A Project Report On

Trash2Points

Submitted in partial fulfillment of the requirement for the award of the degree

Master of Computer Applications
(MCA)

Academic Year 2025 – 26

Vishal Kanani (9240058409)
Prasham Makwana (92400584106)
Manan Solanki (92400584107)

Internal Guide
Dr. Jaypalsinh Gohil

Rajkot-Morbi Road, At &PO :Gauridad, Rajkot 360 003. Gujarat. India.

Marwadi University
Marwadi Chandarana Group

NAAC A+

Faculty of Computer Applications (FoCA)

Certificate

This is to certify that the project work entitled

Trash2Points

submitted in partial fulfillment of the requirement for the award of the degree of

Master of Computer Applications (MCA)

of the

Marwadi University

is a result of the bonafide work carried out by

Vishal Kanani (92400584098)

Prasham Makwana (92400584106)

Manan Solanki (92400584107)

during the academic year 2025-26

Faculty Guide	HOD	Dean

DECLARATION

We hereby declare that this project work entitled Trash2Points is a record done by us.

We also declare that the matter embodied in this project is genuine work done by us and has not been submitted whether to this University or to any other University / Institute for the fulfillment of the requirement of any course of study.

Place:

Date:

Vishal Kanani (92400584098) Signature:_____
Prasham Makwana (92400584106) Signature:_____
Manan Solanki (92400584107) Signature:_____

ACKNOWLEDGEMENT

It is indeed a great pleasure to express our thanks and gratitude to all those who helped us. No serious and lasting achievement or success one can ever achieve without the help of friendly guidance and co-operation of so many people involved in the work.

We are very thankful to our guide **Dr. Jaypalsinh Gohil,** the person who makes us to follow the right steps during our project work. We express our deep sense of gratitude to for his guidance, suggestions and expertise at every stage. A part from that his valuable and expertise suggestion during documentation of our report indeed help us a lot.

Thanks to our friend and colleague who have been a source of inspiration and motivation that helped to us during our project work.

We are heartily thankful to the Dean of our department **Dr. R. Sridaran** sir and HoD **Dr. Sunil Bajeja** sir for giving us an opportunity to work over this project and for their end-less and great support to all other people who directly or indirectly supported and help us to fulfil our task.

Vishal Kanani (92400584098) Signature:_____
Prasham Makwana (92400584106) Signature:_____
Manan Solanki (92400584107) Signature:_____

CONTENTS

TABLE INDEX

FIGURE INDEX

CHAPTER 1
INTRODUCTION TO PROJECT DEFINITION

➢ Trash2Points

Trash2Points is a smart waste-reporting application with two main panels one for users and one for the admin. The user panel is developed using **Android in Kotlin**, allowing users to register, log in, and report unclean areas by uploading images, adding descriptions, and selecting locations. Users can also view their submitted reports and track their status.

The admin panel is developed using the **MERN stack** (MongoDB, Express.js, React.js, Node.js). It allows admins to manage all reports submitted by users. Admins can view, update the status (like “Pending”, “Cleaned”, or “Rejected”), and delete reports as needed. All data is securely stored in **MongoDB Atlas**, a cloud-based database that connects both panels.

This project aims to involve citizens in keeping their surroundings clean by reporting garbage spots directly through their phones. It creates a bridge between users and the authorities, making communication faster and action more effective. In the future, more features like reward points or notifications can be added to improve user engagement.

CHAPTER 2
PREAMBLE

Trash2Points is an initiative to promote cleanliness by encouraging users to report garbage or unclean areas in their surroundings. The application empowers users to take a photo, describe the issue, and mark the location, making it easy for the concerned authorities to take necessary action. It bridges the gap between the public and the local or government authorities through the use of mobile and web technologies.

The project is divided into two main panels:

- **User Panel (Android - Kotlin):** Allows users to register, login, submit reports with images and location, and view the status of their past reports.
- **Admin Panel (Web - MERN Stack):** Admins can view, edit, update, and delete reports. They can also change the status of each report.

2.1 Module description

1. User (Client) Panel

- User Registration and Login
- Submit garbage spot reports (image, location, description)
- View status of submitted reports

2. Admin Panel

- View list of submitted reports
- Update report status (Pending, Cleaned, Rejected)
- Delete reports if needed

3. Report Management System

- Centralized MongoDB Atlas database
- Node.js and Express.js handle backend APIs
- RESTful APIs for smooth data handling between frontend and backend

CHAPTER 3

REVIEW OF LITERATURE

- **Swachhata App**

This app is developed by the Indian Government to let citizens report issues like garbage dumps, overflowing drains, and road damage. Users can upload photos, describe the issue, and mark the location. However, it is mostly limited to areas supported by the municipal corporation and may not work in all cities.

- **IChangeMyCity**

This app by Janaagraha helps people report problems like garbage, broken roads, or water leaks in their area. Users can take a photo, select the location, and send the complaint through the app. They can also check if their problem is being fixed. But the app works only in a few cities in India and may not be useful everywhere.

CHAPTER 4

TECHNICAL DESCRIPTION

4.1 Hardware Requirement

Component	Requirement
User Panel (Android App)	- Android Smartphone (Android 8.0 or above) - Minimum 2 GB RAM - Camera & GPS enabled - Internet connection (Wi-Fi/Mobile Data)
Admin Panel (Web)	- Laptop/Desktop with Intel i3 or above - Minimum 4 GB RAM - Internet Browser (Chrome/Firefox) - Stable Internet connection

Table 4.1 Hardware Requirement

4.2 Software Requirements

Component	Software / Tools Used
Android (User Panel)	- Android Studio (latest version) - Kotlin Language - Google Maps SDK - Internet Permission
Admin Panel (Web)	- Visual Studio Code (VS Code) - React.js (Frontend) - Tailwind CSS / Material UI (Styling) - Node.js & Express.js (Backend) - Retrofit (API calls)
Database	- MongoDB Atlas (Cloud Database) - Mongoose (Database Modeling)
Others	- Git & GitHub (Version Control) - Postman (API Testing)

Table 4.2 Software Requirement

CHAPTER 5

SYSTEM DESIGN AND DEVELOPMENT

5.1 Diagrams

- **Flow Chart**

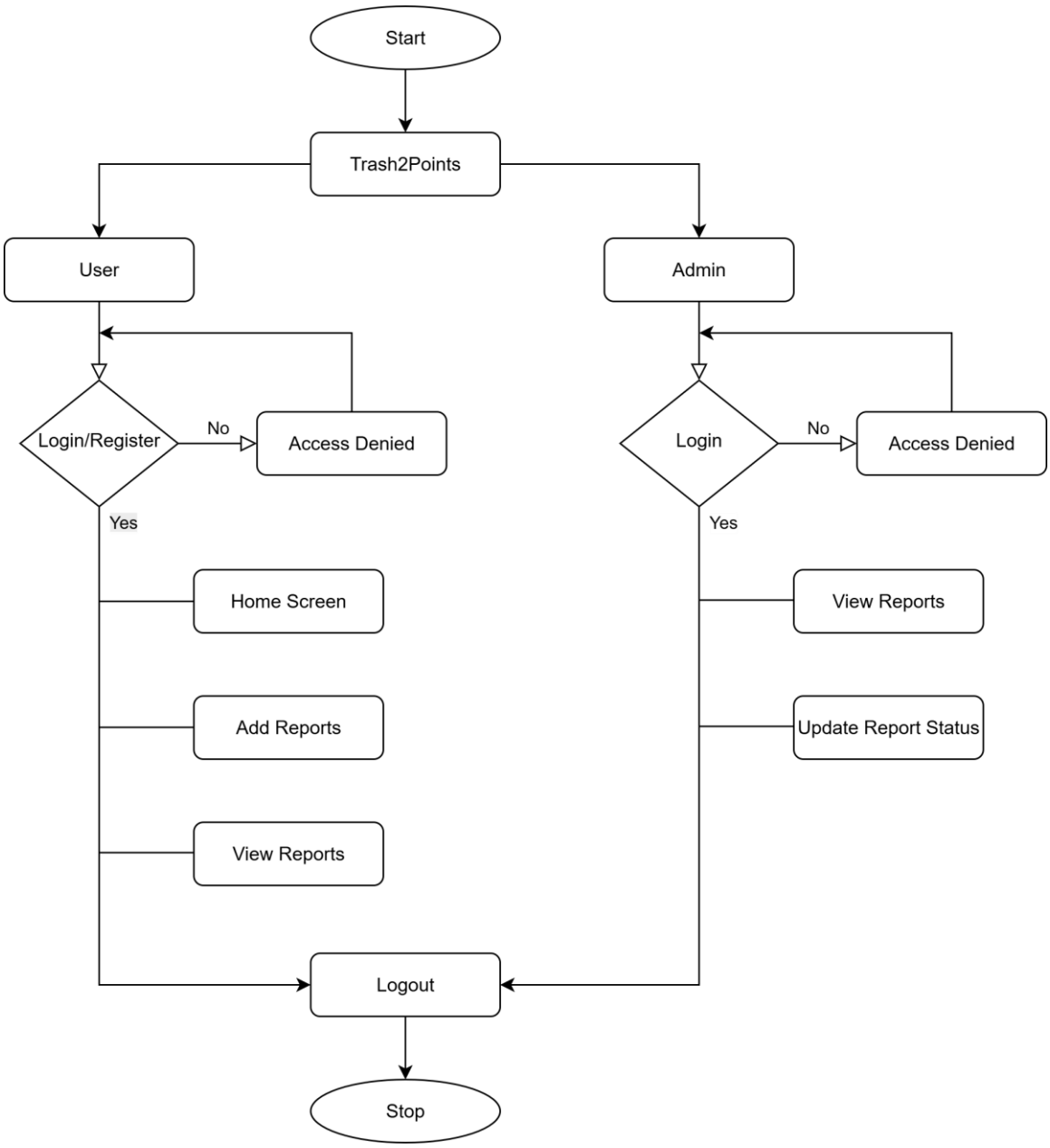


Figure 5.1 Flow Chart

- **Data Flow Diagram**
  - **Context Level DFD**

Figure 5.2 Context Level DFD

  - **User - 1st Level DFD**

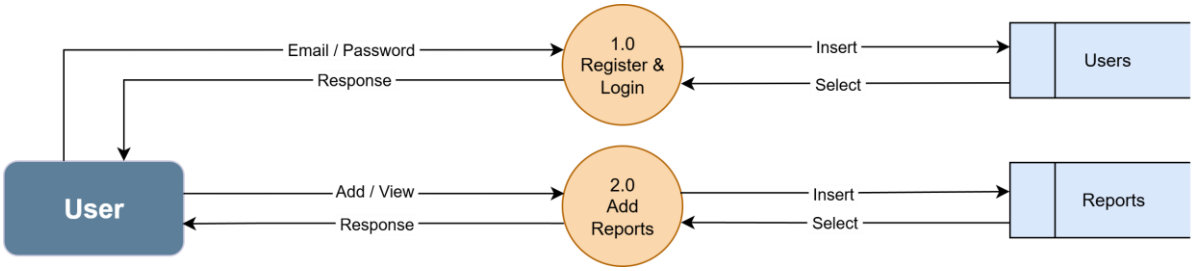


Figure 5.3 User – 1st Level DFD

  - **Admin – 1st Level DFD**

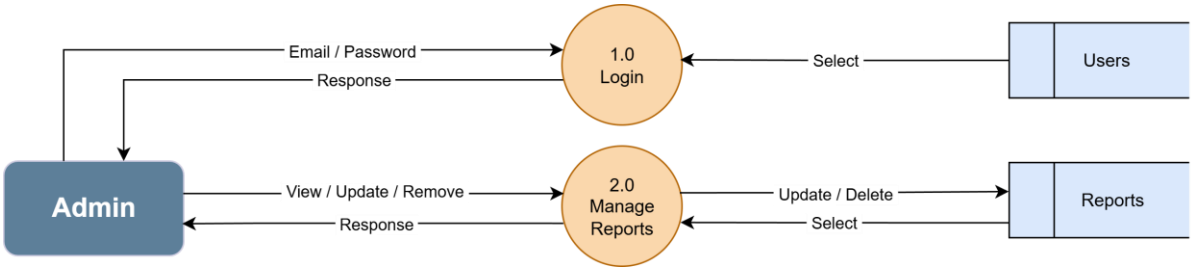


Figure 5.4 Admin – 1st Level DFD

- **Class Diagram**

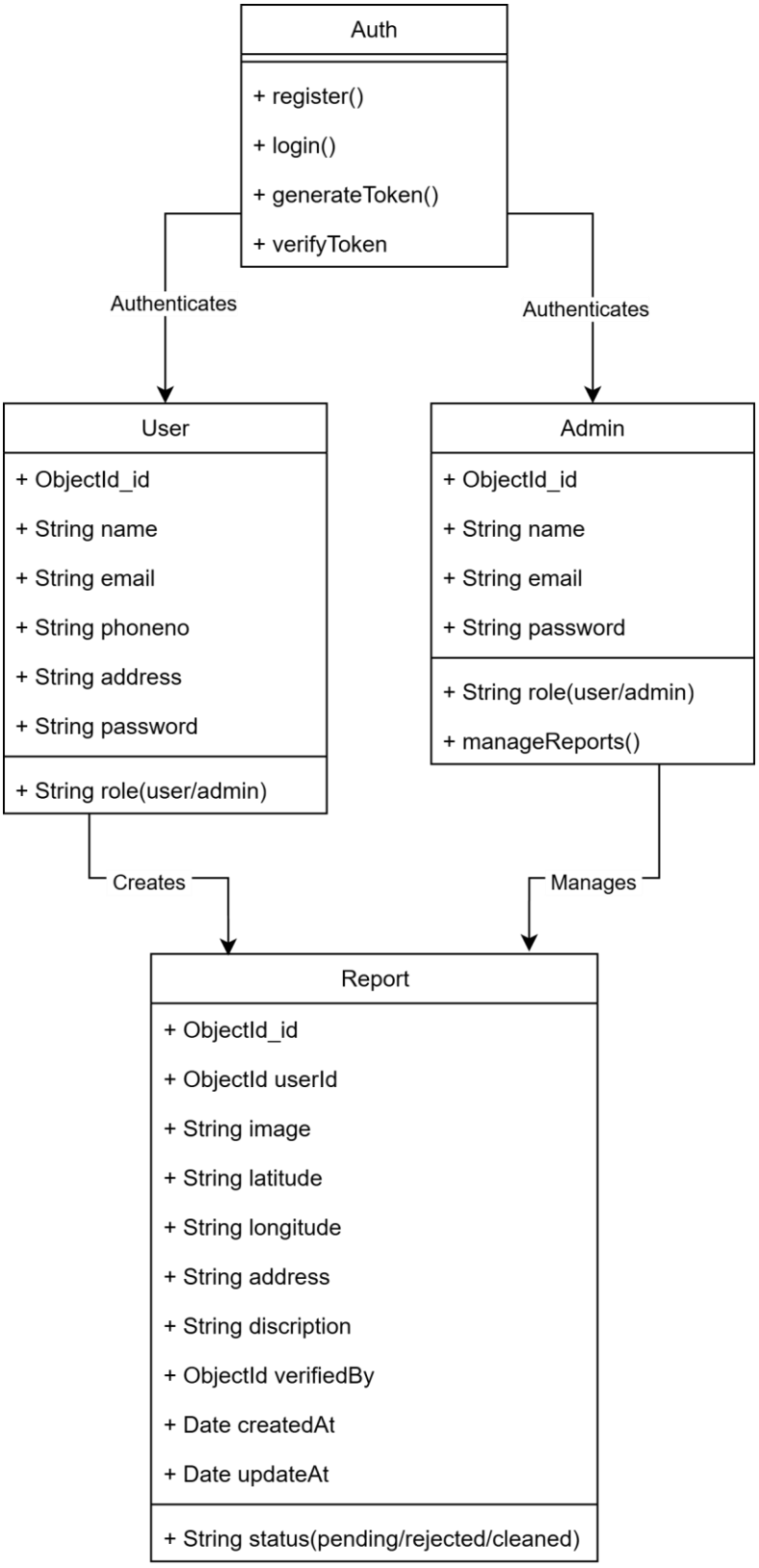


Figure 5.5 Class Diagram

- **Use Case Diagram**
  - **User**

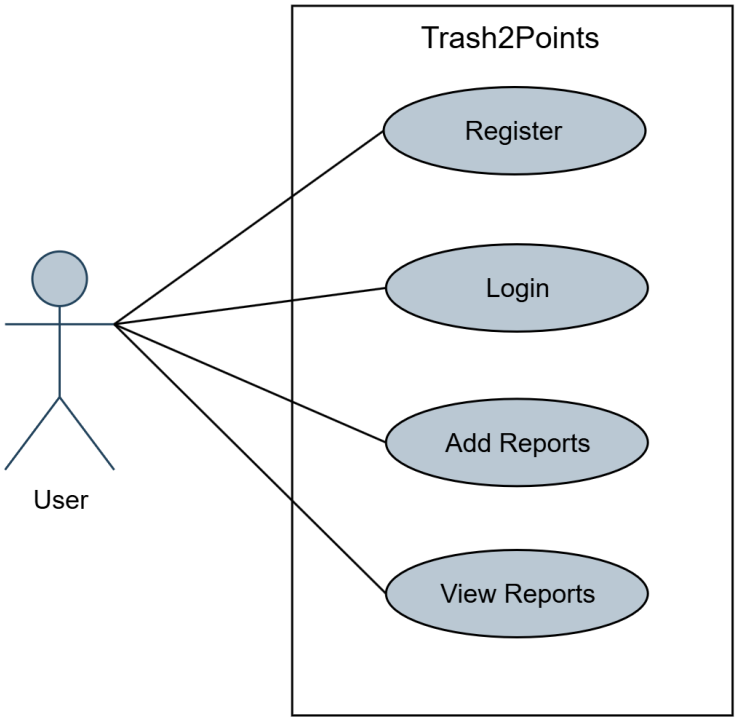


Figure 5.6 User Use Case Diagram

  - **Admin**

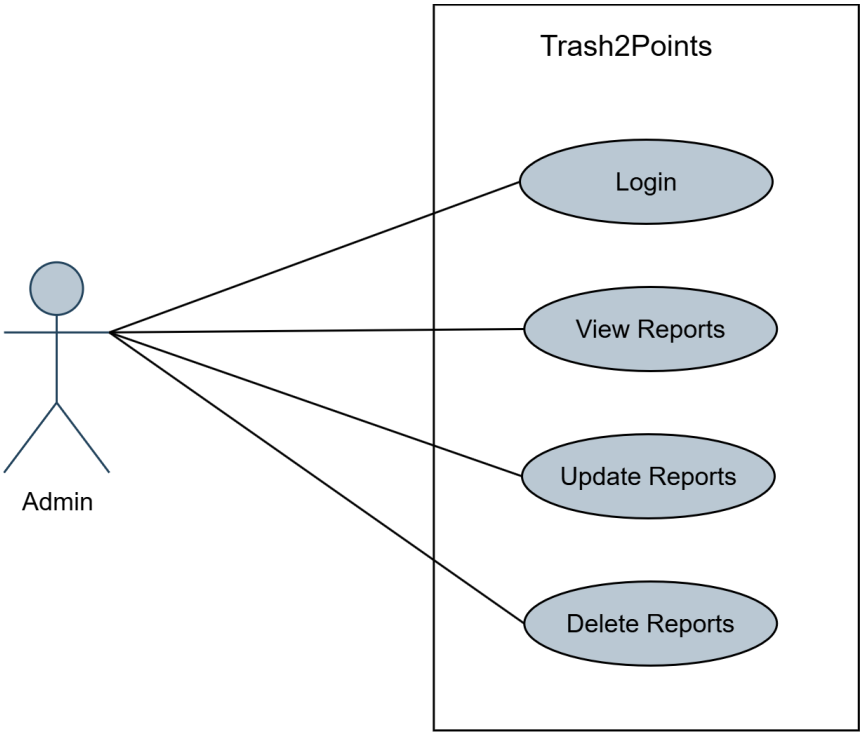


Figure 5.7 Admin Use Case Diagram

- **Sequence Diagram**
  - **User**

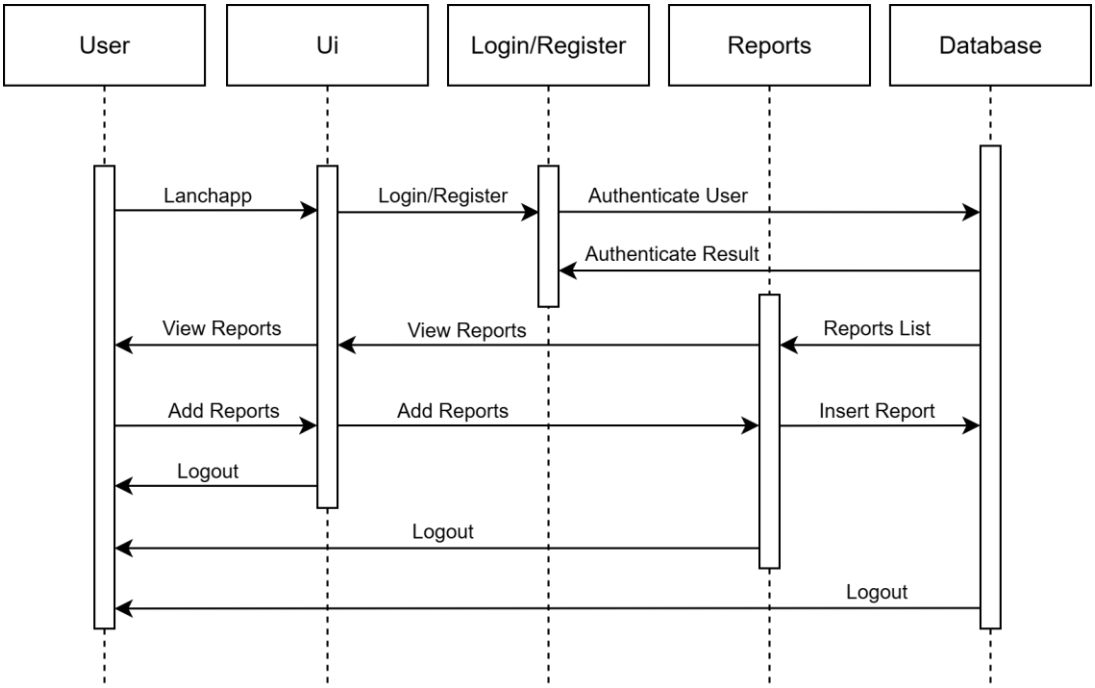


Figure 5.8 User Sequence Diagram

  - **Admin**

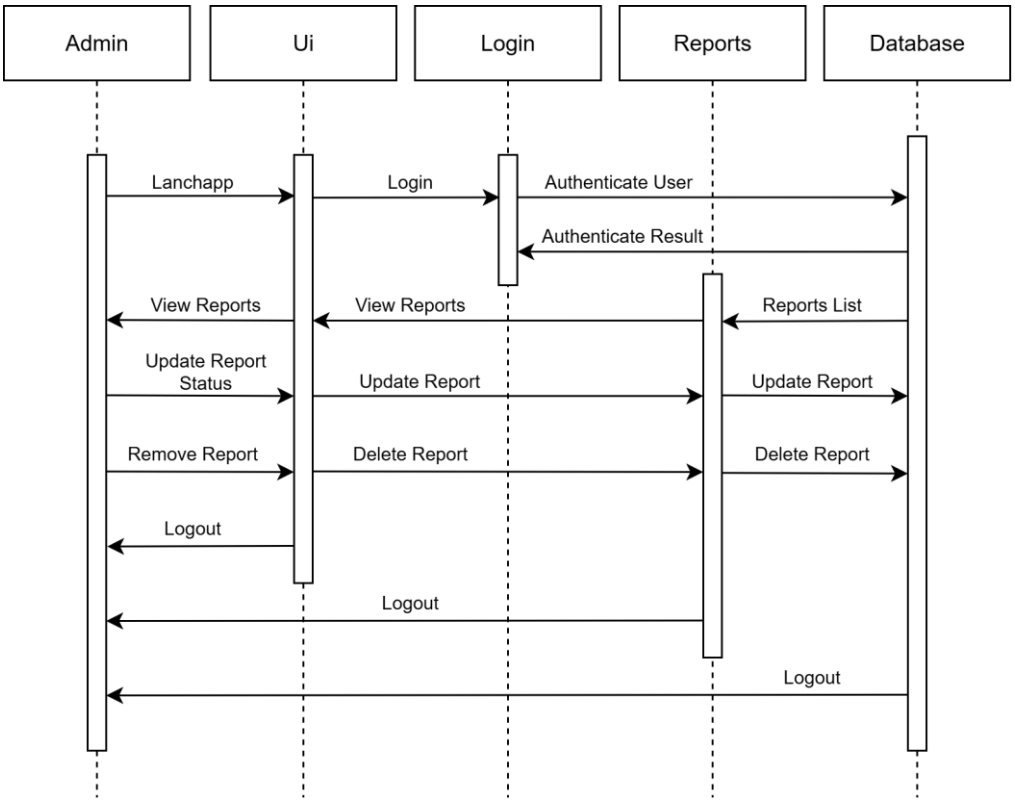


Figure 5.9 Admin Sequence Diagram

- **Activity Diagram**

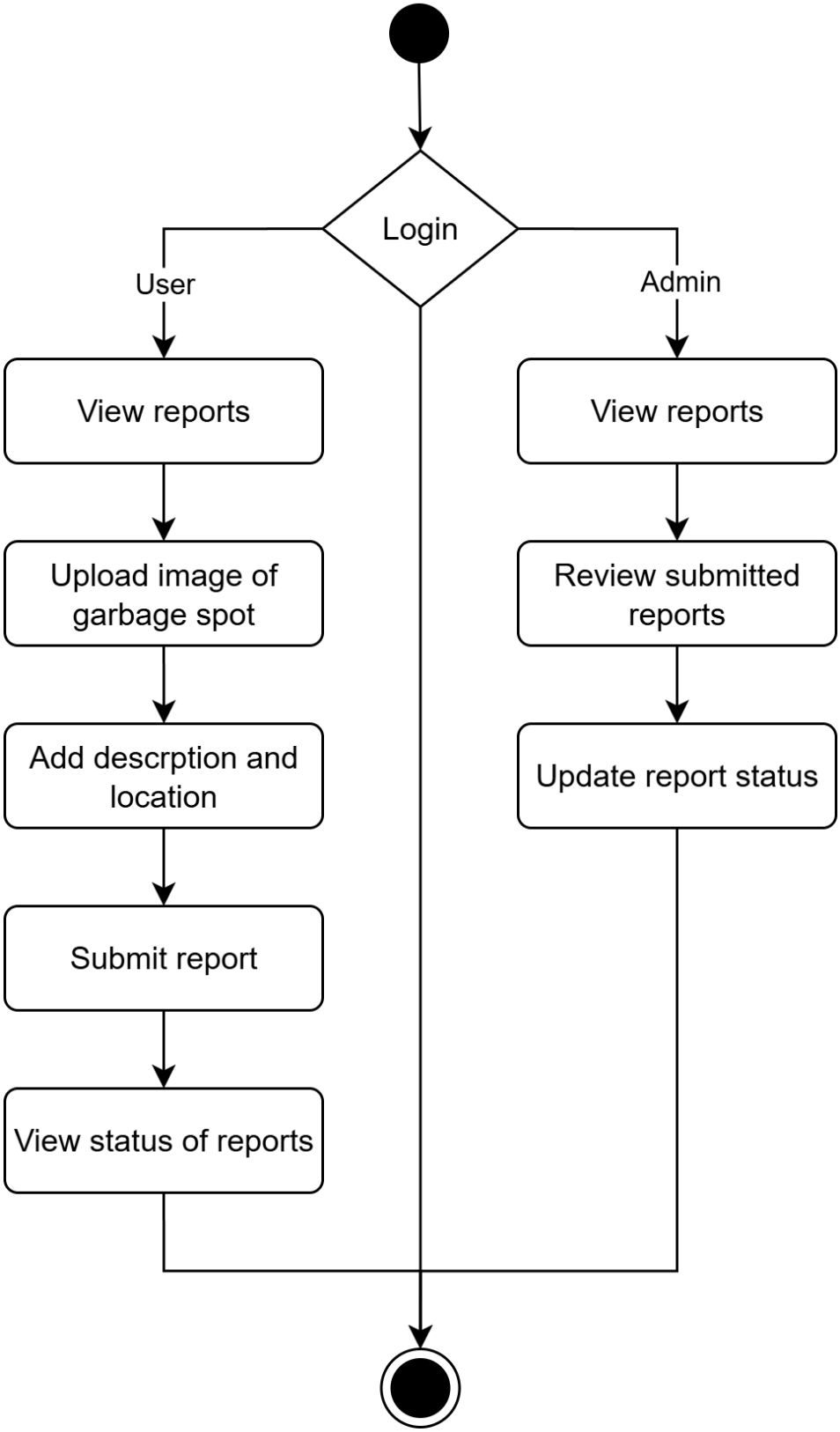


Figure 5.10 Activity Diagram

- **State Diagram**

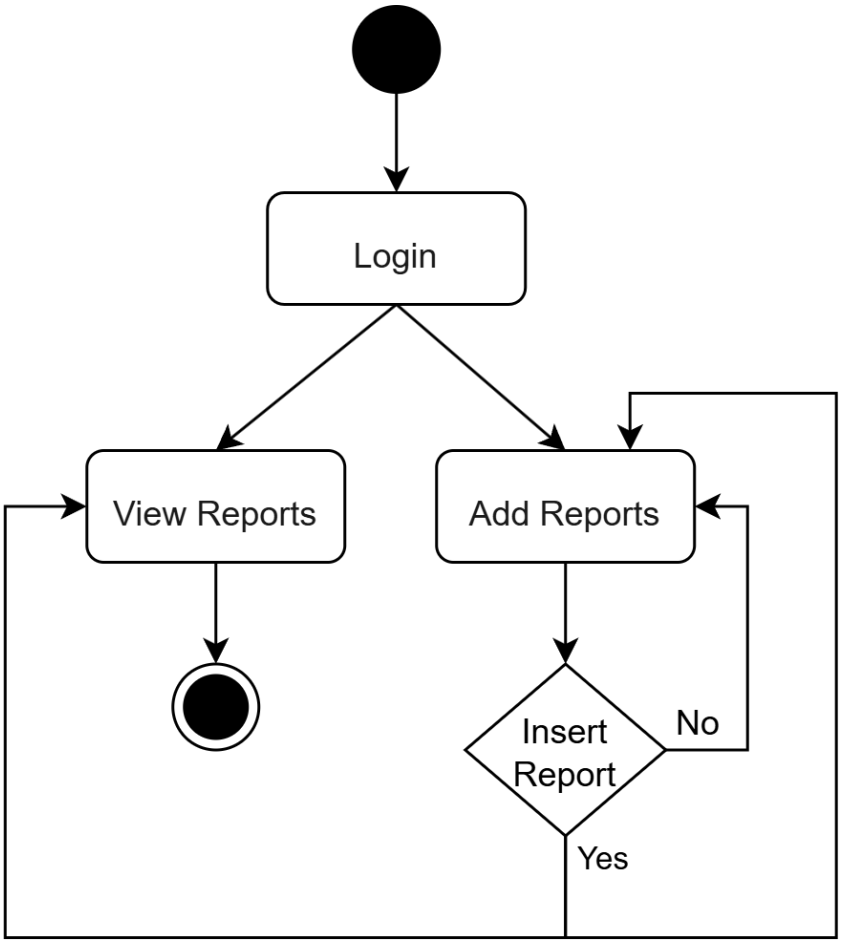


Figure 5.11 State Diagram

5.2 Database Design

- **Users**

Colum Name	Data Type	Constrain
_id	ObjectId	Auto-Generated, Primary Key
name	String	Required
email	String	Required, Unique
phoneno	String(10)	Required, Unique
address	String	Required
password	String(6)	Required
role	String	Default('user'), Required

Table 5.1 Database: Users

- **Reports**

Colum Name	Data Type	Constrain
_id	ObjectId	Auto-Generated, Primary Key
userId	ObjectId	Foreign Key (References Table: users, Colum:_id), Required
image	String	Required
locatation	Object	Required
description	String	Default(' ')
status	String	Default('pending'),
verifiedBy	ObjectId	Foreign Key (References Table: users, Colum:_id), Required
createdAt	Date	Auto-Add, Timestamps
updatedAt	Date	Auto-Add, Timestamps

Table 5.2 Database: Reports

- **Location Object**

```
{
        latitude: {
            type: String,
            require: true
        },
        longitude: {
            type: String,
            require: true
        },
        address: {
            type: String
        }
}
```

5.3 Screen Design

1. User

- **Register Screen & Login Screen**

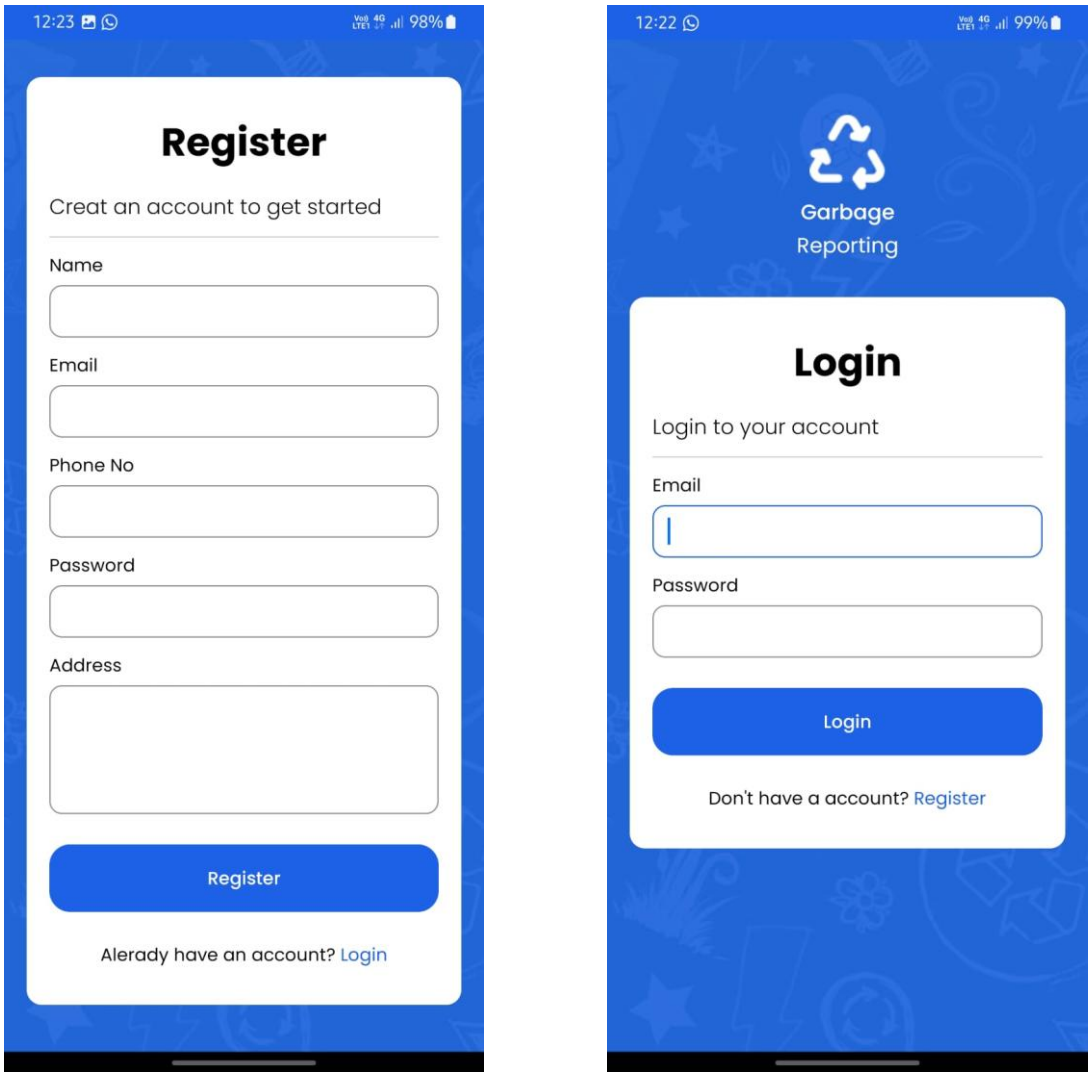


Figure 5.12 User Register & Login Screen

- **User Register Screen:** This screen allows new users to create an account by entering their personal details like name, email, phone number, address, and password.
- **User Login Screen:** This screen lets registered users log in securely using their email and password to access the application.

- **Splash Screen & Home Screen**

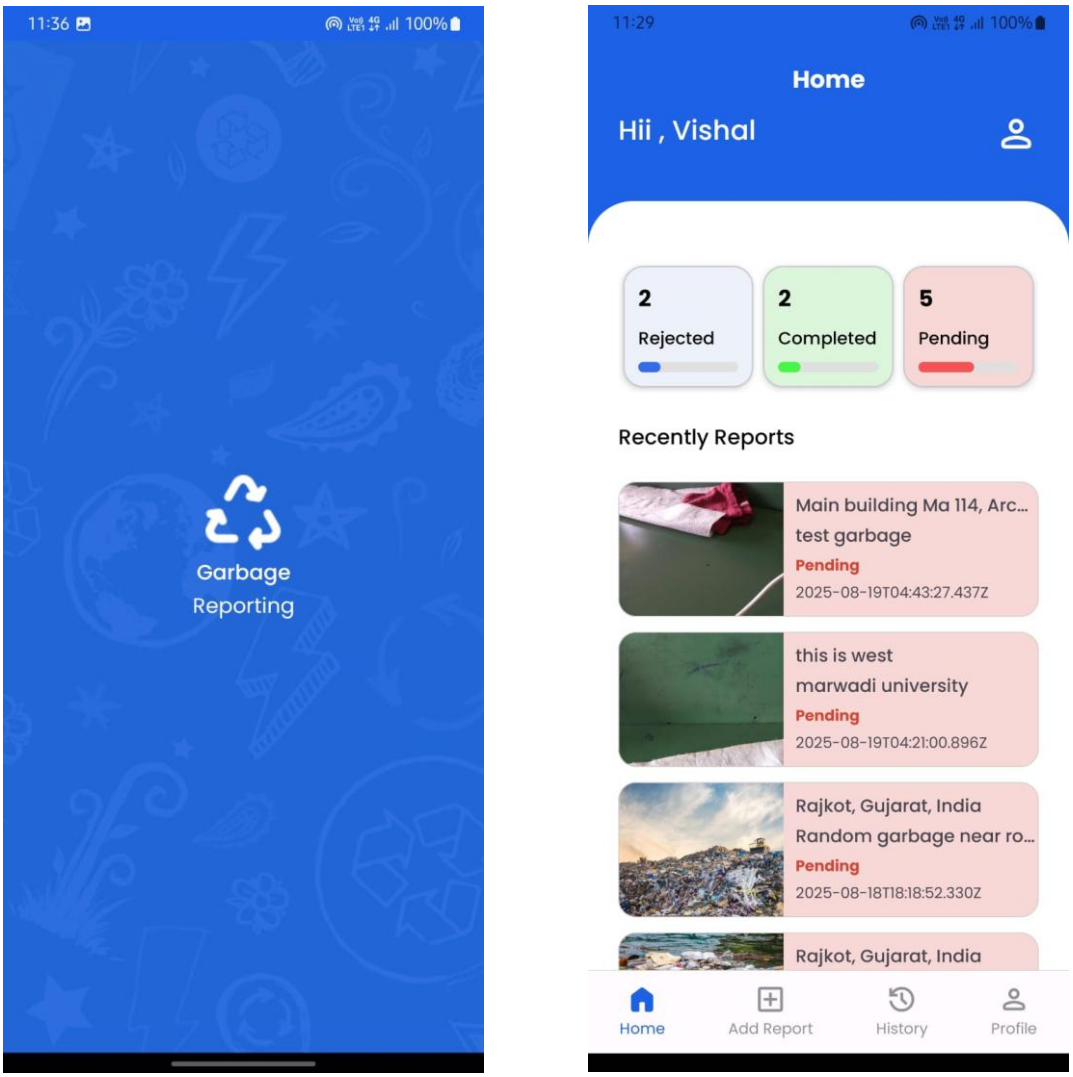


Figure 5.13 User Splash Screen & Home Screen

- **Splash Screen:** This screen is shown for a few seconds when the app starts, displaying the app logo and name.
- **Home Screen:** This screen is the main dashboard where users can quickly access options like adding a report, viewing history, or checking their profile.

- **Report History Screen & Single Report Details Screen**

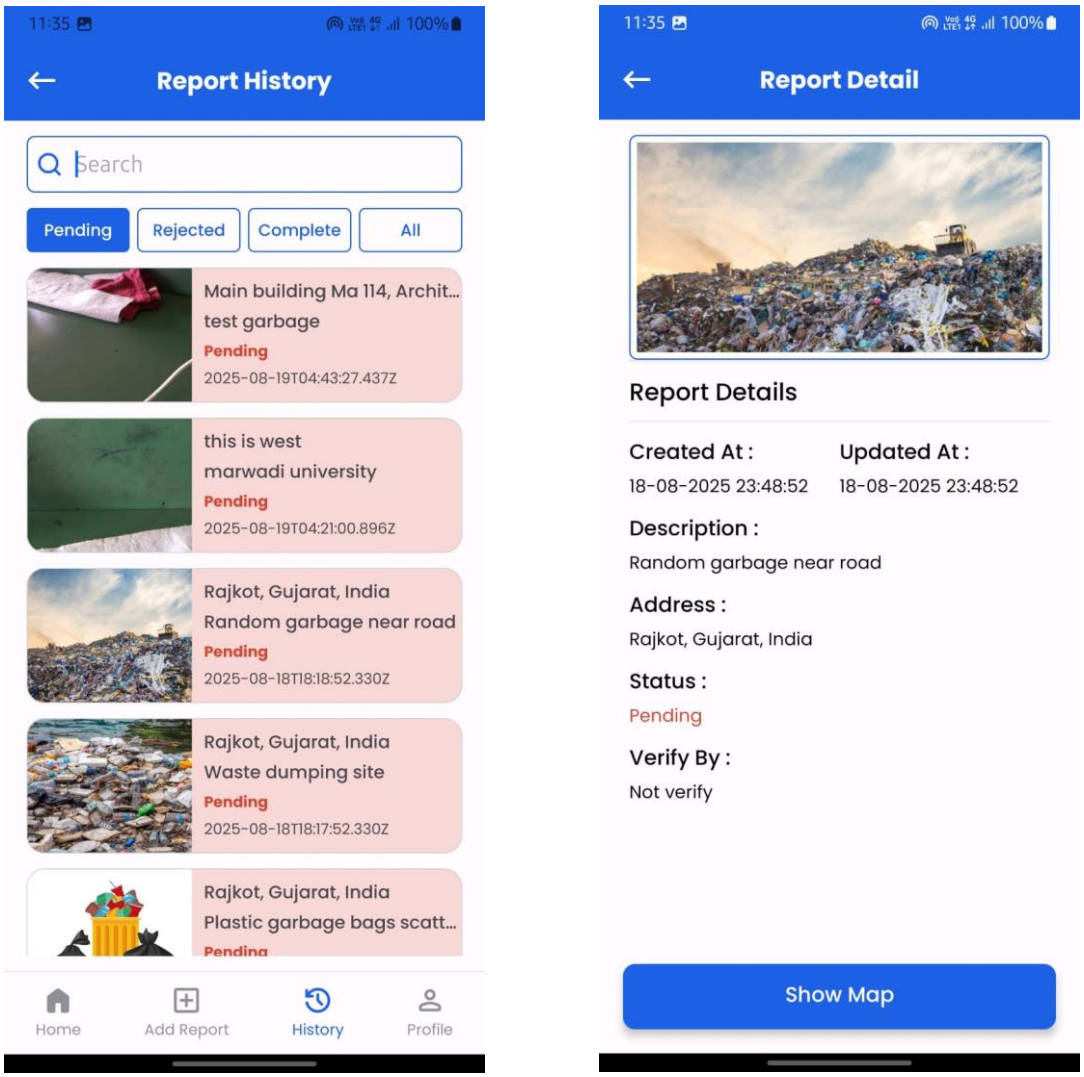


Figure 5.14 User Report History & Single Report Screen

- **Report History Screen:** This screen shows a list of all reports submitted by the user along with their current status.
- **Single Report Details Screen:** This screen displays full details of a selected report including image, location, description, and status.

- **Add Report Screen**

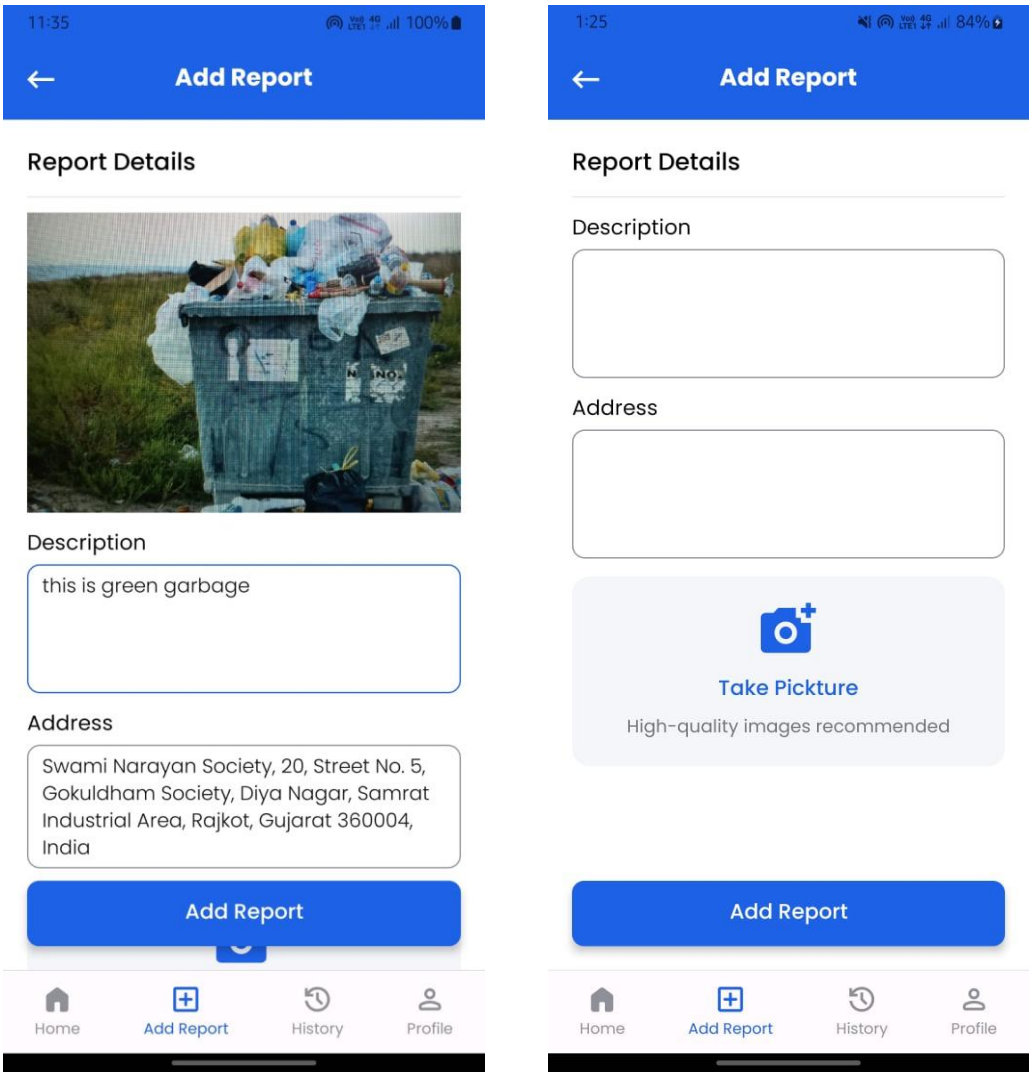


Figure 5.15 User Add Report Screen

- **Add Report Screen:** This screen allows users to submit a new report by uploading an image, adding location, and writing a description.

- **Profile Screen & Edit Profile Screen**

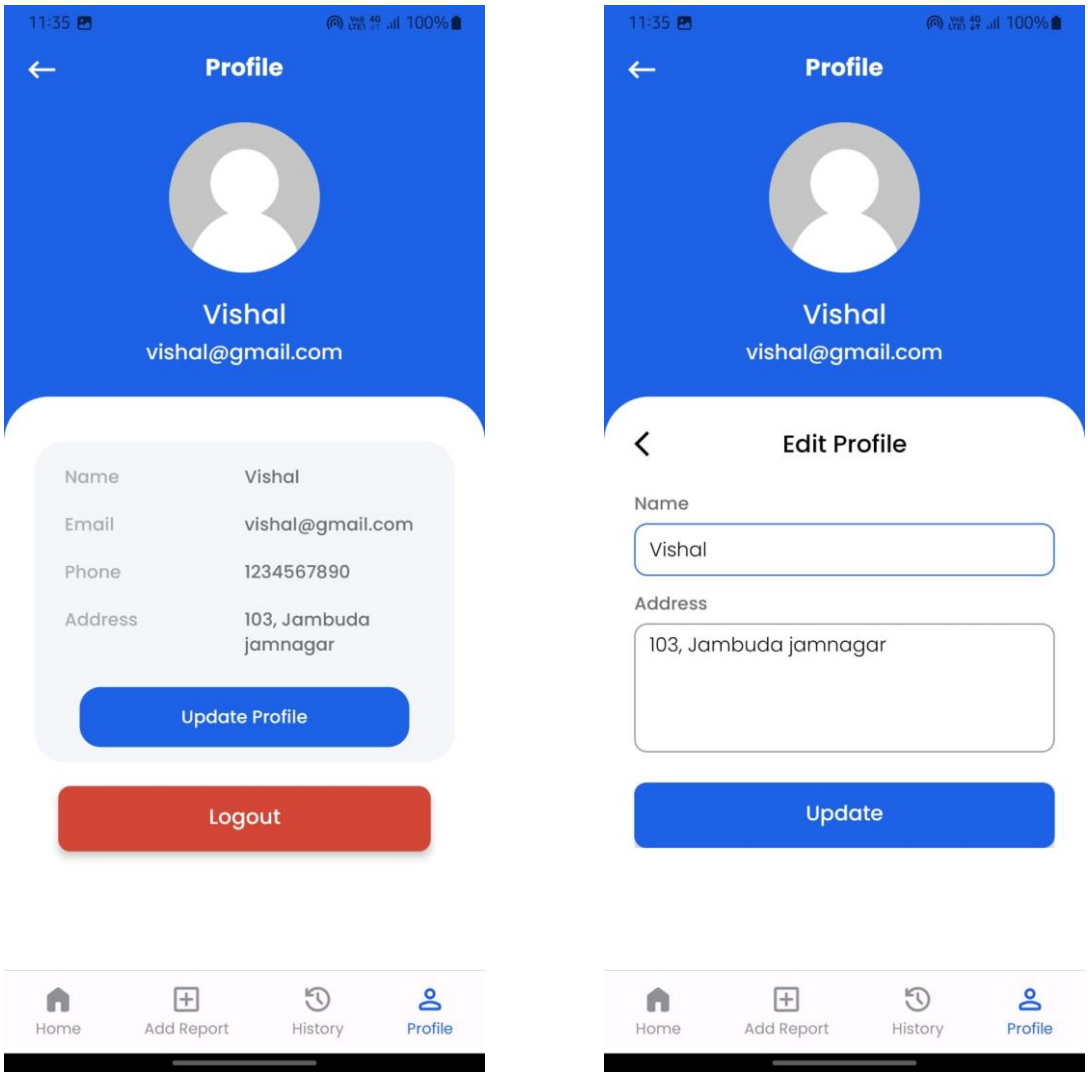


Figure 5.16 User Profile Screen

- **Profile Screen:** This screen displays the user's personal details such as name, email, phone number, and address.
- **Edit Profile Screen:** This screen lets users update or change their personal information.

2. Admin

- **Login Screen**

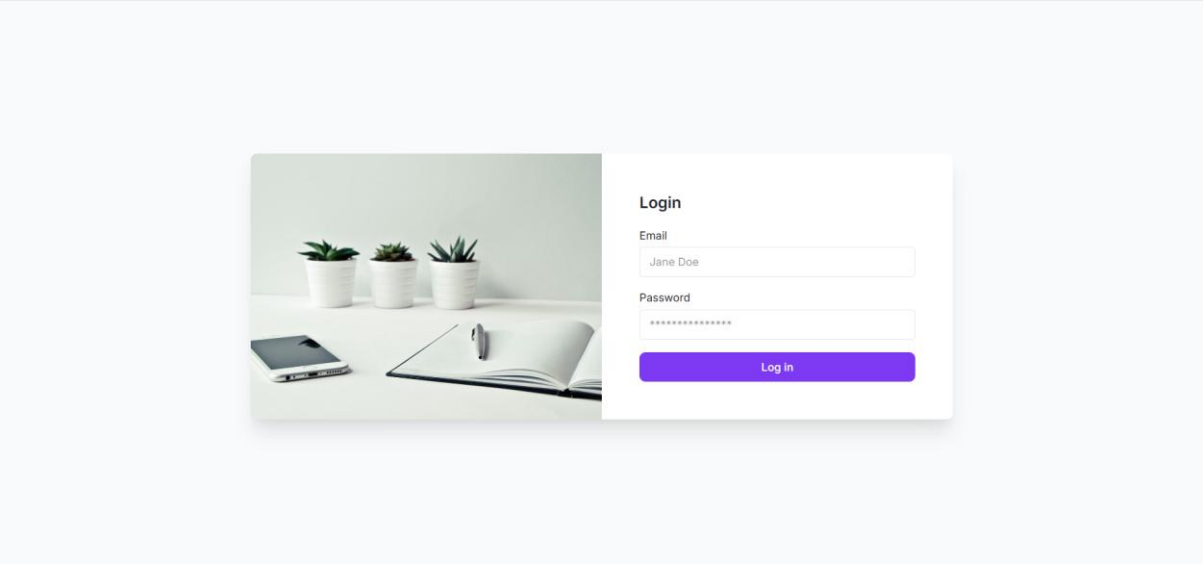


Figure 5.17 Admin Login Screen

- **Login Screen:** This screen allows the admin to log in securely using their email and password to access the panel.

- **Dashboard Screen**

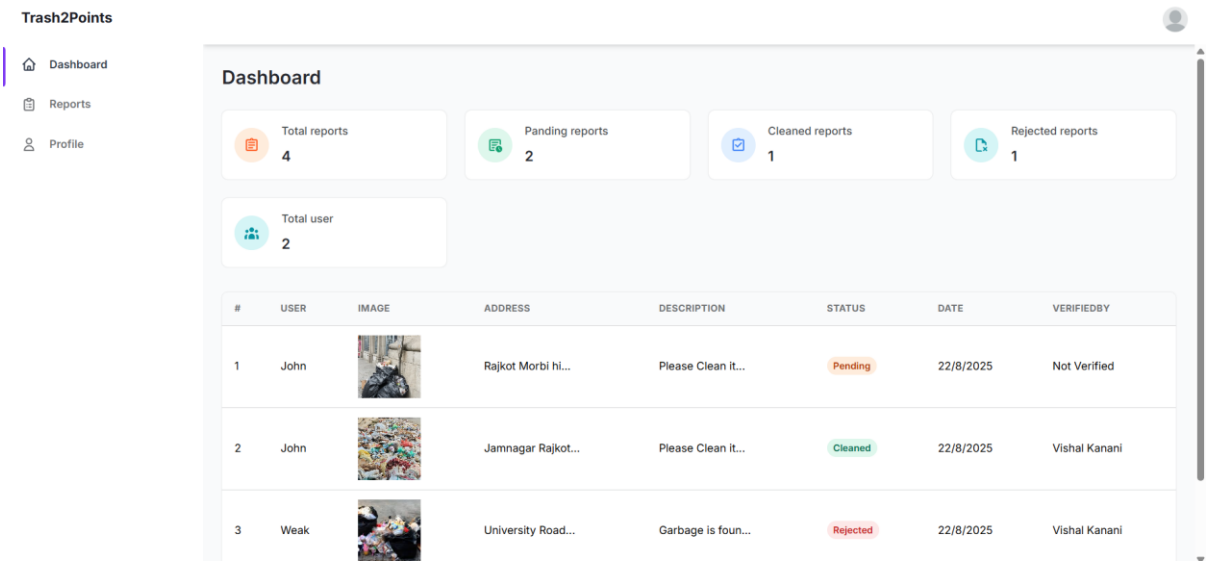


Figure 5.18 Admin Dashboard Screen

- **Dashboard Screen:** This is the main screen where the admin can see an overview of reports and manage actions.

- **Reports Screen**

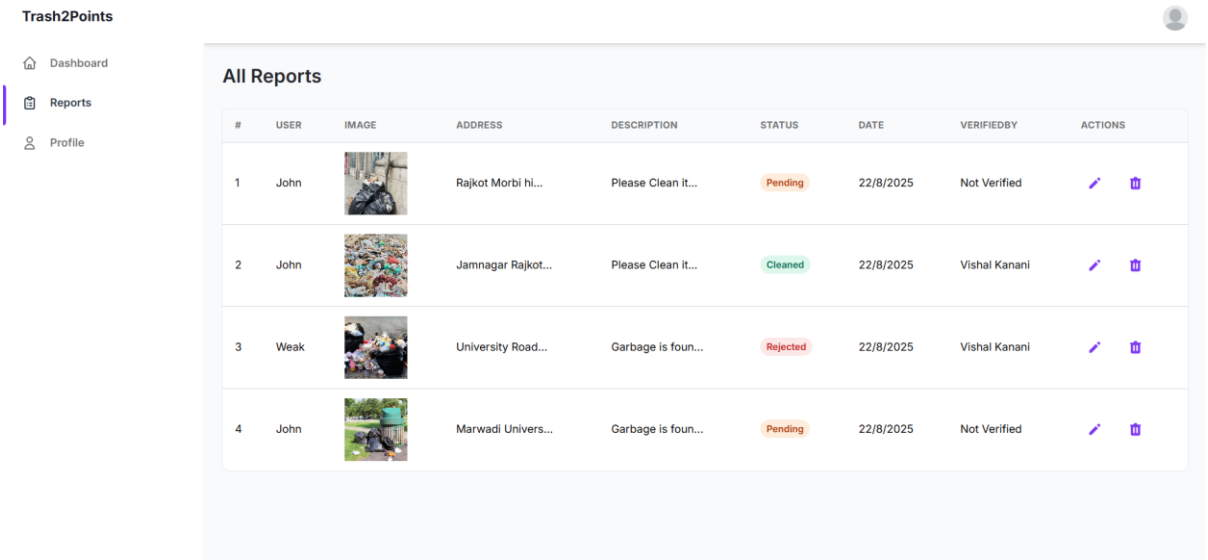


Figure 5.19 Admin Reports Screen

- **Reports Screen:** This screen shows the list of all reports submitted by users with details like image, description, and status.

- **Update Report Modal Screen**

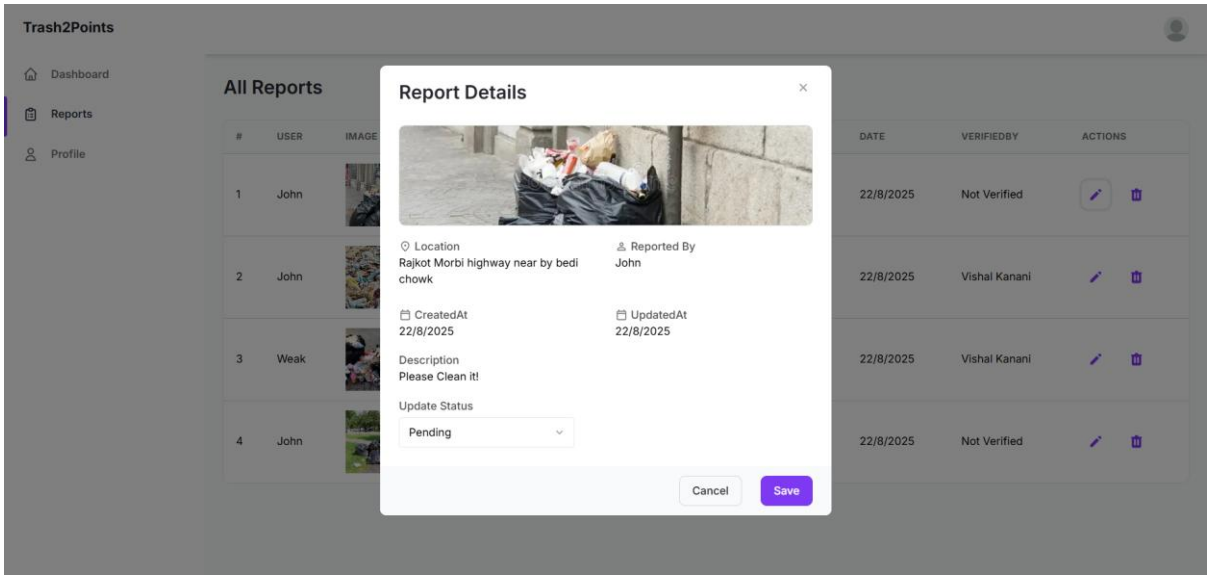
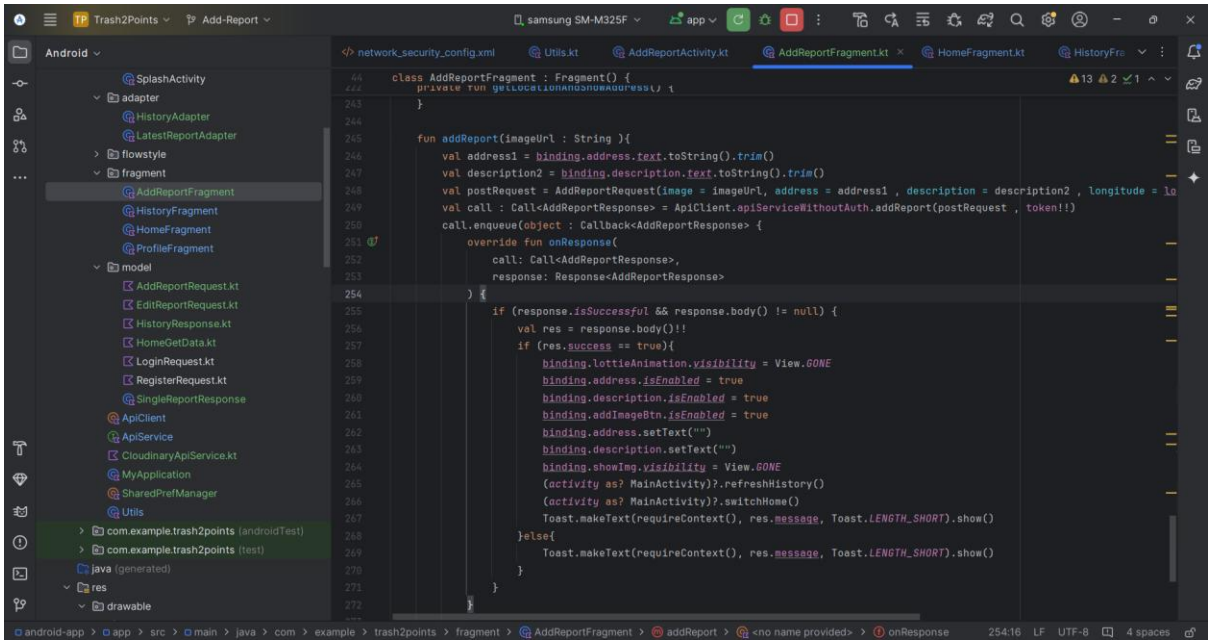


Figure 5.20 Admin Update Report Modal Screen

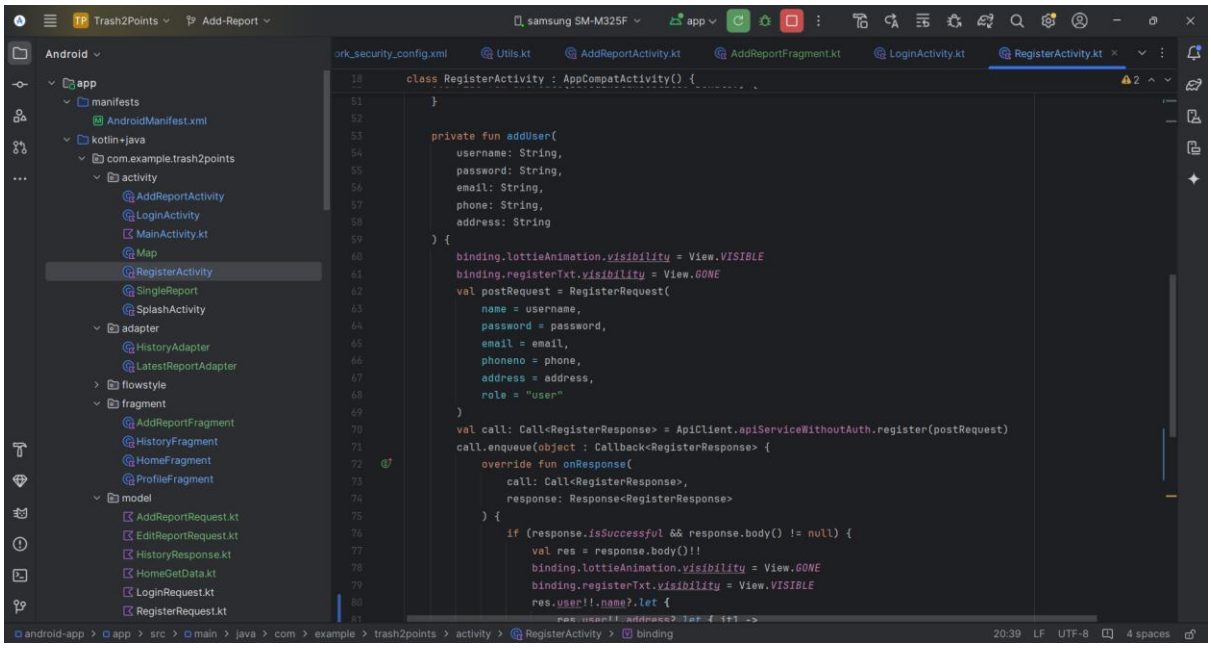
- **Update Report Modal Screen:** Shows full report details (image, location, user name, dates, description) and lets the admin update the status.

5.4 Code of the module

1. User – Android App

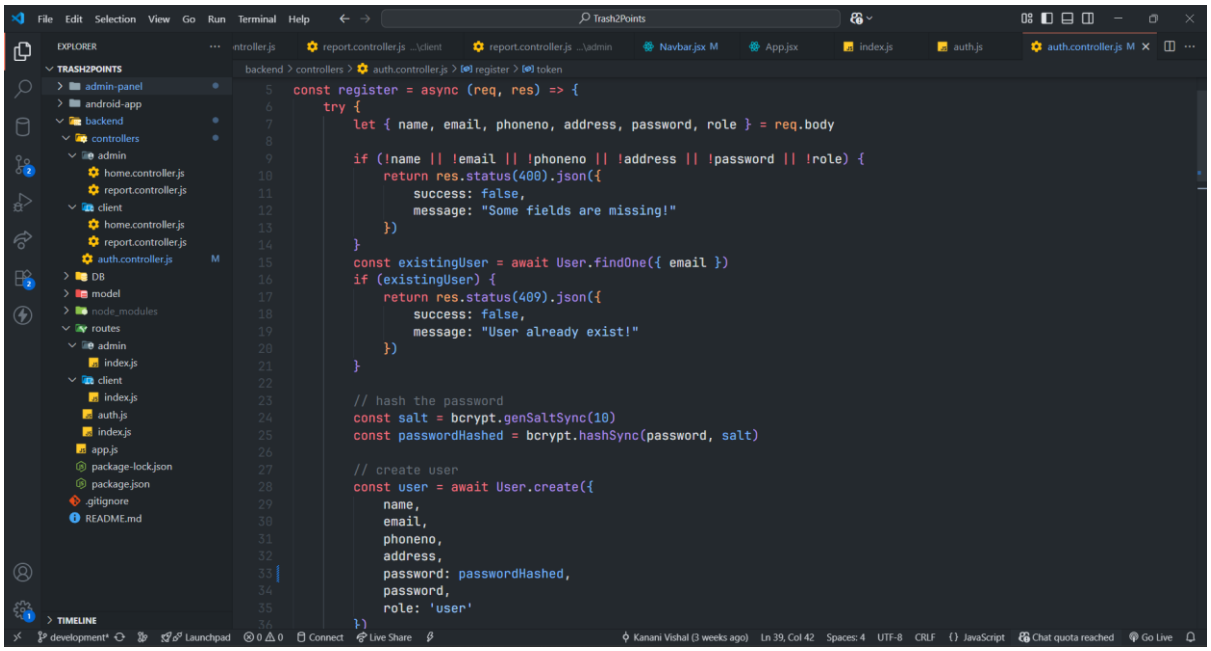


```
class AddReportFragment : Fragment() {
    }

    fun addReport(imageUrl : String ){
        val address1 = binding.address.text.toString().trim()
        val description2 = binding.description.text.toString().trim()
        val postRequest = AddReportRequest(image = imageUrl, address = address1 , description = description2 , longitude = lo
        val call : Call<AddReportResponse> = ApiClient.apiServiceWithoutAuth.addReport(postRequest , token!!)
        call.enqueue(object : Callback<AddReportResponse> {
            override fun onResponse(
                call: Call<AddReportResponse>,
                response: Response<AddReportResponse>
            ) {
                if (response.isSuccessful && response.body() != null) {
                    val res = response.body()!!
                    if (res.success == true){
                        binding.lottieAnimation.visibility = View.GONE
                        binding.address.isEnabled = true
                        binding.description.isEnabled = true
                        binding.addImageBtn.isEnabled = true
                        binding.address.setText("")
                        binding.description.setText("")
                        binding.showImg.visibility = View.GONE
                        (activity as? MainActivity)?.refreshHistory()
                        (activity as? MainActivity)?.switchHome()
                        Toast.makeText(requireContext(), res.message, Toast.LENGTH_SHORT).show()
                    }else{
                        Toast.makeText(requireContext(), res.message, Toast.LENGTH_SHORT).show()
                    }
                }
            }
```

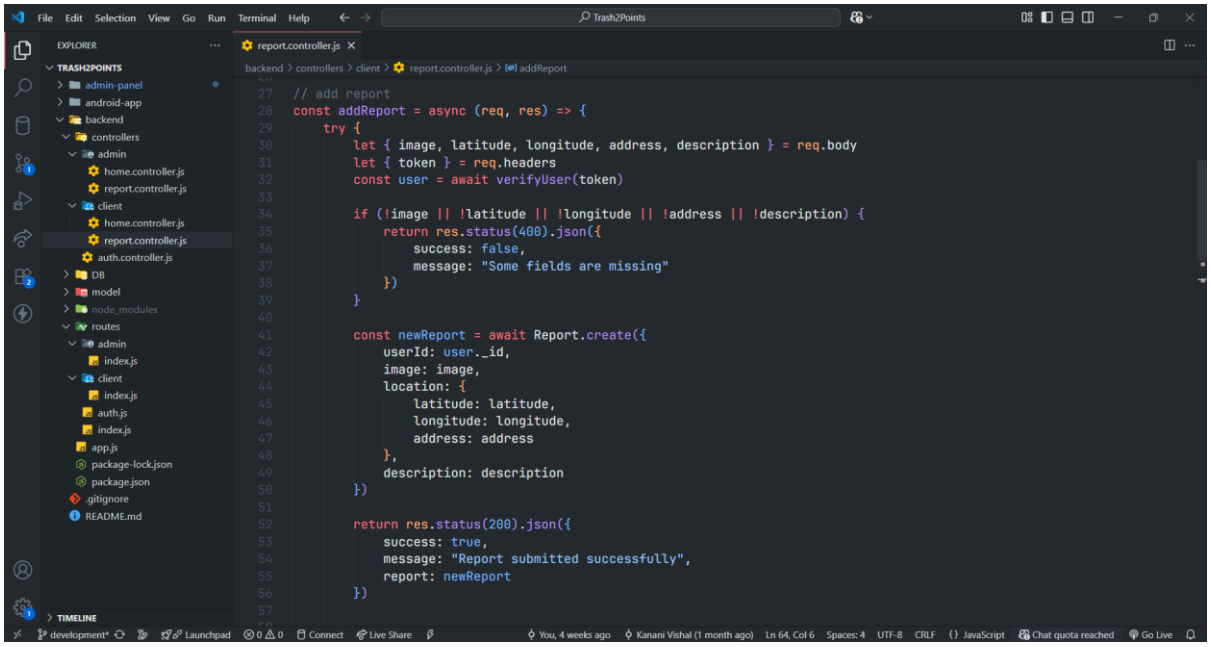


```
class RegisterActivity : AppCompatActivity() {
    }

    private fun addUser(
        username: String,
        password: String,
        email: String,
        phone: String,
        address: String
    ) {
        binding.lottieAnimation.visibility = View.VISIBLE
        binding.registerTxt.visibility = View.GONE
        val postRequest = RegisterRequest(
            name = username,
            password = password,
            email = email,
            phoneno = phone,
            address = address,
            role = "user"
        )
        val call: Call<RegisterResponse> = ApiClient.apiServiceWithoutAuth.register(postRequest)
        call.enqueue(object : Callback<RegisterResponse> {
            override fun onResponse(
                call: Call<RegisterResponse>,
                response: Response<RegisterResponse>
            ) {
                if (response.isSuccessful && response.body() != null) {
                    val res = response.body()!!
                    binding.lottieAnimation.visibility = View.GONE
                    binding.registerTxt.visibility = View.VISIBLE
                    res.user!!.name?.let {
```

2. Backend – Api

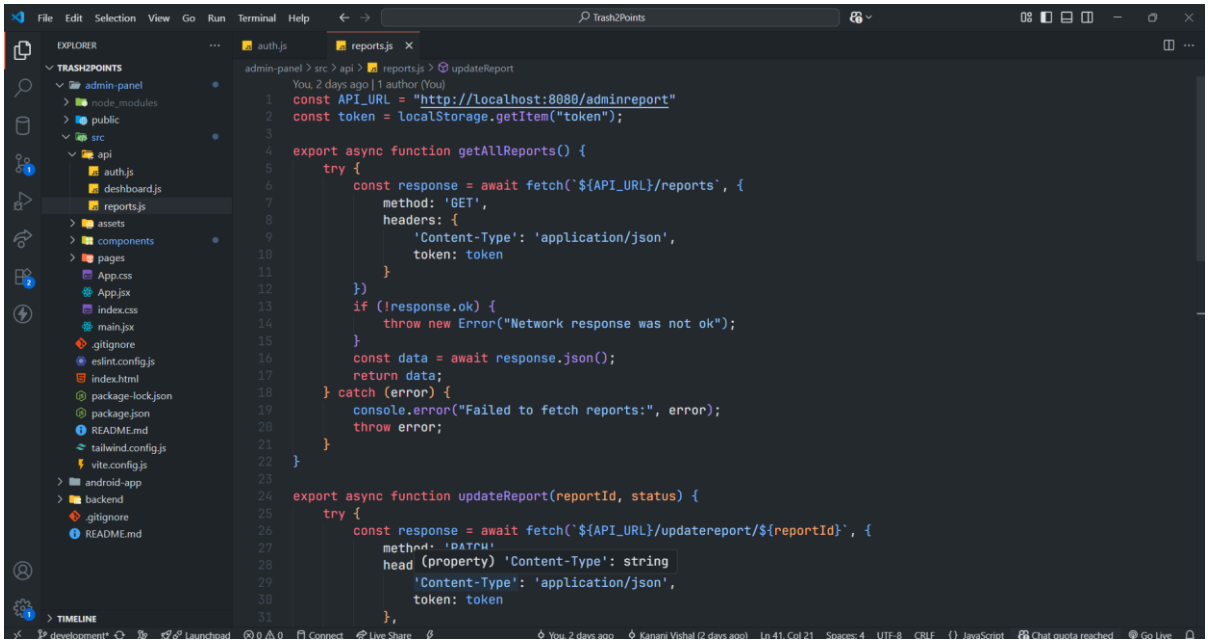


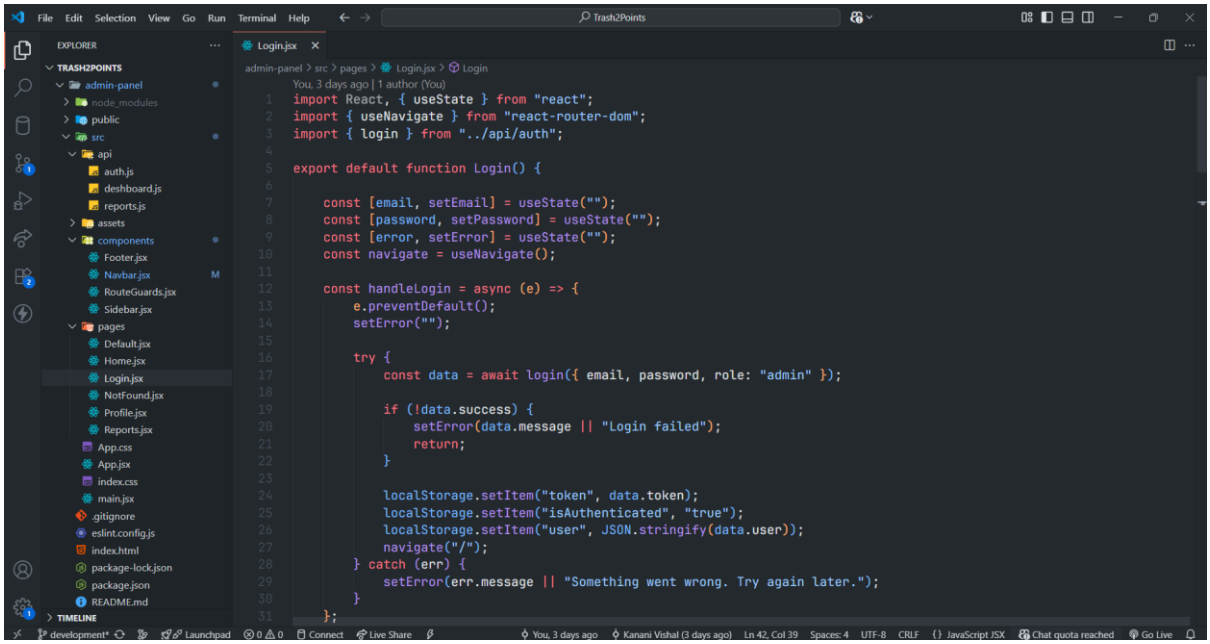
```
const register = async (req, res) => {
    try {
        let { name, email, phoneno, address, password, role } = req.body

        if (!name || !email || !phoneno || !address || !password || !role) {
            return res.status(400).json({
                success: false,
                message: "Some fields are missing!"
            })
        }
        const existingUser = await User.findOne({ email })
        if (existingUser) {
            return res.status(409).json({
                success: false,
                message: "User already exist!"
            })
        }

        // hash the password
        const salt = bcrypt.genSaltSync(10)
        const passwordHashed = bcrypt.hashSync(password, salt)

        // create user
        const user = await User.create({
            name,
            email,
            phoneno,
            address,
            password: passwordHashed,
            password,
            role: 'user'
        })
```



```
// add report
const addReport = async (req, res) => {
    try {
        let { image, latitude, longitude, address, description } = req.body
        let { token } = req.headers
        const user = await verifyUser(token)

        if (!image || !latitude || !longitude || !address || !description) {
            return res.status(400).json({
                success: false,
                message: "Some fields are missing"
            })
        }

        const newReport = await Report.create({
            userId: user._id,
            image: image,
            location: {
                latitude: latitude,
                longitude: longitude,
                address: address
            },
            description: description
        })

        return res.status(200).json({
            success: true,
            message: "Report submitted successfully",
            report: newReport
        })
```

3. Admin Panel





CHAPTER 6

SYSTEM TESTING

1. Test Case of Register Process

Test Case ID	Test Case Description	Test Steps	Input Data	Expected Result	Actual Result	Pass /Fail
TC001	Valid Register	1. Open register page 2. Fill details 3. Click register	Name: Vishal Email: Vishal@gmail.com Phone: 9876543210 Password: vishal@123	Display Dashboard	Display Dashboard	Pass
TC002	Invalid Email	1. Open register page 2. Fill details 3. Click register	Name: Vishal Email: Vishalgmail.com Phone: 9876543210 Password: vishal@123	Display error	Error displayed	Pass
TC003	Invalid Password	1. Open register page 2. Fill details 3. Click register	Name: Vishal Email: Vishal@gmail.com Phone: 9876543210 Password: vishal	Display error	Error displayed	Pass
TC004	Empty Fields	1. Open register page 2. Fill details 3. Click register	Name: Email: Vishal@gmail.com Phone: 9876543210 Password: vishal@123	Display error	Error displayed	Pass
TC005	Duplicate Email	1. Open register page 2. Fill details 3. Click register	Name: Vishal Email: Vishal@gmail.com Phone: 9876543210 Password: vishal@123	Display error	Error displayed	Pass
TC006	Email in emoji	1. Open register page 2. Fill details 3. Click register	Name: Vishal Email: Vishal😊@gmail.com Phone: 9876543210 Password: vishal@123	Display error	Error not shown	Faill

Table 6.1 Test Case: Register Process

2. Test Case of Login Process

Test Case ID	Test Case Description	Test Steps	Input Data	Expected Result	Actual Result	Pass /Fail
TC001	Valid Login	1. Open login page 2. Fill details 3. Click login	Email: Vishal@gmail.com Password: vishal@123	Display Dashboard	Display Dashboard	Pass
TC002	Invalid Email	1. Open login page 2. Fill details 3. Click login	Email: Vishalgmail.com Password: vishal@123	Display error	Error displayed	Pass
TC003	Invalid Password	1. Open login page 2. Fill details 3. Click login	Email: Vishal@gmail.com Password: vishal@546	Display error	Error displayed	Pass
TC004	Empty Fields	1. Open login page 2. Fill details 3. Click login	Email: Vishal@gmail.com Password:	Display error	Error displayed	Pass
TC005	Password Length	1. Open login page 2. Fill details 3. Click login	Email: Vishal@gmail.com Password: Abc@1234567890Abc	Display error	Error not showing	Fail

Table 6.2 Test Case: Login Process

3. Test Case of Report Submission Process(User)

Test Case ID	Test Case Description	Test Steps	Input Data	Expected Result	Actual Result	Pass /Fail
TC001	Valid Report Submission	1. Login as user 2. Go to report form 3. Upload image, add location, description 4. Click submit	Image: photo.jpg Location: Rajkot Desc: Garbage near street	Report saved, status "Pending"	Report created	Pass
TC002	Missing Image	1. Fill all details but leave image blank 2. Click submit	No image	Error "Image required"	Error displayed	Pass
TC003	Invalid Location	1. Fill form with invalid coordinates 2. Click submit	Latitude: ---	Error message "Invalid location"	Error displayed	Pass
TC004	Missing Description	1. Fill all details but leave description blank 2. Click submit	No Description	Error "All Fields required"	Error not showing	Faill

Table 6.3 Test Case: Report Submission Process

4. Test Case of Admin Report Action Process

Test Case ID	Test Case Description	Test Steps	Input Data	Expected Result	Actual Result	Pass /Fail
TC001	Approve Report	1. Login as admin 2. Open pending report 3. Update status	Report ID: 123	Report status updated	Status updated	Pass
TC002	Reject Report	1. Login as admin 2. Reject report 3. Save	Report ID: 123	Status changed to "Rejected"	Status updated	Pass
TC003	Delete Report	3. Login as admin 4. Select report 5. Click delete	Report ID: 123	Report removed from DB	Deleted	Pass

Table 6.4 Test Case: Admin Report Action Process

CHAPTER 7
CONCLUSION

Trash2Points is a small step toward keeping our surroundings clean. This project makes it easy for people to report garbage or dirty places using their mobile phones. They just need to take a picture, add the location, and send the report.

The admin panel helps the authorities see all the reports in one place. They can check the reports, update the status, and take quick action to clean the area. All the information is safely stored in MongoDB Atlas so that both users and admins can access it anytime.

This project helps people and authorities work together for a cleaner city. In the future, more features like reward points and notifications can be added to make the app more useful and encourage more people to report issues.

CHAPTER 8

LEARNING DURING PROJECT WORK

1. **Android Development (Kotlin):** Gained experience in building mobile applications using Kotlin, designing user-friendly interfaces, and managing app navigation and data handling.
2. **Backend Development (Node.js & Express):** Learned how to create RESTful APIs, handle requests and responses, use middleware, and manage errors effectively.
3. **Database Management (MongoDB Atlas):** Designed schemas for users and reports, handled data relationships, and performed CRUD operations on a NoSQL cloud database.
4. **React.js (Admin Panel):** Improved skills in creating dynamic web interfaces for the admin panel using React components, state management, and API integration.
5. **User Authentication:** Implemented secure login and registration processes, learned about password validation and JWT-based authentication for safe access.
6. **Image Handling:** Understood how to handle file uploads and store image paths in the database for efficient storage and retrieval.
7. **Version Control (GitHub):** Learned to manage code with multiple team members using branches, commits, pull requests, and merge conflict resolution.
8. **Team Collaboration:** Developed skills in dividing tasks, working together efficiently, and communicating progress within a three-member team.

8.1 Future Enhancement

1. **Reward System:** Giving users points or badges for reporting garbage spots to encourage more participation.
2. **Push Notifications:** Sending alerts to users when their report status is updated or when nearby areas are cleaned.
3. **Multiple Language Support:** Allowing people to use the app in their local language for better accessibility.
4. **Image Recognition:** Using AI to automatically detect garbage in uploaded images for faster verification.
5. **Admin Analytics Dashboard:** Give admins a report on how many garbage spots were reported, cleaned, or still pending.
6. **Offline Reporting:** Allow users to create reports even without the internet, which will be uploaded once they are online.

CHAPTER 9

BIBLIOGRAPHY

9.1 Online References

[1] Android Developers, “Official documentation for Android and Kotlin development.” [Online]. Available: https://developer.android.com. [Accessed: Aug. 3, 2025].

[2] Node.js Foundation, “Node.js documentation for backend development concepts.” [Online]. Available: https://nodejs.org. [Accessed: Aug. 6, 2025].

[3] Express.js, “Express.js documentation for building RESTful APIs.” [Online]. Available: https://expressjs.com. [Accessed: Aug. 10, 2025].

[4] MongoDB Inc., “MongoDB Atlas documentation for database setup and management.” [Online]. Available: https://www.mongodb.com/docs/atlas. [Accessed: Aug. 16, 2025].

[5] React, “React documentation for creating the admin panel.” [Online]. Available: https://react.dev. [Accessed: Aug. 20, 2025].

[6] JWT, “JSON Web Token reference for implementing user authentication.” [Online]. Available: https://jwt.io. [Accessed: Aug. 26, 2025].

9.2 Offline References

[1] J. Horton, *Android Programming for Beginners*, 2nd ed. Birmingham, U.K.: Packt Publishing, 2018.

[2] A. Mead, *Learning Node.js Development*. Birmingham, U.K.: Packt Publishing, 2019.

[3] E. Hahn, *Express in Action*. Shelter Island, NY, USA: Manning Publications, 2016.

[4] Z. Gordon, *React Explained*. Orlando, FL, USA: Go Make Things, 2019.