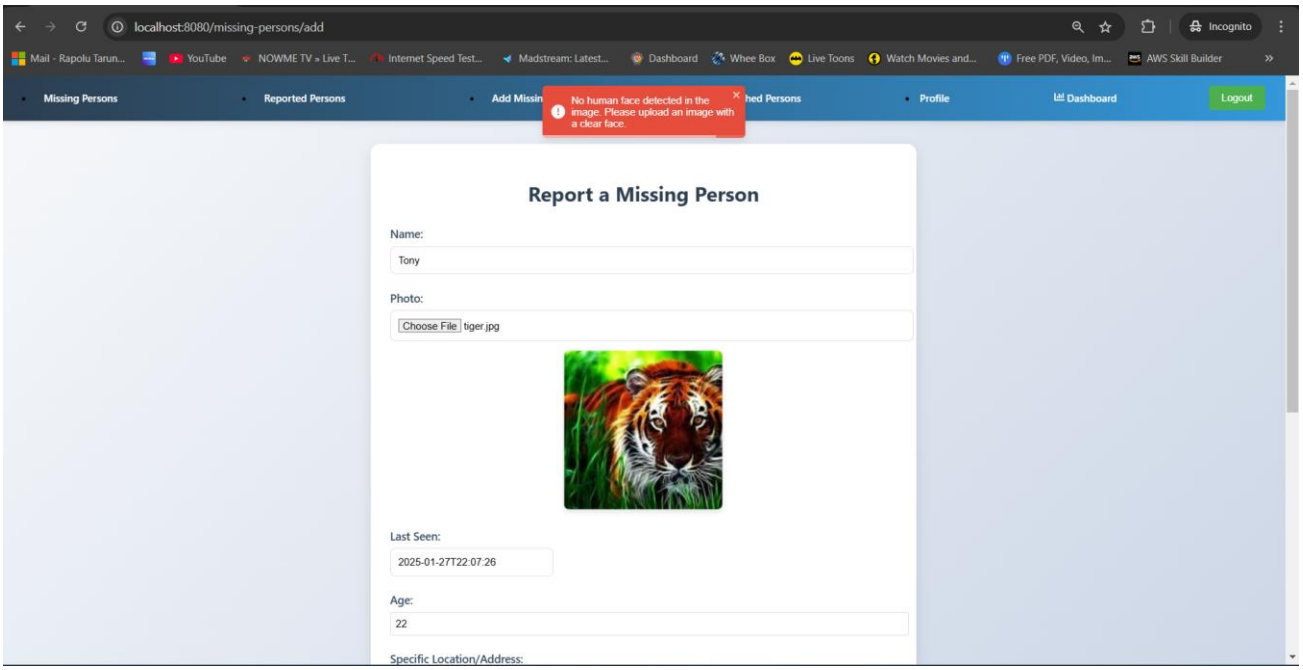

Fig.21 Getting error to upload the human face photo

2. View Missing Persons:

- Frontend: List view with filters and search functionality
- Backend: GET /missing-persons with pagination
- Implements sorting and filtering options

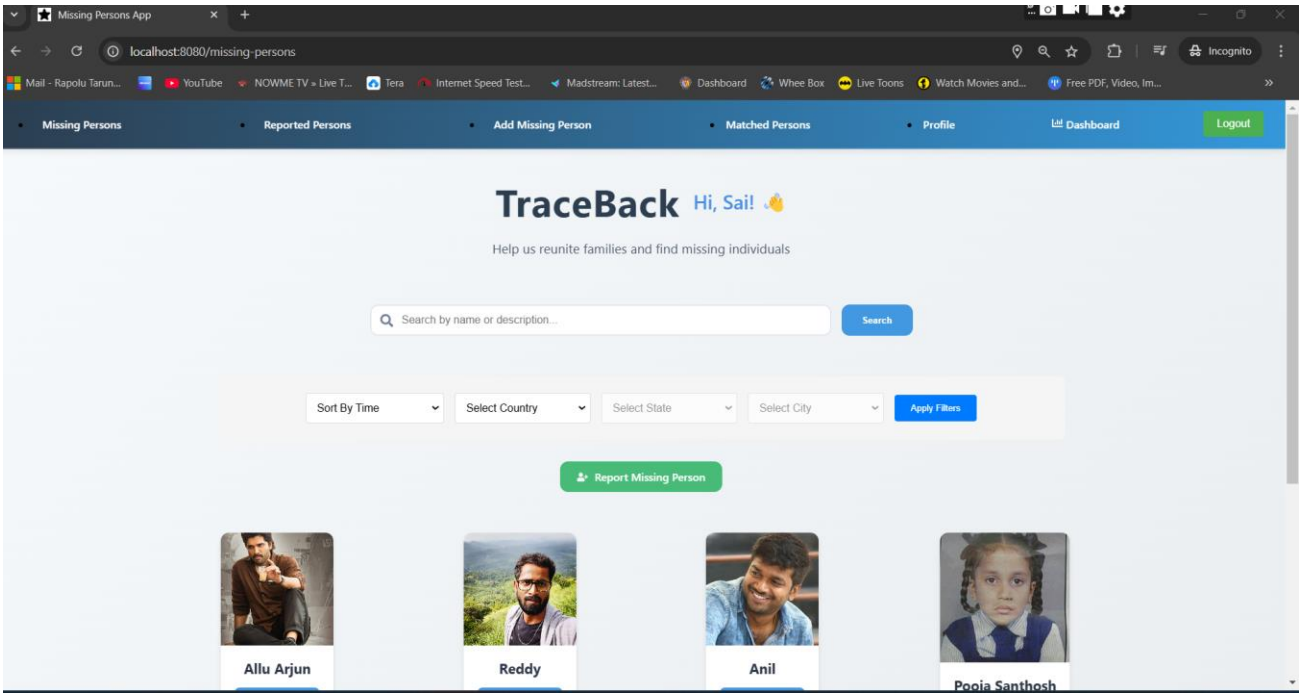

Fig.22 Accessing all Missing Person Profiles

3. View Single Missing Person:

- Frontend: Detailed case view with full information and image
- Backend: GET /missing-persons/{id} fetches specific case details
- Returns complete case data with reporter information

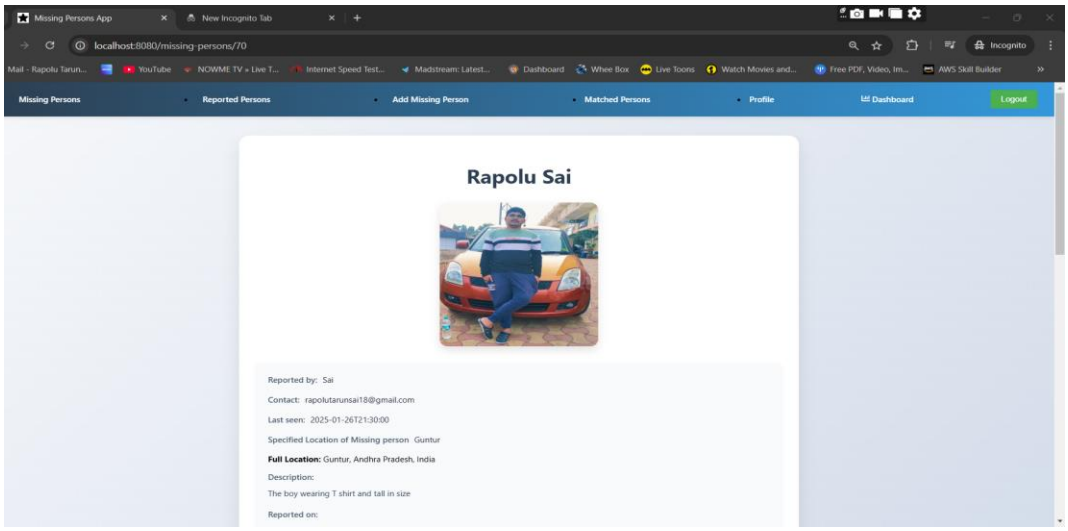

Fig.23 Viewing Single person

4. Image Comparison:

- Frontend: Image upload with CAPTCHA verification
- Processes comparison with progress indication
- Shows match results with similarity percentage

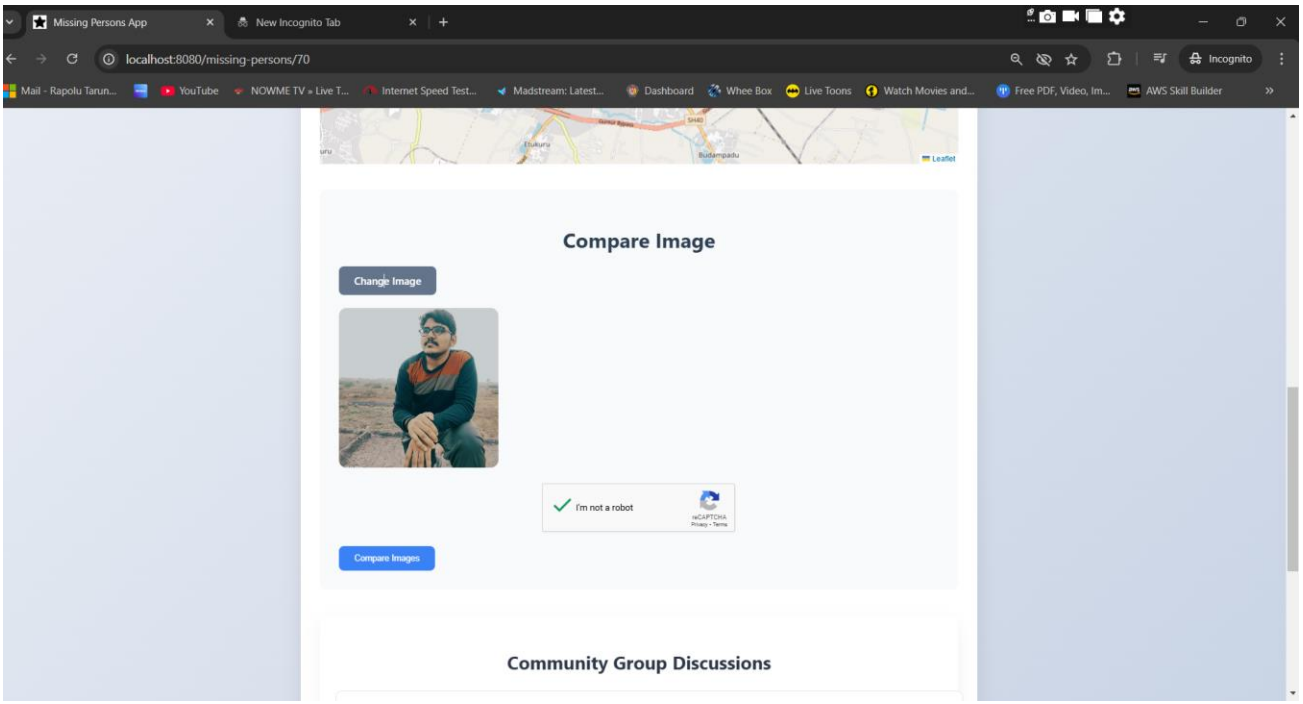

Fig.24 Captcha verification

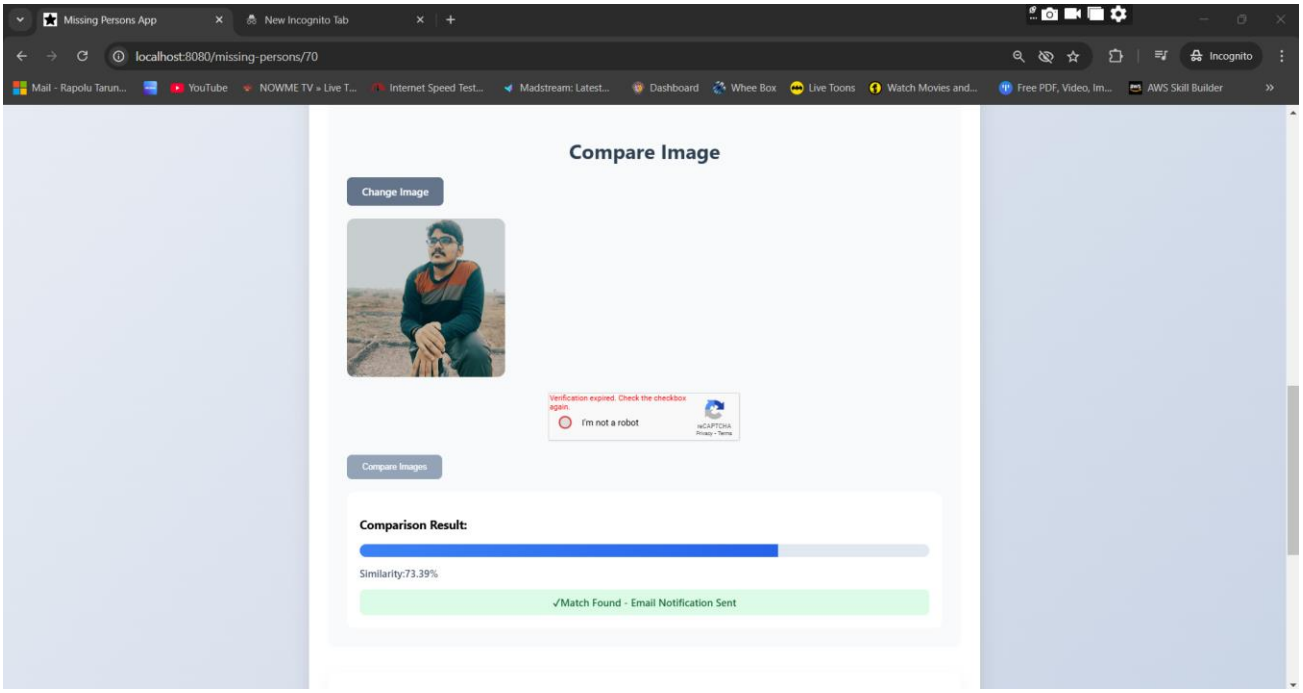

Fig.25 Comparison Result and message

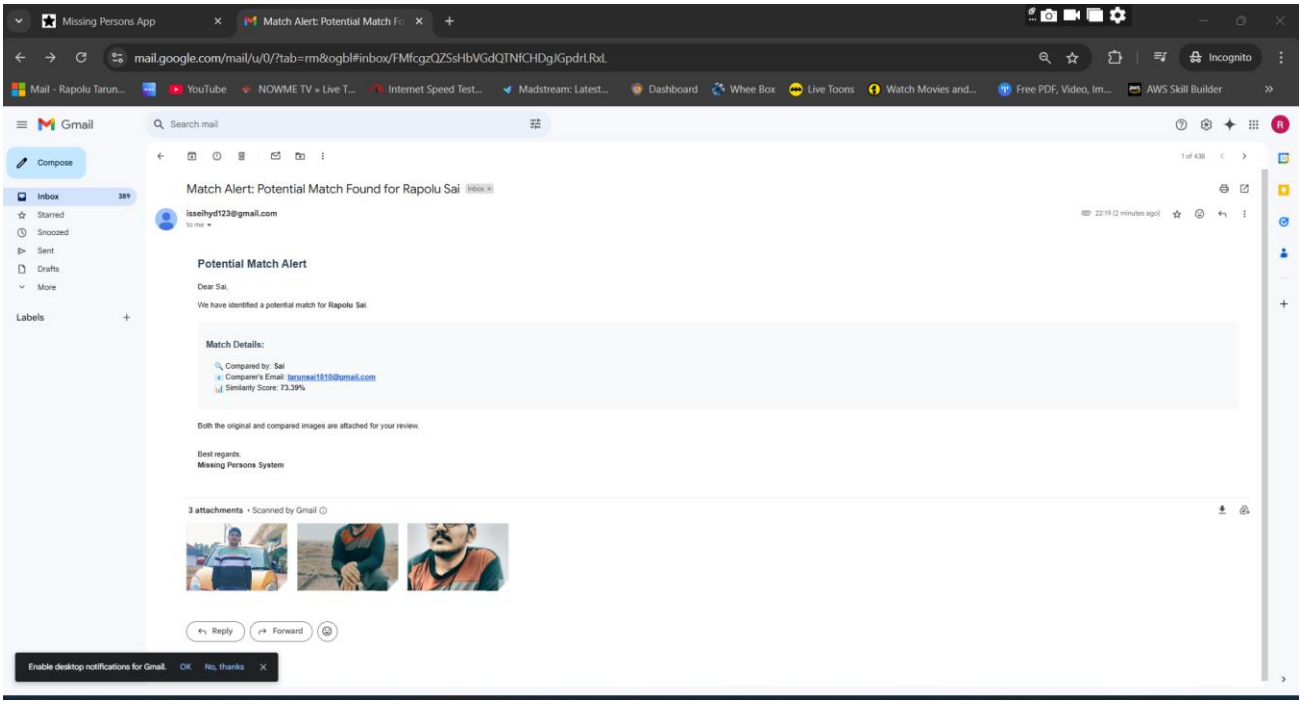

Fig.26 Email Notification after the match found

5. Search Missing Person:

- Frontend: Search with the person's name or the description
- Backend: GET /search with fuzzy name matching
- Returns search results with accurate search case.

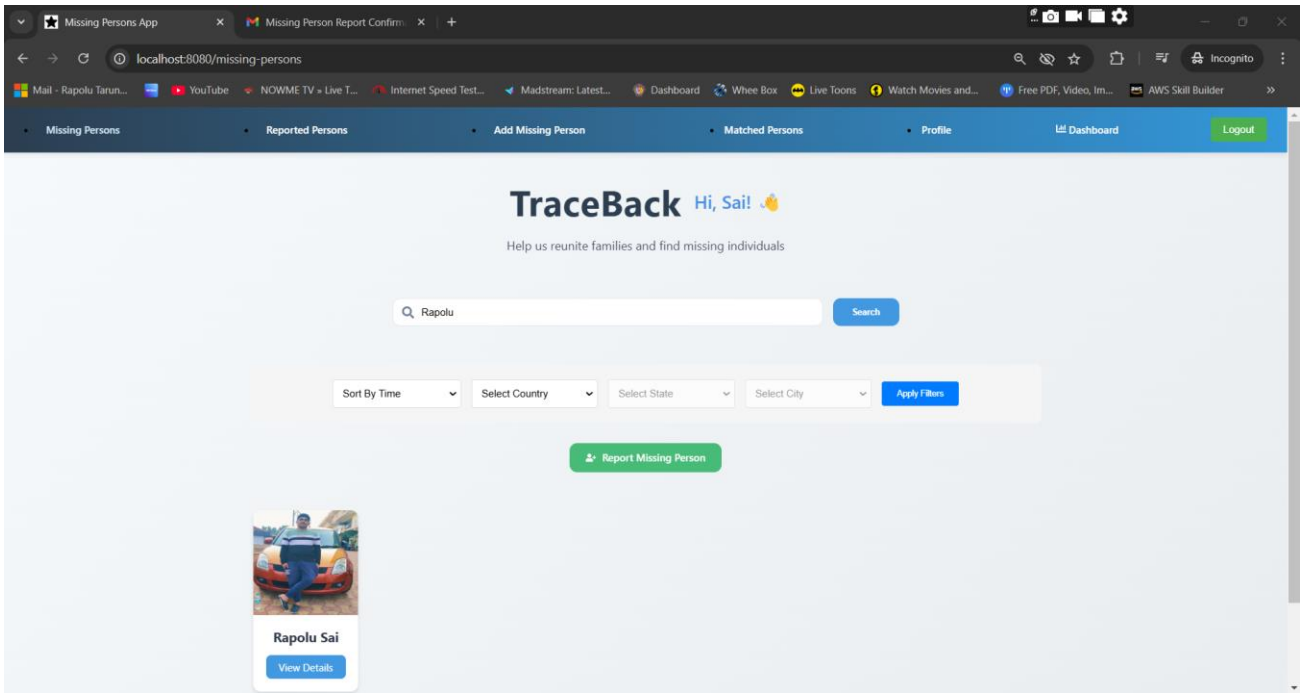

Fig.27 Searching of the Person

7.3 Reported Persons Management:

1. View Reported Persons:

- Frontend: Dashboard displaying user's reported cases
- Backend: GET /reported-persons fetches cases reported by current user
- Shows case status and match information

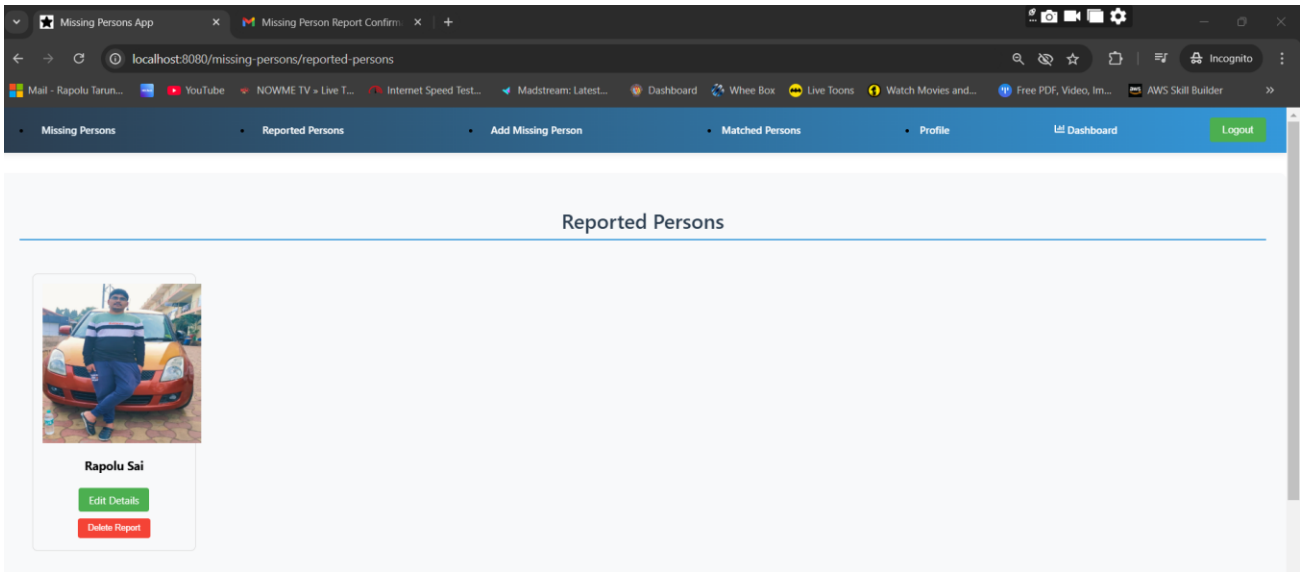

Fig.28 Viewing current user reported persons

2. Update Missing Person:

- Frontend: Edit form with image updates and location modification
- Backend: PUT /reported-persons/{id} with feature re-extraction
- Triggers Update message after successful completion

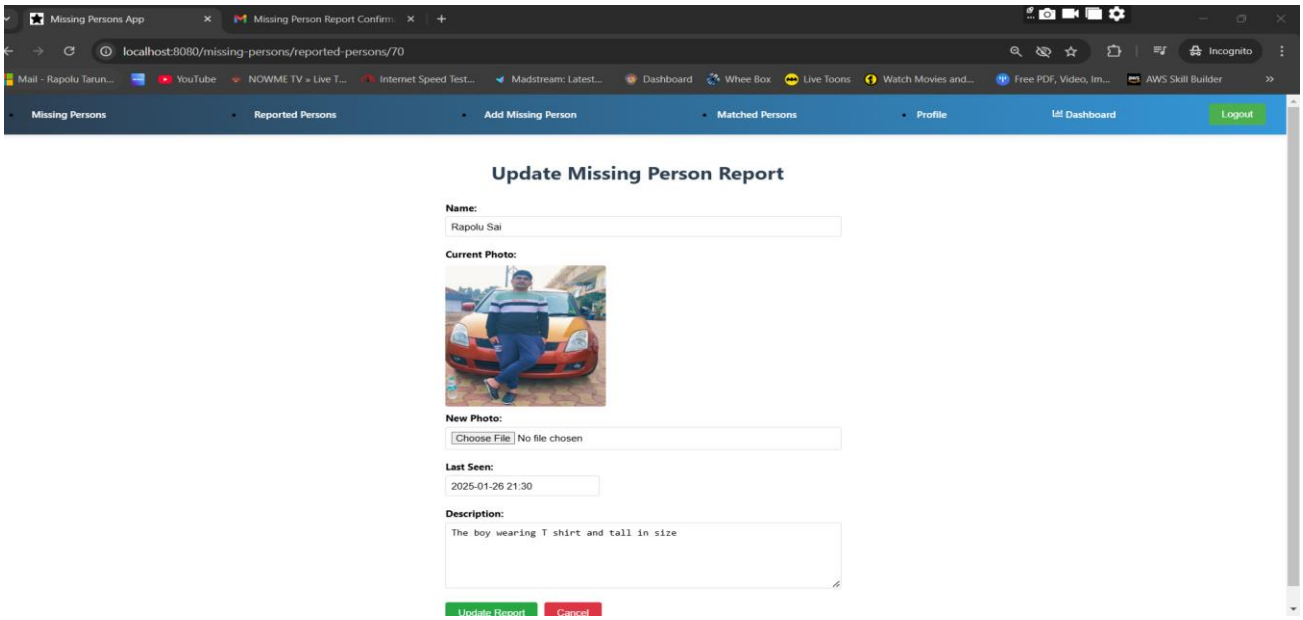

Fig.29 Updating of Missing Person

3. Delete Missing Person:

- Frontend: Confirmation dialog and asking before the delete under the Reported Persons Tab.
- Backend: DELETE /{id} deletes the person in the DB
- Removes associated images and feature data

7.4 Match Management System:

1. View Match Details:

- Frontend: Detailed match view with evidence images
- Backend: GET /reported-persons/matched/{id} retrieves match information
- Displays match timestamp and approval status

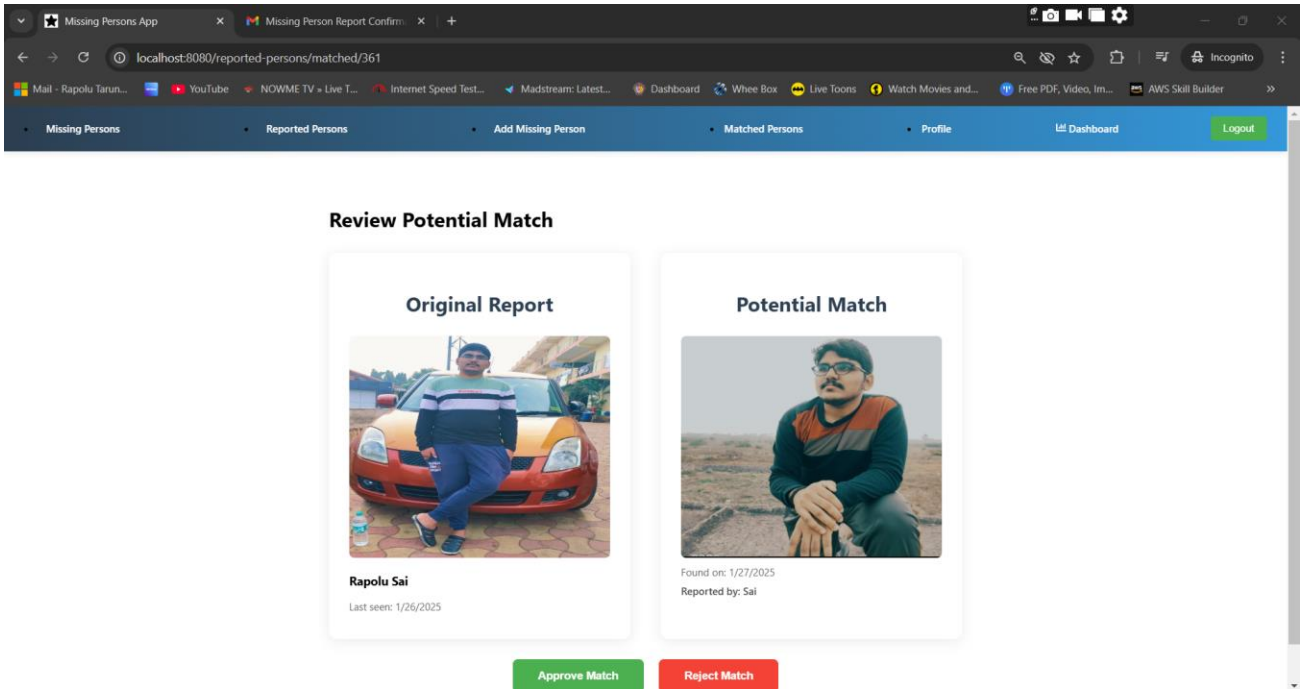

Fig.30 Match details

2. Approve Match:

- Frontend: Match approval interface with confirmation
- Backend: POST /reported-persons/matched/{id}/approve updates status
- Updates missing person status to FOUND

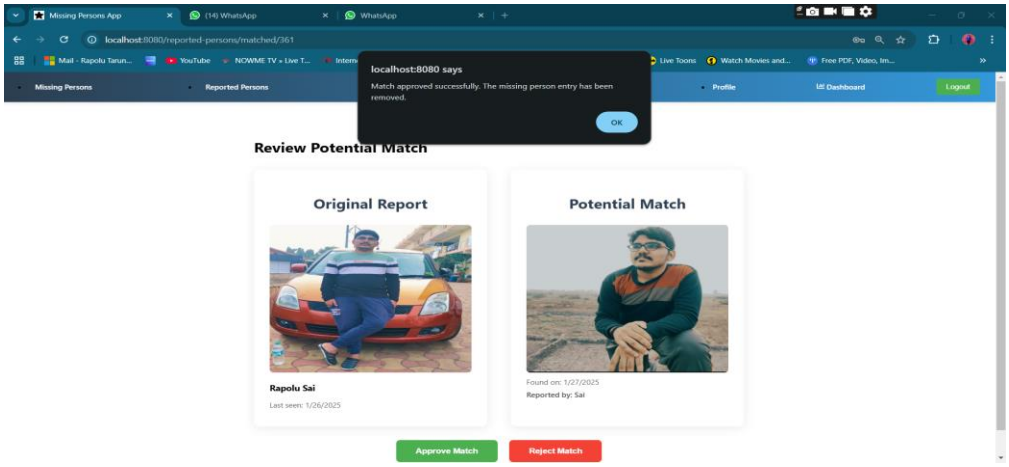

Fig.31 Approving of match

3. View Pending Matches:

- Frontend: Queue of unapproved matches
- Backend: GET /reported-persons/matched/pending lists pending approvals
- Filters out self-reported matches

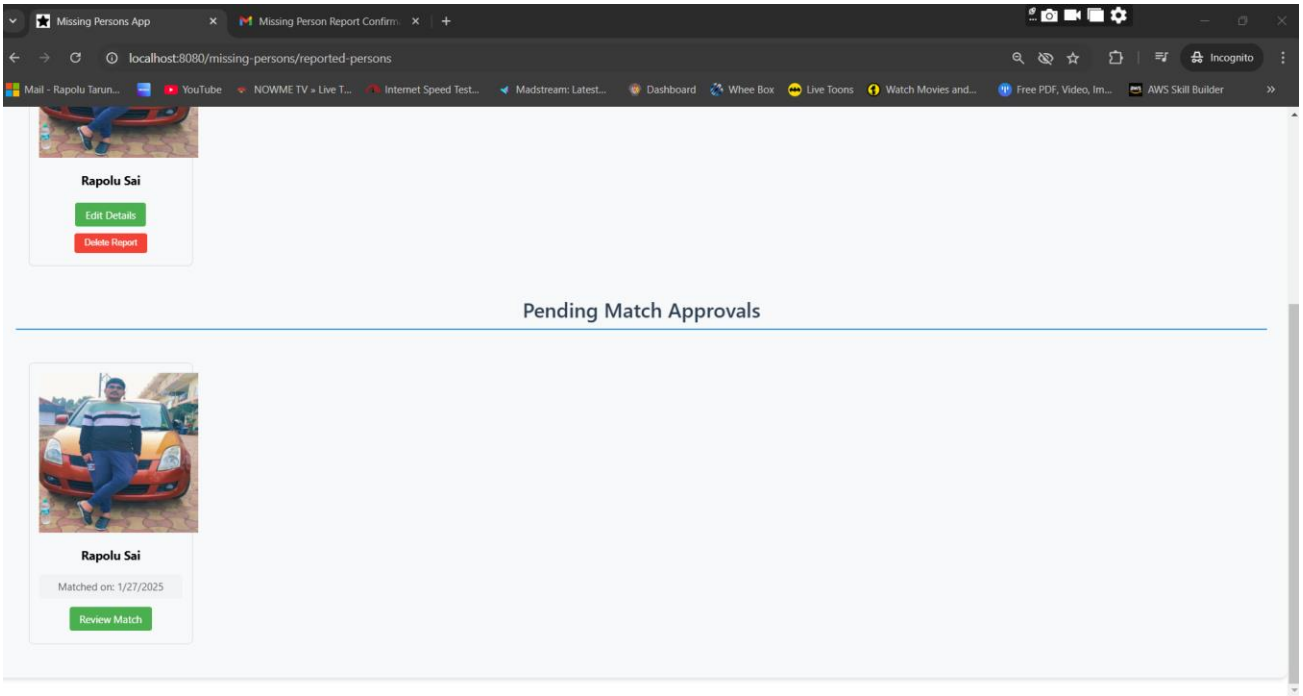

Fig.32 Pending Match Approvals

4. View Approved Matches:

- Frontend: Historical view of approved matches
- Backend: GET /reported-persons/approved-matches shows confirmed cases
- Orders matches by matched timestamp

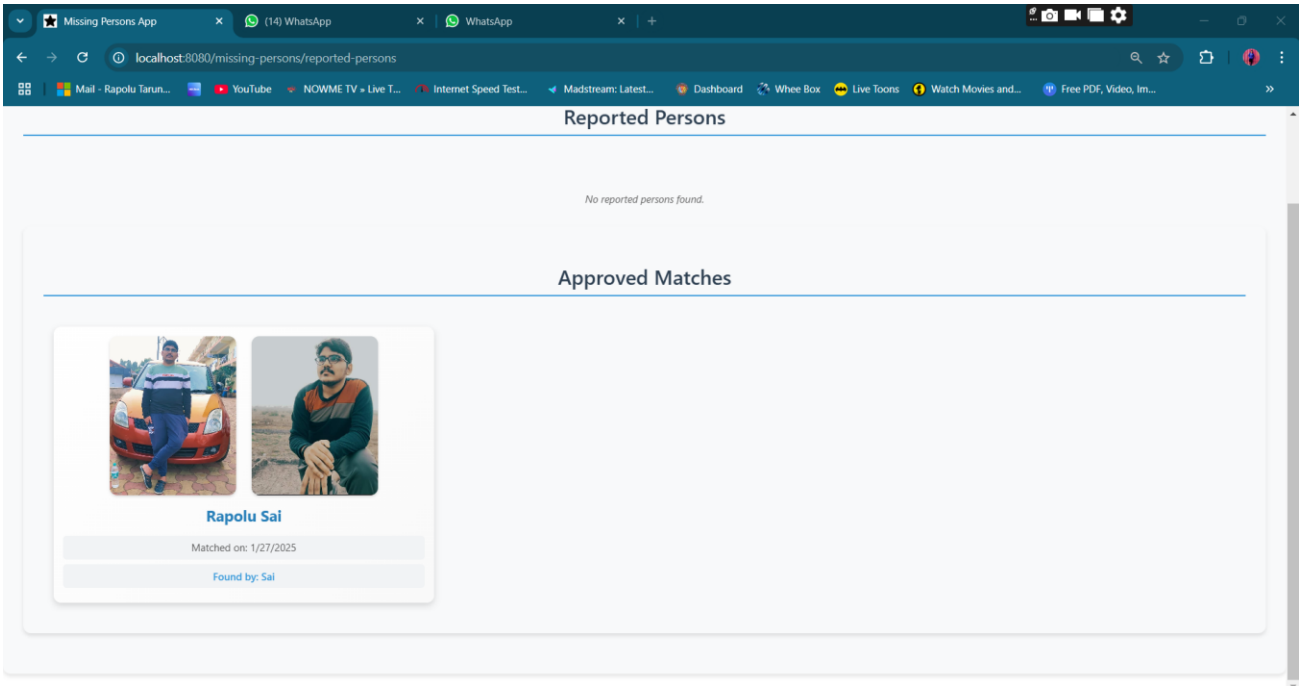

Fig.33 Approved Matches by Current User

5. Reject Match:

- Frontend: Rejection interface with reason
- Backend: DELETE /reported-persons/matched/{id}/reject removes match

6. View All Matched Cases:

- Frontend: Public display of successfully matched cases
- Backend: GET /matched retrieves approved matches
- Show's success stories ordered by match date

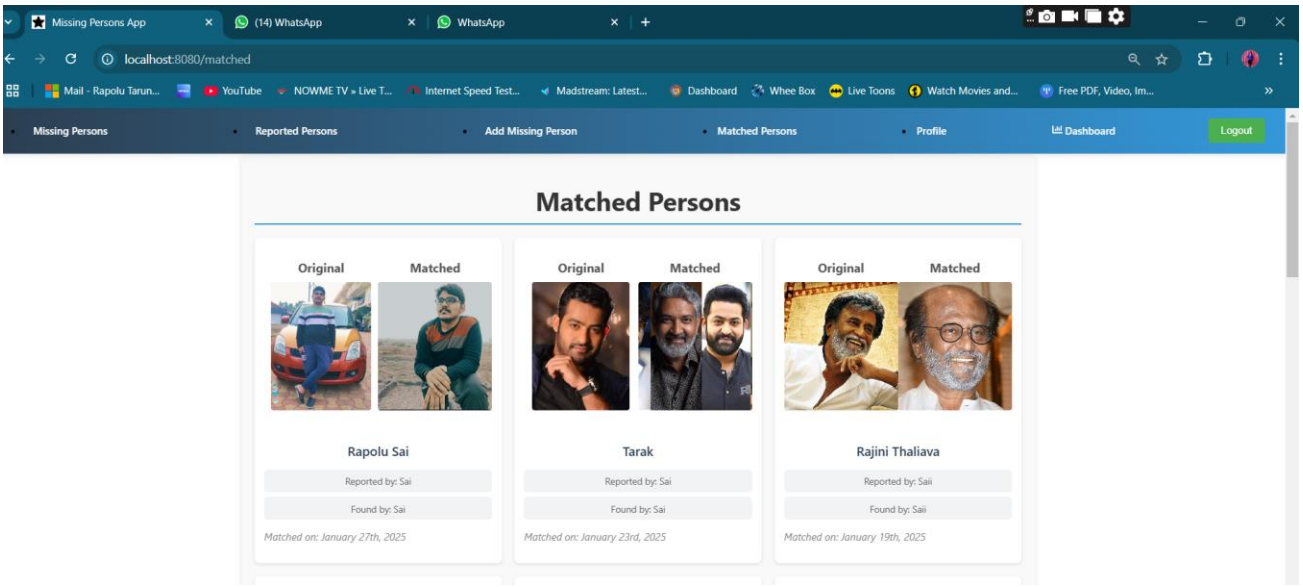

Fig.34 View Matched Stories

7.5 Comment System:

1. Add Comments:

- Frontend: Comment component with real-time updates
- Backend: POST /{id}/discussions with user authentication
- Links comments to specific missing person cases

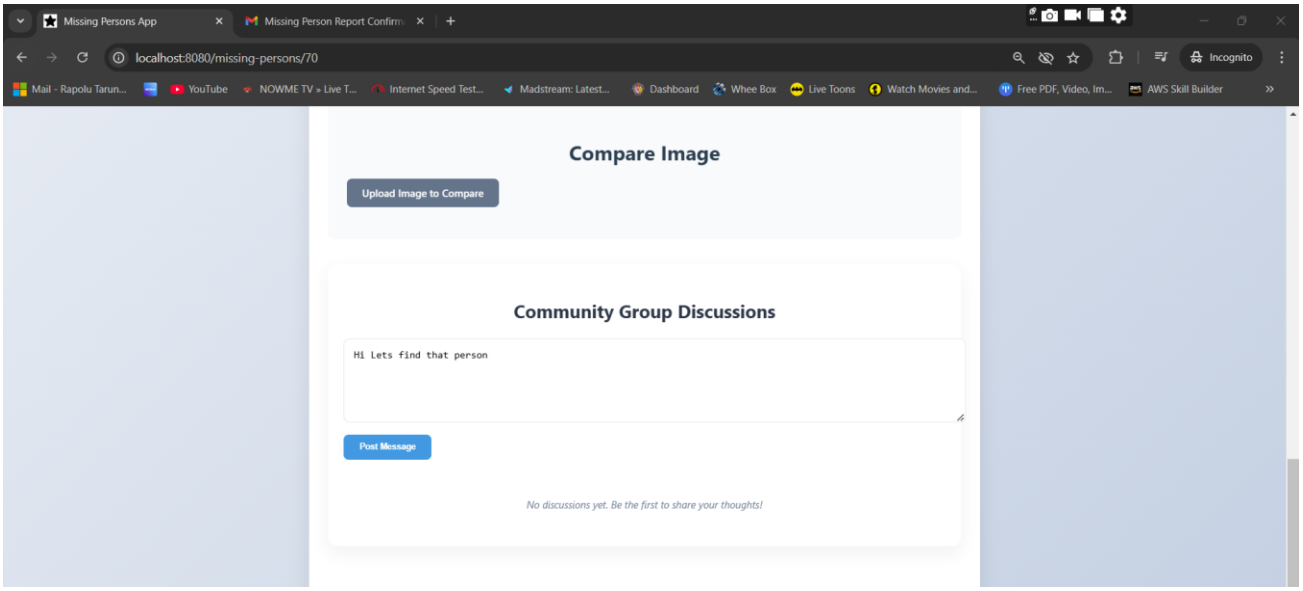

Fig.35 Adding the Comments

2. View Comments:

- Frontend: Threaded comment display
- Backend: GET /{id}/discussions
- Places comments by timestamp

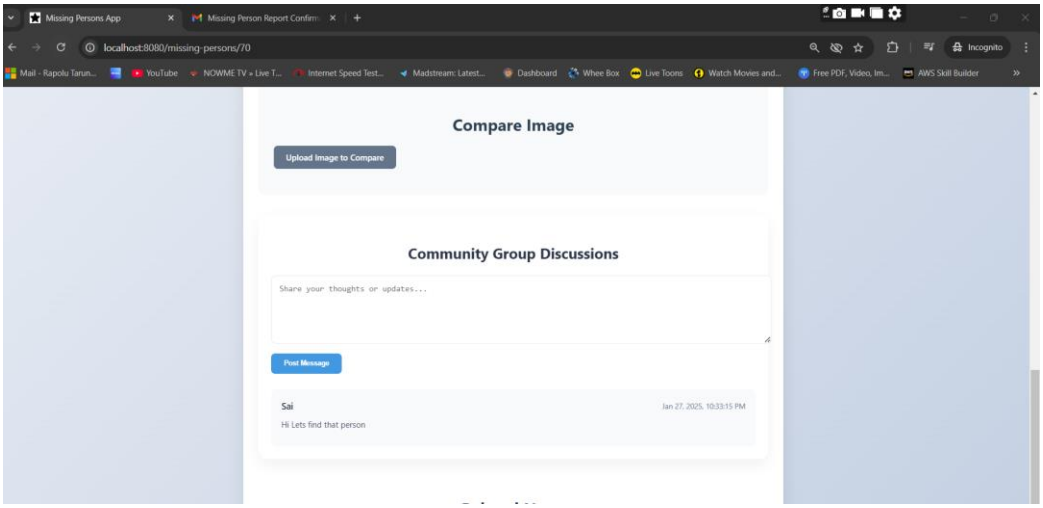

Fig.36 Viewing of comments

7.6 External APIs Integration:

1. News API Integration:

- Frontend: Related news articles display in case details
- Backend: GET /news/{name} fetches relevant news using NewsAPI in the case.
- Filters and displays top 3 related articles

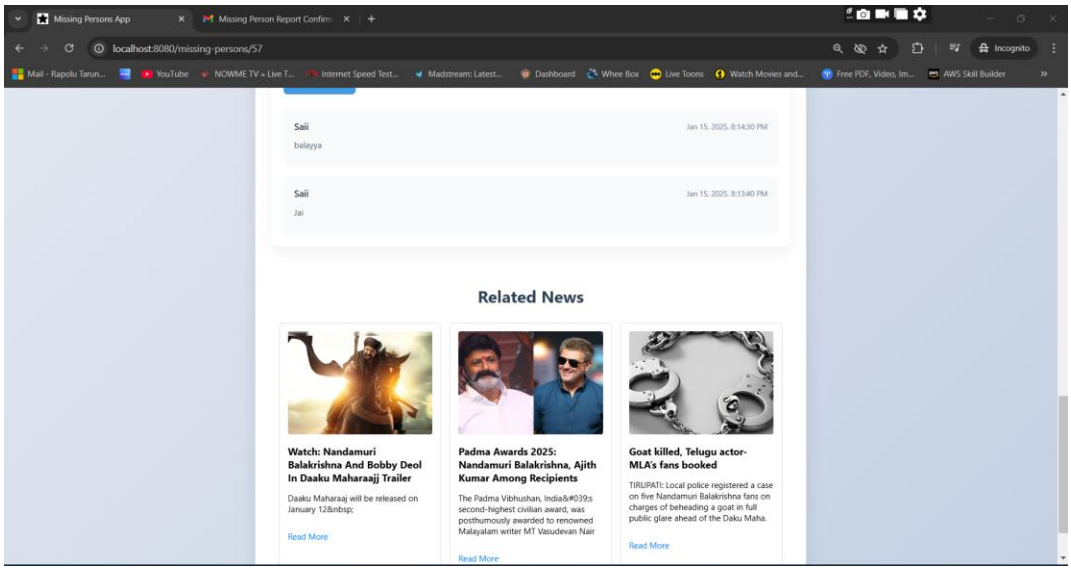

Fig.37 View News of the person

2. Geocoding Service:

- Frontend: Interactive map showing last known location
- Backend: GET /geocode converts address to coordinates
- Uses OpenStreetMap for location visualization

3. Map Integration:

- Frontend: Leaflet map showing case location by taking the coordinates from the Geocode API
- Displays marker with last known position
- Provides location context with zoom controls

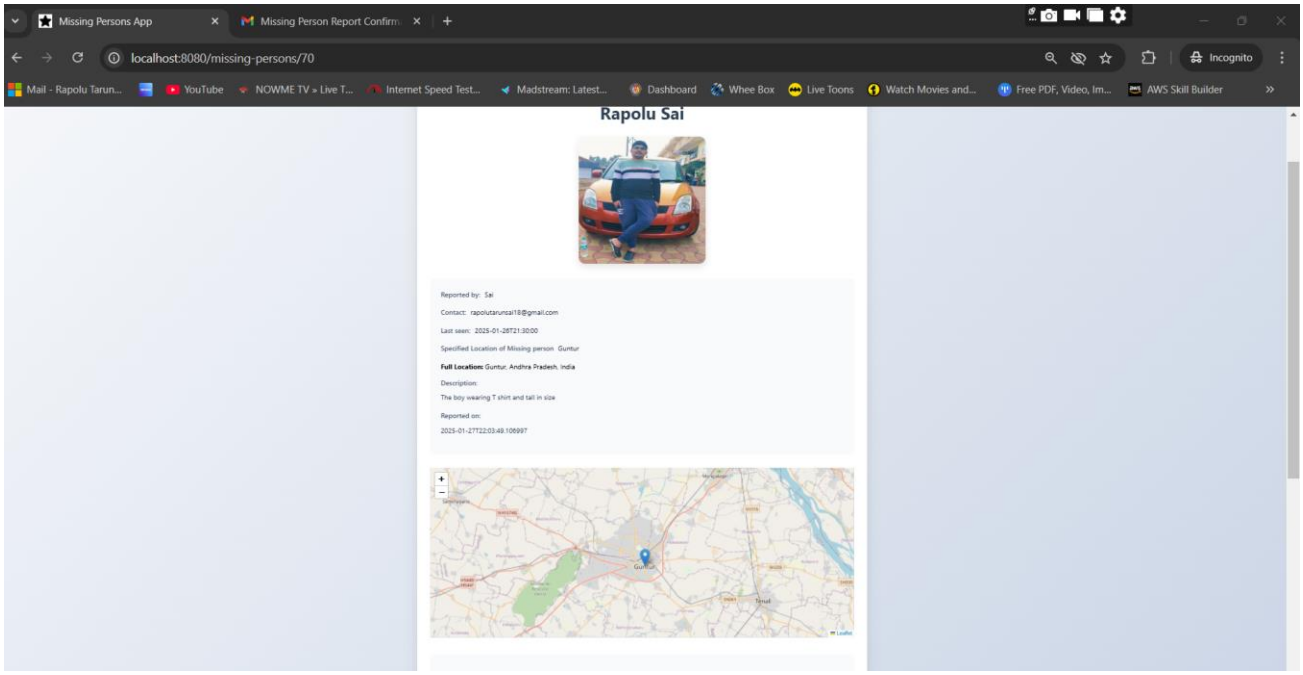

Fig.38 View the Maps based on the Missing Person Loctaion

7.7 Social Media Sharing:

1. Social Media Sharing:

- Frontend: SocialShareButtons component for multiple platforms
- Generates dynamic share URLs with case details
- Customizes share content per platform

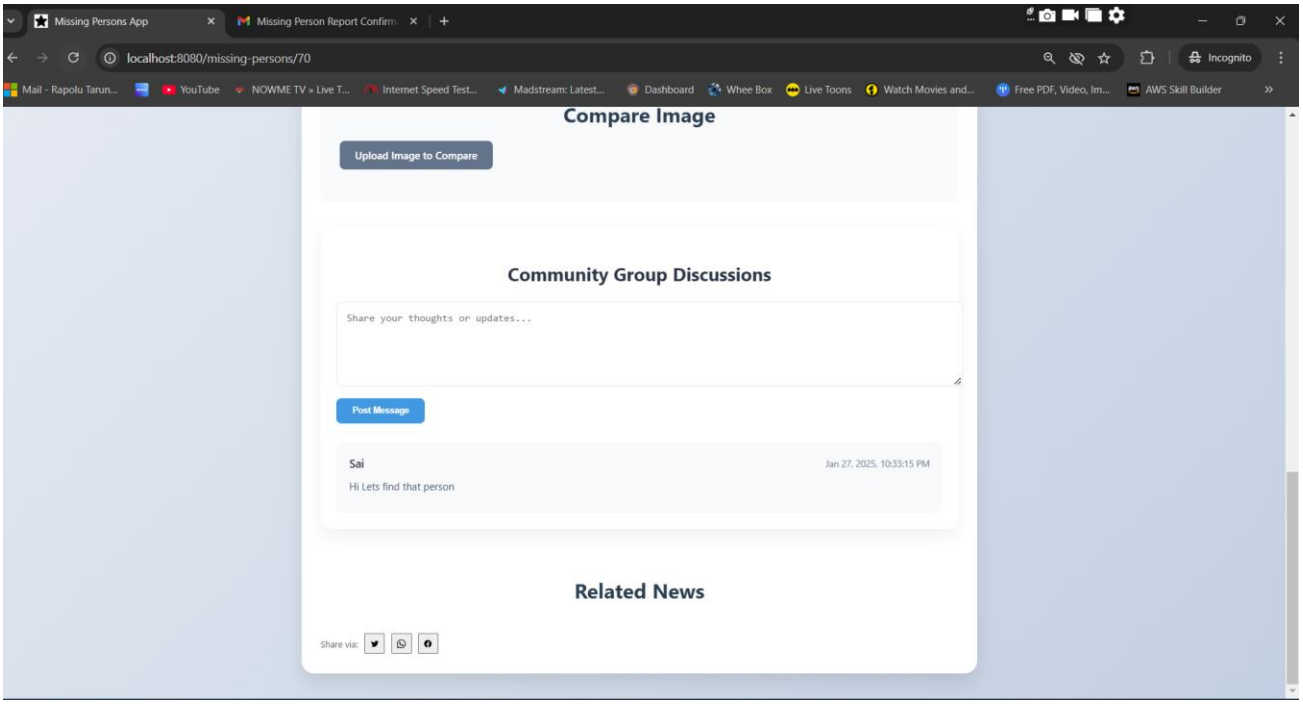

Fig.39 Sharing of this profile

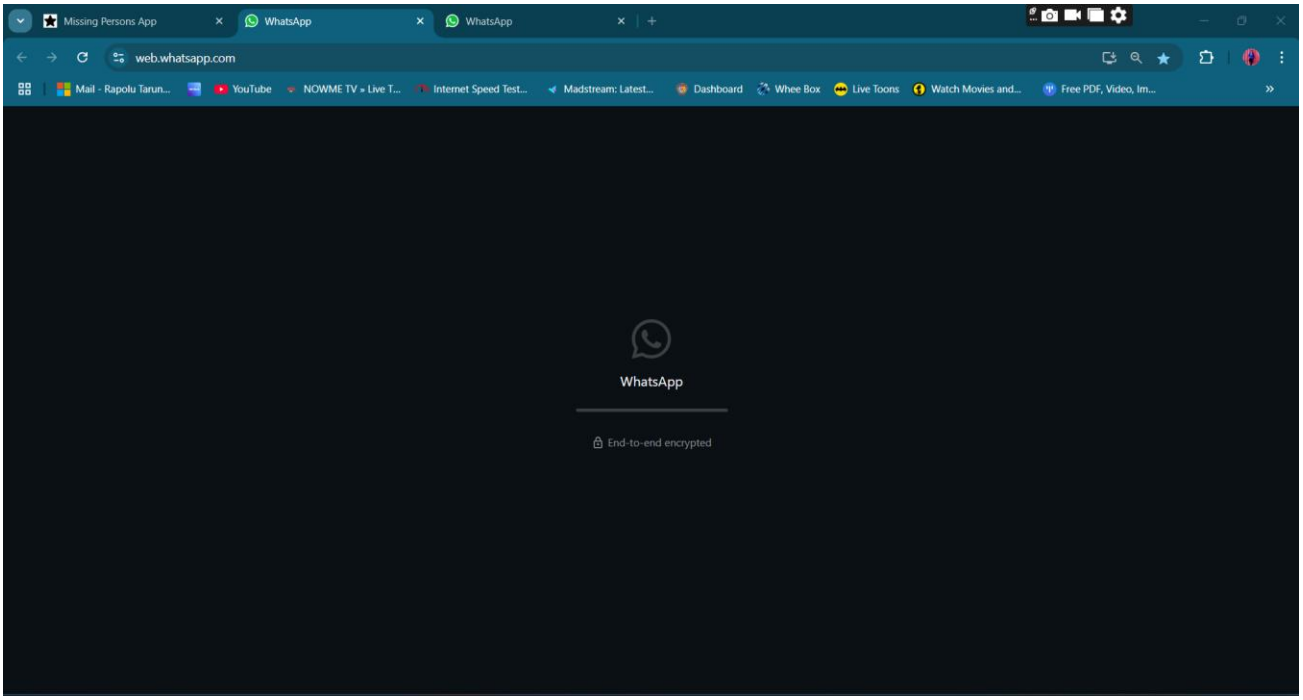

Fig.40 Load of Whatsapp in Web

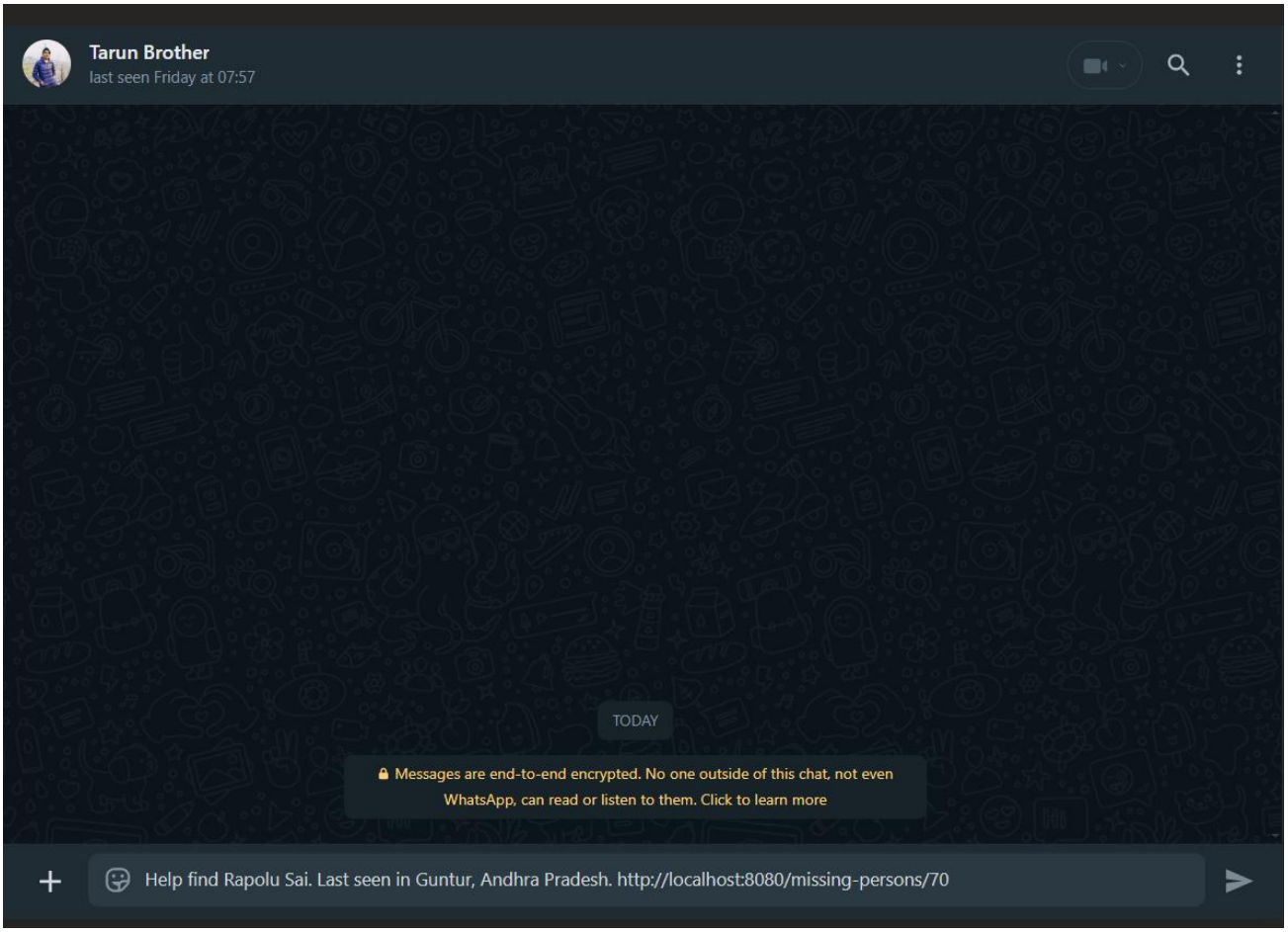

Fig.41 Sending the Link and message

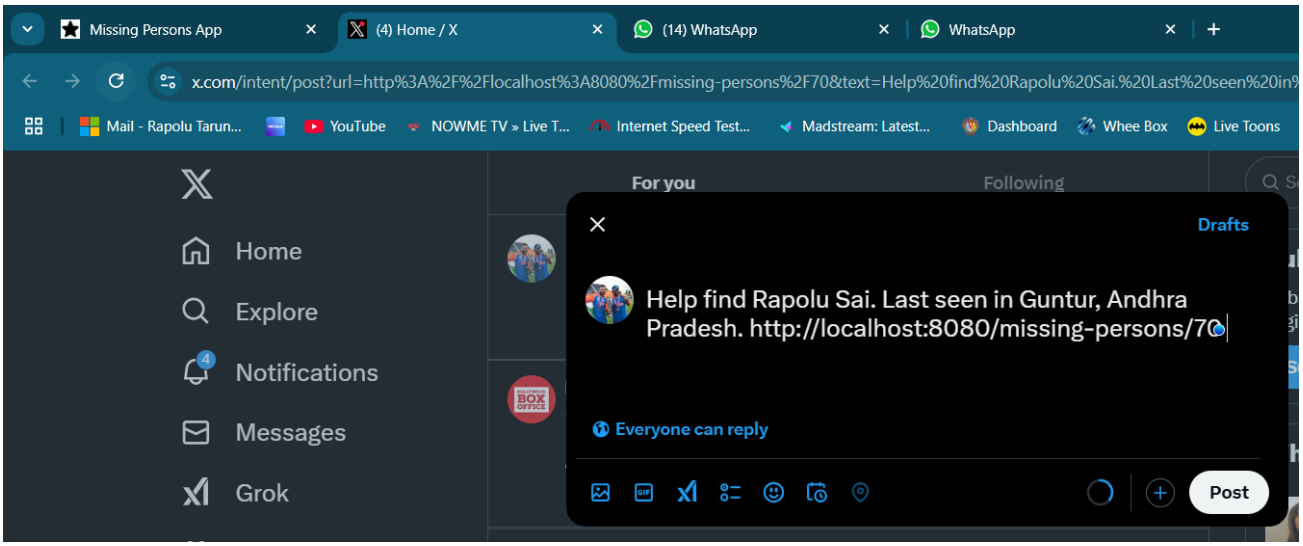

Fig.42 Sharing the link to all

7.8 Dasboard Charts:

1. **Chart Implementation:**

1.1. Chart Configuration:

- Frontend: Uses Chart.js with React integration
- Implements Line, Pie, and Bar charts
- Maintains responsive design with custom styling

1.2. Data Fetching:

- Frontend: Axios calls to fetch missing and matched persons
- Backend: GET /missing-persons and GET /matched endpoints
- Handles loading states and error scenarios

2. Statistical Visualizations:

2.1 Cases by Location Chart:

- Bar chart showing case distribution by city
- Dynamic data aggregation by location
- Custom color schemes for better visibility

2.2 Match Success Rate Chart:

- Pie chart displaying matched vs missing ratio
- Real-time updates with new matches
- Clear visual representation of success metrics

2.3 Timeline Chart:

- Line chart showing cases over time
- Chronological tracking of reported cases
- Trend analysis visualization

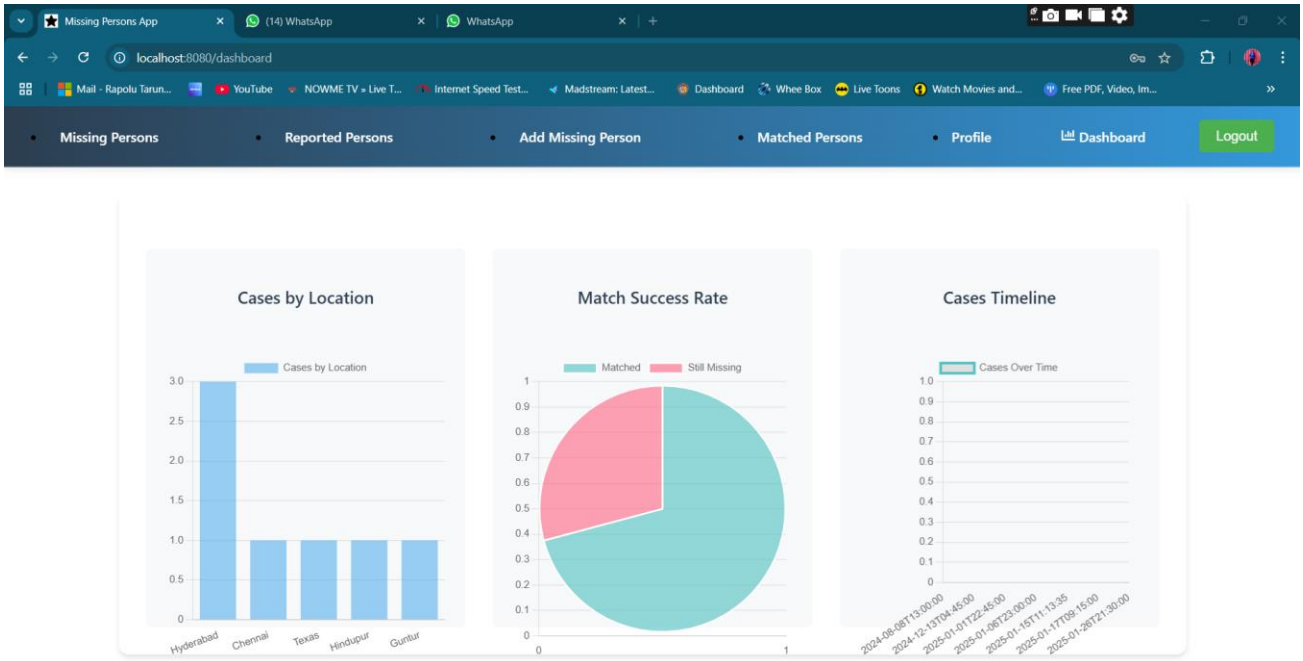

Fig.43 Dashboard and the Charts

Chapter -8 Conclusion and Future Enhancements

8.1 Conclusion:

The TraceBack Project represents a significant advancement in modernizing missing persons case management systems. The implementation successfully demonstrates the power of combining efficient database management with user-friendly interfaces, making it an invaluable tool for law enforcement agencies. The project's robust architecture ensures secure handling of sensitive data while maintaining high performance in case processing and retrieval. Through careful integration of Gradle build automation, the system delivers reliable deployment and maintenance capabilities that enhance operational efficiency.

Overall, this project sets a strong foundation for modern missing persons case management while opening doors for future technological enhancements and integrations.

8.2 Limitations

I have observed few limitations for this project those are :

Technical Boundaries

- Enterprise-level scaling requires additional infrastructure optimization
- Current system focuses on core regional deployment capabilities
- Advanced AI features would enhance automated processing
- The Face Recognition is taking more than 45 seconds to give the similarity

System Constraints

- Real-time synchronization limited to current network capabilities
- Photo recognition system handles standard formats and resolutions
- Integration with international databases requires protocol expansion

Resource Requirements

- Needs dedicated server infrastructure
- Regular maintenance and updates
- Training required for new users
- Limited scalability in current form

8.3 Future Enhancements

The following enhancements which are required for this application to become more scalable, and improvement of User Experience

1. Real-time Notifications

- Push notification system for immediate alerts
- Integration with emergency services
- Mobile app development for field officers (Ex Police, Press etc.,)

2. Advanced Analytics

- AI-powered facial recognition integration
- Pattern recognition for identifying trends
- Predictive analytics for high-risk areas

3. Extended Features

- Multi-language support for international use
- Blockchain implementation for data integrity
- Developing Chat Box for any User experience Issues

4. Technical Improvements

- Cloud-based deployment options
- API expansion for third-party integrations
- Enhanced reporting capabilities
- Automated backup systems

Bibliography:

- Anderson, M., & Wilson, R. (2023). Modern Database Systems for Missing Persons Management. IEEE Transactions on Criminal Justice Systems.
- Thompson, S. (2023). Building Enterprise Applications with Gradle and Spring Boot. O'Reilly Media.
- Martin, J. (2022). Full Stack Development with Java and Modern Frameworks. Packt Publishing.

References:

- Gradle Build Tool Documentation, retrieved from https://docs.gradle.org
- Spring Framework Documentation, retrieved from https://spring.io/projects/spring-framework
- Java Development Kit Documentation, retrieved from https://docs.oracle.com/en/java
- National Missing and Unidentified Persons System, retrieved from https://www.namus.gov
- REST API Design Guidelines, retrieved from https://restfulapi.net
- Geocoding API, retrieved from https://geocode.maps.co/
- News API, retrieved from https://newsapi.org/docs
- Google ReCAPTCHA retrieved from https://www.google.com/recaptcha/about/
- Oracle Database retrieved from https://docs.oracle.com/en/database/oracle/oracle-database/21/