

Car recovR

**PROJECT SUPERVISOR**

Ms. Syeda Rubab Jaffar

**PROJECT CO-SUPERVISOR**

Mr. Shahbaz Siddiqui

**PROJECT TEAM**

Uzair Khan K19-1477

Muhammad Eahtesham K19-0298

Muhammad Hassaan Ahsan Qureshi K19-0272

Submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

FAST SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES KARACHI CAMPUS

May 2023

| Project Supervisor | Mr. Supervisor Name |
| --- | --- |
| Project Team | Uzair Khan K19-1477  Muhammad Eahtesham K19-0298  Muhammad Hassaan Ahsan Qureshi K19-0272 |
| Submission Date | May 18, 2023 |

Ms. Syeda Rubab Jaffar

**Supervisor**

Mr. Shahbaz Siddiqui

**Co-Supervisor**

Dr. Zulfiqar Ali Memon \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Head of Department**

FAST SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES KARACHI CAMPUS

**Acknowledgement**

We begin this document by thanking **Allah** with our praise towards the energy, mental capacity, and the positive inclination on working towards the completion of this project. Surely, nothing good can happen if **Allah** does not will it, and no progress can be made if **Allah** does not give us a sense of concern and progress. All glory is to Him, for Him.

Secondly, this report is dedicated to the spirits of passion, ambition, and good will that keep motivating the team to focus on delivering quality work, giving us a sense of self-awareness of where things can and might go wrong, and how we can improve that. We thank the Software Engineers and our Seniors for giving us a fire to want to build more and more interesting products that have some meaning instead of projects that just pass by.

We like to thank the engineers working in the **#OpenSource** community. All of our libraries and technologies come from the **Open Source** world. Had we sat down to implement any of the things we needed to develop this project, we would be scratching ourselves over getting a hackish version of a library working.

We would like to express our sincere gratitude to our supervisor, **Ms. Syeda Rubab Jaffar** and co-supervisor **Mr. Shahbaz Siddiqui**, for their guidance and support throughout the course of this project. Their feedback, insights, and expertise were invaluable in shaping our ideas and helping us overcome the challenges we faced.

We would also like to thank the staff at **NUCES FAST Karachi** for providing us with access to resources and equipment that were essential to the success of this project.

We would like to acknowledge the value inputs of respected **Jury** members **Dr. Farooque Hassan Khumber** and **Mr. Abuzar** for their suggestions and feedback throughout the development and evaluation phase. Their input and assistance played a key role in helping us achieve our project objectives.

Finally, we would like to thank each other - **Muhammad Eahtesham**, **Muhammad Hassaan Ahsan Qureshi**, and **Uzair Khan** - for our individual and collective contributions to this project. We worked collaboratively and overcame obstacles together to deliver this project successfully.

We are grateful for the unwavering support and encouragement of our **family** and **friends**, who provided us with the motivation to see this project through to completion.

Thank you all for your contributions to this project.

**Abstract**

This project involves the development of a web-based car recovery portal that is designed to facilitate the process of recovering stolen cars. The portal consists of two roles: an admin module and a user. The admin manages all the user details and their complaints. Users register their cars to the database. The admin is responsible for reviewing and approving the request before the data is stored in the database. In the event of a car theft, the user can launch a complaint on the portal, which provides the admin module with relevant information that can be used to locate the stolen car through surveillance and number plate recognition using cameras installed in the area. If the car is found, the user is notified about its location via the portal. The application is built with a focus on data security and confidentiality, with user authentication and authorization implemented to ensure that data can only be accessed by authorized users.

# **Table of Contents**

**Page No.**

[**Table of Contents 4**](#)

[**1. Introduction 8**](#_30j0zll)

[1.2 Project Overview 8](#_wgchktm6eik)

[**2. Related Work 9**](#_1fob9te)

[2.1 Project Scope 9](#_f1224vks8234)

[2.2 Not In Scope 10](#_7e5ptz6bkc0l)

[2.3 Project Stakeholders 11](#_17waj1a3qcb0)

[**3. Requirements 12**](#_3znysh7)

[3.1 Functional Requirements 12](#_pd3kqdo6lx9p)

[3.2 Use Cases 13](#_5pavz0ua183v)

[3.2.1 Car Registration 15](#_c72dtkicormh)

[3.2.2 Complaint Registration 17](#_oiapfmtum64t)

[3.2.3 Car Recovery 20](#_5x5oxfsdi4tw)

[3.3 Non-functional Requirements 23](#_2bn6wsx)

[3.3.1 Performance Requirements 23](#_qsh70q)

[3.3.2 Safety Requirements 23](#_3as4poj)

[3.3.3 Security Requirements 23](#_1pxezwc)

[3.3.4 User Documentation 24](#_49x2ik5)

[**4. Design Considerations 25**](#_2et92p0)

[4.1 Assumptions and Dependencies 26](#_pnsk145owku6)

[4.1.1 System Assumptions 26](#_ac4hd3u8wrx2)

[4.1.2 Project Delivery Details 26](#_kuekoerl7wcu)

[4.2 Risks & Volatile Areas 27](#_ju9bfjhn72dn)

[**5. Design Strategy 28**](#_gbvpfyqy7ye)

[**6. Design Details 29**](#_b8c8z8rcy8hf)

[6.1 Process Model 29](#_tsh85ikypr8x)

[6.2 Sequence Diagram 30](#_t6fd7kzeb3a6)

[6.2.1 Car Registration 30](#_fm0ylgprfry4)

[6.2.2 Complaint Registration 31](#_47xzxwia3099)

[6.2.3 Surveillance 32](#_tqh6qk1j3gdy)

[6.3 State Diagram 32](#_z2i0tkg5mmsm)

[6.3.1 Car Registration 32](#_67tsb66cmgf8)

[6.3.2 Complaint Registration 33](#_i4fn7pylsnhr)

[6.3.3 Surveillance 34](#_7axm018fdgbf)

[6.4 Database Design 35](#_m3b8486ghuwl)

[6.4.1 ER Model 35](#_r8fuzz5iqgcj)

[6.4.2 Entity Relation Diagram 35](#_cgqjem90920w)

[6.4.3 Data Dictionary 36](#_m4gbhittdeem)

[6.4.3.1 Admin 36](#_807s7rpkc7mb)

[6.4.3.2 User 37](#_s3kkwgboshbh)

[6.4.3.3 Car 37](#_jeol3od9wwlz)

[6.4.3.4 Location 38](#_39cqquyyyvuj)

[6.4.3.5 Complaint 39](#_g8lutvgvzqz8)

[6.5 System Architecture 40](#_wkidgw422xh0)

[6.5.1 System Level Architecture 40](#_q1bx7sgrvxa6)

[6.5.2 Software Level Architecture 41](#_eibzsujinqfv)

[**7. External Interfaces 41**](#_wi222j2zguvl)

[7.1 Hardware Interfaces 42](#_tx9bqtlnrcsk)

[7.2 Software Interfaces 43](#_81zy4zwx8uvl)

[7.3 Communications Interfaces 44](#_oq6lv4xpph3r)

[**8. Implementation 45**](#_tyjcwt)

[8.1 Frontend 45](#_nk5pbpoqdk4b)

[8.1.1 ReactJS 45](#_bvg6i0u3h0v9)

[8.1.2 Material UI 45](#_uwlt35fsnm1b)

[8.2 Backend 46](#_re88el8m80zj)

[8.2.1 FastAPI 46](#_xu0cmkio8wqa)

[8.2.2 Websocket 46](#_e23lqqm5ac3p)

[8.2.3 JWT Token 47](#_la8dqvlq2dzi)

[8.2.4 Email Validator 48](#_5er0ip44edkf)

[We have used email validator to validate user input when creating or updating a user account. 48](#_7zy50qlks7d)

[8.3 Databases 48](#_95200awx97ij)

[8.3.1 MongoDB 48](#_p3rgps7sp1r2)

[**9. Testing & Evaluation 49**](#_x0p4zxnfy0ng)

[9.1 Testing 49](#_z7wq7ez13m81)

[9.1.1 Purpose of Testing 49](#_n3e28h4bgxdm)

[9.1.2 Testing Approaches 49](#_ojat76oghh9n)

[9.2 Test Cases 49](#_oj95oofnpnaq)

[**10. User Interfaces 52**](#_adpp65g3jgqm)

[**USER 55**](#_anduli36c896)

[**User dashboard 55**](#_e3avsrdrm9u6)

[**Conclusion 57**](#_3dy6vkm)

[**References 57**](#_1t3h5sf)

**List of Figures**

1. Functional Hierarchy
2. Use case diagram 1
3. Use case diagram 2
4. Use case diagram 3
5. Car Registration
6. Complaint Registration
7. Car recovery
8. Sequence Diagram 1
9. Sequence Diagram 2
10. Sequence Diagram 3
11. Sequence Diagram 4
12. State Diagram 1
13. State Diagram 2
14. State Diagram 3
15. ER Model
16. ER Diagram
17. SystemLevel Architecture
18. Software Level Architecture
19. External Interfaces
20. UI Interfaces

# **Introduction**

**1.1 Project Background**

Street crimes, as one of the most common-seen malefactions, poses huge threats to the individual’s safety as well as the social stability and is increasing day by day in our society. To be more specific, there has been an increasing interest in the emerging trends of stealing cars which are then being used in mugging incidents and can also be used in bomb blasting or other serious crimes and will not be recovered. Preventions were made but no proper solution regarding the recovery i.e., recovery rate has dropped significantly as these cars then move within an hour in other cities/territory and government officers are unable to track them down. In light of this issue, the team proposes a solution with the help of artificial intelligence and cybersecurity and desire to build a car recovery portal that will be used by the provincial governments’ security forces and officials to share the details and information regarding the unauthorized or stolen car with each other.

## **1.2 Project Overview**

Vehicle theft is a persistent problem in many parts of the world, and the process of recovering stolen vehicles can often be slow and difficult. To address this issue, we have developed a web-based car recovery portal that is designed to facilitate the process of recovering stolen cars. The portal consists of two roles: an admin module and a user module. The admin module is responsible for managing all user details and complaints, while the user module is used by car owners to register their cars to the database and launch complaints in the event of theft.

The aim of this project is to create a system that streamlines the process of recovering stolen cars by providing a centralized platform where users can report theft, and administrators can manage and review these complaints. The portal is built with a focus on data security and confidentiality, ensuring that user data can only be accessed by authorized users.

In this report, we will discuss the design and implementation of the car recovery portal. We will detail the features of the system, including user authentication and authorization, surveillance and number plate recognition, and notification mechanisms. We will also discuss the challenges we faced during development and the solutions we implemented to overcome them.

Overall, we believe that this project has the potential to make a significant contribution to the fight against vehicle theft, by providing law enforcement authorities with the tools they need to quickly and efficiently recover stolen vehicles.

# **Related Work**

To the best of our knowledge, there is currently no existing web-based car recovery portal that provides the same features and functionality as our proposed system. While there are some commercial vehicle tracking and recovery systems available via trackers, these solutions are often costly and may require specialized hardware or software to operate, and can be compromised via jammers.

In recent years, there have been some advancements in surveillance and number plate recognition technology for number plate detection. However, no one has thought to use this technology for this use case as it can be a better fit in a centralized environment with roadside units (RSU).

Therefore, our project is unique in that it provides a centralized platform that is accessible to car owners and administrators alike, without the need for specialized equipment or expertise. Our system is also designed with a focus on data security and confidentiality, ensuring that user data is protected at all times.

Overall, we believe that our car recovery portal has the potential to make a significant contribution to the field of vehicle recovery and could serve as a model for future development in this area.

# 

## **2.1 Project Scope**

The scope of this project is to develop a web-based car recovery portal that provides a centralized platform for car owners to register their vehicles and launch complaints in the event of theft, while also providing administrators with the tools they need to manage and review these complaints efficiently. The portal will consist of two modules: an admin module and a user module.

The user module will allow car owners to register their cars to the database by providing relevant information such as the make, model, color, and license plate number. Users will also be able to update their vehicle information and launch a complaint if their car is stolen.

The admin module will be responsible for managing all user details and complaints. The admin will have access to a dashboard that provides an overview of all registered users and their complaints. The admin can view each complaint in detail, update its status, and take necessary actions such as notifying the relevant authorities about the incident.

The system will include a surveillance and number plate recognition mechanism that uses cameras installed in the area to detect and recognize license plate numbers (**Just Concept**). The data captured by the cameras will be processed by an image recognition algorithm, which can identify the stolen car based on its license plate number. Once the car is located, the user will be notified about its location via the portal.

The project will be developed using Python, NoSQL (MongoDB), HTML & CSS, FAST API, ReactJS and Javascript, commonly used web development technologies. The system will be designed with a focus on data security and confidentiality, ensuring that user data is protected at all times.

The project will not include any specialized hardware or software, and it will not require any specialized training or expertise to operate. The system will be accessible through a web browser and will be compatible with a wide range of devices and operating systems.

Overall, the scope of this project is to develop a user-friendly and secure car recovery portal that streamlines the process of recovering stolen vehicles and provides car owners with the peace of mind they deserve.

## **2.2 Not In Scope**

1. Vehicle tracking beyond the use of cameras and number plate recognition technology: Our system relies solely on the use of High-Quality images captured via camera to detect and recognize license plate numbers. While the system may be able to locate a stolen car based on its license plate number, it does not include any advanced tracking capabilities beyond this.
2. Integration with third-party applications or services: Our system is designed to function as a standalone application and does not include any integration with third-party applications or services. While it may be possible to integrate the system with other applications in the future, this is outside the scope of this project.
3. Physical installation of cameras or other hardware: Our system relies on the use of High-Quality images that are already present in the system, and does not include the physical installation of any hardware like cameras or RSUs.
4. Legal or law enforcement procedures related to vehicle theft: Our system is designed to facilitate the process of recovering stolen vehicles, but it does not include any features or functionalities related to legal or law enforcement procedures related to vehicle theft.
5. Mobile applications or other non-web-based interfaces: Our system is a web-based application that can be accessed through a web browser on any device with an internet connection. It does not include any mobile applications or other non-web-based interfaces.
6. Additional languages other than English: Our system is developed in English and does not include any support for additional languages.
7. Integration with social media platforms or messaging applications: Our system is designed to function as a standalone application and does not include any integration with social media platforms or messaging applications.
8. Payment processing or financial transactions: Our system is designed to facilitate the recovery of stolen vehicles and does not include any features or functionalities related to payment processing or financial transactions.

## **2.3 Project Stakeholders**

In any software development project, it is important to identify the stakeholders who are impacted by the system or have a vested interest in its success. The stakeholders for the Car recovR portal include end-users, administrators, law enforcement agencies, insurance companies, and system developers. Each stakeholder group has unique interests and requirements that need to be considered during the development and implementation of the system. Understanding the perspectives of each stakeholder group can help ensure the success of the project and the satisfaction of all parties involved.

**End-users:** The primary stakeholders are the end-users who register their vehicles on the portal and use it to report stolen vehicles. They are interested in the timely and efficient recovery of their stolen vehicles.

**Administrators:** The administrators are responsible for managing user accounts, verifying requests, and coordinating with law enforcement agencies for stolen vehicle recovery. They are interested in the smooth functioning of the system, ensuring user privacy and data security, and efficient handling of complaints.

**Law Enforcement Agencies:** The law enforcement agencies are responsible for taking action on the stolen vehicle reports submitted on the portal. They are interested in the accuracy and effectiveness of the information provided by the portal for the recovery of stolen vehicles.

**Insurance Companies:** The insurance companies are interested in the timely recovery of the stolen vehicles, as this can help minimize the financial losses incurred due to the theft.

**System Developers:** The system developers are responsible for designing, developing, and maintaining the web-based portal. They are interested in the successful implementation of the system, ensuring user satisfaction, and providing ongoing technical support.

# **3. Requirements**

## **3.1 Functional Requirements**

**3.1.1 Functional Hierarchy**

### 

As per above, there is a functional hierarchy described by each of the levels of hierarchy. In terms of user and system interaction, these can be written as the following,

1. Admin should be able to login into the system.
2. Admin should be able to reset their passwords.
3. Admin should be able to register a car.
4. Registration of cars must be only done after following some set of rules.
5. Admin should be able to file a complaint for stolen/missing vehicles.
6. Admin should be able to check the complaint records.
7. Admin should be able to monitor the system that is to see footage (in case of real-time implementation)
8. Admin should be able to enter details in complaints and registration of cars.
9. Admin should be able to monitor vehicle movement in particular locations.
10. Admin should be able to see all alerts and total complaints, pending complaints, and solved complaints.
11. The Detection module should be able to fetch license plates from vehicles.
12. The Detection module should be able to trigger alerts in case of fake and unregistered license plates.
13. RSUs should be able to fetch and throw signals in terms of velocities to the detection system (in case of real-time implementation).
14. Security module should be able to encrypt the license plate and send it to the database.
15. Security module should provide a tag to every car registered in the system.
16. Security module should provide a secure communication channel between services and web-portal..
17. Security module should maintain data integrity between processes.

## **3.2 Use Cases**

The primary uses of concerns right now falls into three categories:

1. Car registration
2. Complaint registration
3. Car recovery

Each of the following use cases are further categorized into sub-functions as follows:

1. Car registration
2. Register
   1. CNIC Authorization
   2. Detect Car No. plate
3. Rules
   1. Attribute-based rules
   2. Authentication rules
4. Complaint registration
   1. File complaint
      1. User Details
      2. Get Number plate
      3. Authentication

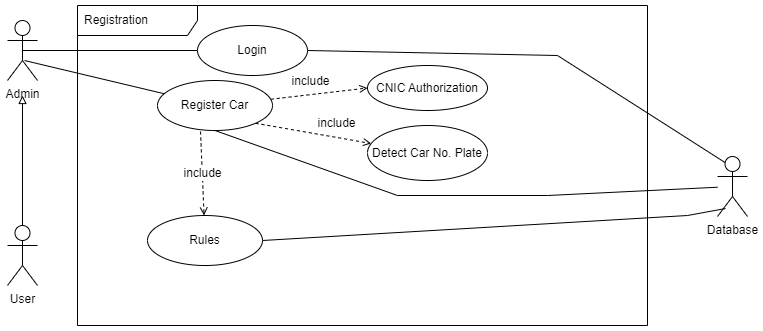
1. Car recovery
   1. Number plate
      1. Match plate
      2. Fake/Unregistered plate

### 

### 

### 

### **3.2.1 Car Registration**

**

Admin, user, and database will utilize the car registration system. Admin will log in to the system and then the user will be telling the information to the admin when registering the user's car. This provided information will then be authenticated on the basis of attribute-based rules.

| **Login: 1.1** | | | | |
| --- | --- | --- | --- | --- |
| **Use case Id:** | | 1.1 | | |
| **Actors:**  User, Admin, Database | | | | |
| **Feature:** To access web portal | | | | |
| **Pre-condition:** | | Admin must have credentials | | |
| **Scenarios:** Admin opens a web browser and enters the web url then enters the credentials and click on login to successfully log in into the system. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | Enter email and password | | | Home page is rendered |
| **Alternate Scenarios:** NA | | | | |
| **Post Conditions :** Successful login | | | | |
| **Step#** | **Description** | | | |
| **1.** | Successful login after providing correct credentials | | | |
| **Use Case Cross referenced** | | | NA | |

| **Register Car: 1.2** | | | | |
| --- | --- | --- | --- | --- |
| **Use case Id:** | | 1.2 | | |
| **Actors:**  User, Admin, Database | | | | |
| **Feature:** To register number plate of car | | | | |
| **Pre-condition:** | | Admin must be signed in into the system. | | |
| **Scenarios:** User asked the admin to register the car by providing the details. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | Admin enters user basic information | | | Backend validate the information on the basis of prescribed rules and then register number plate after validation |
| **Alternate Scenarios:** NA | | | | |
| **Post Conditions :** Car will register after fulfilling the registering criteria. | | | | |
| **Step#** | **Description** | | | |
| **1.** | Name, CNIC#, phone# , and email will be provided through the web form and image of number plate will be extracted by the camera | | | |
| **Use Case Cross referenced** | | | NA | |

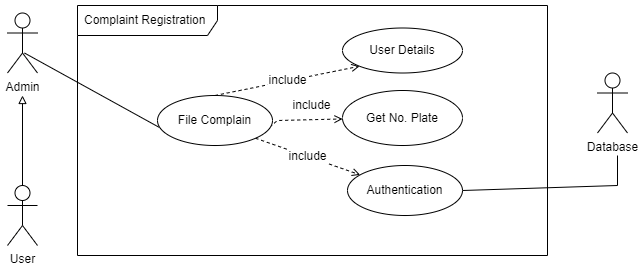
| **Rules: 1.3** | | | | |
| --- | --- | --- | --- | --- |
| **Use case Id:** | | 1.3 | | |
| **Actors:**  User, Admin, Database | | | | |
| **Feature:** Attribute based rules to authenticate information provided through form | | | | |
| **Pre-condition:** | | Information must be provided to trigger the rules which will be checked in this use case. | | |
| **Scenarios:** Admin enters the data in the form and submit the form to the server, server triggers the rule to authenticate the vehicle. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | Validates information provided on car registration form | | | If rules are satisfied then, database is updated with record |
| **Alternate Scenarios:** NA | | | | |
| **Step#** | **Description** | | | |
| **1.** | Hash will be placed in an excel file or database with respect to the car number plate. | | | |
| **Use Case Cross referenced** | | | Register car : 1.2 | |

| **CNIC Authorization: 1.4** | | | | |
| --- | --- | --- | --- | --- |
| **Use case Id:** | | 1.4 | | |
| **Actors:**  User, Admin, Database | | | | |
| **Feature:** A unique CNIC# with a particular regex is needed to register a car | | | | |
| **Pre-condition:** | | CNIC must be unique for each user | | |
| **Scenarios:** While registering, CNIC of the car owner must be provided and it will be authorize via rules. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | Validates information provided on car registration form | | | If rules are satisfied then, database is updated with record |
| **Alternate Scenarios:** NA | | | | |
| **Step#** | **Description** | | | |
| **1.** | CNIC number will be provided with respect to the car owner to register a car on a CNIC number. | | | |
| **Use Case Cross referenced** | | | Register car : 1.2 | |

| **Detecting Car No. Plate : 1.5** | | | | |
| --- | --- | --- | --- | --- |
| **Use case Id:** | | 1.5 | | |
| **Actors:**  User, Admin, Database | | | | |
| **Feature:** A unique car no. plate is needed to register a car | | | | |
| **Pre-condition:** | | Car no. plate must be unique for each user | | |
| **Scenarios:**  While in registration, a camera will detect the car’s number plate and send it to the database via rules. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | Validates information provided on car registration form | | | If rules are satisfied then, database is updated with record |
| **Alternate Scenarios:** NA | | | | |
| **Post Conditions :** Camera successfully detected car’s number plate. | | | | |
| **Step#** | **Description** | | | |
| **1.** | Camera detects a number plate via an AI model and it will extract the string to store in the database. | | | |

| **Use Case Cross referenced** | | | Register car : 1.2 | |
| --- | --- | --- | --- | --- |

### **3.2.2 Complaint Registration**



In the complaint registration system,a complaint of stolen car is launched and before filing the complaint, the system will validate the information from the database and. The complaint will then launch on successful authentication.

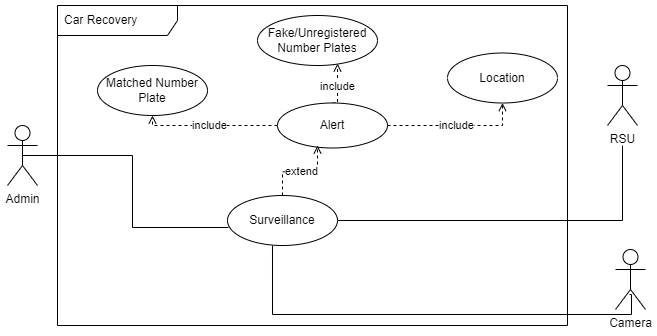
| **File Complain : 2.1** | | | | |
| --- | --- | --- | --- | --- |
| **Use case Id:** | | 2.1 | | |
| **Actors:**  User, Admin, Database | | | | |
| **Feature:** User complain if his car is stolen | | | | |
| **Pre-condition:** | | Car no. plate must exist in the database | | |
| **Scenarios:** User comes to the admin to register a car stolen, theft, lost complaint and admin launches a complaint after filling a web form. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | Complain information is extracted through form | | | Authenticates and show results i.e success or fail |
| **Alternate Scenarios:** NA | | | | |
| **Post Conditions :** User filed a complaint after fulfilling the criteria. | | | | |
| **Step#** | **Description** | | | |
| **1.** | Complain form is filled | | | |
| **2.** | Backend authentication | | | |
| **3.** | Updates on database or vice versa | | | |
| **Use Case Cross referenced** | | | NA | |

| **User Details: 2.2** | | | | |
| --- | --- | --- | --- | --- |
| **Use case Id:** | | 2.2 | | |
| **Actors:**  User, Admin, Database | | | | |
| **Feature:** User details is necessary to file a complaint | | | | |
| **Pre-condition:** | | NA | | |
| **Scenarios:** Admin enters user details provided by user while filing a complaint for the car | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | Credentials such as email, name, CNIC is provided | | | Validates from a set of rules |
| **Alternate Scenarios:** NA | | | | |
| **Post Conditions :** Admin successfully entered the details for a user to file a complaint. | | | | |
| **Step#** | **Description** | | | |
| **1.** | User details is entered in a form | | | |
| **2.** | Backend validates from database | | | |
| **Use Case Cross referenced** | | | Complaint Register:2.1 |  |

| **Get no. plate: 2.3** | | | | |
| --- | --- | --- | --- | --- |
| **Use case Id:** | | 2.2 | | |
| **Actors:**  User, Admin, Database | | | | |
| **Feature:** Car no. plate is necessary to file a complaint | | | | |
| **Pre-condition:** | | Number plate must be registered in the database. | | |
| **Scenarios:** While launching a complaint for a car, the number late must be provided by user and it will validate via authentication. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | Car no. plate is provided | | | Validates from database |
| **Alternate Scenarios:** NA | | | | |
| **Post Conditions :** Car number plate and details provided by the user and already present in the database. | | | | |
| **Step#** | **Description** | | | |
| **1.** | User details is entered in a form | | | |
| **2.** | Backend validates from database | | | |
| **Use Case Cross referenced** | | | Complaint Register:2.1 |  |

| **Authentication: 2.4** | | | | |
| --- | --- | --- | --- | --- |
| **Use case Id:** | | 2.4 | | |
| **Actors:**  User, Admin, Database | | | | |
| **Feature:** All the details provided by admin is verified from a web form | | | | |
| **Pre-condition:** | | User Details must be provided by the admin via web form. | | |
| **Scenarios:** Admin launched the complaint, while successfully registering a complaint, authentication module validate and authenticate the details to complete the process. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | Checks each field of information with rules | | | Satisfied(returns true) if all the rules matches and vice versa |
| **Alternate Scenarios:** NA | | | | |
| **Post Conditions :** User filed a complaint after fulfilling the criteria. | | | | |
| **Step#** | **Description** | | | |
| **1.** | Validating information via database. | | | |
| **2.** | Sending data to nodes. | | | |
| **3.** | Receiving response from Database | | | |
| **Use Case Cross referenced** | | | Complaint Register:2.1 | |

### **3.2.3 Car Recovery**



On car recovery, surveillance will be done through camera and RSU and alert will be launched if the number plate matches or the number plate is fake/unregistered. The alert feature will provide location to the admin by returning latitude and longitude of the car.

| **Surveillance : 3.1** | | | | |
| --- | --- | --- | --- | --- |
| **Use case Id:** | | 3.1 | | |
| **Actors:**  Admin, RSU, Camera | | | | |
| **Feature:** Look for the stolen car and monitor vehicles. | | | | |
| **Pre-condition:** | | Cameras must be installed to monitor traffic. | | |
| **Scenarios:** Web Page displays option to monitor the vehicles. Admin clicks on the button, webpage displays traffic visuals. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | Traverse through each camera | | | If car found then tells location(longitude, latitude) |
| **Alternate Scenarios:** NA | | | | |
| **Post Conditions :** Admin successfully able to monitor the traffic. | | | | |
| **Step#** | **Description** | | | |
| **1.** | Admin enter the number plate | | | |
| **2.** | a real time table is shown which is being update | | | |
| **Use Case Cross referenced** | | | NA | |

| **Alert : 3.2** | | | | |
| --- | --- | --- | --- | --- |
| **Use case Id:** | | 3.2 | | |
| **Actors:**  Admin, RSU, Camera | | | | |
| **Feature:** System alerts when the location of stolen or fake/unregistered car is found | | | | |
| **Pre-condition:** | | There must be an unregistered vehicle or complaint vehicle for the alert. | | |
| **Scenarios:** System generate alerts for unregistered, stolen vehicles. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | This feature triggers as soon as location of car is found | | | notification keeps on generating until the user does not responds to it |
| **Alternate Scenarios:** NA | | | | |
| **Post Conditions :** System generate alerts successfully. | | | | |
| **Step#** | **Description** | | | |
| **1.** | System keeps on checking the car | | | |
| **2.** | Triggers alert feature once the car is found | | | |
| **Use Case Cross referenced** | | | Surveillance : 3.1 | |

| **Matched No. Plate : 3.3** | | | | |
| --- | --- | --- | --- | --- |
| **Use case Id:** | | 3.3 | | |
| **Actors:**  Admin, RSU, Camera | | | | |
| **Feature:** Alert will trigger if number plate matches | | | | |
| **Pre-condition:** | | There should be a complaint vehicle to get matched from monitored vehicle. | | |
| **Scenarios:** From camera when a complaint vehicle matches, it triggers the alert module. Blockchain system will send and receive the data from nodes. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | Keeps on searching the location of number plate | | | Alert will trigger if matched |
| **Alternate Scenarios:** NA | | | | |
| **Post Conditions :** A number plate matches from the monitored or complaint vehicle number plate successfully. | | | | |
| **Step#** | **Description** | | | |
| **1.** | Nodes send and receive number plates to get matched. | | | |
| **Use Case Cross referenced** | | | Alert: 3.2 | |

| **Location : 3.4** | | | | |
| --- | --- | --- | --- | --- |
| **Use case Id:** | | 3.4 | | |
| **Actors:**  Admin, RSU, Camera | | | | |
| **Feature:** returns the location of the stolen car or is fake/unregistered | | | | |
| **Pre-condition:** | | RSU and camera should be functional | | |
| **Scenarios:** When a camera sends a signal via RSU to the system it will send the vehicle location. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | Returns location of a car | | | Triggers alert function if stolen car is found or is fake/unregistered |
| **Alternate Scenarios:** NA | | | | |
| **Post Conditions :** Vehicle location send successfully accurate up to 80%. | | | | |
| **Step#** | **Description** | | | |
| **1.** | RSU and Camera work as a unit to send the data to the system for a particular vehicle. | | | |
| **Use Case Cross referenced** | | | Alert: 3.2 | |

| **Fake/Unregistered Number plate : 3.5** | | | | |
| --- | --- | --- | --- | --- |
| **Use case Id:** | | 2.1 | | |
| **Actors:**  Admin, RSU, Camera | | | | |
| **Feature:** Alert will trigger if number plate is not in the systems database | | | | |
| **Pre-condition:** | | A camera should detect fake number plate, unregistered number plate or a car with no number plate. | | |
| **Scenarios:** RSU and Camera work as a unit to detect and send data for unregistered and fake number plates. | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | Keeps on validating the number plate | | | Alert will trigger if number plate is not in database |
| **Alternate Scenarios:** NA | | | | |
| **Post Conditions :** Data for unregistered and fake number plates sent to the system successfully to trigger alerts. | | | | |
| **Step#** | **Description** | | | |
| **1.** | RSU and Camera work as a unit to send the data to the system for a particular vehicle. | | | |
| **Use Case Cross referenced** | | | Alert: 3.2 | |

## **3.3 Non-functional Requirements**

### **3.3.1 Performance Requirements**

The system should be prepared to handle large amounts of incoming traffic and not falter. It should cater to increasing and decreasing demand, and provide a solution that does not prevent users from being unable to browse and use the features smoothly. The application should be able to perform requests and services as intended, providing a robust and correct set of inputs and outputs, and being able to cater to user mistakes. It should be fast in loading the correct pages and the data in a smooth fashion, and should not overwhelm the user’s machine with its software requirements.

The application should, thus, provide,

1. **Speed** **-** Fast for loading, reloading, content fetching, and not putting limitations on the client end and instead on the user’s end.
2. **Precision -** The software should be able to tolerate mistakes, user problems, and deliver the right results without compromise, and discrepancy in the output provided.
3. **Concurrency -** The software should be able to handle multiple requests and multiple “activated” functionalities concurrently, where needed.
4. **Capacity -** The software should be able to handle and deal with multiple concurrent users requesting for the same resources.
5. **Safety -** The complexity of the software is minimal, for both safety and security purposes, for administrative and normal user procedures. It does not pose harm to the end user. Security modes of software and system are addressed in the design of the software.
6. **Reliability -** The software should exhibit failure free functionality and work with correctness and reliability.

### **3.3.2 Safety Requirements**

The software does not include functionalities or operations that would result in possible loss, damage, or harm from the use of our system. It does not pose a physical, financial or otherwise cause of harm to the end user. It does not put the interface in such a way that would affect the user negatively, or put any user in any risk or case of discrimination through poor interface design and color sensitivity.

### 

### **3.3.3 Security Requirements**

Obvious issues such as data privacy, integrity, confidentiality, protection of data generated by users through their content, protection of internal private information, and exposure to data to only what is needed when is the primary concern of the software project. The data modeling aspect concerns itself with what is populated by what, and who has access to what kind of information and at what level. We also define what resources are defined under what users and at what capacity to ensure unauthorized users and data are redirected accordingly. Unauthenticated requests for data are prevented from accessing any part of the system through the use of guards, proper digital signatures and self-signed certificates assure the confidentiality, and integrity of the requests, and data is ensured by the proper management of sessions and through password encryption.

The system should focus on setting up its probable infrastructure as securely as possible and making sure that communication between different services is done securely and data is trusted to be stored as expected. Authentication, and authorization, are all important concerns and are implemented via security layered protocol.

### **3.3.4 User Documentation**

Our requirements do not yet specify the development of user documentation to act as a helping guide. We assume that the user interfaces will be simple and enough to be used at an MVP level. This is because the scope of an FYP is limited to development as a main priority. If time allows and if needed, user documentation can be developed after the development has been completed and has ceased

# **4. Design Considerations**

The system should be in all cases reliable, robust, adaptive, and flexible for development and use purposes. For this intent, we spent a considerable amount of time trying to work towards a better design flow and structure for the project, in order to solve major problems that existed with the design of the system on a finer-grained level.

The issues related to design were

1. What is the architecture we would like to develop our front-end and back-end on? What frameworks/libraries we believe align with the workflow we have and how would we like to break the project down into smaller components
2. How would we go about solving the remote code execution problem and what tools would we use to enable real-time collaboration? It is a niche and difficult problem to solve - what approaches could we use to counter these problems?
3. What would we use to build rapid UI, and how would we manage user states and sessions on the frontend?
4. How would we deal with authentication?
5. How would we talk to the database?

A few other affecting considerations can be about,

1. **Compatibility -** since it is a web application, we care about compatibility on different browsers enough so that it does not break for our users.
2. **Modularity -** modules are designed to be well-defined, independent components which would lead to better maintainability. Dependencies are clearly defined where needed and are well-intentioned and well-contained.
3. **Fault-tolerance -** the application is not yet planned to be beyond local development environments, so this consideration does not really apply.
4. **Maintainability -** the codebase can be broken down into modules, so bugs if any are module-contained, easy to trace and detect.
5. **Reliability -** because we are working on the local environment, it is easy to check for reliability for the software to do as it is expected within stated conditions for a specified period of time
6. **Reusability -** we are using already developed software and packages where we believe we can use the functionality that they provide us, reducing our time to develop from scratch.
7. **Robustness -** that the software operates under the stress of users and inputs or tolerates unpredictable or invalid input.
8. **Security -** the software is able to withstand and resist hostile acts and influences.
9. **Usability -** the interface is designed to be simple and easy to use without the need to involve user documentation manuals.
10. **Performance -** the software will be usable across a number of platforms of mobile, desktop, and laptops.
11. **Portability -** since it is a web application, the software is perfectly portable.
12. **Scalability -** the software adapts well to increasing data or added features or a number of users

## **4.1 Assumptions and Dependencies**

### **4.1.1 System Assumptions**

Going with the assumption (and generally the will) of putting the application finally into production once everything is completed and set up, we would like to cater to certain architectures to better adjust to our economic needs and finance.

For example, we would like to develop a system that is easy to modularize and break into components and microservices to be deployed on machines. As such, we would like to write a backend that supports such service related breakdown and adjustment.

As for the environment, it is a fact that users on different platforms and screens will use our platform.

Our dependencies will rely on the packages we use for development. We are assuming that they are long supported, lasting, and are easy enough to develop, modify as per our needs, and quick to learn to use in our project directly.

### **4.1.2 Project Delivery Details**

As per concern with the team’s idea of system and software design, we imagine design to be concerned with data modeling, entities, abstractions, architectural patterns, and strategies (design models being a strategy to achieve those goals). Our sense of assumptions encapsulates notions about the project’s timeline, deliverables, and the workflow that we assume will be in place, in order to help the project run but cannot be guaranteed.

It is a want to have those assumptions never proven wrong, otherwise, there will be a direct impact on the project.

And so, what were our assumptions starting FYP I. They were as the following,

1. We would be able to finish all of the required engineering and analysis before Mid I (FYP defense) in order to not stumble into the unforeseen territory and have a team consensus on what must be done and what should be left for now, as per a team timeline.
2. We would be able to finish our Registration module completely, all of the backend as well as the integration with the user flow before Mid 2.
3. We would be able to finish all of the Complaint System prior to the finals completely, all of the back-end, as well as the integration with the user flow before the finals.
4. We will not have any changes and will be fixated on a set of requirements at the start of the semester.
5. We would be able to manage personal time to handle this large project as per our expectations - we realize realistically and responsibly that this project is of large scope and cannot be implemented here in Pakistan but its importance is gigantic, and that missing deadlines would keep spiraling into other deadlines.

For each of the above assumptions, many were altered,

1. A change occurred after the FYP defense where we were asked to implement frontend in FYP-I deliverable and we have a plan to develop frontend after FYP-I as we have extensive work to do on backend. Moreover, we were also asked to remove the idea of blockchain from the project and add pure cybersecurity aspects for security concerns such as encryption, digital signatures and self signed certificates,
2. While most of our work related to the Registration module was completed before Mid 2 in terms of back-end, we could not work on integration and have been working on it since.
3. A bit of uncertainty with data modeling led us to exploring Remote Code Execution, Real Time Collaboration solutions, and we were generally unwilling to start implementation until our defense was approved. We were scared of “what if this does not get approved?”, or “what if we have to make changes at the core of requirements?”. While these fears were not of fruition, it led the team to realize document related deadlines/timelines and how much they can affect a project’s progress.

For dependencies, we were blocked by a few factors we were waiting on/for to pass,

1. FYP Defense - was the reason that we could not start work prior to the defense, and if we had the opportunity of getting our project approved prior to the fixed date, we might have taken that option.
2. Data modeling had to be altered at different times to cater to requirements

## 

## **4.2 Risks & Volatile Areas**

There is little to no time to consider, implement, and practice a contingency path. As such, we should not be facing any further changes in the requirements at the moment and our choice of technology is likely to remain consistent throughout the project - we have not faced any conditions where we would not use our current set of tools and implement functionalities further with those.

It is also important to note that any further developments and requirements would further alter our timeline as altered it already is. If any developments do arise, our initial approach would be as always to understand if it’s a priority, and if so, when does it become a priority to get done first.

# 

# 

# **5. Design Strategy**

The front end of the website is done using the React framework which is a Javascript library as it allows developers to easily create fast user interfaces for websites and applications. It is a user-friendly and highly compelling framework that provides a number of possibilities to the developers to make them more creative.

Why React?

* React is a component-based framework that manages its own state, then composes them to make complex UIs.
* The JS library consists of several functions including one that converts the HTML components into required functions and transforms the entire project so that it is easy to understand.
* Provides reusability i.e. complex and detailed components can further be divided into sub-components. Also, one can develop new features in React without rewriting existing code.
* Provides simplicity through one-way data binding. This means that absolutely anyone can track all the changes made to any particular segment of the data.

The backend and all its functionalities are designed in FastAPI FastAPI is an open-source, mature, and well-supported library that aims at building sophisticated web APIs. In regards to functionality, we use the Python language to design the functions therefore to successfully enable these functionalities..

Reason for using FastAPI;

* High Performance: FastAPI is built on top of the Starlette framework, which is optimized for high performance. As a result, FastAPI can handle high volumes of traffic with low latency and high throughput.
* Easy Integration: FastAPI is designed to be easily integrated with other Python libraries and frameworks.
* Built-in Testing: FastAPI includes built-in support for automated testing, making it easy to test your API endpoints and ensure that they are working as expected.
* FastAPI is designed to be scalable and can handle large applications with ease. It is built on top of the ASGI specification, which allows for high-performance, asynchronous web servers.
* HTTP response handling, content-type are easily changed using HTTP Accept headers.

For the database storage, the data will be stored in MongoDB, a document-oriented NoSQL database that is designed for high performance, scalability, and flexibility. MongoDB is a distributed database that can be deployed across multiple data centers and cloud providers. Also, MongoDB provides a rich query language with support for complex queries, including text search, geospatial queries, and aggregation queries.

The software architecture used will be Layered Architecture which is the most common pattern otherwise known as n-tier architecture. Components within the layered architecture pattern are organized into horizontal layers, each layer performing a specific role within the application (e.g., presentation logic or business logic)

One of the powerful features of the layered architecture pattern is the separation of concerns among components. Components within a specific layer deal only with logic that pertains to that layer. For example, components in the presentation layer deal only with presentation logic, whereas components residing in the business layer deal only with business logic.

# **6. Design Details**

Following are the design details of this project.

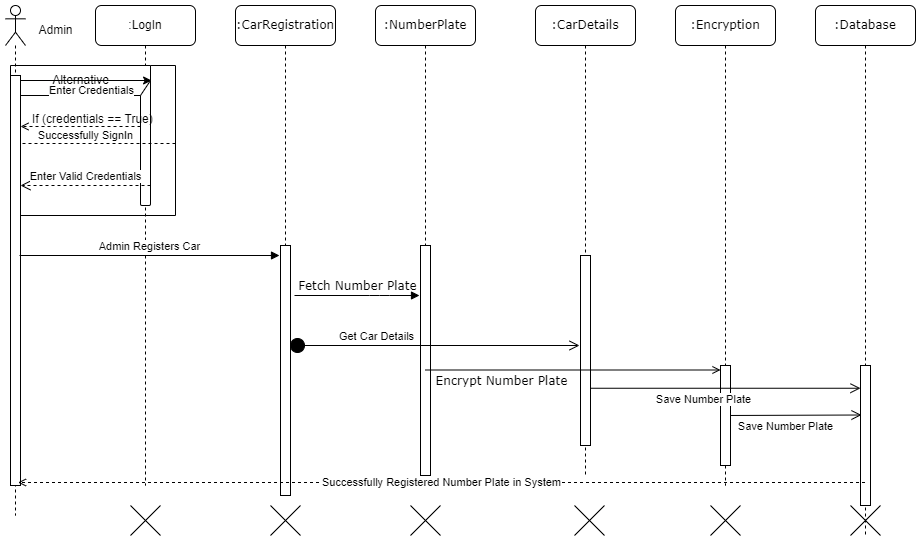
## **6.1 Process Model**

While the Agile Scrum framework was used as the basis for the process model for the car recovery portal project, it was not always possible to strictly adhere to it. There were several factors that made it difficult to follow the process model completely, such as unexpected changes in requirements, unanticipated technical challenges, and team member availability issues. As a result, some sprints were extended, and some features were postponed to subsequent sprints to allow more time for development and testing. Despite these challenges, the team remained committed to the Agile Scrum framework and was able to successfully complete the project within the given timeline, delivering a high-quality product that met the user requirements.

## **6.2 Sequence Diagram**

Following is the sequence of activities for the project.

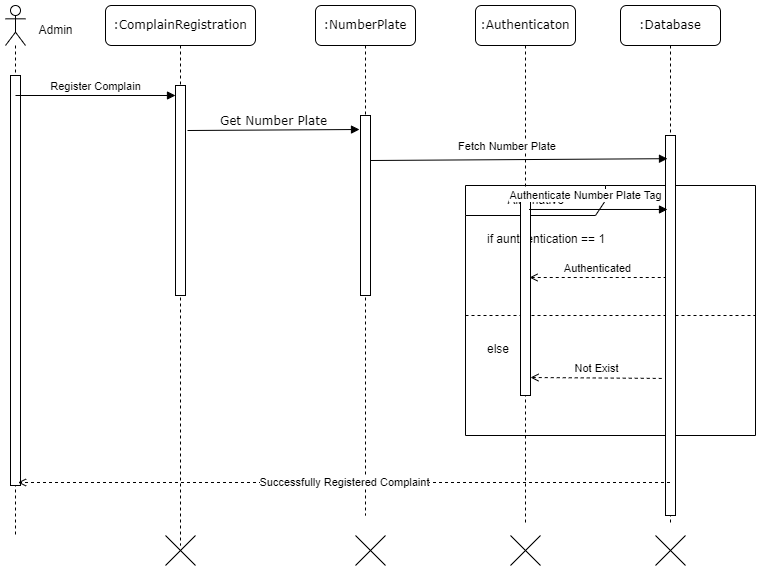
### **6.2.1 Car Registration**



This sequence diagram depicts the overall registration process.

* After login, the admin submits the registration form.
* The backend then fetch number plate , encrypt it
* The encrypted number plate and all the information is saved in the database
* The system also deals the wrong credentials i.e on incorrect credentials, it will request the user to enter a valid credentials

### **6.2.2 Complaint Registration**



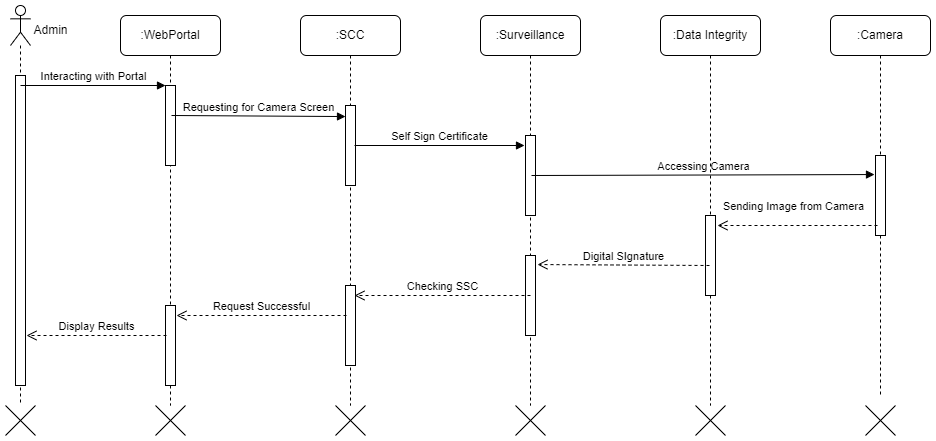
This sequence diagram depicts the overall complaint registration process.

* Firstly, the admin registered the complaint in the complaint module.
* But for registering a complaint of the stolen car, admin has to have stolen car number plate against which action should be taken.
* Then, it is being verified that if the given number plate of stolen car belongs to the database of registered cars then

- if the car belongs to the database, successfully register a complaint of the stolen car.

- if not, unsuccessfully by giving a response as the car is not registered or exists.

### **6.2.3 Surveillance**



This sequence diagram depicts the overall response/monitor from the camera to the web portal.

* Admin is interacting with the surveillance of a web portal to see the result of the camera.
* To communicate with the camera, the web portal must perform some security infrastructure i.e.

- A self-signed certificate to authenticate that it's the registered web portal that is trying to communicate with the camera.

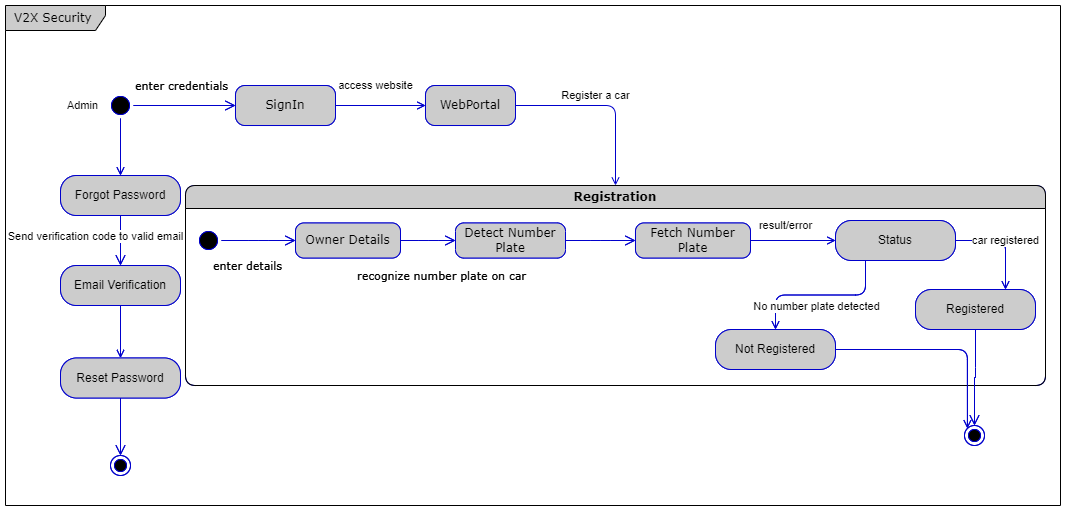
- In response to the web portal, images were sent in the data format from the camera to the web portal. In order to prevent the integrity of data, data is assigned with digital signatures.

- It shows the result in the web portal after the successful transfer of data.

## **6.3 State Diagram**

Following are the state diagrams of this project.

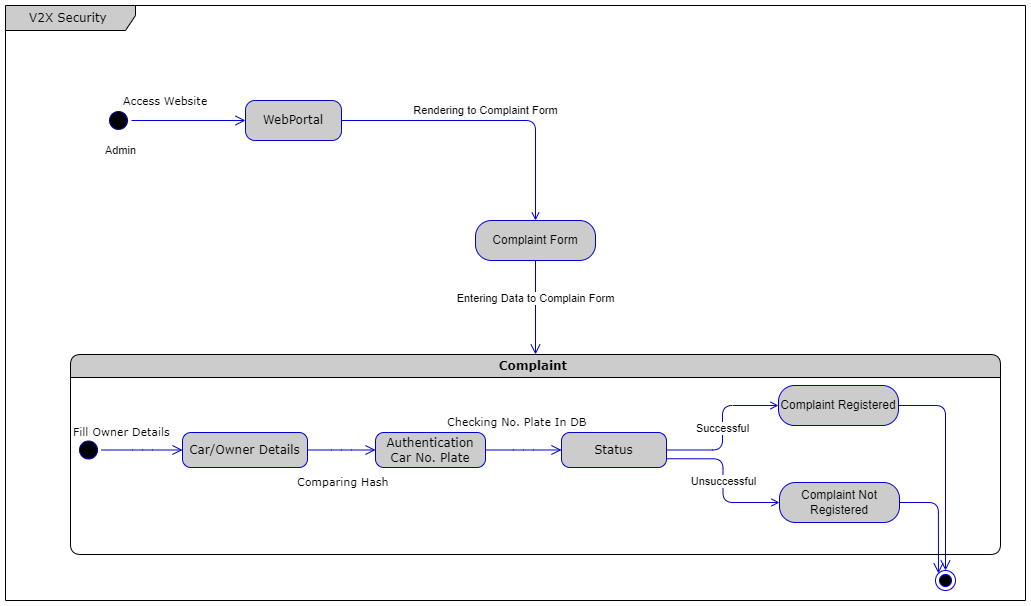
### **6.3.1 Car Registration**



The above state diagram illustrates all the states of a car registration;

* Admin enters the credentials in the login page.
* On successful login;
  + For registration, the owner details are provided through a web form.
  + The backend detects the textual number plate from an image and fetches it.
  + Then it decides whether to save all the information in the database or not. The decision depends upon the text extraction.
  + If the number plate is extracted, the data is then stored in the database
  + If the image is not clear or if the captured image does not contain number plate, then the data will not be stored in the database
* If the admin forget his/her password;
  + The system then ask for email to verify/authenticate and the system resets the password.

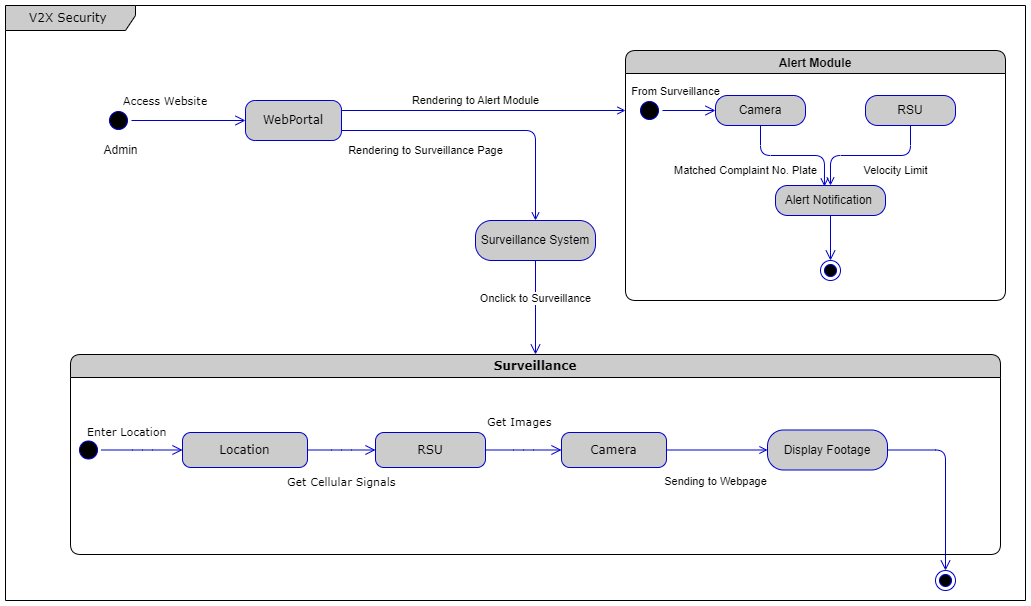
### **6.3.2 Complaint Registration**



The above state diagram illustrate all the states of filing a complaint;

* The admin access the website through login process and goes to the complain section
* Data will be entered in the complaint form.
* Backend then authenticates the car number plate.
* If the number plate matches the number plate of the database, the complaint is then registered.
* On an unsuccessful match, a complaint will not be launched.

### **6.3.3 Surveillance**

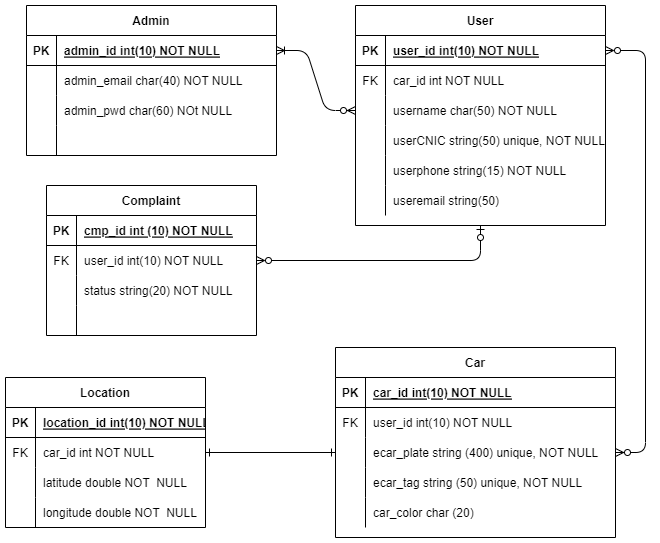


The above state diagram illustrate all the states of surveillance module;

* To reach a surveillance state, admin access the website through login process and goes to the surveillance section
* To view the information of a particular location, the admin enters the location name
* A table appears containing the information of all the location which the backend fetches from the images captured by the camera from RSU
* Alert module will be launched, i.e system will send the notification to the admin if the number plate is matched with the complaint number plate.

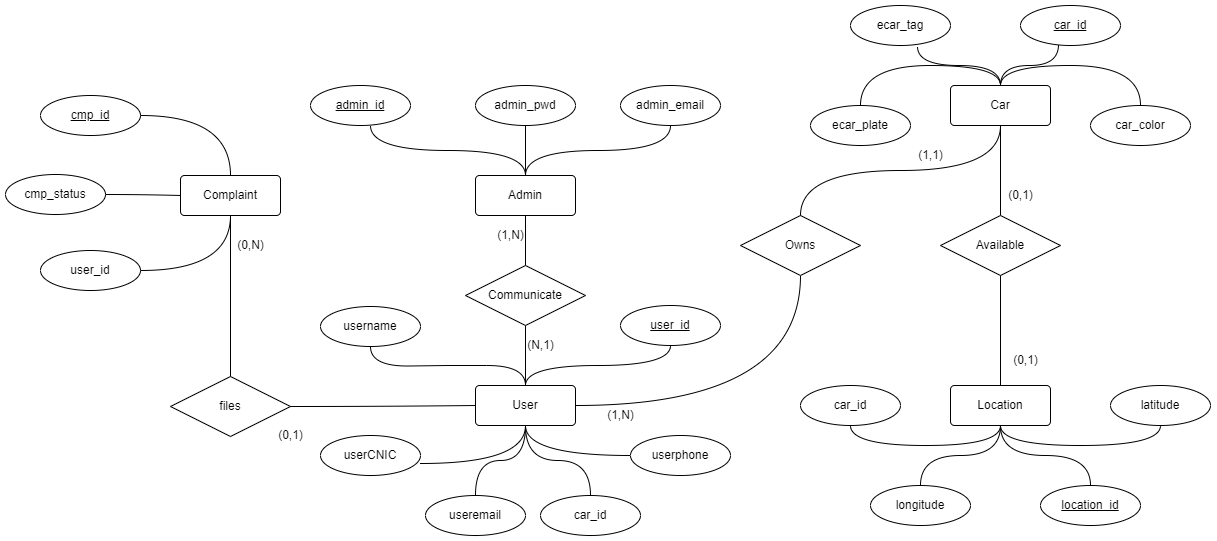
## **6.4 Database Design**

## **6.4.1 ER Model**



## **6.4.2 Entity Relation Diagram**

We have used ERD to describe the relationship of how we have structured the users, their attributes, and the functions in the database. All the relationships are made in such a way that helps us in implementing the functionalities of our project in an effective way.



In the entity relationship diagram, it is been included that,

* User is an entity that contains the attributes like User\_id(PK), Username, UserCNIC, Userphone, User email, Car\_id(FK)
* Another entity named Admin which contains the attributes named admin\_id(PK), admin\_email, admin\_password.
* These two entities have a one-to-many relationship i.e. users are communicating to one admin at a time.
* Furthermore, an entity named Car contains attributes like ecar\_plate, Ecar\_tag, car\_id(FK), car\_color.
* Entity named CAr and User has a relation i.e. user owns a car
* Locations as entities have location\_id(PK), longitude, car\_id(FK), and latitude. Location entity help in detecting the car availability on the specific location i.e. car can be found or not on the location being monitored
* Complaint entities with cmp\_id(PK), cmp\_status, user\_id(FK), have a relationship with user for filing none or many complaints of the stolen car.

## **6.4.3 Data Dictionary**

#### 6.4.3.1 Admin

| **Admin** | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | | Admin | | | | | |
| **Alias** | | N/A | | | | | |
| **Where-used/how-used** | | Used for performing activities like complain register, registration of cars | | | | | |
| **Content description** | | = | | | | | |
|  | | | | | | | |
| **Column Name** | **Description** | | **Type** | **Length** | **Null able** | **Default Value** | **Key Type** |
| admin\_id | the admin identity | | integer | 10 | NA | NA | Primary Key |
| admin\_email | the email of the admin | | character | 40 | NA | NA |  |
| admin\_pwd | the password to mail register to admin | | character | 60 | NA | NA |  |

#### 6.4.3.2 User

| **User** | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | | User | | | | | |
| **Alias** | | N/A | | | | | |
| **Where-used/how-used** | | Used in response from the admin | | | | | |
| **Content description** | | = | | | | | |
|  | | | | | | | |
| **Column Name** | **Description** | | **Type** | **Length** | **Null able** | **Default Value** | **Key Type** |
| user\_id | the user identity | | integer | 10 | NA | NA | Primary Key |
| username | the username of the user | | character | 50 | NA | NA |  |
| user\_email | the email address of the mail | | string | 50 |  | NA |  |
| user\_phone\_number | the phone number of the user | | string | 15 | NA | NA |  |
| user\_CNIC | CNIC of the user | | string | 50 | NA | NA |  |
| car\_id | the car identity | | integer | 10 | NA | NA | Foreign Key |

#### 6.4.3.3 Car

| **Car** | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | | Car | | | | | |
| **Alias** | | N/A | | | | | |
| **Where-used/how-used** | | Used in the recovery of the stolen car and register of cars | | | | | |
| **Content description** | | = | | | | | |
|  | | | | | | | |
| **Column Name** | **Description** | | **Type** | **Length** | **Null able** | **Default Value** | **Key Type** |
| car\_id | the car identity | | integer | 10 | NA | NA | Primary Key |
| e\_car\_plate | the number plate of the car | | string | 400 | NA | NA |  |
| e\_car\_tag | tag assigned to each car number plate | | string | 50 | NA | NA |  |
| car color | the color of the car | | character | 20 |  | NA |  |
| user\_id | the user identity | | integer | 10 | NA | NA | Foreign key |

#### 6.4.3.4 Location

| **Location** | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | | Location | | | | | |
| **Alias** | | N/A | | | | | |
| **Where-used/how-used** | | Used in getting the location of the car been seen | | | | | |
| **Content description** | | = | | | | | |
|  | | | | | | | |
| **Column Name** | **Description** | | **Type** | **Length** | **Null able** | **Default Value** | **Key Type** |
| location\_id | the location identity | | integer | 10 | NA | NA | Primary Key |
| latitude | the latitude of the location | | double | 50 | NA | NA |  |
| longitude | the longitude of the location | | double | 50 | NA | NA |  |
| car\_id | the car identity | | integer | 10 | NA | NA | Foreign Key |

### 

### 

### 

### 

### 

### 

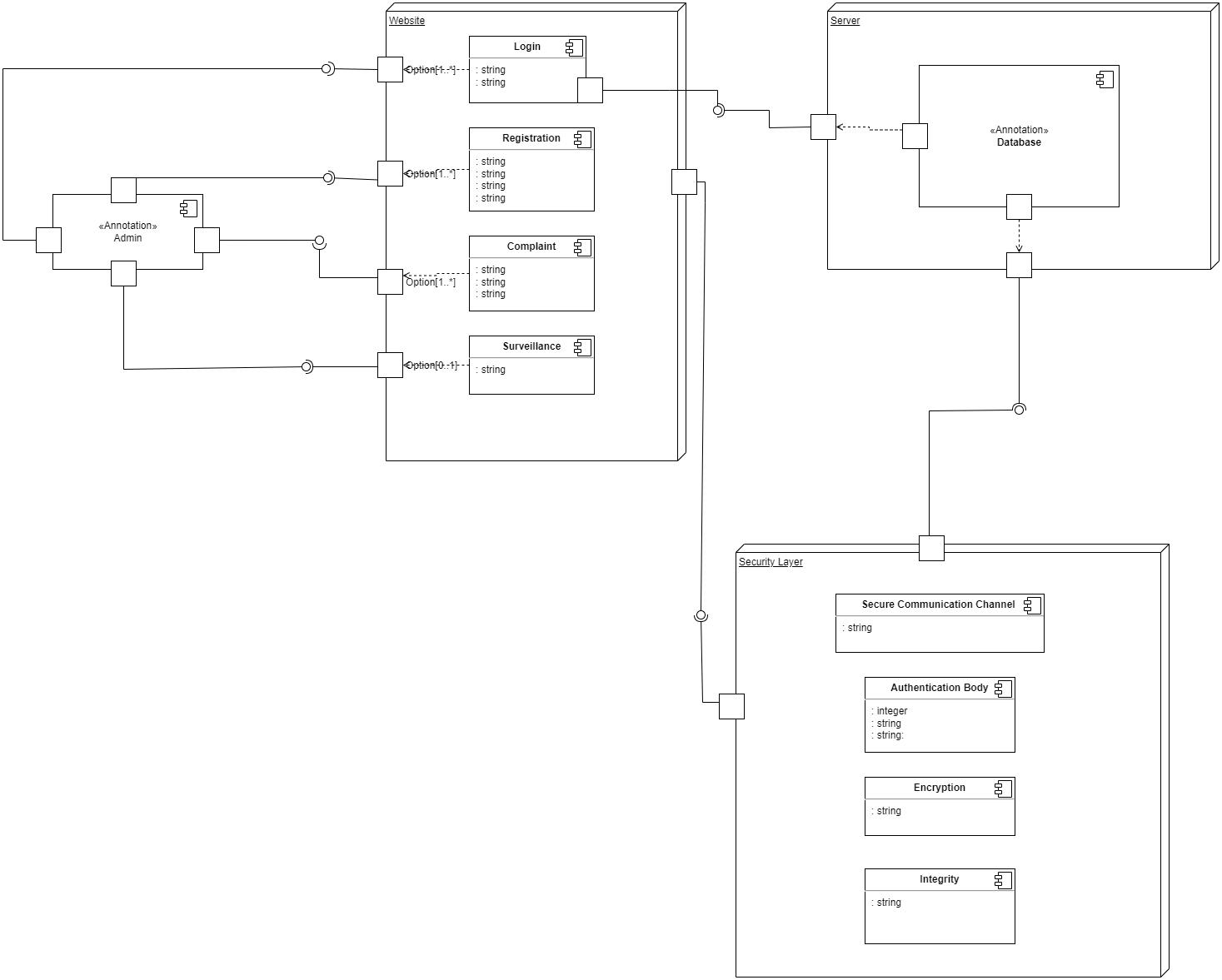
#### 6.4.3.5 Complaint

| **Complaint** | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | | Complaint | | | | | |
| **Alias** | | N/A | | | | | |
| **Where-used/how-used** | | Used in filing complain of stolen car | | | | | |
| **Content description** | | = | | | | | |
|  | | | | | | | |
| **Column Name** | **Description** | | **Type** | **Length** | **Null able** | **Default Value** | **Key Type** |
| cmp\_id | the complain identity | | integer | 10 | NA | NA | Primary Key |
| user\_id | the user identity | | integer | 10 | NA | NA | Foreign Key |
| status | the complaint status | | string | 20 | NA | NA |  |

## 

## **6.5 System Architecture**

### **6.5.1 System Level Architecture**



# 

### **6.5.2 Software Level Architecture**

# 

# 

# 

# 

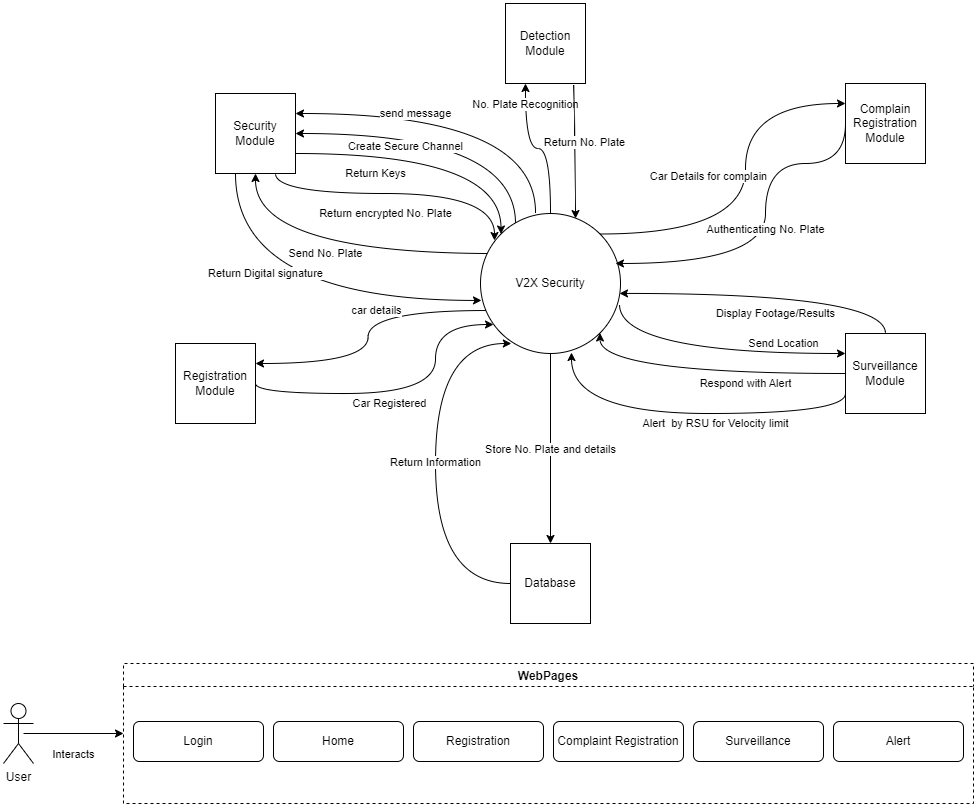
# 

# 

# 

# 

# **7. External Interfaces**



## **7.1 Hardware Interfaces**

As such, there is not really a place for hardware interfaces currently as all of the system lies entirely on the developer’s machine for now. Thus, as a result, the client will only need a web browser to view the pages on demand. The only hardware in use is just the machine that the user is using to run the web application.

Moreover, there are two main hardware components that will interact with the software interfaces.

1. Road Side Unit
2. CCTV Camera

These two devices only be used in a real-time environment but the project is purely based on simulation.

## **7.2 Software Interfaces**

There are 4 primary flows of data in the system,

1. **Services** - Services provide the backend logic needed to connect with the database (MongoDB) and offer modular services to the pages that need them
2. **API**- Calls the back-end resources on the front-end and brings applications together in order to perform a designed function built around sharing data and executing pre-defined processes.
3. **Pages** - Pages are the views that the user actually sees.
4. **Components** - Components are the parts of a view that are developed individually and integrated with each of the pages to provide a complete view of the page.
5. **Web Sockets** - A protocol that allows for bi-directional, real-time transfer of data between the client and server.

Externally, we are depending on NPM packages to support our application. They help in reuse as in a library but do not exactly act as a way for us to store the data. All the data is stored in the database. Since the library does not store or transfer data (to make a data flow), they are not well suited here in these diagrams.

The database in question is MySQL, connected with Django.

There are no commercial components as of such.

Each of the services either talks to other services OR the APIs to provide services/responses based on the requests given to it. The purpose of these data items is to provide business logic on how to deal with incoming requests and to manage the transfer of data between the database and the actual views.

The services needed are described in the diagram above:

1. **Detection Module -** For all services related to the detection of number plates, recognition of number plates, fetching number plates as a string, and sending the data back to the registration module to register the vehicle in the system.
2. **Registration Module -** For all services regarding the registration of the vehicle including message passing to the detection module to export the number plate from the vehicle and send data to the database.
3. **Complain Registration Module -** For all services concerning the registration of complaints from the user including services to send the data to the surveillance module.
4. **Surveillance Module -** For all the services under the module regarding capturing of data from the RSU and CCTV (in case of real-time implementation of the system) receiving location and responding with alert signals with a location.
5. **Security Module -**  For all the services provided to all other modules including the website to provide a secure communication channel, saving data to the database in an encrypted manner, digital signatures and self signed certificates.

All these services provide an end point that with the right information provide a response. It also has built-in integration of error handling for different error codes based on the request given.

## **7.3 Communications Interfaces**

A secure communication channel used between the web portal and server will be used at first i.e. exchanging of keys between the web and server using public key encryption for exchanging, to be more specific using the Diffie-Hellman algorithm. Then for authentication between the user and web, self-signed certificates will be generated as for the authenticated web. Furthermore, the data exchange between the web and the server will be in the form of digital signatures therefore no result in the integrity of the data in the middleware i.e. message formatting in the electronic manner using digital signatures. The system will use the HTTP protocol for the data transfer.

# **8. Implementation**

Collectively, Car recovR uses 1. frontend 2 backend 3. database and OpenSource tools to develop a fully functional product. They are as follows:

## **8.1 Frontend**

### **8.1.1 ReactJS**

React is one of the most popular and widely used JavaScript libraries for building user interfaces. It has a large and active community of developers, with a wealth of resources available online. This includes documentation, tutorials, and libraries that helped us build better applications faster.



### **8.1.2 Material UI**

MUI, or Material UI provides many default components to come right out of the box that can be used on the frontend, so that much time is not wasted actually styling custom-made components.

## **8.2 Backend**

Backend refers to the server side of a web application that connects frontend and database. Its main purpose is to implement the core business logic of the entire application by serving as a gate through which the main logic for many features of the application is defined and ruled. This includes;

• Authentication & Authorization

• Database Connectivity & Organizational Structure

• Authorization Checks Over Permission For Specific Controller Routes

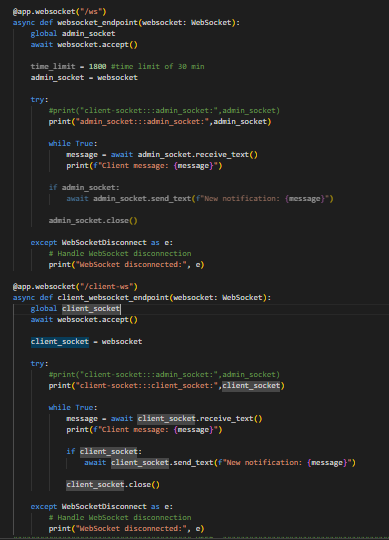
• Supports Controllers, Services, and Modules

### **8.2.1 FastAPI**

The core purpose of using Fastapi is its high performance and ease to use and the inclusion of a number of features that make it easy to build APIs quickly, such as automatic request and response validation, automatic generation of JSON schemas, and built-in support for common tasks like CORS and authentication.

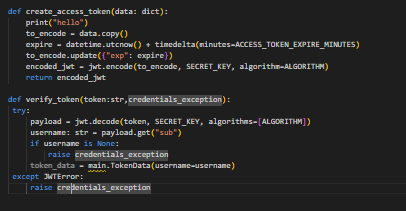
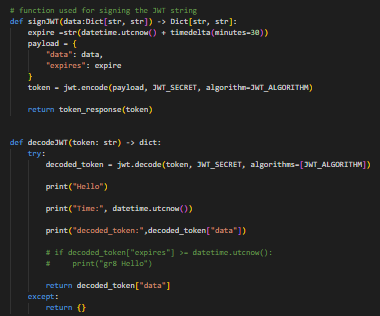
### **8.2.2 Websocket**

We have used WebSockets for real-time communication between the server side and frontend as it reduces latency and lowers overhead as compared to traditional HTTP requests, as messages are sent in a binary format rather than as text, and only headers are sent after the initial connection is established.



### **8.2.3 JWT Token**

We have used JWT Token as it is a popular way to authenticate users and provide authorization in web applications



### **8.2.4 Email Validator**

### We have used email validator to validate user input when creating or updating a user account.

### **8.3 Databases**

### 

### **8.3.1 MongoDB**

We have used MongoDB because it is a popular NoSQL document database that is used by many organizations and developers as it is scalable, flexible and easy to develop

## 

# **9. Testing & Evaluation**

## **9.1 Testing**

### **9.1.1 Purpose of Testing**

The purpose of testing a final year project is to ensure that it meets the intended requirements, functions correctly, and achieves the desired outcomes. Testing is an essential part of the project development process as it helps identify and resolve any issues, errors, or defects before the project is deployed or presented to users.

### **9.1.2 Testing Approaches**

The selection of the approach depends on factors such as project requirements, resources, time constraints, and the nature of the project.

* **Unit Testing**: This approach focuses on testing individual components or units of the project in isolation. It helps ensure that each unit functions correctly and meets the expected behavior.
* **Integration Testing**: Integration testing is performed to verify the interaction and compatibility of different components or modules of the project. It tests the interfaces and interactions between units to ensure they work together seamlessly.
* **System Testing**: System testing involves testing the entire system as a whole to verify that it meets the specified requirements.

## **9.2 Test Cases**

| TEST CASE ID: 1  DESCRIPTION: Testing the SignUp Page | | | | | |
| --- | --- | --- | --- | --- | --- |
| **No.s** | **Test Cases** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1. | Check if user signup successfully | Username,  Email, Password, Confirm Password | User signup successfully | Sign Up successfully | Pass |
| 2. | Validate each input field individually for proper validation. Entering invalid characters, exceeding the maximum character limit. | Username, Email, Password, Confirm Password | Appropriate error messages are displayed for each field with invalid input. | Invalid Input | Fail |
| 3. | Enter different values in the password and confirmation password fields | Password, Confirm Password | Displays an error message or visual indication to the user. | The confirmation password field is case-sensitive | Fail |

| TEST CASE ID: 2  DESCRIPTION: Testing the Login Page | | | | | |
| --- | --- | --- | --- | --- | --- |
| **No.s** | **Test Cases** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1. | Enter valid login credentials (username/email and password) | Email, Password | System successfully logs the user in and redirects them to the appropriate page/dashboard. | Login Successfully | Pass |
| 2. | Submit the login form with empty email and password fields. | Email, Password | Displays appropriate error messages indicating that both fields are mandatory. | Invalid Input | Fail |

| TEST CASE ID: 3  DESCRIPTION: Testing the Admin Dashboard Page | | | | | |
| --- | --- | --- | --- | --- | --- |
| **No.s** | **Test Cases** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1. | Verify that all the widgets or modules present on the dashboard page are displayed correctly. | Widgets like Navigation bar, Menu bar | Widgets are working properly | Widgets are align and properly working | Pass |
| 2. | Test the navigation within the dashboard page. | Links, Menus | Redirect the user to the corresponding pages within the application. | Redirect to pages successfully | Pass |

| TEST CASE ID: 4  DESCRIPTION: Testing the User Dashboard Page | | | | | |
| --- | --- | --- | --- | --- | --- |
| **No.s** | **Test Cases** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1. | Enter valid registration information regarding stolen cars. | Owner email, Owner CNIC, Car VIN, Car model, Car image | System successfully registered the car. | Car registered successfully | Pass |
| 2. | Check if data already fills upon hitting the complaint button. | Complaint button | Complain file successfully | Your complaint register successfully | Pass |
| 3. | Test the navigation within the dashboard page. | Links, Menus | Redirect the user to the corresponding pages within the application. | Redirect to pages successfully | Pass |

# 

# 

# 

# 

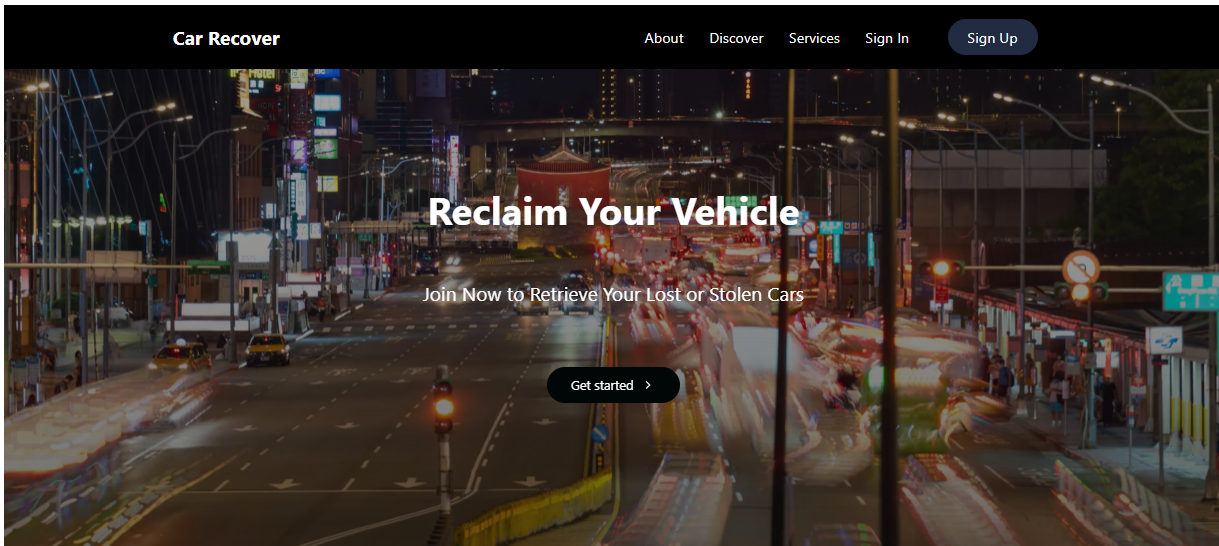
# 

# 

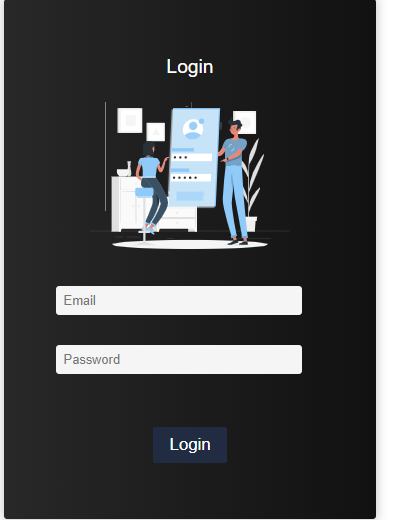
# 

# **10. User Interfaces**

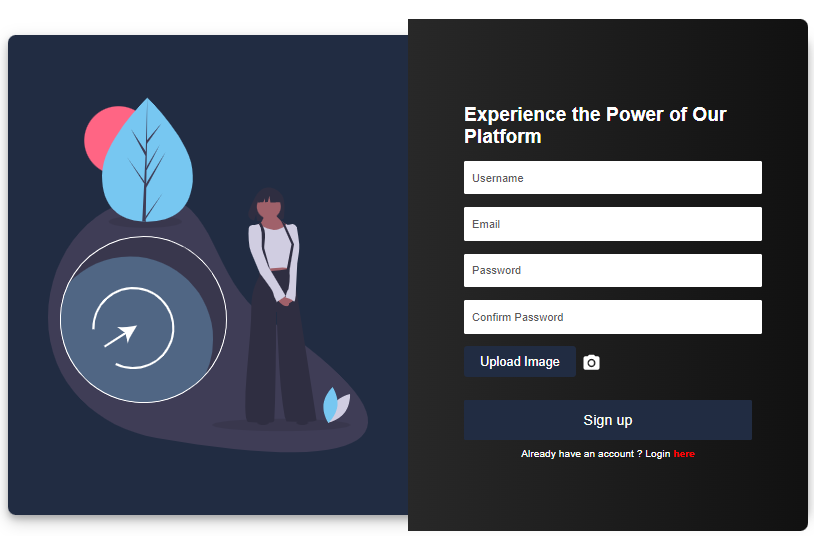
Home page



Login page

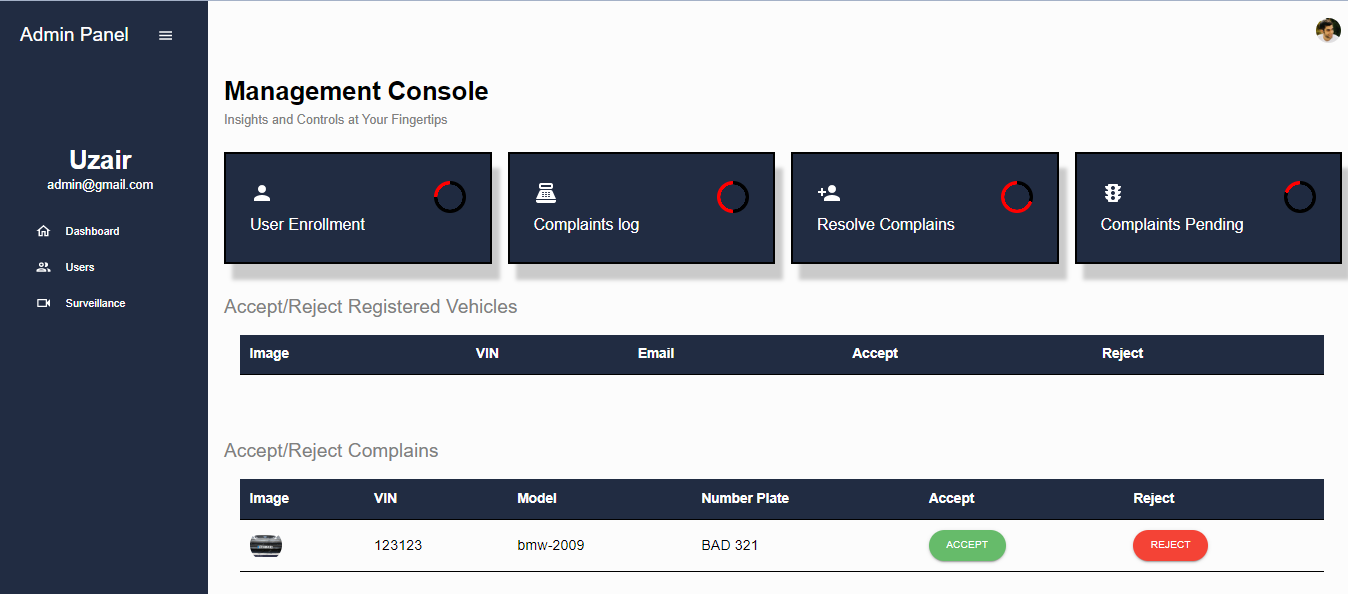


Sign Up UI

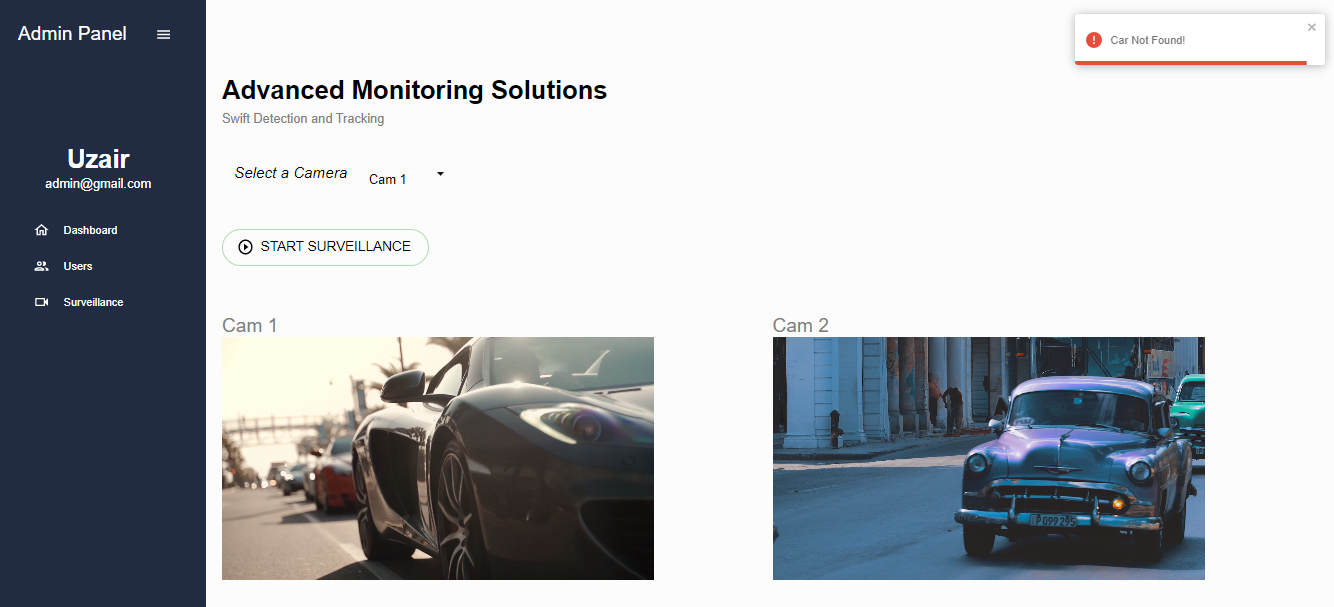


**Admin**

Admin Dashboard

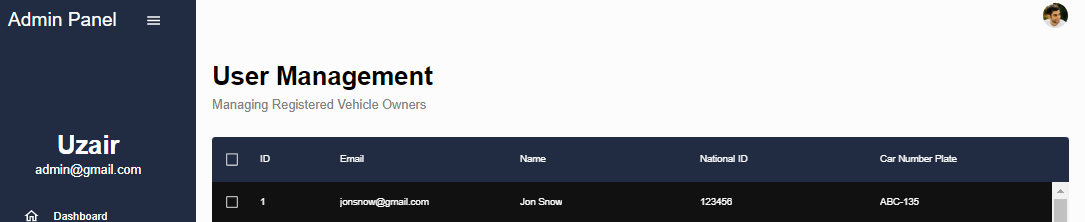


Surveillance





User Management



# 

# 

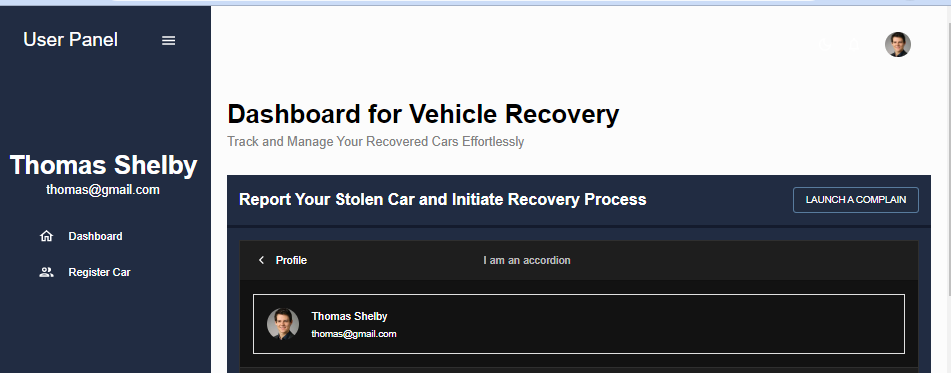
# 

# 

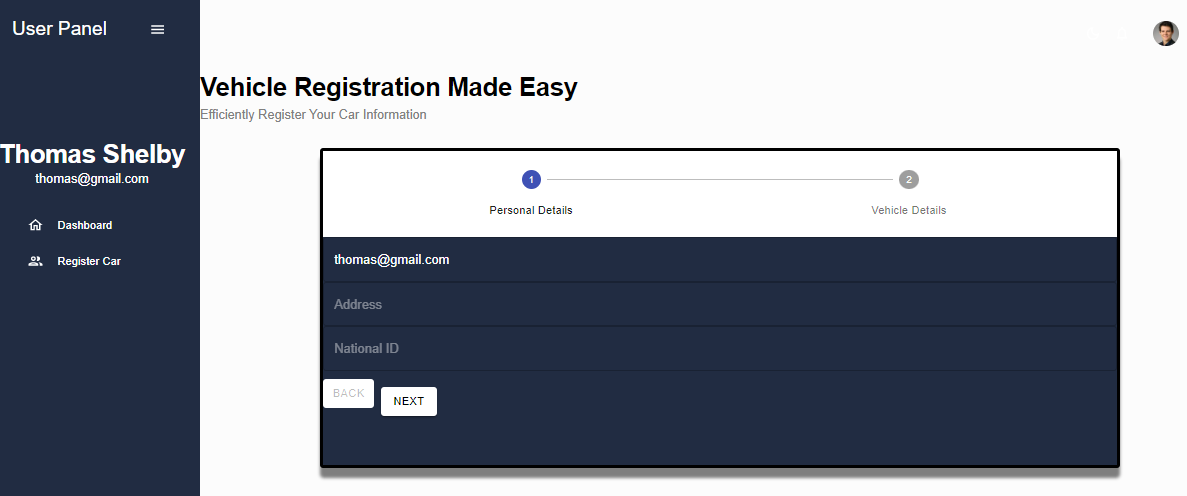
# 

# **USER**

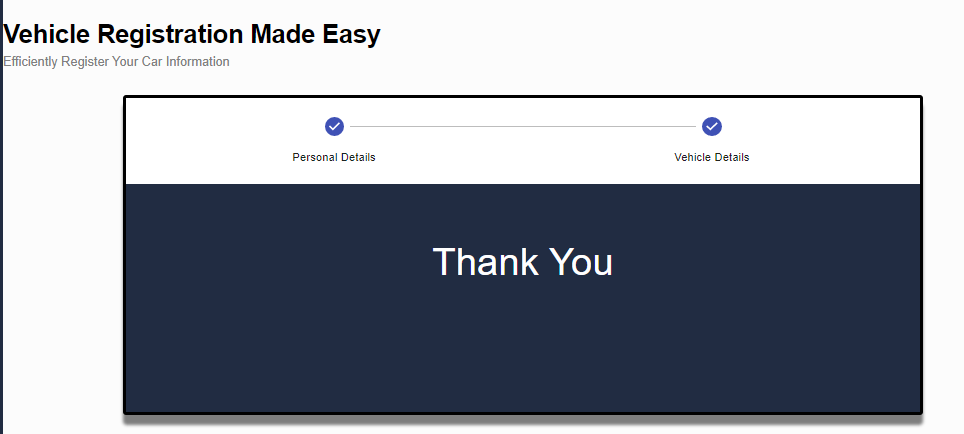
# User dashboard



Car registration



# 



# **Conclusion**

In conclusion, the development of the car recovery portal i.e. **Car recovR** was a complex and challenging project that required a comprehensive approach to software engineering. The project aimed to create a secure and efficient web-based portal that facilitates the process of recovering stolen cars, with two roles: an admin module and a user. The system was designed to ensure data security and confidentiality, with user authentication and authorization implemented to ensure that data could only be accessed by authorized users.

The process model for the project was based on the Agile Scrum framework, which provided a flexible and iterative approach to software development. While there were some challenges that made it difficult to completely adhere to the Scrum framework, the project was ultimately a success, and the car recovery portal was delivered on time and met the requirements of all stakeholders. The system has the potential to greatly improve the efficiency of car recovery processes, and it is hoped that it will prove useful in reducing car theft and increasing the recovery rate of stolen vehicles.

Overall, the project has been a valuable learning experience for all team members, allowing us to gain practical experience in software engineering, software testing and project management.

# **References**

Anshuman Dash, Satyajit Pal, and Chinmay Hegde (2018). Machine Learning-Based Ransomware Auto-Detection in IoT Devices.

R. Kumar, X. Zhang, W. Wang, R. U. Khan, J. Kumar, and A. Sharif, “A Multimodal Malware Detection Technique for Android IoT Devices Using Various Features,” IEEE Access, vol. 7, no. 6, pp. 64411-64430, 2019, Doi: 10.1109/ACCESS.2019.2916886.

H. Alasmary and colleagues, “Analyzing and Detecting Emerging Internet of Things Malware: A Graph-Based Approach” IEEE Internet of Things Journal, vol. 6, no. 5, pp. 8977-8988, October 2019, Doi: 10.1109/JIOT.2019.2925929.