

ReportSafer

Online Portal for Reporting and Filing Crimes

Mini Project Report

Submitted by

Midhu J H

Reg. No.: AJC19MCA-I039

In Partial fulfillment for the Award of the Degree of

INTEGRATED MASTER OF COMPUTER APPLICATIONS

(INMCA)

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY



AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC with 'A' grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2023-2024

DEPARTMENT OF COMPUTER APPLICATIONS
AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY



CERTIFICATE

This is to certify that the Project report, “**ReportSafer**” is the bona fide work of **MIDHU J H (Regno: AJC19MCA-I039)** in partial fulfillment of the requirements for the award of the Degree of Integrated Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2023-24.

Ms. Shelly Shiju George

Internal Guide

Ms. Meera Rose Mathew

Coordinator

Rev. Fr. Dr. Rubin Thottupurathu Jose

Head of the Department

DECLARATION

I hereby declare that the project report “**REPORTSAFER**” is a bona fide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Integrated Master of Computer Applications (INMCA) from APJ Abdul Kalam Technological University, during the academic year 2023-2024.

Date:

MIDHU J H

KANJIRAPPALLY

Reg: AJC19MCA-I039

ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our Manager **Rev. Fr. Dr. Mathew Paikatt** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department **Rev. Fr. Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **Ms. Meera Rose Mathew** for her valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Ms. Shelly Shiju George** for her inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

MIDHU J H

ABSTRACT

In an era of increasing digitalization, the "ReportSafer," an Online Portal for Reporting and Filing Crimes project seeks to revolutionize the way crimes are reported, investigated, and managed. This project introduces a comprehensive online platform designed to foster seamless communication between law enforcement agencies and the public. It not only simplifies the process of reporting crimes but also offers valuable tools for researchers and government officials to monitor crime statistics within specific regions. The system's core functionalities cater to various types of users, including administrators who oversee user accounts and data management, registered users who can submit crime reports, law enforcement personnel who review and investigate these reports, control room staff responsible for handling emergency responses, and prison wardens who manage inmates and report incidents. By embracing advanced technologies and streamlining manual processes, the proposed system promises increased reliability, reduced manual effort, and a user-friendly interface, ultimately enhancing the overall efficiency of the criminal filing system.

CONTENT

SL. NO	TOPIC	PAGE NO
1	INTRODUCTION	1
1.1	PROJECT OVERVIEW	2
1.2	PROJECT SPECIFICATION	2
2	SYSTEM STUDY	4
2.1	INTRODUCTION	5
2.2	EXISTING SYSTEM	5
2.3	DRAWBACKS OF EXISTING SYSTEM	6
2.4	PROPOSED SYSTEM	7
2.5	ADVANTAGES OF PROPOSED SYSTEM	7
3	REQUIREMENT ANALYSIS	9
3.1	FEASIBILITY STUDY	10
3.1.1	ECONOMICAL FEASIBILITY	10
3.1.2	TECHNICAL FEASIBILITY	10
3.1.3	BEHAVIORAL FEASIBILITY	11
3.1.4	FEASIBILITY STUDY QUESTIONNAIRE	11
3.2	SYSTEM SPECIFICATION	15
3.2.1	HARDWARE SPECIFICATION	15
3.2.2	SOFTWARE SPECIFICATION	15
3.3	SOFTWARE DESCRIPTION	15
3.3.1	HTML	15
3.3.2	CSS	15
3.3.3	DJANGO	16
3.3.4	SQLITE	16
3.3.5	JQUERY	16
3.3.6	JAVASCRIPT	16
4	SYSTEM DESIGN	17
4.1	INTRODUCTION	18
4.2	UML DIAGRAM	18
4.2.1	USE CASE DIAGRAM	19
4.2.2	SEQUENCE DIAGRAM	20
4.2.3	STATE CHART DIAGRAM	23

4.2.4	ACTIVITY DIAGRAM	23
4.2.5	CLASS DIAGRAM	25
4.2.6	OBJECT DIAGRAM	26
4.2.7	COMPONENT DIAGRAM	27
4.2.8	DEPLOYMENT DIAGRAM	28
4.3	USER INTERFACE DESIGN USING FIGMA	29
4.4	DATABASE DESIGN	32
5	SYSTEM TESTING	37
5.1	INTRODUCTION	38
5.2	TEST PLAN	38
5.2.1	UNIT TESTING	39
5.2.2	INTEGRATION TESTING	39
5.2.3	VALIDATION TESTING	40
5.2.4	USER ACCEPTANCE TESTING	40
5.2.5	AUTOMATION TESTING	40
5.2.6	SELENIUM TESTING	41
6	IMPLEMENTATION	53
6.1	INTRODUCTION	54
6.2	IMPLEMENTATION PROCEDURE	54
6.2.1	USER TRAINING	55
6.2.2	TRAINING ON APPLICATION SOFTWARE	55
6.2.3	SYSTEM MAINTENANCE	56
7	CONCLUSION & FUTURE SCOPE	57
7.1	CONCLUSION	58
7.2	FUTURE SCOPE	58
8	BIBLIOGRAPHY	59
9	APPENDIX	61
9.1	SAMPLE CODE	62
9.2	SCREEN SHOTS	87

List of Abbreviation

HTML - Hyper Text Markup Language

CSS - Cascading Style Sheet

SQLite - Structured Query Language Lite

UML - Unified Modelling Language

MVC - Model-View-Controller

MVT - Model-View-Template

DRY - Don't Repeat Yourself

JS - JavaScript

AJAX - Asynchronous JavaScript and XML Environment

DOM - Document Object Model

DBMS - Database Management System

PY - Python

1NF - First Normal Form

2NF - Second Normal Form

3NF - Third Normal Form

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

"ReportSafer," an Online Portal for Reporting and Filing Crimes is a cutting-edge project that aims to transform the process of reporting and managing crimes. It introduces a user-friendly and transparent online platform for both the public and law enforcement agencies, fostering efficient communication and accountability. By streamlining crime reporting, providing legal support information, and enabling online fine payments, this project enhances the efficiency of the criminal filing system and reduces opportunities for corruption. The system's functionalities cater to various user types, from administrators overseeing data management to registered users filing crime reports, and from law enforcement officers investigating cases to control room staff managing emergency responses. By leveraging advanced technologies and analytical tools, this project promises to improve the safety and justice of society while promoting a user-centric approach to crime management.

1.2 PROJECT SPECIFICATION

ReportSafer, an Online Portal for Reporting and Filing Crimes is a web-based platform with the following specifications:

- **User Authentication**

The system allows users to create an account with a unique username and password. Users must be authenticated before accessing the system.

- **Crime Reporting**

Registered users can submit crime reports through the online portal. They can provide details about the incident, location, date, time, and any other relevant information.

- **Crime Report Review and Investigation**

Law enforcement personnel have access to the portal to review and investigate crime reports submitted by registered users. They can prioritize and assign cases for further action.

- **Evidence Collection and Documentation**

Law enforcement officers can collect additional information, evidence, and documentation related to reported crimes through the portal. This may include photographs, videos, witness statements, and other relevant data.

- **Incident Reporting**

In the event of any criminal activities or incidents occurring within the prison, the prison warden may use the portal to report such incidents to the appropriate authorities or law enforcement.

- **Case Management**

Law enforcement can manage and update the status of reported cases within the portal, keeping track of the investigation's progress and any actions taken.

- **Reporting to Higher Authorities (Control room staff)**

Law enforcement can generate and submit comprehensive reports to higher authorities or agencies as required for official records and further actions.

- **System Configuration**

Admins have the authority to configure and customize the portal's settings.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

The portal facilitates an array of essential functionalities that empower investigators and users alike in the process of reporting and managing crimes. Investigators are provided with the flexibility to seamlessly modify case details, document their investigation findings, and closely monitor the status of each case. Users, on the other hand, are offered a user-friendly platform to initiate the reporting of criminal cases, further streamlining the process by enabling online fine payments. Additionally, this dynamic portal introduces the convenience of scheduling appointments with jailors, centralizing crucial interactions and enhancing the overall efficiency of the criminal filing system.

This comprehensive system goes beyond simply bridging the gap between the public and law enforcement agencies; it is a pivotal step towards a more modern, accountable, and transparent approach to crime management. Users can effortlessly submit intricate crime reports, keeping tabs on the progress of their cases, and access vital legal support information within a secure and efficient online environment. Additionally, the project equips law enforcement with powerful analytical tools, enabling in-depth crime trend analysis, which aids in proactive resource allocation and enhances the effectiveness of crime prevention strategies. The incorporation of a chatbot for user assistance further elevates the portal's accessibility, making it a holistic solution that holds the potential to revolutionize society's approach to crime reporting, investigation, and management.

This innovative project aims to leverage technology to foster a safer and more just society. By offering a seamless and transparent crime reporting platform, it seeks to enhance the accountability and efficiency of the criminal filing system. With features such as online fine payments and legal support information, this system not only streamlines processes but also reduces opportunities for corruption. It empowers both law enforcement agencies and the public, making crime management more user-centric and data-driven, ultimately contributing to a society where crimes are reported, investigated, and managed with greater ease and fairness.

2.2 EXISTING SYSTEM

The primary focus of the existing system lies in facilitating communication between law enforcement and the public, but it lacks provisions for legal support, such as contact information for advocates. In the traditional way of reporting crimes, individuals often relied on phone calls or physically visiting a police station to report incidents, which could be time-consuming and sometimes challenging, especially during non-business hours. These paper-based procedures not

only consumed valuable time but also led to the risk of errors and inefficiencies in processing and managing the information.

2.2.1 NATURAL SYSTEM STUDIED

The natural system studied encompasses the complex dynamics of criminal incidents within a given region or area. It delves into the intricate interplay of various elements, including the nature of reported crimes, their geographic distribution, temporal patterns, and the impact of socio-economic factors. Through an examination of these natural systems, the project seeks to not only facilitate the reporting and investigation of crimes but also contribute to a deeper understanding of criminal behaviors and trends. This comprehensive insight into the natural system enables more effective resource allocation for law enforcement agencies and promotes proactive crime prevention strategies, aligning the project with the broader goal of fostering a safer and more secure society.

2.2.2 DESIGNED SYSTEM STUDIED

The focus is on the innovative platform created to revolutionize crime reporting, investigation, and management. This system has been meticulously designed to ensure optimal efficiency and effectiveness. It includes features that allow investigators to seamlessly modify case details, document findings, and track case statuses. Additionally, it provides registered users with an intuitive interface for reporting crimes and making fine payments online, enhancing transparency and accountability in the process. The integration of advanced analytical tools empowers law enforcement agencies to study crime trends and patterns, thereby enabling more data-driven decision-making and resource allocation for crime prevention. The designed system's chatbot feature enhances user accessibility and support, making it a comprehensive solution poised to transform the way society approaches crime management.

2.3 DRAWBACKS OF EXISTING SYSTEM

- **Limited User Access:** The existing system primarily caters to communication between law enforcement agencies and the public, but it lacks the provisions to allow users to gain insights into the progress of their reported cases.
- **Lack of Legal Support Information:** The system does not provide users with crucial legal support information or contact details for advocates, limiting their ability to navigate the legal aspects of their cases.
- **Manual and Time-Consuming Processes:** The existing system relies on manual processes, leading to inefficiencies in reporting, investigation, and management. This manual work can

be time-consuming and prone to errors.

- **Lack of User-Centric Features:** The existing system does not prioritize user-centric features, making it less convenient and user-friendly for individuals reporting crimes.

2.4 PROPOSED SYSTEM

The proposed system represents a significant advancement over the existing system, with a core objective of reducing manual efforts and promoting efficiency. By automating various processes, it streamlines the entire crime reporting and management workflow, significantly minimizing the need for manual data entry and processing. This not only expedites the handling of cases but also decreases the likelihood of errors, ensuring that critical information is accurately recorded and readily accessible to authorized personnel.

In addition to its focus on automation, the proposed system incorporates measures to enhance transparency and reduce opportunities for corruption. For instance, by enabling online reporting of crimes, the system reduces the chances of intermediary corruption as the fines for petty offenses can be paid directly through a secure payment gateway on the website. This not only ensures that the funds are channeled directly to the government but also mitigates the potential for bribery or manipulation within the law enforcement department. This innovative approach aligns with the broader objective of instilling public trust and accountability in the justice system.

Furthermore, the proposed system acknowledges the importance of providing legal support to users involved in criminal cases. To this end, it includes a dedicated feature that offers access to contact details of legal advocates. This feature not only empowers users with valuable resources but also bolsters their confidence in navigating the legal aspects of their cases. It is a testament to the system's commitment to delivering user-centric solutions that facilitate a fair and just experience for all stakeholders involved in the crime reporting and management process.

2.5 ADVANTAGES OF PROPOSED SYSTEM

- **Reduced Manual Work:** The proposed system streamlines the entire crime reporting and management process, significantly reducing the need for manual data entry and processing. This not only saves time but also minimizes the likelihood of errors.
- **Enhanced Transparency:** By enabling users to track the progress of their reported cases and providing access to case updates, the system enhances transparency. Users can have greater confidence in the handling of their cases, promoting trust in the justice system.
- **Legal Support Feature:** The incorporation of a legal support feature, offering access to

contact details of legal advocates, empowers users to navigate the legal aspects of their cases effectively. This feature ensures that users have the necessary resources to address legal issues associated with their reports.

- **Efficient Case Handling:** With automated processes and access to a comprehensive platform, the proposed system expedites case handling, allowing law enforcement agencies to prioritize and manage cases more efficiently.
- **User-Centric Solutions:** The proposed system is designed with a user-centric approach, focusing on providing a user-friendly interface and convenient features that make the reporting and management of crimes more accessible and efficient for all stakeholders.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

A feasibility study is a preliminary exploration of a proposed project or undertaking to determine its merits and viability. A feasibility study aims to provide an independent assessment that examines all aspects of a proposed project, including technical, economic, financial, legal, and environmental considerations. Sometimes called a feasibility analysis or feasibility report—is a way to evaluate whether a project plan could be successful. A feasibility study evaluates the practicality of your project plan to judge whether you can move forward with the project.

3.1.1 ECONOMIC FEASIBILITY

An economic feasibility study plays a pivotal role in assessing the cost-effectiveness of a project, particularly in the context of introducing a new system like the one proposed here. In this case, the advantages of the proposed system over the existing manual process are evident. The new system not only streamlines the workflow, reducing the extensive paperwork associated with traditional methods but also significantly saves time, both for users and law enforcement agencies. Notably, one of the substantial cost-saving aspects of the proposed system is the reduction in paper-related expenses and the potential for office space and storage cost reductions.

Moreover, the economic feasibility study highlights that law enforcement agencies need to allocate resources primarily for the initial software implementation cost, which can be a cost-effective investment in the long run. This stands in contrast to the high expenses associated with developing and maintaining packaged hardware and software in today's technology landscape. By providing a comprehensive assessment, the feasibility study demonstrates the financial viability of the project, making a compelling case for the adoption of the new system as a cost-effective and efficient alternative to the existing manual processes.

3.1.2 TECHNICAL FEASIBILITY

The foundation of this study lies in the comprehensive design of system requirements, serving as a litmus test to gauge the developer's technical acumen and capability to successfully execute the proposed project. In this context, it's crucial to recognize that the project represents a forward-thinking web-based solution, offering the flexibility for users to access the system remotely, fostering accessibility and convenience. The selection of web technologies for system development, including Python Django, MySQL, JavaScript, and Bootstrap, underscores the project's commitment to staying aligned with industry standards and contemporary web

development practices. These technologies not only facilitate robust web application development but also offer cost-efficiency, as they are open-source and widely supported, providing developers with access to a vast community of resources and expertise. Additionally, the inclusion of machine learning tools within the project framework indicates a commitment to harnessing cutting-edge technology to improve the overall efficacy of the proposed system. Overall, the integration of these technologies is a strategic move, demonstrating the feasibility of the project from a technical standpoint while leveraging cost-effective and powerful tools for its successful implementation.

3.1.3 BEHAVIORAL FEASIBILITY

The study examines the readiness and willingness of the end-users, both within law enforcement agencies and the public, to embrace and effectively utilize the proposed online portal for reporting and filing crimes. This evaluation considers the human element of the project, focusing on the users' receptiveness to the new system, their ability to adapt to the technology, and any potential resistance to change. It also assesses the impact of the system on user behavior, addressing issues related to ease of use, training requirements, and the perceived benefits of the system. Understanding the behavioral aspects is essential, as it directly influences the successful adoption and integration of the proposed solution into the daily routines of all involved parties.

3.1.4 FEASIBILITY STUDY QUESTIONNAIRE

1. Project Overview?

The "ReportSafer", Online Portal for Reporting and Filing Crimes is an initiative aimed at enhancing public safety and streamlining the process of reporting and filing crimes. The portal will provide a secure and user-friendly platform for individuals to report various types of crimes to law enforcement agencies. This project is aiming to develop an online portal for law-enforcement officials and common man to report crimes which will act as a communication platform. It will be useful for researchers and other government officials to track the count of crimes in a particular area.

2. To what extend the system is proposed for?

The proposed system seeks to leverage technology to improve crime reporting and response, but it should complement existing crime reporting mechanisms and may need to integrate with other law enforcement systems to ensure seamless information sharing and collaboration. The portal is intended to support the reporting of various types of crimes, such as theft, assault, cybercrime,

property damage, and more.

3. Specify the Viewers/Public which is to be involved in the System?

- **Admin:** Admin can design the interface and manage the databases. Implements the communication with the law-enforcement, registered user, control room staff and prison warden.
- **Common Citizens:** Common citizens, as potential victims, or witnesses of crimes, can use the online portal to report criminal incidents and will provide information to law enforcement. They may access the portal to seek help or report criminal activities they have witnessed or experienced.
- **Researchers, Government Officials and Law-enforcement:** Researchers and various government officials can utilize the data collected through the portal for statistical analysis, crime trend identification, and resource allocation. This data can be valuable for formulating policies and strategies to combat crime effectively. Law-enforcement bodies can track the status and count of crimes happening in a particular area.
- **Advocates and Legal Professionals:** The proposed system will include a legal support feature that provides contact information for advocates and legal professionals. Advocates may use the portal to aid victims or accused individuals involved in criminal cases.

4. List the Modules included in your System?

- **User Management Module:**

It will allow the admin to create, update, and deactivate accounts of law-enforcement, control room staff and prison warden. The admin can assign specific roles and access privileges to different users such as Law-enforcement, Control room staff and Prison warden.

- **Crime Reporting Module:**

This module will enable registered users to submit crime reports through the online portal. Users can provide details about the incident, location, date, time, and other relevant information. Users may have options to report crimes anonymously or keep certain personal details confidential, depending on the portal's setup.

- **Case Management Module:**

This module will allow law enforcement personnel to review and investigate crime reports submitted by registered users. Law enforcement officers can prioritize and assign cases for further action. They can update the status of reported cases and keep track of the investigation's progress.

- **Payment Gateway Module:**

This module will enable users to make online payments for fines related to petty offences. The system will ensure that fines are paid directly to the government through the payment gateway,

reducing the possibility of bribery.

- **Legal Support Module:**

This module will provide contact details of advocates and legal professionals. Users can seek legal assistance and support through the portal.

- **Inmate Management Module:**

The prison warden can use this module to view information about inmates, including their records and behavior. They can report incidents within the prison to the appropriate authorities.

- **System Configuration Module:**

The admin has the authority to configure and customize the portal's settings, such as defining types of crimes that can be reported and managing notification preferences.

5. Identify the users in your project?

Admin

Registered User

Law-enforcement

Control room staff

Prison warden

6. Who owns the system?

Midhu J H, student of Amal Jyothi College of Engineering, Kanjirappally, Kottayam.

7. System is related to which firm/industry/organization?

The system is related to law-enforcement organization.

8. Details of person that you have contacted for data collection?

Mr. Jayakumar S,

Sub Inspector of Police,

Marayamuttom Police Station, Thiruvananthapuram District.

9. Questionnaire to collect details about the project?

Question 1: What is the purpose of the online portal, and what are the main objectives it aims to achieve?

The online portal aims to provide a convenient and accessible platform for citizens to report crimes,

file complaints, and seek assistance from law enforcement.

Question 2: What types of crimes can be reported through the online portal? Are there any specific categories or limitations?

You can report various types of crimes through the portal, including theft, vandalism, assault, harassment, and other non-emergency incidents. However, for emergencies or in-progress crimes, please call our emergency helpline.

Question 3: What information and details are necessary when reporting a crime through the portal?

When filing a crime report, it is essential to provide as much relevant information as possible, such as the location, date, time, a description of the incident, and any available evidence or witnesses.

Question 4: How does the portal ensure the security and confidentiality of the reported data?

Our online portal uses encryption and secure data storage to protect the confidentiality of the reported information. Access to the data is restricted to authorized personnel only.

Question 5: How does the police department handle emergency crime reports submitted through the online portal?

For emergency crime reports requiring immediate attention, it is crucial to call our emergency helpline (e.g., 911 or a specific emergency number for your area). The online portal is primarily intended for non-emergency incidents and complaints.

Question 6: Are there any plans for future upgrades or enhancements to the online portal?

Yes, we are continually looking to improve our services. Future upgrades may include enhancements to the user interface, adding new features, and integrating advanced technologies to improve crime reporting and investigation.

Question 7: Is there a feedback mechanism for users to share their opinions and reviews about their experience with the online portal?

Yes, we value user feedback. There is a feedback form available on the portal where users can share their opinions, suggestions, and any issues they may have encountered while using the platform.

Question 8: Is there chatbot feature included in the website?

No, we do not have chatbot facility yet in the website.

Question 9: Can you provide any success stories or real-life examples of how the portal has been

instrumental in solving or preventing crimes?

We have successfully solved several cases with the help of crime reports filed through the portal, leading to arrests and the recovery of stolen property. The portal has also facilitated timely interventions in critical situations.

Question 10: How is the portal maintained and updated to ensure it remains functional and secure? Our IT department regularly monitors and maintains the portal, performing updates and security patches to ensure its functionality and security are up to date.

3.1 SYSTEM SPECIFICATION

3.2.1 HARDWARE SPECIFICATION

Processor Intel core i3

RAM 4 G B

Hard disk 1 T B

3.2.2 SOFTWARE SPECIFICATION

Front End HTML, CSS

Back End Python Django

Database SQLite

Client on PC Windows 7 and above.

Technologies used - JS, HTML5, AJAX, Python Django, CSS

3.3 SOFTWARE DESCRIPTION

3.2.1 HTML

HTML (Hypertext Markup Language) is the most basic building block of the Web. HTML is a simple language made up of elements, which can be applied to pieces of text to give them different meaning in a document, structure a document into logical sections, and embed content such as images and videos into a page. In the proposed system, HTML 5 is used to carry out the design part of the webpage. HTML validations are also used.

3.3.2 CSS

While HTML is used to define the structure and semantics of your content, CSS is used to style it and lay it out. For example, you can use CSS to alter the font, colour, size, and spacing of your

content, split it into multiple columns, or add animations and other decorative features. In the proposed system CSS 3 is used.

3.3.3 DJANGO

Django is a high-level, open-source web framework written in Python. It is designed to simplify and speed up the development of web applications by providing a robust set of tools, libraries, and features. Django follows the model-view-controller (MVC) architectural pattern, which is often referred to as the model-view-template (MVT) pattern in the context of Django. Additionally, Django follows the principle of the Don't Repeat Yourself (DRY) and encourages the use of reusable components, reducing redundancy and promoting code maintainability.

3.3.4 SQLITE

SQLite is a self-contained, serverless, and zero-configuration relational database management system (RDBMS). It is often described as a "lightweight" and "embedded" database because it is designed to be simple to use and does not require a separate server process. SQLite is self-contained in the sense that it does not rely on a separate database server to operate. It is a library that can be linked into a program, allowing applications to interact with the database directly.

3.3.5 jQuery

jQuery is a lightweight, "write less, do more", JavaScript library. The purpose of jQuery is to make it much easier to use JavaScript on your website. jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code. jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation. The jQuery library contains the following features: HTML/DOM manipulation, CSS manipulation, HTML event methods, Effects and animations, AJAX.

3.3.6 JAVASCRIPT

JavaScript was initially created to "make web pages alive". The programs in this language are called scripts. They can be written right in a web page's HTML and run automatically as the page loads. In the proposed system loading screen, different popup animations are implemented using JavaScript.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

The system design phase is a pivotal juncture in the project's development journey, where the conceptual vision of the online portal for reporting and filing crimes is meticulously transformed into a tangible, functional reality. It involves the detailed architectural planning and technical blueprinting of the system, ensuring that it can effectively address the multifaceted challenges faced by law enforcement agencies and the public. This phase not only maps out the framework of the proposed system but also sets the stage for the implementation of features and functionalities that streamline the crime reporting and management process, enhance transparency, and promote operational efficiency. The design decisions, encompassing technology choices, data management strategies, user interfaces, and security measures, are carefully analyzed, reflecting a user-centric approach that aims to reduce manual work, improve data accessibility, and contribute to a safer and more just society. The system design phase acts as a vital link between the project's conceptualization and realization, underscoring how the proposed solution aspires to revolutionize the landscape of crime management and reporting.

4.2 UML DIAGRAM

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997. UML stands for Unified Modeling Language. UML is different from the other common programming languages such as C++, Java, COBOL, etc. UML is a pictorial language used to make software blueprints. UML can be described as a general-purpose visual modeling language to visualize, specify, construct, and document software system. Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object-oriented analysis and design. After some standardization, UML has become an OMG standard. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system. The visual effect of the UML diagram is the most important part of the entire process. UML includes the following nine diagrams.

- Class diagram
- Object diagram
- Use case diagram

- Sequence diagram
- Activity diagram
- State chart diagram
- Deployment diagram
- Component diagram

4.2.1 USE CASE DIAGRAM

A use case diagram is a visual representation of the interactions between system components. An approach for identifying, outlining, and organizing system requirements is called a use case. The word "system" here refers to a thing that is being created or run, like a website for mail-order product sales and services. UML (Unified Modeling Language), a standard language for the modelling of real-world objects and systems, uses use case diagrams. The planning of general requirements, the validation of a hardware design, the testing and debugging of a software product in development, the creation of an online help reference, or the completion of a job focused on customer support are all examples of system objectives. For instance, use cases in a product sales context can involve customer service, item ordering, catalogue updating, and payment processing. There are four elements in a use case diagram.

- The boundary, which defines the system of interest in relation to the world around it.
- The actors, usually individuals involved with the system defined according to their roles.
- The use cases, which are the specific roles are played by the actors within and around the system.
- The relationships between and among the actors and the use cases. Use case diagrams are drawn to capture the functional requirements of a system. After identifying the above items, we must use the following guidelines to draw an efficient use case diagram.
- The name of a use case is very important. The name should be chosen in such a way so that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships, as the main purpose of the diagram is to identify the requirements.
- Use notes whenever required to clarify some important points.

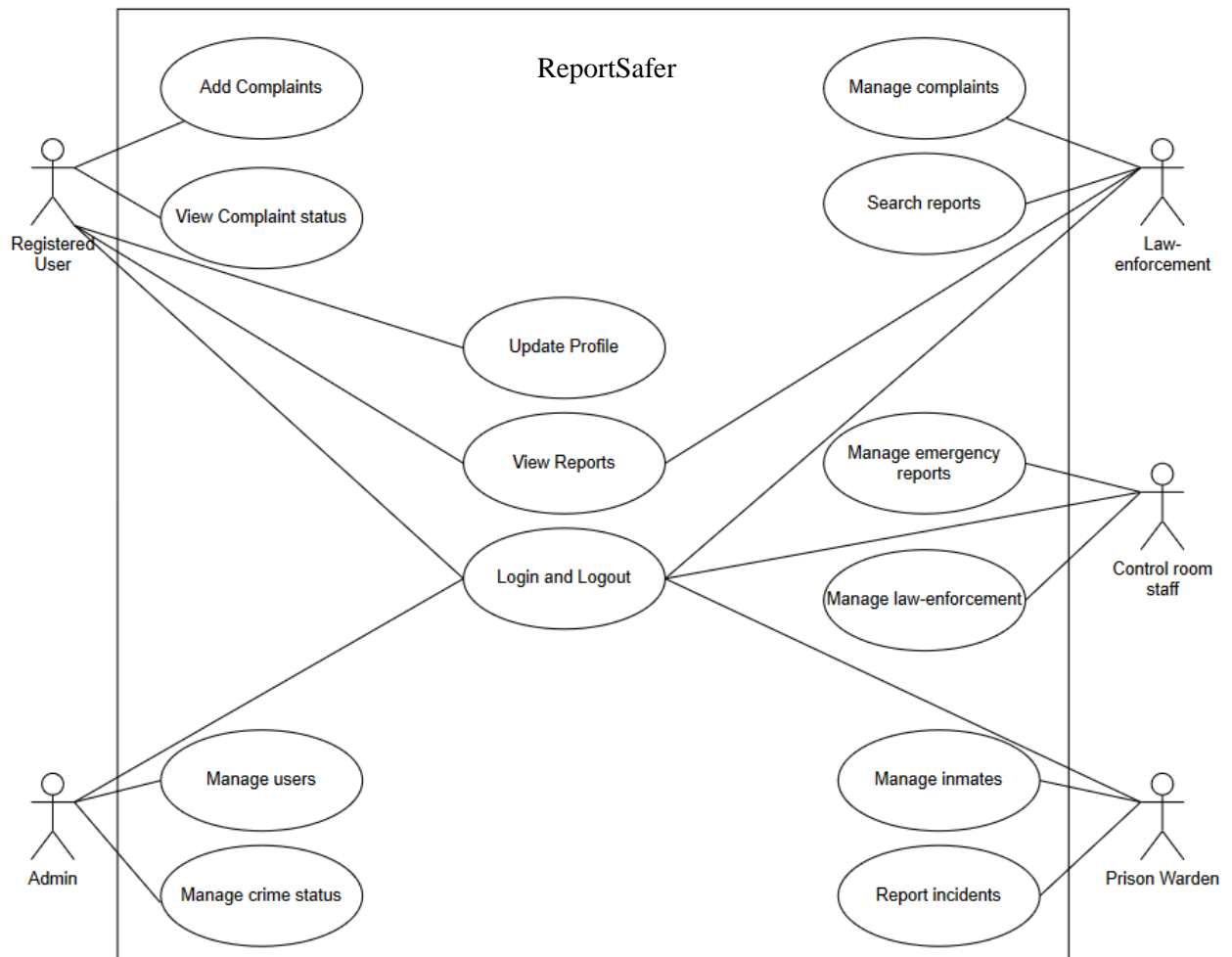


Figure 1: Use case diagram

4.2.2 SEQUENCE DIAGRAM

A sequence diagram essentially shows how things interact with one another sequentially, or the order in which these interactions occur. A sequence diagram can also be referred to as event diagrams or event scenarios. Sequence diagrams show the actions taken by the components of a system in chronological order. Business people and software engineers frequently use these diagrams to record and comprehend the requirements for new and current systems. Sequence Diagram Notations –

i. Actors – In a UML diagram, an actor represents a particular kind of role that interacts with the system and its objects. An actor is always beyond the purview of the system that we want to use the UML diagram to represent. We employ actors to portray a variety of roles, including those of human users and other outside subjects. In a UML diagram, an actor is represented using a stick person notation. In a sequence diagram, there might be several actors.

ii. Lifelines – A lifeline is a named element in a sequence diagram that represents an individual participant. So, in a sequence diagram, each incident is represented by a lifeline. A sequence diagram's lifeline elements are at the top.

iii. Messages – Messages are used to show how objects communicate with one another. The messages are displayed on the lifeline in chronological sequence. Arrows are how messages are represented. A sequence diagram's main components are lifelines and messages. Messages can be broadly classified into the following categories:

- Synchronous messages
- Asynchronous Messages
- Create message
- Delete Message
- Self-Message
- Reply Message
- Found Message
- Lost Message

iv. Guards – To model conditions we use guards in UML. They are used when we need to restrict the flow of messages on the pretext of a condition being met. Guards play an important role in letting software developers know the constraints attached to a system or a particular process.

Uses of sequence diagrams –

- Used to model and visualize the logic behind a sophisticated function, operation or procedure.
- They are also used to show details of UML use case diagrams.
- Used to understand the detailed functionality of current or future systems.
- Visualize how messages and tasks move between objects or components in a system.

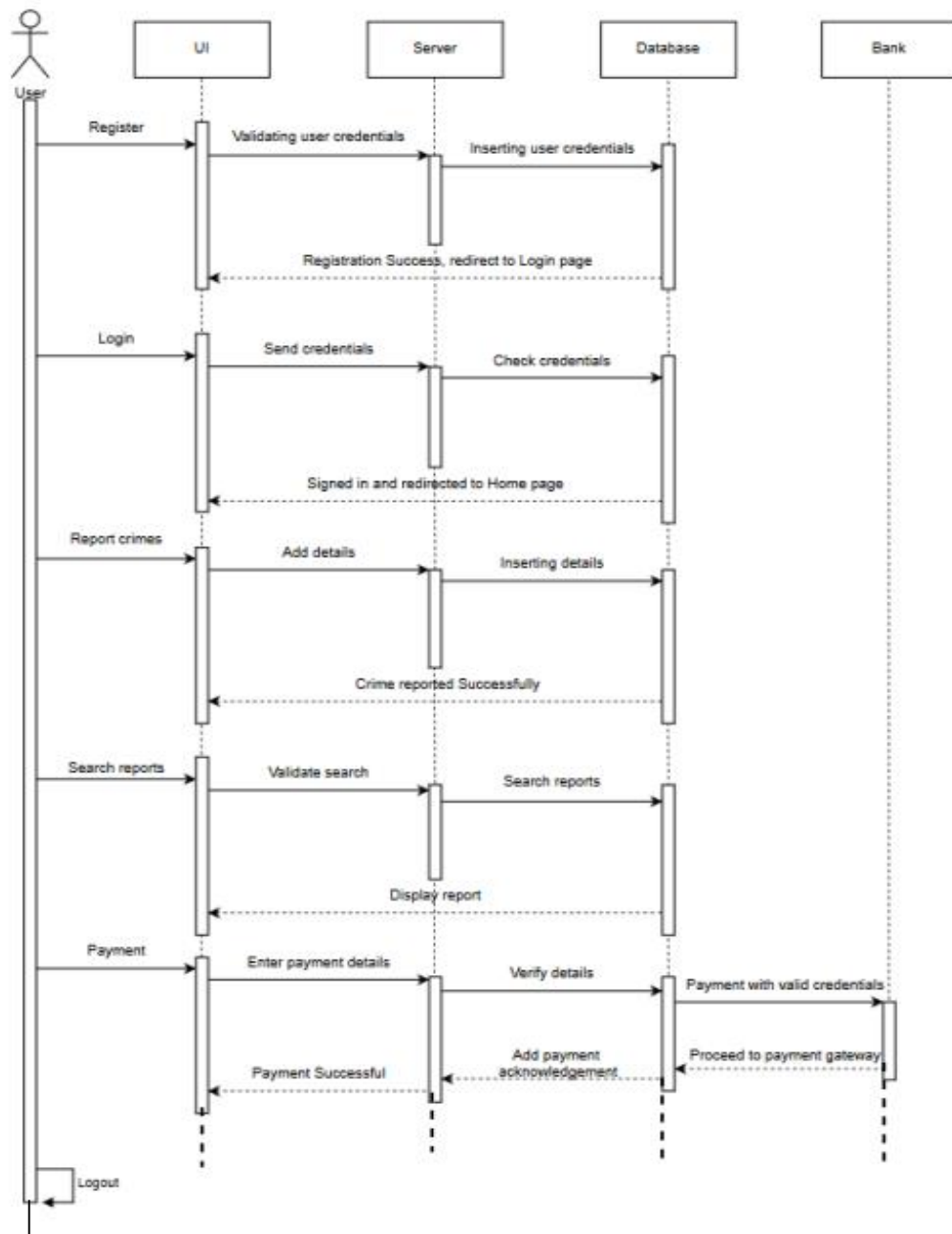


Figure 2: Sequence diagram

4.2.3 STATE CHART DIAGRAM

A state diagram, also known as a state machine diagram or state chart diagram, is an illustration of the states an object can attain as well as the transitions between those states in the Unified Modeling Language (UML). In this context, a state defines a stage in the evolution or behavior of an object, which is a specific entity in a program or the unit of code representing that entity. A state diagram resembles a flowchart in nature; however, a flowchart shows the processes within a system that alters the state of an object rather than the actual state changes themselves. The first step to creating a state chart diagram is identifying the initial and final states of a system. Then, all the possible existing states are placed in relation to the beginning and the end. Lastly, all of the events that trigger state changes are labelled as transition elements.

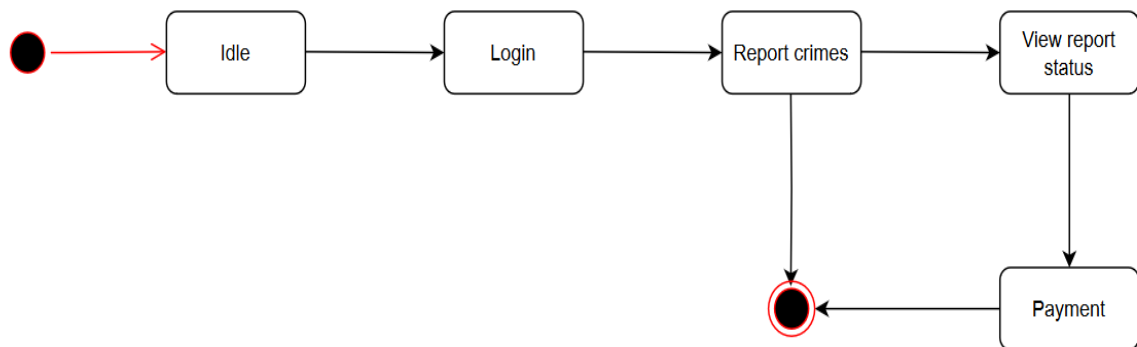


Figure 3: State chart diagram

4.2.4 ACTIVITY DIAGRAM

An activity diagram is essentially a flowchart that shows how one activity leads to another. The action might be referred to as a system operation. One operation leads to the next in the control flow. This flow may be parallel, contemporaneous, or branched. Activity diagrams use many features, such as fork, join, etc., to cope with all types of flow control. An activity diagram is a behavioral diagram i.e., it depicts the behavior of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

Activity diagram is another important behavioral diagram in UML diagram to describe dynamic aspects of the system. Activity diagram is essentially an advanced version of flow chart that modeling the flow from one activity to another activity.

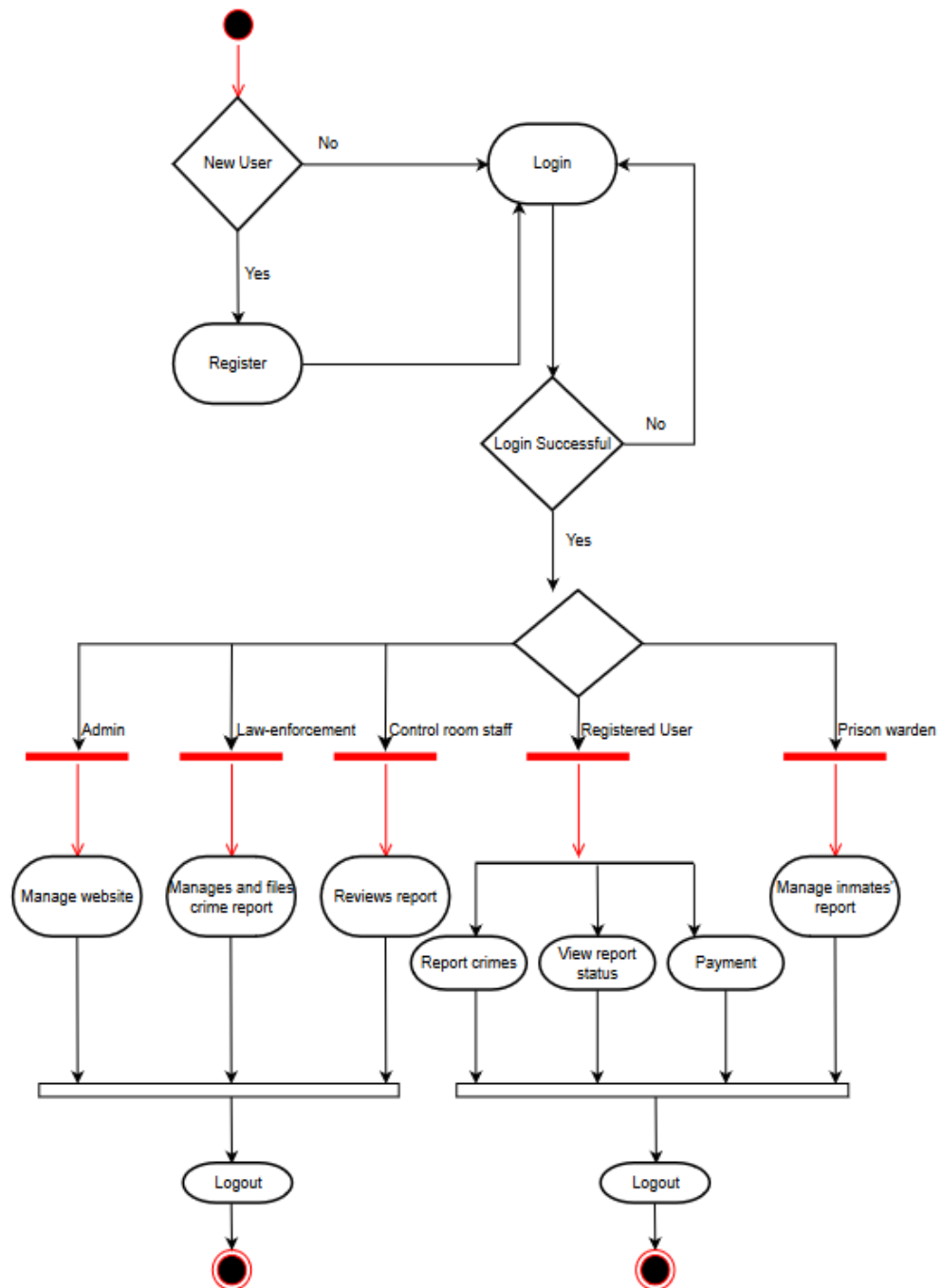


Figure 4: Activity diagram

4.2.5 CLASS DIAGRAM

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling, translating the models into programming code. Class diagrams can also be used for data modeling. Class diagrams are the blueprints of your system or subsystem. You can use class diagrams to model the objects that make up the system, to display the relationships between the objects, and to describe what those objects do and the services that they provide. Class diagrams are useful in many stages of system design. In the analysis stage, a class diagram can help you to understand the requirements of your problem domain and to identify its components. In an object-oriented software project, the class diagrams that you create during the early stages of the project contain classes that often translate into actual software classes and objects when you write code. Later, you can refine your earlier analysis and conceptual models into class diagrams that show the specific parts of your system, user interfaces, logical implementations, and so on. Your class diagrams then become a snapshot that describes exactly how your system works, the relationships between system components at many levels, and how you plan to implement those components.

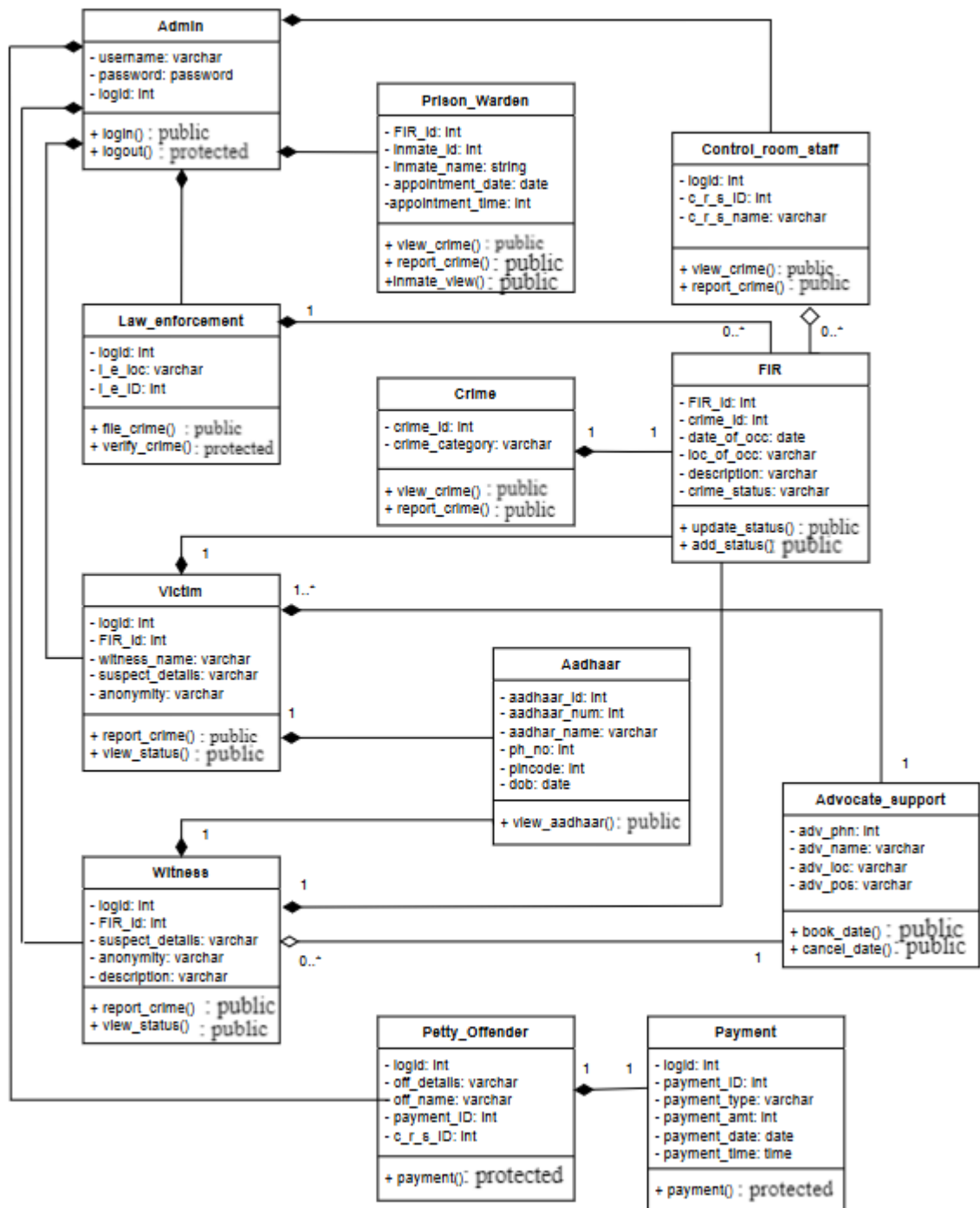


Figure 5: Class diagram

4.2.6 OBJECT DIAGRAM

Since class diagrams are the source of object diagrams, class diagrams are a prerequisite for object diagrams. An instance of a class diagram is represented by an object diagram. Class and object diagrams both use the same fundamental ideas. The static view of a system is also represented by object diagrams, but this static view represents a momentary snapshot of the system. To represent a group of items and their connections as an instance, object diagrams are employed.

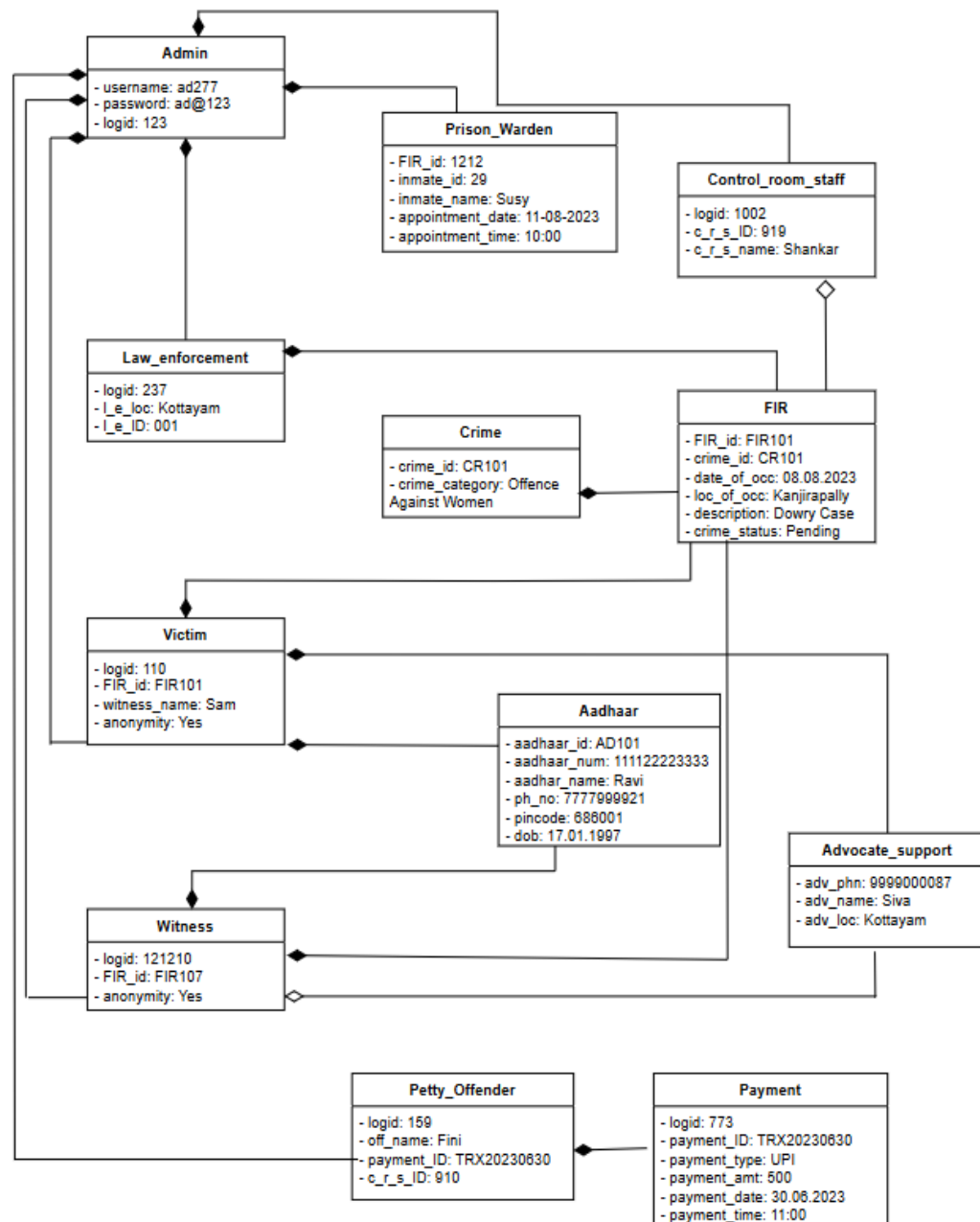


Figure 6: Object diagram

4.2.7 COMPONENT DIAGRAM

UML diagrams are used to represent and describe the behavior of object-oriented systems, assisting in visualization, explanation, and thorough documentation. Particularly in class diagrams, where they record the static behavior of a system, these diagrams play a vital role. On the other hand, graphics can impair the efficiency of real multitasking systems. In order to maintain coordination, each part within the broader process has a distinct duty and only communicates with other essential components.

A component diagram depicts how components are wired together to form larger components or software systems. They are used to illustrate the structure of arbitrarily complex systems.

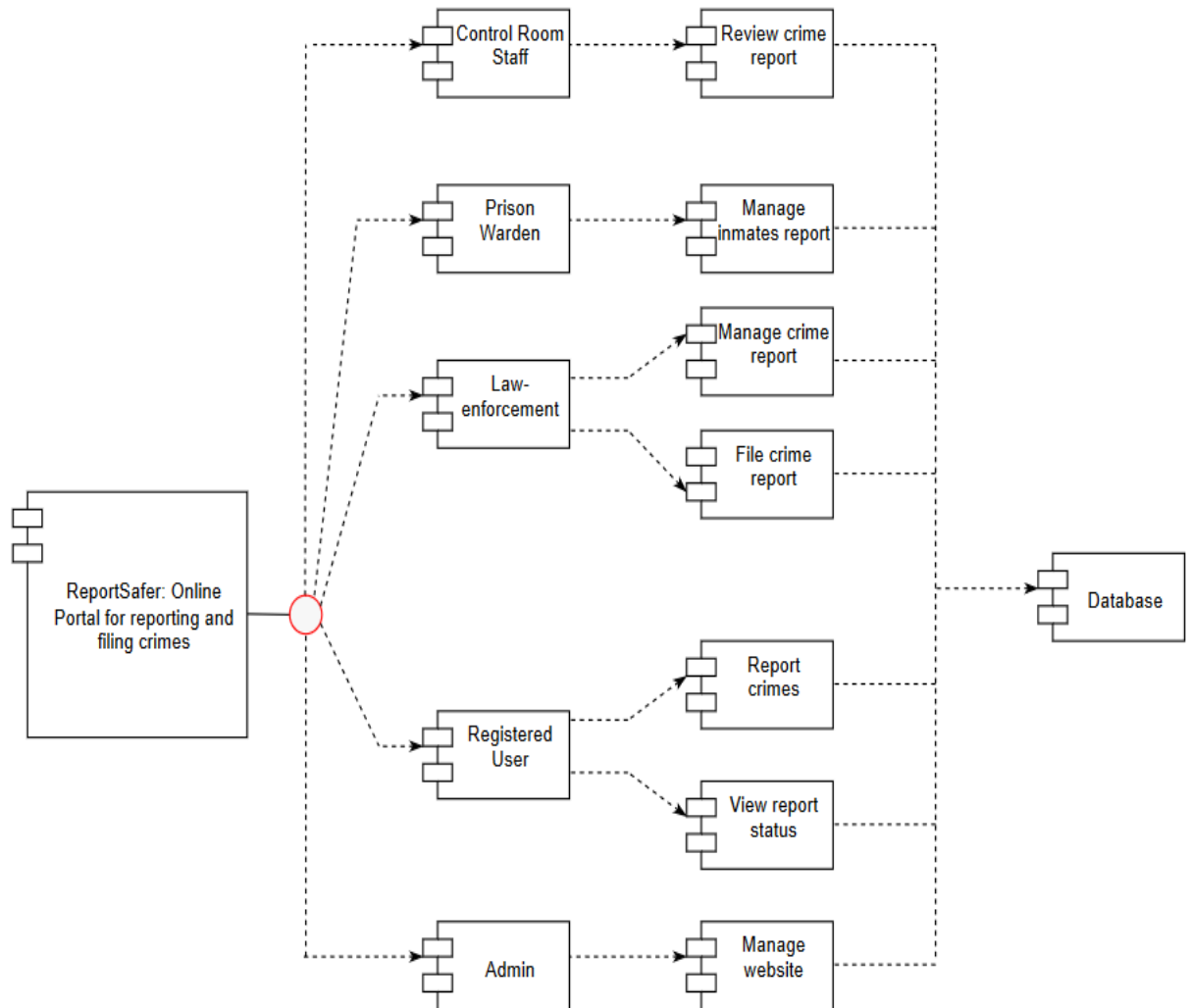


Figure 7: Component diagram

4.2.8 DEPLOYMENT DIAGRAM

The deployment diagram visualizes the physical hardware on which the software will be deployed. It portrays the static deployment view of a system. It involves the nodes and their relationships. It ascertains how software is deployed on the hardware. It maps the software architecture created in design to the physical system architecture, where the software will be executed as a node. Since it involves many nodes, the relationship is shown by utilizing communication paths. In contrast to other UML diagram types, which primarily depict the logical components of a system. The deployment diagram does not focus on the logical components of the system, but it put its attention on the hardware topology.

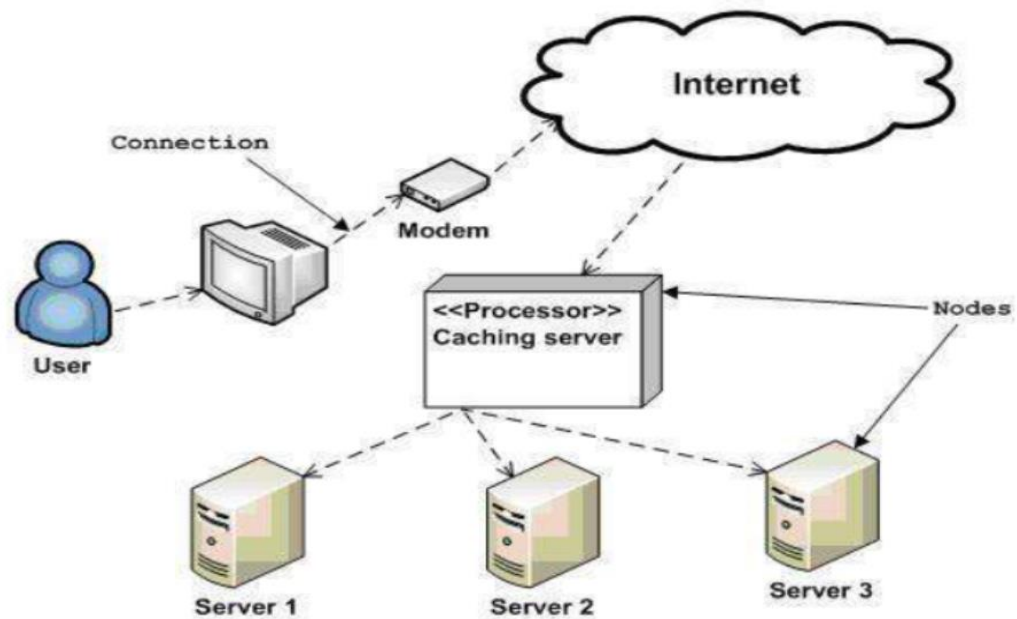


Figure 8: Deployment diagram

4.3 USER INTERFACE DESIGN USING FIGMA

Login Form

Login

Your e-mail ID

Password

[Forgot Password?](#)

LOGIN

Don't have an account? [Register](#)

Registration Form

Registration Form

Title

First Name

Second Name

Date of Birth

Aadhaar Number

Phone Number

e-mail

Address

Password

Confirm Password

Already have an account? [Login](#)

REGISTER AS VICTIM

REGISTER AS WITNESS

Crime Report Form:

REPORT A CRIME

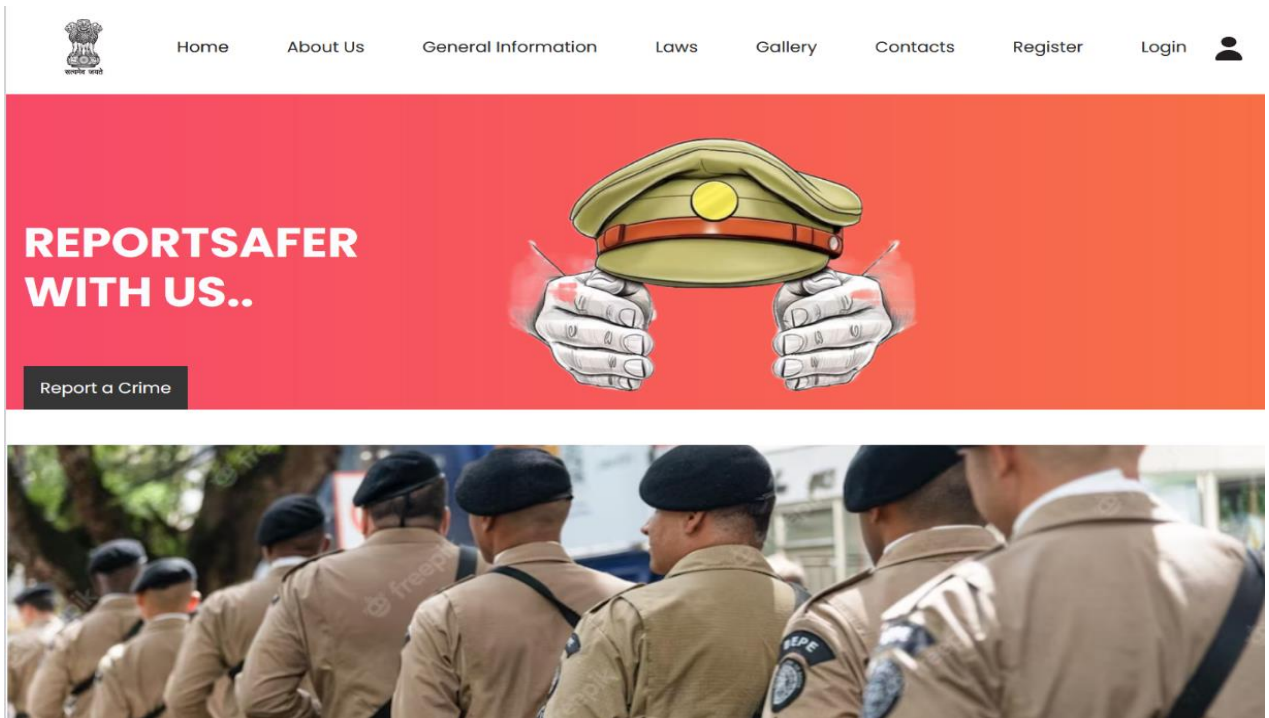
NAME

AADHAAR NUMBER

EVIDENCE

DESCRIPTION OF INCIDENT

SUBMIT

Index Page:

4.4 DATABASE DESIGN

A database is a set of information that has been arranged so that it can be easily accessed, managed, and updated. Any database could have information security as one of its primary goals. There are two stages in the database design process. To build a database that as clearly as possible satisfies user requirements, user requirements are obtained in the first stage. This is done independently of any DBMS and is referred to as information-level design. The design is changed from an information-level design to a particular DBMS design that will be utilized to build the system in the second step. The physical-level design stage is where the properties of the particular DBMS are taken into account.

4.4.1 RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS)

A Relational Database Management System (RDBMS) is a critical software application for organizing and managing data in a structured manner. It stores data in tables with rows and columns, where each row represents a record, and each column is a specific attribute or field. RDBMS systems like MySQL, Oracle, or Microsoft SQL Server ensure data integrity, enforce relationships between tables, and allow for efficient data retrieval and manipulation through Structured Query Language (SQL). These systems are widely used in various applications, such as e-commerce websites, financial systems, and inventory management, due to their robustness, scalability, and ability to handle complex data structures, making them essential for modern data-driven environments.

A Relational Database Management System (RDBMS) is a cornerstone of modern data management. It structures data into tables, where each row represents a unique entry, and each column corresponds to a specific attribute, ensuring a logical and organized storage system. RDBMS systems employ complex algorithms for data retrieval and manipulation, guaranteeing data consistency and adherence to predefined relationships between tables. Popular RDBMS software, including PostgreSQL, MySQL, and Microsoft SQL Server, provides robust features for transactions, data security, and scalability. These systems are integral to a myriad of applications, from healthcare records and customer databases to inventory control and financial platforms, supporting the efficient storage, retrieval, and analysis of vast datasets, making them essential tools in today's data-driven world.

4.4.2 NORMALIZATION

Normalization is a database design technique used in relational database management systems (RDBMS) to eliminate data redundancy and improve data integrity. It involves organizing data in a way that reduces data duplication and ensures that relationships between tables are defined and maintained.

The primary goals of normalization are:

Minimizing Data Redundancy: By breaking down data into separate tables and ensuring that each piece of data is stored only once, normalization helps reduce the chances of data inconsistencies or errors.

Ensuring Data Integrity: Normalization enforces the rules of referential integrity, which means that relationships between tables are well-defined and maintained. This ensures that data remains accurate and consistent.

First Normal Form (1NF):

- Each table has a primary key.
- All columns contain atomic (indivisible) values.
- There are no repeating groups or arrays in columns.

Second Normal Form (2NF):

- The table is in 1NF.
- All non-key attributes are fully functionally dependent on the entire primary key. In other words, all non-key attributes must be dependent on the entire primary key, not just part of it.

Third Normal Form (3NF):

- The table is in 2NF.
- There is no transitive dependency, meaning that non-key attributes are not dependent on other non-key attributes.

4.4.3 SANITIZATION

In Python Django, sanitization refers to the process of cleaning and validating data to ensure that it is safe and free from malicious content before it is used or stored in a database. Sanitization is a crucial security practice to prevent various forms of attacks, such as cross-site scripting (XSS) and SQL injection, which can compromise the security and integrity of a web application.

4.4.4 INDEXING

Indexing in Django is a fundamental database optimization technique. It involves creating data structures, or indexes, on specific fields to expedite data retrieval. These indexes act as signposts that enable the database to swiftly locate and fetch relevant data, especially in tables with substantial amounts of information. Django offers automatic index creation for primary keys, unique fields, and foreign keys. Additionally, developers can define custom indexes for fields frequently used in filtering, sorting, or searching. The choice of proper indexing plays a pivotal role in improving query performance, resulting in more responsive and scalable web applications. It's essential to consider application-specific query patterns and create indexes accordingly, as well as to understand the capabilities and limitations of the chosen database backend. Effective indexing is a cornerstone of efficient database operations in Django, contributing to enhanced application performance.

4.5 TABLE DESIGN

1. Tbl_login

Primary key: **loginID**

Foreign key: **regID** references table **Tbl_register**

FieldName	Datatype	Key Constraint	Description
loginID	Int (10)	PK	Login ID of users
Email	Varchar (70)	Not	Email of users
Password	Varchar (100)	Not null	Password of users
Role	Varchar (50)	Not null	Role of users
regID	Int (10)	FK	regID of users

2. Tbl_register

Primary key: **regID**

Foreign key: **AadharID** references table **Tbl_Aadhar**

FieldName	Datatype	Key Constraint	Description
regID	Int (10)	PK	regID of users
Name	Varchar (70)	Not null	Name of user
Dob	Date (10)	Not null	Date of birth of user
Ph_no	Int (10)	Not null	Phone number of user
Address	Varchar (30)	Not null	Address of users
Gender	Varchar (15)	Not null	Gender of users
Aadhar_ID	Int (10)	FK	Unique Aadhar_ID

3. Tbl_Aadhar

Primary key: **AadharID**

FieldName	Datatype	Key Constraint	Description
Aadhar_ID	Int (10)	PK	Unique Aadhar_ID
Aadhar_Num	Int (12)	Not null	Aadhar_Num of user
Aadhar_Name	Varchar (20)	Not null	Aadhar_Name of user
Ph_No	Int (10)	Not null	Phone number of user
Pin_code	Int (6)	Not null	Pin code of user
Dob	Date (10)	Not null	Date of birth

4. Tbl_crime

Primary key: **crimeID**

FieldName	Datatype	Key Constraint	Description
Crime_ID	Int (20)	PK	Unique Crime_ID
Crime_category	Varchar (40)	Not null	Category of crime

5. Tbl_victim

Foreign key: **loginID** references table **Tbl_login**

Foreign key: **FIR_ID** references table **Tbl_FIR**

FieldName	Datatype	Key Constraint	Description
loginID	Int (10)	FK	loginID of victim
FIR_ID	Int (20)	FK	FIR_ID of FIR
Witness_name	Varchar (20)	Null	Name of witness
Witness_relationship	Varchar (10)	Null	Relation with witness
Evidence	Varchar (10)	Null	Evidence to upload
Suspect_name	Varchar (20)	Null	Name of suspect
Suspect_age	Int (2)	Null	Age of suspect
Suspect_loc	Varchar (30)	Null	Location of suspect
Suspect_attributes	Varchar (30)	Null	Attributes of suspect
Description	Varchar (100)	Not null	Description of incident
Anonymity	Varchar (3)	Not null	Anonymity needed or not

6. Tbl_FIR

Primary key: **FIR_ID**

Foreign key: **Crime_ID** references table **Tbl_Crime**

FieldName	Datatype	Key Constraint	Description
FIR_ID	Int (10)	PK	FIR_ID of FIR
Crime_ID	Int (20)	FK	Crime_ID of Crime
Date_of_occ	Date (10)	Not null	Date of Occurrence
Loc_of_occ	Varchar (20)	Not null	Location of occurrence
Description	Varchar (100)	Not null	Description of incident
Crime_status	Varchar (20)	Not null	Status of crime

7. Tbl_Prisonwarden

Primary key: **Inmate_ID**

Foreign key: **FIR_ID** references table **Tbl_FIR**

FieldName	Datatype	Key Constraint	Description
Inmate_ID	Int (10)	PK	ID of Inmate
Inmate_Name	Varchar (20)	Not null	Name of Inmate
FIR_ID	Int (10)	FK	FIR_ID of IFR
Appointment_date	Date (10)	Null	Date of Appointment
Appointment_time	Time (10)	null	Time of Appointment

8. Tbl_Controlroomstaff

Primary key: **Staff_ID**

Foreign key: **loginID** references table **Tbl_login**

FieldName	Datatype	Key Constraint	Description
Staff_ID	Int (10)	PK	ID of Staff
Staff_name	Varchar (20)	Not null	Name of Staff
loginID	Int (10)	FK	loginID of Staff

9. Tbl_Lawenforcement

Primary key: **Lawenforcement_ID**

Foreign key: **loginID** references table **Tbl_login**

FieldName	Datatype	Key Constraint	Description
loginID	Int (10)	FK	loginID of officer
Lawenforcement_ID	Int (10)	PK	ID of officer
Lawenforcement_loc	Varchar (20)	Not null	Location of officer

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

Software testing is a way to check if the computer program works like it's supposed to. We use testing to make sure the software does what it's supposed to do. Validation means checking or testing things like software to make sure they meet the requirements and standards they are supposed to follow. Software testing is a way to check if a program works well. It goes along with other methods like checking and walking through the program. Validation means making sure that what the user wanted is what they got. There are several rules that can serve as testing objectives.

They are:

Testing is a process of executing a program with the intent of finding an error.

- A good test case is one that has high possibility of finding an undiscovered error.
- A successful test is one that uncovers an undiscovered error.

If a test works well and follows its goals, it can find mistakes in the software. The test showed that

the computer program is working like it's supposed to and is doing well.

There are three ways to test program.

- For correctness
- For implementation efficiency
- For computational complexity

5.2 TEST PLAN

A test plan suggests several required steps that need be taken to complete various testing methodologies. The activity that is to be taken is outlined in the test plan. A computer program, its documentation, and associated data structures are all created by software developers. Testing's precise goals should be laid forth in quantifiable language. So that the mean time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test work-hours per regression test all should be stated within the test. The levels of testing include:

- Unit testing
- Integration Testing
- Data validation Testing & Output Testing

5.2.1 UNIT TESTING

Unit testing concentrates verification efforts on the software component or module, which is the smallest unit of software design. The component level design description is used as a guide when testing crucial control paths to find faults inside the module's perimeter. The level of test complexity and the untested area determined for unit testing. Unit testing is white-box focused, and numerous components may be tested simultaneously. To guarantee that data enters and exits the software unit under test properly, the modular interface is tested. To make sure that data temporarily stored retains its integrity during each step of an algorithm's execution, the local data structure is inspected. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling paths are tested.

Before starting any other test, tests of data flow over a module interface are necessary. All other tests are irrelevant if data cannot enter and depart the system properly. An important duty during the unit test is the selective examination of execution pathways. Error circumstances must be foreseen in good design, and error handling paths must be put up to cleanly reroute or halt work when an error does arise. The final step of unit testing is boundary testing. Software frequently fails at its limits.

Unit testing was carried out by treating each module as a distinct entity and subjecting them to a variety of test inputs. The internal logic of the modules had some issues, which were fixed. Each module is tested and run separately after coding. All unused code was eliminated, and it was confirmed that every module was functional and produced the desired outcome.

5.2.2 INTEGRATION TESTING

Integration testing is a methodical approach for creating the program's structure while also carrying out tests to find interface issues. The goal is to construct a program structure that has been determined by design using unit tested components. The program is tested. Correction is challenging since the size of the overall program makes it challenging to isolate the causes. As soon as these mistakes are fixed, new ones arise, and the process repeats itself in an apparently unending cycle. All the modules were integrated after unit testing was completed in the system to check for an interface inconsistency. A distinctive program structure also developed when discrepancies in program structures.

5.2.3 VALIDATION TESTING OR SYSTEM TESTING

The testing process comes to an end here. This involved testing the entire system in its entirety, including all forms, code, modules, and class modules. Popular names for this type of testing include system tests and black box testing. The functional requirements of the software are the main emphasis of the black box testing approach. That example, using Black Box testing, a software engineer can create sets of input conditions that will fully test every program requirement. The following sorts of problems are targeted by black box testing: erroneous or missing functions, interface errors, data structure or external data access errors, performance errors, initialization errors, and termination errors.

5.2.4 OUTPUT TESTING OR USER ACCEPTANCE TESTING

The system considered is tested for user acceptance; here it should satisfy the firm's need. The software should keep in touch with perspective system; user at the time of developing and making changes whenever required.

This done with respect to the following points:

- Input Screen Designs,
- Output Screen Designs,

The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using that test data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future.

5.2.5 AUTOMATION TESTING

Automation Testing is a software testing technique that performs using special automated testing software tools to execute a test case suite. Essentially, it's a test to double-check that the equipment or software does exactly what it was designed to do. It tests for bugs, defects, and any other issues that can arise with product development. Although some types of testing, such as regression or functional testing can be done manually, there are greater benefits of doing it automatically. Automation testing can be run at any time of the day. It uses scripted sequences to examine the software. It then reports on what has been found, and this information can be compared with earlier test runs. Automation developers generally write in the following programming languages: C#, JavaScript, and Ruby.

5.2.6 SELENIUM TESTING

Selenium is an open-source automated testing framework used to verify web applications across different browsers and platforms. Selenium allows for the creation of test scripts in various programming languages such as Java, C#, and Python. Jason Huggins, an engineer at Thought Works, developed Selenium in 2004 while working on a web application that required frequent testing. He created a JavaScript program called "JavaScriptTestRunner" to automate browser actions and improve testing efficiency. Selenium has since evolved and continues to be developed by a team of contributors. In addition to Selenium, another popular tool used for automated testing is Cucumber. Cucumber is an open-source software testing framework that supports behavior-driven development (BDD). It allows for the creation of executable specifications in a human-readable format called Gherkin. One of the advantages of using Cucumber is its ability to bridge the gap between business stakeholders and technical teams. By using a common language, Cucumber facilitates effective communication and collaboration during the testing process. It promotes a shared understanding of the requirements and helps ensure that the developed software meets the intended business goals. Cucumber can be integrated with Selenium to combine the benefits of both tools. Selenium is used for interacting with web browsers and automating browser actions, while Cucumber provides a structured framework for organizing and executing tests. This combination allows for the creation of end-to-end tests that verify the behavior of web applications across different browsers and platforms, using a business-readable and maintainable format.

Example:

Test Case 1: User Login

Code

```
from datetime import datetime
from django.test import TestCase
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
class Hosttest(TestCase):

    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.implicitly_wait(10)
```

```

self.live_server_url = 'http://127.0.0.1:8000/'

def tearDown(self):
    self.driver.quit()

def test_01_login_page(self):
    driver = self.driver
    driver.get(self.live_server_url)
    driver.maximize_window()
    time.sleep(1)
    login=driver.find_element(By.CSS_SELECTOR,"a.nav-
link[href='http://127.0.0.1:8000/login']")
    login.click()
    time.sleep(1)
    email=driver.find_element(By.CSS_SELECTOR,"input#email[name='email']")
    email.send_keys("midhujh27@gmail.com")
    password=driver.find_element(By.CSS_SELECTOR,"input#password[name='pas
sword']")
    password.send_keys("Midhu1.jayan")
    time.sleep(1)
    driver.execute_script("window.scrollTo(0,
document.body.scrollHeight);")
    time.sleep(1)
    submit = driver.find_element(By.CSS_SELECTOR, "button#login")
    submit.click()
    time.sleep(1)

if __name__ == '__main__':
    import unittest
    unittest.main()

```

Screenshot

```

(env) F:\FINAL_PROJECT\PROJECT\crimeproject>python manage.py test
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).

DevTools listening on ws://127.0.0.1:64376/devtools/browser/0d61271e-2d92-40ef-a6b9-69592d034e8a
[32740:8564:1024/141855.152:ERROR:cert_issuer_source_aia.cc(36)] Error parsing cert retrieved from AIA (as DER):
ERROR: Couldn't read tbsCertificate as SEQUENCE
ERROR: Failed parsing Certificate
[]

[32740:8564:1024/141856.381:ERROR:cert_issuer_source_aia.cc(36)] Error parsing cert retrieved from AIA (as DER):
ERROR: Couldn't read tbsCertificate as SEQUENCE
ERROR: Failed parsing Certificate

.
-----
Ran 1 test in 17.036s

OK
Destroying test database for alias 'default'...

```

Test Report

Test Case 1					
Project Name: REPORTSAFER					
Login Test Case					
Test Case ID: 1			Test Designed By: Midhu J H		
Test Priority (Low/Medium/High): High			Test Designed Date: 13-10-2023		
Module Name: Login Screen			Test Executed By: Ms. Shelly Shiju George		
Test Title: User Login			Test Execution Date: 14-10-2023		
Description: Verify login with valid email and password					
Pre-Condition: User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigation to Login Page		Login Page should be displayed	Login Page be displayed	Pass
2	Provide valid Email	Email: midhujh27@gmail.com	User should be able to login	User is logged in and navigated to corresponding Home Page	Pass
3	Provide Valid Password	Password: Midhu1.jayan			
4	Click on Login button				
Post-Condition: User is validated with database and successfully login into account. The Account session details are logged in database					

Test Case 2: Report a Crime**Code**

```

from datetime import datetime
from django.test import TestCase
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
class Hosttest(TestCase):

```

```

def setUp(self):
    self.driver = webdriver.Chrome()
    self.driver.implicitly_wait(10)
    self.live_server_url = 'http://127.0.0.1:8000/'

def tearDown(self):
    self.driver.quit()

def test_01_login_page(self):
    driver = self.driver
    driver.get(self.live_server_url)
    driver.maximize_window()
    time.sleep(1)
    login=driver.find_element(By.CSS_SELECTOR,"a.nav-
link[href='http://127.0.0.1:8000/login']")
    login.click()
    time.sleep(1)
    email=driver.find_element(By.CSS_SELECTOR,"input#email[name='email']")
    email.send_keys("midhujh27@gmail.com")
    password=driver.find_element(By.CSS_SELECTOR,"input#password[name='password']")
    password.send_keys("Midhu1.jayan")
    time.sleep(1)
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
    time.sleep(1)
    submit = driver.find_element(By.CSS_SELECTOR, "button#login")
    submit.click()
    time.sleep(1)
    activity = driver.find_element(By.CSS_SELECTOR, "a#activity")
    activity.click()
    time.sleep(1)
    reportcrime = driver.find_element(By.CSS_SELECTOR, "a#reportcrime")
    reportcrime.click()
    time.sleep(1)
    slt = driver.find_element(By.CSS_SELECTOR, "select#crimeCategory")
    slt.click()
    time.sleep(1)
    slt_human = driver.find_element(By.CSS_SELECTOR,
"option[value='off_doc']")
    slt_human.click()
    time.sleep(1)
    chk = driver.find_element(By.CSS_SELECTOR, "label[for='termsCheckbox']")
    chk.click()
    time.sleep(1)
    sub = driver.find_element(By.CSS_SELECTOR,
"button[onclick='redirectToPage()']")
    sub.click()
    time.sleep(1)
    aadh=driver.find_element(By.CSS_SELECTOR,"input#ano[type='text']")

```

```

aadh.send_keys("987605479087")
time.sleep(1)
aadh=driver.find_element(By.CSS_SELECTOR,"input#email[type='email']")
aadh.send_keys("midhujh27@gmail.com")
time.sleep(1)
rad=driver.find_element(By.CSS_SELECTOR,"input[type='radio'][name='threat'
][value='Yes']")
rad.click()
time.sleep(1)
des=driver.find_element(By.CSS_SELECTOR,"textarea#descri")
des.send_keys("Manu Joseph, Phone: 9999999999, On 11-10-2023, I discovered
a potentially forged document while reviewing SSLC Certificates. The document
exhibited suspicious alterations, including questionable signatures and
discrepancies in dates.")
time.sleep(1)
driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
time.sleep(1)
file_path = 'C:\\Users\\Midhu J H\\Downloads\\user.jpg'
evi_image_input = driver.find_element(By.CSS_SELECTOR,
"input#id_evidence_image[type='file']")
evi_image_input.send_keys(file_path)
time.sleep(3)
btn=driver.find_element(By.CSS_SELECTOR,"button[type='submit']")
btn.click()
time.sleep(3)
log = driver.find_element(By.CSS_SELECTOR,"a.nav-link[href='/logout/']")
log.click()
time.sleep(2)
if __name__ == '__main__':
    import unittest
    unittest.main()

```

Screenshot

```

(env) F:\FINAL_PROJECT\PROJECT\crimeproject>python manage.py test
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).

DevTools listening on ws://127.0.0.1:64601/devtools/browser/7ac1c920-20d5-4090-b024-b3490346373d
[26200:9420:1024/143955.103:ERROR:cert_issuer_source_aia.cc(36)] Error parsing cert retrieved from AIA (as DER):
ERROR: Couldn't read tbsCertificate as SEQUENCE
ERROR: Failed parsing Certificate

[26200:9420:1024/143955.606:ERROR:cert_issuer_source_aia.cc(36)] Error parsing cert retrieved from AIA (as DER):
ERROR: Couldn't read tbsCertificate as SEQUENCE
ERROR: Failed parsing Certificate

.
-----
Ran 1 test in 39.085s

OK
Destroying test database for alias 'default'...

```

Test report

Test Case 2					
Project Name: REPORTSAFER					
Report Crime Test Case					
Test Case ID: 2			Test Designed By: Midhu J H		
Test Priority (Low/Medium/High): High			Test Designed Date: 15-10-2023		
Module Name: Crime Reporting			Test Executed By: Ms. Shelly Shiju George		
Test Title: Report a Crime			Test Execution Date: 16-10-2023		
Description: User reporting a Crime related to Documents					
Pre-Condition: User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigation to Login Page		Login Page should be displayed	Login Page be displayed	Pass
2	Provide valid Email	Email: midhujh27@gmail.com	User should be able to login	User is logged in and navigated to corresponding Home Page	Pass
3	Provide Valid Password	Password: Midhu1.jayan			
4	Click on Login button				
5	Click on Report a Crime option	Input the particular crime category option		Navigated to Crime Category form	Pass
6	Click on Submit button			Successfully navigated to Document Crime Report Form	Pass
7	Enter the required form details			Successfully added to the database	Pass
8	Click on Submit button			Response downloaded	Pass
Post-Condition: A new crime related to documents is reported successfully by the User.					

Test Case 3: View Reported Crimes

Code

```
from datetime import datetime
from django.test import TestCase
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
class Hosttest(TestCase):

    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.implicitly_wait(10)
        self.live_server_url = 'http://127.0.0.1:8000/'

    def tearDown(self):
        self.driver.quit()

    def test_01_login_page(self):
        driver = self.driver
        driver.get(self.live_server_url)
        driver.maximize_window()
        time.sleep(1)
        login=driver.find_element(By.CSS_SELECTOR,"a.nav-
link[href='http://127.0.0.1:8000/login']")
        login.click()
        time.sleep(1)
        email=driver.find_element(By.CSS_SELECTOR,"input#email[name='email']")
        email.send_keys("midhujh27@gmail.com")
        password=driver.find_element(By.CSS_SELECTOR,"input#password[name='passwor
d']")
        password.send_keys("Midhu1.jayan")
        time.sleep(1)
        driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
        time.sleep(1)
        submit = driver.find_element(By.CSS_SELECTOR, "button#login")
        submit.click()
        time.sleep(1)
        activity = driver.find_element(By.CSS_SELECTOR, "a#activity")
        activity.click()
        time.sleep(1)
        reportcrime = driver.find_element(By.CSS_SELECTOR, "a#viewcrime")
        reportcrime.click()
        time.sleep(1)
        log = driver.find_element(By.CSS_SELECTOR,"a.nav-link[href='/logout/']")
```



```
log.click()
time.sleep(2)

if __name__ == '__main__':
    import unittest
    unittest.main()
```

Screenshot

```
(env) F:\FINAL_PROJECT\PROJECT\crimeproject>python manage.py test
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).

DevTools listening on ws://127.0.0.1:65026/devtools/browser/262ffaf0-323d-4d27-bc09-5ec276006c35
[11692:22396:1024/145532.416:ERROR:cert_issuer_source_aia.cc(36)] Error parsing cert retrieved from AIA (as DER):
ERROR: Couldn't read tbsCertificate as SEQUENCE
ERROR: Failed parsing Certificate

[11692:22396:1024/145533.679:ERROR:cert_issuer_source_aia.cc(36)] Error parsing cert retrieved from AIA (as DER):
ERROR: Couldn't read tbsCertificate as SEQUENCE
ERROR: Failed parsing Certificate

.
-----
Ran 1 test in 49.285s

OK
Destroying test database for alias 'default'...
```

Test report

Test Case 3					
Project Name: REPORTSAFER					
View Reported Crimes					
Test Case ID: 3			Test Designed By: Midhu J H		
Test Priority (Low/Medium/High): High			Test Designed Date: 17-10-2023		
Module Name: View reported crimes			Test Executed By: Ms. Shelly Shiju George		
Test Title: View Crimes			Test Execution Date: 18-10-2023		
Description: User viewing the reported crimes.					
Pre-Condition: User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigation to Login Page		Login Page should be displayed	Login Page be displayed	Pass
2	Provide valid Email	Email: midhujh27@gmail.com	User should be able to login	User is logged in and navigated to corresponding Home Page	Pass
3	Provide Valid Password	Password: Midhu1.jayan			
4	Click on Login button				
5	Select the View Reported Crimes option	Click on View reported crimes button		Navigated to the Reported Crimes listing Page	Pass
6	Click the Logout option from the Navbar			Successfully logged out and Navigated to the Home Page	Pass
Post-Condition: A crime is reported successfully by the User and stored in the database.					

Test Case 4: Upload the Documents by Law Enforcement

Code

```
from datetime import datetime
from django.test import TestCase
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
class Hosttest(TestCase):

    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.implicitly_wait(10)
        self.live_server_url = 'http://127.0.0.1:8000/'

    def tearDown(self):
        self.driver.quit()

    def test_01_login_page(self):
        driver = self.driver
        driver.get(self.live_server_url)
        driver.maximize_window()
        time.sleep(1)
        login2=driver.find_element(By.CSS_SELECTOR, "a.nav-
link[href='http://127.0.0.1:8000/login']")
        login2.click()
        time.sleep(1)
        email=driver.find_element(By.CSS_SELECTOR, "input#email[name='email']")
        email.send_keys("kanjirappallylawenforcement@gmail.com")
        password=driver.find_element(By.CSS_SELECTOR, "input#password[name='password']")
        password.send_keys("Midhu1.jayan")
        time.sleep(1)
        driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
        time.sleep(1)
        submit = driver.find_element(By.CSS_SELECTOR, "button#login")
        submit.click()
        time.sleep(1)
        activity2 = driver.find_element(By.CSS_SELECTOR, "a.nav-
link#activity[data-toggle='dropdown']")
        activity2.click()
        time.sleep(1)
        managecrime = driver.find_element(By.CSS_SELECTOR,
"a#managecrime[href='/law_page']")
        managecrime .click()
```

```

        time.sleep(1)
        btndetails = driver.find_element(By.CSS_SELECTOR,
"a[href='/view_crime/37'] > button.btn.btn-light.edit-button")
        btndetails .click()
        time.sleep(1)
        driver.execute_script("window.scrollTo(0, 350);")
        time.sleep(3)
        file_path_ = 'C:\\\\Users\\Midhu J H\\Downloads\\Report_9.pdf'
        file_in = driver.find_element(By.CSS_SELECTOR,
"input[type='file'][name='document_arrest'][id='id_document_arrest']")
        file_in.send_keys(file_path_)
        time.sleep(5)
        btnup1 = driver.find_element(By.CSS_SELECTOR, "button#charge-arrest")
        btnup1.click()
        time.sleep(2)
        activity3 = driver.find_element(By.CSS_SELECTOR, "a.nav-link#acti[data-
toggle='dropdown']")
        activity3.click()
        time.sleep(1)
        managecrimestatus = driver.find_element(By.CSS_SELECTOR, "a.dropdown-
item[href='/law_page']")
        managecrimestatus.click()
        time.sleep(2)
        log2 = driver.find_element(By.CSS_SELECTOR, "a.nav-link[href='/logout/']")
        log2.click()
        time.sleep(2)

# Add more test methods as needed

if __name__ == '__main__':
    import unittest
    unittest.main()

```

Screenshot

```

(env) F:\FINAL_PROJECT\PROJECT\crimeproject>python manage.py test
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).

DevTools listening on ws://127.0.0.1:65501/devtools/browser/51c06311-b816-487b-87cf-d7e34cf05062
[5724:32072:1024/150706.628:ERROR:cert issuer source_aia.cc(36)] Error parsing cert retrieved from AIA (as DER):
ERROR: Couldn't read tbsCertificate as SEQUENCE
ERROR: Failed parsing Certificate

[5724:32072:1024/150707.086:ERROR:cert issuer source_aia.cc(36)] Error parsing cert retrieved from AIA (as DER):
ERROR: Couldn't read tbsCertificate as SEQUENCE
ERROR: Failed parsing Certificate

.
-----
Ran 1 test in 72.711s

OK
Destroying test database for alias 'default'...

(env) F:\FINAL_PROJECT\PROJECT\crimeproject>

```

Test report

Test Case 4					
Project Name: REPORTSAFER					
Upload Documents by Law Enforcement					
Test Case ID: 4			Test Designed By: Midhu J H		
Test Priority (Low/Medium/High): High			Test Designed Date: 23-10-2023		
Module Name: Upload Documents			Test Executed By: Ms. Shelly Shiju George		
Test Title: Upload Documents and update Status			Test Execution Date: 24-10-2023		
Description: Law enforcement uploading documents and updating the crime status.					
Pre-Condition: Law enforcement has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigation to Login Page		Login Page should be displayed	Login Page is displayed	Pass
2	Provide valid Email	Email: kanjirappallylawenforcement@gmail.com	Law enforcement should be able to login	Law enforcement is logged in and navigated to corresponding Home Page	Pass
3	Provide Valid Password	Password: Midhu1.jayan			
4	Click on Login button				
5	From the navbar, click on Manage Crime Status	Clicking on View documents button		Law enforcement can upload documents which results in the update of crime status	Pass
6	Click on Logout option from the Navbar			Successfully logged out and Navigated to the Home Page	Pass
Post-Condition: One or more crimes had been reported successfully by the User and stored in the database.					

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

Implementation is the stage of the project where the theoretical design is turned into a working system. It can be the most crucial stage in achieving a successful new system gaining the users confidence that the new system will work and will be effective and accurate. It is primarily concerned with user training and documentation. Conversion usually takes place about the same time the user is being trained or later. Implementation simply means convening a new system design into operation, which is the process of converting a new revised system design into an operational one.

At this stage the main work load, the greatest upheaval and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned or controlled, it can create chaos and confusion. Implementation includes all those activities that take place to convert from the existing system to the new system. The new system may be a totally new, replacing an existing manual or automated system or it may be a modification to an existing system. Proper implementation is essential to provide a reliable system to meet organization requirements. The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from the old system to the new system. The system can be implemented only after thorough testing is done and if it is found to be working according to the specifications. The system personnel check the feasibility of the system. The more complex the system being implemented, the more involved will be the system analysis and design effort required to implement the three main aspects: education and training, system testing and changeover. The implementation state involves the following tasks:

- Careful planning.
- Investigation of system and constraints.
- Design of methods to achieve the changeover.

6.2 IMPLEMENTATION PROCEDURES

The implementation procedures for this project involve a systematic and phased approach to bring the online crime reporting portal to fruition. This complex system, designed to cater to the needs of various stakeholders, requires careful planning and execution.

To commence the implementation, the project team will initiate a detailed requirements analysis phase. This stage involves comprehensive discussions with law enforcement officials, control room staff, prison wardens, and potential end-users to determine their specific needs and expectations.

The results of this analysis will inform the development of a detailed system specification and design.

The development phase will follow, involving the creation of the portal's architecture, databases, and user interfaces. It's during this stage that the functionalities outlined in the project's scope, including user management, crime reporting, and communication features, will be integrated into the system. The portal will be designed with a user-friendly interface to ensure ease of use for all categories of users.

Simultaneously, a dedicated team will work on the incorporation of advanced technologies, such as machine learning capabilities for crime trend analysis and predictive features. These technologies will be implemented carefully to ensure the portal's robustness and security.

Once the development phase is complete, rigorous testing will be conducted to identify and rectify any bugs or issues. The portal will undergo extensive quality assurance and user acceptance testing to ensure that it meets all the necessary requirements and is free of vulnerabilities.

Deployment of the system will be carried out in a controlled manner, ensuring minimal disruption to the existing processes, and allowing for gradual adaptation by the involved stakeholders. Comprehensive training sessions will be conducted to familiarize law enforcement personnel, control room staff, prison wardens, and registered users with the portal's functionality and features. Post-deployment, the project team will continue to provide support and maintenance, addressing any issues that may arise and making necessary improvements as the system matures. Regular updates and security measures will be implemented to safeguard user data and maintain the portal's efficiency.

6.2.1 USER TRAINING

User training is designed to prepare the user for testing and converting the system. To achieve the objective and benefits expected from computer-based system, it is essential for the people who will be involved to be confident of their role in the new system. As system becomes more complex, the need for training is more important. By user training the user comes to know how to enter data, respond to error messages, interrogate the database, and call up routine that will produce reports and perform other necessary functions.

6.2.2 TRAINING ON THE APPLICATION SOFTWARE

After providing the necessary basic training on computer awareness the user will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the data entered. It should then cover information needed by the specific user/ group to use the system or part of the system while imparting the training of the program on the application. This training may be different across different user groups and across different levels of hierarchy.

6.2.3 SYSTEM MAINTENANCE

Maintenance is the enigma of system development. The maintenance phase of the software cycle is the time in which a software product performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is for it to make adaptable to the changes in the system environment. Software maintenance is of course, far more than “Finding Mistakes”.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

In conclusion, this project endeavors to create an online portal that serves as a pivotal communication platform between law-enforcement officials, registered users, control room staff, and prison wardens. By addressing the shortcomings of the existing system and introducing innovative features such as online fine payments, legal support, and machine learning capabilities, this portal aims to streamline and modernize the process of reporting and managing crimes. It offers a user-friendly interface, enhances transparency, and fosters greater efficiency in the criminal filing system. The diverse set of user roles and their respective functionalities provide a comprehensive solution that not only empowers users to report crimes but also enables law enforcement to respond swiftly and effectively, ultimately contributing to the overall safety and security of society.

7.2 FUTURE SCOPE

1. **Future Development:** The project offers substantial opportunities for continuous development and enhancement, focusing on the integration of advanced analytics and artificial intelligence to create a predictive tool for law enforcement.
2. **Collaboration and Expansion:** The project can expand by facilitating collaboration with other government agencies and departments, including emergency services, to provide a more comprehensive response to emergencies and incidents.
3. **Public Awareness and Education:** The portal's future scope includes public awareness and education initiatives, disseminating information about crime prevention, legal rights, and safety tips to empower individuals and promote a safer community.
4. **Mobile Accessibility and Security:** The project's future involves the development of a mobile application for easier reporting, optimized for various devices, and a strong commitment to cybersecurity to protect sensitive information and maintain user trust.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

- Pankaj Jalote, “Software engineering: a precise approach”.
- Gary B. Shelly, Harry J. Rosenblatt, “System Analysis and Design”, 2009.
- Ken Schwaber, Mike Beedle, “Agile Software Development with Scrum”, Pearson (2008).
- Roger S Pressman, “Software Engineering”
- IEEE Std 1016 Recommended Practice for Software Design Descriptions

WEBSITES:

- <https://keralapolice.gov.in/>
- www.w3schools.com

CHAPTER 9

APPENDIX

9.1 SAMPLE CODE

listcrime.html:

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- basic -->
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- mobile metas -->
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="viewport" content="initial-scale=1, maximum-scale=1">
    <!-- site metas -->
    <title>ReportSafer | Reported Crimes</title>
    <link rel="icon" href="{% static 'images/logo2.jpg' %}" type="image/x-icon">
    <meta name="keywords" content="">
    <meta name="description" content="">
    <meta name="author" content="">
    <!-- bootstrap css -->
    <link rel="stylesheet" type="text/css" href="{% static
'css/bootstrap.min.css' %}">
    <!-- style css -->
    <link rel="stylesheet" type="text/css" href="{% static 'css/style.css' %}">
    <!-- Responsive-->
    <link rel="stylesheet" href="{% static 'css/responsive.css' %}">
    <!-- favicon -->
    <link rel="icon" href="{% static 'images/fevicon.png' %}" type="image/gif" />
    <!-- Scrollbar Custom CSS -->
    <link rel="stylesheet" href="{% static 'css/jquery.mCustomScrollbar.min.css'
%}">
    <!-- Tweaks for older IEs-->
    <link rel="stylesheet" href="https://netdna.bootstrapcdn.com/font-
awesome/4.0.3/css/font-awesome.css">
    <!-- fonts -->
    <link
href="https://fonts.googleapis.com/css?family=Poppins:400,700&display=swap"
rel="stylesheet">
    <!-- owl stylesheets -->
    <link href="https://cdnjs.cloudflare.com/ajax/libs/owl-
carousel/1.3.3/owl.carousel.css" rel="stylesheet" />
  </head>
  <body style="font-family:Poppins,sans-serif;">
    <!-- header section start -->
    <div class="header_section">
      <div class="container-fluid">
        <nav class="navbar navbar-expand-lg navbar-light bg-light">
```

```

        <a class="logo" href="{% url 'index' %}"></a>
        <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarSupportedContent">
            <ul class="navbar-nav mr-auto mt-3">
                {% if user.is_authenticated %}
                <li class="nav-item active">
                    <a class="nav-link" href="{% url 'index' %}">Home</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="{% url 'about' %}">About Us</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="{% url 'general' %}">General
Information</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="{% url 'laws' %}">Laws</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="{% url 'gallery' %}">Gallery</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="{% url 'contact' %}">Contacts</a>
                </li>
                <div class="navbar-nav ml-auto">
                    <div class="nav-item dropdown">
                        <a href="#" class="nav-link dropdown-toggle" data-
toggle="dropdown">Your Activity</a>
                        <div class="dropdown-menu">
                            <a href="{% url 'crime_category' %}" class="dropdown-
item">Report a Crime</a>
                            <a href="{% url 'listcrime' %}" class="dropdown-
item">View Crime Status</a>
                        </div>
                    </div>
                </div>
                <li class="nav-item">
                    <a class="nav-link" href="{% url 'logout' %}">Logout</a><br>
                    {% else %}
                </li>
                <li class="nav-item">
                    <a class="nav-link"
href="http://127.0.0.1:8000/userregister">Register</a>
                </li>
            </ul>
        </div>
    </div>

```



```

        {% comment %} <div class="navbar-nav ml-auto">
            <div class="nav-item dropdown">
                <a href="#" class="nav-link dropdown-toggle" data-
toggle="dropdown">User Account</a>
                <div class="dropdown-menu">
                    <a href="http://127.0.0.1:8000/witnessregister"
class="dropdown-item">Register as Witness</a>
                    <a href="http://127.0.0.1:8000/victimregister"
class="dropdown-item">Register as Victim</a>
                </div>
            </div>
        </div> {% endcomment %}

        {% comment %} <li class="nav-item">
            <a class="nav-link"
href="http://127.0.0.1:8000/register">Register</a>
        </li> {% endcomment %}
        <li class="nav-item">
            <a class="nav-link"
href="http://127.0.0.1:8000/login">Login</a>
        </li>
    {% endif %}
    <br>
    {% comment %} <h2>User: {{user.name}}</h2> {% endcomment %}
</ul>
<!-- <form class="form-inline my-2 my-lg-0">
    <div class="login_menu">
        <ul>
            <li><a href="#">Login</a></li>
            <li><a href="#">Register</a></li>
            <li><a href="#"></a></li>
            <li><a href="#"></a></li>
        </ul>
    </div>
</form> -->
</div>
</nav>
</div>
</div>

<div style="padding:50px;">
    <h1 style="text-align: center;">Reported Crimes</h1>
    <section id="crime-list">
        {% block content %}
        <div class="row">
            {% for i in combined_reports %}
            <div class="col-md-4 mb-4"> <!-- Adjust the column
size as needed -->

```

```

        <div class="card text-black bg-custom rounded-
3" style="background-color: #d8e1e8; height: 100%;">
            <div class="card-body">
                <strong>{{ i.reporter_location
}}</strong><br>

                Status: {{ i.status }}<br>
                <!-- You can add more crime details
here if needed -->

            </div>
            <div class="card-footer bg-custom">
                <!-- Additional content or buttons can
go here -->

            </div>
        </div>
    </div>
    <div class="row">
        <div class="col-lg-3 col-md-6">
            <div class="col-lg-3 col-md-6">
                <div class="social_icon">
                    <ul>
                        <li><a href="#"></a></li>
                        <li><a href="#"></a></li>
                        <li><a href="#"></a></li>
                        <li><a href="#"></a></li>
                        <li><a href="#"></a></li>
                    </ul>
                </div>
            </div>
        </div>
    </div>
</div>

```

```
</div>
<!-- footer section end -->

    <!-- copyright section start -->
    <div class="copyright_section">
        <div class="container">
            <p class="copyright_text">© 2020 All Rights Reserved.</p>
        </div>
    </div>
    <!-- copyright section end -->

    <!-- Javascript files-->
    <script src="{% static 'js/jquery.min.js' %}"></script>
    <script src="{% static 'js/popper.min.js' %}"></script>
    <script src="{% static 'js/bootstrap.bundle.min.js' %}"></script>
    <script src="{% static 'js/jquery-3.0.0.min.js' %}"></script>
    <script src="{% static 'js/plugin.js' %}"></script>
    <!-- sidebar -->
    <script src="{% static 'js/jquery.mCustomScrollbar.concat.min.js' %}"></script>
    <script src="{% static 'js/custom.js' %}"></script>
    <!-- javascript -->
    <script src="{% static 'js/owl.carousel.js' %}"></script>
    <!-- owl carousel -->
    <script>
        $($('.owl-carousel')).owlCarousel({
            loop:true,
            margin:30,
            nav:true,
            responsive:{
                0:{
                    items:1
                },
                600:{
                    items:3
                },
                1000:{
                    items:4
                }
            }
        })
    </script>
</body>
</html>
```

views.py:

```
from django.contrib import messages
from django.shortcuts import render, redirect, get_object_or_404
#from .models import User
from .forms import CrimeReportForm, AnonyReportForm, DocReportForm, PublicForm,
PublicForm, EvidenceCrimeForm, PrisonReportForm
from .models import CustomUser, CrimeReport, DocReport, Jailor, SpecLoc, FIRFile,
PublicReport, EvidenceCrimeReport, PrisonReport, Inmate
import re
from django.contrib.auth import authenticate, login as auth_login
from django.contrib.auth.models import User, auth
from django.contrib.auth.decorators import login_required
from django.contrib.auth import logout as auth_logout
from django.http import HttpResponse
from django.template.loader import get_template
from xhtml2pdf import pisa
from django.core.mail import send_mail
from django.utils.crypto import get_random_string
from django.http import JsonResponse, FileResponse
from django.template.loader import get_template
from django.urls import reverse
from django.http import Http404
from django.core.mail import send_mail
from django.contrib import messages

# Create your views here.

def index(request):
    return render(request, 'index.html')

def logout(request):
    auth_logout(request)
    return redirect('/')

def userregister(request):
    if request.method == 'POST':
        name = request.POST.get('name')
        email = request.POST.get('email')
        password = request.POST.get('password')
        cpwd = request.POST.get('cpd')

        if CustomUser.objects.filter(email=email).exists():
            messages.error(request, "Email already exists.")
        elif password != cpwd:
            messages.error(request, "Password does not match!")
        elif name and email and password:
            user = CustomUser(name=name, email=email)
            token = get_random_string(length=32)
```

```
        user.verification_token = token
        user.is_verified = False
        user.set_password(password)
        user.is_normal= True
        user.save()
        send_mail(
            'Email Verification',
            f'Click the following link to verify your email:
{request.build_absolute_uri("/verify/")}?token={token}',
            'reportsafer@gmail.com',
            [email],
            fail_silently=False,
        )

        return redirect('/')
    return render(request, 'userregister.html')

def witnessregister(request):
    if request.method == 'POST':
        name = request.POST.get('name')
        email = request.POST.get('email')
        aadhaar = request.POST.get('aadhaarno')
        password = request.POST.get('password')
        cpwd = request.POST.get('cpd')

        if CustomUser.objects.filter(email=email).exists():
            messages.error(request, "Email already exists.")
        elif password != cpwd:
            messages.error(request, "Password does not match!")
        elif name and email and aadhaar and password:
            user = CustomUser(name=name,aadhaarno=aadhaar, email=email)
            user.set_password(password)
            user.is_witness=True
            user.save()
            return redirect('/')
    return render(request, 'witnessregister.html')

def login(request):
    if request.method == 'POST':
        email = request.POST.get('email')
        password = request.POST.get('password')
        if email and password:
            user = authenticate(request, email=email, password=password)
            if user is not None and user.is_verified:
                auth_login(request, user)
                if user.is_normal:
                    return redirect('/')
                if user.is_law:
                    return redirect('law_index')
```

```
        if user.is_prison:
            return redirect('prisonstaff')
        if user.is_control:
            return redirect('control')
        return redirect('/')
    else:
        try:
            user = CustomUser.objects.get(email=email)
            messages.error(request, "Email not Verified or Incorrect
password")
        except CustomUser.DoesNotExist:
            messages.error(request, "Email not registered")
    else:
        messages.error(request, "Please provide both email and password")
    return render(request, 'login.html')

def verify(request):
    token = request.GET.get('token')
    user = CustomUser.objects.filter(verification_token=token).first()
    if user:
        user.is_verified = True
        user.verification_token = None
        user.save()
        return redirect('/') # Redirect to login page after successful
verification
    else:
        return render(request, 'invalid_token.html') # Handle invalid token

def report_crime(request):
    location_options = ["", "Changanassery", "Chethipuzha", "Kangazha",
"Karukachal", "Kurichy", "Madappally", "Nedumkunnam", "Payippad", "Thottackad",
"Thrikkodithanam", "Vakathanam", "Vazhappally East", "Vazhappally West", "Vazhoor",
"Vellavoor", "Cheruvally", "Chirakkadavu", "Edakkunnam", "Elamgulam", "Elikkulam",
"Erumeli North", "Erumeli South", "Kanjirappally", "Koottickal", "Koovappally",
"Koruthodu", "Manimala", "Mundakkayam"]
    if request.method == 'POST':
        form = CrimeReportForm(request.POST)
        if form.is_valid():
            instance = form.save(commit=False)
            instance.list_user=request.user
            spec_station_name = request.POST['spec_station']
            reporter_location = request.POST['reporter_location']
            spec_locs = SpecLoc.objects.filter(enforcement_loc=spec_station_name)
            for i in spec_locs:
                if i.reporter_loc==reporter_location:
                    instance.spec_location = i
            instance.save()
            crime_report = form.save()
            fir_id = crime_report.id
```

```
template_path = 'report_template.html' # Path to your PDF template
context = {'crime_report': crime_report, 'fir_id': fir_id}
response = HttpResponse(content_type='application/pdf')
response['Content-Disposition'] = f'inline;
filename="Report_{fir_id}.pdf"'

template = get_template(template_path)
html = template.render(context)

# Create PDF document
pdf_response = pisa.CreatePDF(html, dest=response)

if not pdf_response.err:
    return response
    return redirect('/')
else:
    form = CrimeReportForm()

print(form.errors)
messages.error(request, form.errors)

return render(request, 'report_crime.html', {'form': form,
'location_options':location_options})

def reported_crimes(request):
    return render(request, 'reported_crimes.html')

def law_page(request):
    crime_reports = CrimeReport.objects.all()
    doc_reports = DocReport.objects.all()
    public_reports = PublicReport.objects.all()
    # Assuming you have models associated with the forms
    data1 = CrimeReport.objects.all()
    data2 = DocReport.objects.all()
    data3 = PublicReport.objects.all()
    return render(request, 'law_page.html', {
        'crime_reports': crime_reports,
        'doc_reports': doc_reports,
        'public_reports': public_reports,
        'data_from_model1': data1,
        'data_from_model2': data2,
        'data_from_model3': data3,
    })

# def generate_pdf(request):
#     if request.method == 'POST':
#         form = CrimeReportForm(request.POST)
#         if form.is_valid():
```

```
#         cr=form.save(commit=False)
#         cr.list_user=request.user
#         cr.save()
#         crime_report = form.save()
#         fir_id = crime_report.id
#         template_path = 'reported_crimes.html' # Replace with your template
path
#         context = {'form': form, 'fir_id': fir_id}
#         response = HttpResponse(content_type='application/pdf')
#         response['Content-Disposition'] = f'attachment;
filename="FIR_{fir_id}.pdf"' # Use f-string to include fir_id

#         template = get_template(template_path)
#         html = template.render(context)

#         pisa_status = pisa.CreatePDF(html, dest=response)
#         if not pisa_status.err:
#             return response
#         return redirect('/')
#     else:
#         form = CrimeReportForm()

#     return render(request, 'report_crime.html', {'form': form})

def report_doc(request):
    if request.method == 'POST':
        form = DocReportForm(request.POST,request.FILES)
        if form.is_valid():
            instance = form.save(commit=False)
            instance.list_user=request.user
            instance.save()
            doc_report = form.save()
            fir_id = doc_report.id
            template_path = 'doc_template.html' # Path to your PDF template
            context = {'doc_report': doc_report, 'fir_id': fir_id}
            response = HttpResponse(content_type='application/pdf')
            response['Content-Disposition'] = f'attachment;
filename="Report_{fir_id}.pdf"

            template = get_template(template_path)
            html = template.render(context)

            # Create PDF document
            pdf_response = pisa.CreatePDF(html, dest=response)

            if not pdf_response.err:
                return response
            return redirect('/')
        else:
```



```
        form = DocReportForm()

    print(form.errors)
    messages.error(request, form.errors)

    return render(request, 'report_doc.html', {'form': form})

    # if request.method == 'POST':
    #     # Process the form submission
    #     form = DocReportForm(request.POST, request.FILES) # Use your form if you
have one
    #     if form.is_valid():
    #         # Create a new DocumentReport object and save it to the database
    #         report = form.save(commit=False)
    #         report.user = request.user # Assuming you have a user associated
with the report
    #         report.save()

    #     # Define the context for your template
    #     context = {'report': report}
    #     messages.success(request, 'Your report has been submitted
successfully.')
    #     return redirect('index') # Redirect to the desired page after
successful submission
    #     else:
    #         messages.error(request, form.errors)
    # else:
    #     # Render the initial form
    #     form = DocReportForm()
    #     return render(request, 'report_doc.html', {'form': form})

def about(request):
    return render(request, 'about.html')

def general(request):
    return render(request, 'general.html')

def laws(request):
    return render(request, 'laws.html')

def contact(request):
    return render(request, 'contact.html')

def gallery(request):
    return render(request, 'gallery.html')
```

```
def anony_report(request):
    return render(request, 'anony_report.html')

def anony_pdf(request):
    if request.method == 'POST':
        form = AnonyReportForm(request.POST)
        if form.is_valid():
            anony_report = form.save()
            fir_id = anony_report.id
            template_path = 'reported_crimes.html'
            context = {'form': form, 'fir_id': fir_id}
            response = HttpResponse(content_type='application/pdf')
            response['Content-Disposition'] = f'inline;
filename="FIR_{fir_id}.pdf"' # Use f-string to include fir_id

            template = get_template(template_path)
            html = template.render(context)

            pisa_status = pisa.CreatePDF(html, dest=response)
            if not pisa_status.err:
                return response

        else:
            form = AnonyReportForm()

    return render(request, 'anony_report.html', {'form': form})

def report_public(request):
    location_options = ["", "Changanassery", "Chethipuzha", "Kangazha",
"Karukachal", "Kurichy", "Madappally", "Nedumkunnam", "Payippad", "Thottackad",
"Thrikkodithanam", "Vakathanam", "Vazhappally East", "Vazhappally West", "Vazhoor",
"Vellavoor", "Cheruvally", "Chirakkadavu", "Edakkunnam", "Elamgulam", "Elikkulam",
"Erumeli North", "Erumeli South", "Kanjirappally", "Koottickal", "Koovappally",
"Koruthodu", "Manimala", "Mundakkayam"]
    if request.method == 'POST':
        form = PublicForm(request.POST)
        if form.is_valid():
            instance = form.save(commit=False)
            instance.list_user=request.user
            spec_station_name = request.POST['spec_station']
            reporter_location = request.POST['reporter_location']
            spec_locs = SpecLoc.objects.filter(enforcement_loc=spec_station_name)
            for i in spec_locs:
                if i.reporter_loc==reporter_location:
                    instance.spec_location = i
            instance.save()
            public_report = form.save()
            fir_id = public_report.id
            template_path = 'public_template.html' # Path to your PDF template
```

```
        context = {'public_report': public_report, 'fir_id': fir_id}
        response = HttpResponse(content_type='application/pdf')
        response['Content-Disposition'] = f'inline;
filename="Report_{fir_id}.pdf"'

        template = get_template(template_path)
        html = template.render(context)

        # Create PDF document
        pdf_response = pisa.CreatePDF(html, dest=response)

        if not pdf_response.err:
            return response
        return redirect('/')
    else:
        form = CrimeReportForm()

        print(form.errors)
        messages.error(request, form.errors)

        return render(request, 'report_public.html', {'form': form,
'location_options':location_options})

@login_required
def listcrime(request):
    user_crime=request.user
    crime_dict=CrimeReport.objects.filter(list_user=user_crime)
    doc_dict=DocReport.objects.filter(list_user=user_crime)
    public_dict=PublicReport.objects.filter(list_user=user_crime)
    combined_reports = list(crime_dict) + list(doc_dict) +list(public_dict)
    return render(request, 'listcrime.html', {'combined_reports':
combined_reports})

def law_index(request):
    return render(request, 'law_index.html')

def law_login(request):
    return render(request, 'law_login.html')

# def law_update_status(request, crime_id):
#     if request.method == 'POST':
#         new_status = request.POST.get('status')
#         crime_report = CrimeReport.objects.get(pk=crime_id)
#         crime_report.status = new_status
#         crime_report.save()
#         return redirect('list_crimes')

def law_update_status(request):
    crime_reports = CrimeReport.objects.all()
```

```
doc_reports = DocReport.objects.all()
public_reports = PublicReport.objects.all()
# Assuming you have models associated with the forms
data1 = CrimeReport.objects.all()
data2 = DocReport.objects.all()
data3 = PublicReport.objects.all()
return render(request, 'law_update_status.html', {
    'crime_reports': crime_reports,
    'doc_reports': doc_reports,
    'public_reports': public_reports,
    'data_from_model1': data1,
    'data_from_model2': data2,
    'data_from_model3': data3,
})

def update_status(request):
    if request.method == 'POST':
        report_id = request.POST.get('report_id')
        doc_id = request.POST.get('doc_id')
        public_id = request.POST.get('public_id')
        new_status = request.POST.get('status')
        if report_id:
            report = CrimeReport.objects.get(pk=report_id)
            report.status = new_status
            report.save()

            # Redirect to the view_crime page for CrimeReport with the updated
status
            return redirect('view_crime', crime_id=report_id)

        elif doc_id:
            report_ = DocReport.objects.get(pk=doc_id)
            report_.status = new_status
            report_.save()

            # Redirect to the view_crime page for DocReport with the updated status
            return redirect('view_crime', crime_id=doc_id)

        elif public_id:
            report_p = PublicReport.objects.get(pk=public_id)
            report_p.status = new_status
            report_p.save()

            # Redirect to the view_crime page for PublicReport with the updated
status
            return redirect('view_crime', crime_id=public_id)

        # Handle invalid request method here
        return JsonResponse({'status': 'error', 'message': 'Invalid request method'})
```

```
# def update_crime_status(request):
#     if request.method == 'POST':
#         crime_id = request.POST.get('crime_id')
#         status = request.POST.get('status')

#         # Update the status in your database for the specified crime_id
#         try:
#             crime = CrimeReport.objects.get(pk=crime_id)
#             crime.status = status
#             crime.save()
#             return JsonResponse({'success': True})
#         except CrimeReport.DoesNotExist:
#             return JsonResponse({'success': False, 'error': 'Crime report not found'})

#     return JsonResponse({'success': False, 'error': 'Invalid request method'})

def check_reporter_loc(request):
    if request.method == 'GET':
        reporter_loc = request.GET.get('reporter_loc', None)
        if reporter_loc:
            try:
                # Assuming you have a SpecLoc model
                spec_location = SpecLoc.objects.get(reporter_loc=reporter_loc)
                data = {'valid': True, 'enforcement_loc':
spec_location.enforcement_loc}
            except SpecLoc.DoesNotExist:
                data = {'valid': False}
            else:
                data = {'valid': False}
            return JsonResponse(data)
        else:
            return JsonResponse({'valid': False})

def crime_category(request):
    return render(request, 'crimecategory.html')

def control(request):
    return render(request, 'control.html')

def view_crime(request, crime_id):
    try:
        task = CrimeReport.objects.get(id=crime_id)
        form = CrimeReportForm(request.POST or None, instance=task)
        try:
            evidence = EvidenceCrimeReport.objects.get(crime_idnum_id=crime_id)
```

```

        file_evidence = EvidenceCrimeForm(request.POST or None,
instance=evidence)
        return render(request, 'view_crime.html', {'form': form, 'form_id':
crime_id, 'form_status': task.status, 'file_evidence': file_evidence})
    except EvidenceCrimeReport.DoesNotExist:
        return render(request, 'view_crime.html', {'form': form, 'form_id':
crime_id, 'form_status': task.status, 'file_evidence': None})
    except CrimeReport.DoesNotExist:
        task = DocReport.objects.get(id=crime_id)
        form = DocReportForm(request.POST or None, instance=task)
        return render(request, 'view_crime.html', {'form': form, 'form_id':
crime_id})

# def up_final(request):
#     if request.method == 'POST':
#         # Create a new instance of CrimeReport and save it
#         crime_report_form = EvidenceCrimeForm(request.POST)
#         if crime_report_form.is_valid():
#             crime_report = crime_report_form.save()

#         # Get the uploaded "Final Report" file and store it in document_final
#         field
#         final_report_file = request.FILES.get('evidence_final')
#         if final_report_file:
#             evidence_report = EvidenceCrimeReport(
#                 crime_idnum=crime_report,
#                 document_final=final_report_file,
#             )
#             evidence_report.save()

#         return redirect('success_url') # Replace 'success_url' with your
#         desired URL

#     # Handle GET request or form validation errors
#     else:
#         crime_report_form = EvidenceCrimeForm()

#     return render(request, 'view_crime.html', {'crime_report_form':
# crime_report_form})
def up_final(request):
    if request.method == 'POST':
        crime = request.POST.get('crime_idnum')

        if crime is not None:
            try:
                crime_report = CrimeReport.objects.get(id=int(crime))
            except (CrimeReport.DoesNotExist, ValueError):
                crime_report = None # Handle the case where the CrimeReport
# doesn't exist or the ID is not valid

```

```
        else:
            crime_report = None
        if crime_report:
            try:
                evidence =
EvidenceCrimeReport.objects.get(crime_idnum=crime_report)
            except EvidenceCrimeReport.DoesNotExist:
                evidence = None

        form = EvidenceCrimeForm(request.POST, request.FILES,
instance=evidence)

        if form.is_valid():
            instance = form.save(commit=False)
            instance.crime_idnum = crime_report
            instance.save()
            return redirect(reverse('view_crime', kwargs={'crime_id':
crime_report.id})) # Corrected this line
        else:
            return redirect('/')
    else:
        form = EvidenceCrimeForm()

    # You should pass the crime_report ID here; this line assumes you have the ID
    from somewhere
    return redirect(reverse('view_crime', kwargs={'crime_id': crime_report.id,
'form': form})))

from django.shortcuts import render, redirect
from .models import CrimeReport, EvidenceCrimeReport
from .forms import EvidenceCrimeForm

def fir(request):
    if request.method == 'POST':
        crime_id = request.POST.get('crime_idnum')
        crime_report = CrimeReport.objects.get(id=crime_id) # Assuming the
crime_id is valid
        try:
            evidence =
EvidenceCrimeReport.objects.get(crime_idnum=crime_report)
        except EvidenceCrimeReport.DoesNotExist:
            evidence = None
        # Create an instance of the EvidenceCrimeForm
        form = EvidenceCrimeForm(request.POST, request.FILES,instance=evidence)

        if form.is_valid():
            # Save the form data without committing to the database
            instance = form.save(commit=False)
```

```
        instance.crime_idnum = crime_report # Associate the evidence with the
crime report
        instance.save() # Commit to the database

        # Update the status of the crime report to 'Preliminary Investigation
completed'
        crime_report.status = 'Preliminary Investigation completed'
        crime_report.save()

        return redirect('view_crime', crime_id=crime_id)
    else:
        form = EvidenceCrimeForm()

    return render(request, 'view.html', {'form': form})

def witness(request):
    if request.method == 'POST':
        crime_id = request.POST.get('crime_idnum')
        crime_report = CrimeReport.objects.get(id=crime_id) # Assuming the
crime_id is valid

        try:
            evidence =
EvidenceCrimeReport.objects.get(crime_idnum=crime_report)
        except EvidenceCrimeReport.DoesNotExist:
            evidence = None
        # Create an instance of the EvidenceCrimeForm
        form = EvidenceCrimeForm(request.POST, request.FILES, instance=evidence)

        if form.is_valid():
            # Save the form data without committing to the database
            instance = form.save(commit=False)
            instance.crime_idnum = crime_report # Associate the evidence with the
crime report
            instance.save() # Commit to the database

            # Update the status of the crime report to 'Preliminary Investigation
completed'
            crime_report.witness()
            crime_report.save()

            return redirect('view_crime', crime_id=crime_id)
        else:
            form = EvidenceCrimeForm()

    return render(request, 'view.html', {'form': form})

def forensic(request):
    if request.method == 'POST':
```



```
        crime_id = request.POST.get('crime_idnum')
        crime_report = CrimeReport.objects.get(id=crime_id) # Assuming the
crime_id is valid

        try:
            evidence =
EvidenceCrimeReport.objects.get(crime_idnum=crime_report)
        except EvidenceCrimeReport.DoesNotExist:
            evidence = None
        # Create an instance of the EvidenceCrimeForm
        form = EvidenceCrimeForm(request.POST, request.FILES,instance=evidence)

        if form.is_valid():
            # Save the form data without committing to the database
            instance = form.save(commit=False)
            instance.crime_idnum = crime_report # Associate the evidence with the
crime report
            instance.save() # Commit to the database

            # Update the status of the crime report to 'Preliminary Investigation
completed'
            crime_report.forensic()
            crime_report.save()

            return redirect('view_crime', crime_id=crime_id)
        else:
            form = EvidenceCrimeForm()

        return render(request, 'view.html', {'form': form})
def arrest(request):
    if request.method == 'POST':
        crime_id = request.POST.get('crime_idnum')
        crime_report = CrimeReport.objects.get(id=crime_id) # Assuming the
crime_id is valid

        try:
            evidence =
EvidenceCrimeReport.objects.get(crime_idnum=crime_report)
        except EvidenceCrimeReport.DoesNotExist:
            evidence = None
        # Create an instance of the EvidenceCrimeForm
        form = EvidenceCrimeForm(request.POST, request.FILES,instance=evidence)

        if form.is_valid():
            # Save the form data without committing to the database
            instance = form.save(commit=False)
            instance.crime_idnum = crime_report # Associate the evidence with the
crime report
            instance.save() # Commit to the database
```

```
        # Update the status of the crime report to 'Preliminary Investigation
completed'
        crime_report.arrest()
        crime_report.save()

        return redirect('view_crime', crime_id=crime_id)
    else:
        form = EvidenceCrimeForm()

        return render(request, 'view.html', {'form': form})
def charge(request):
    if request.method == 'POST':
        crime_id = request.POST.get('crime_idnum')
        crime_report = CrimeReport.objects.get(id=crime_id) # Assuming the
crime_id is valid

        try:
            evidence =
EvidenceCrimeReport.objects.get(crime_idnum=crime_report)
        except EvidenceCrimeReport.DoesNotExist:
            evidence = None
        # Create an instance of the EvidenceCrimeForm
        form = EvidenceCrimeForm(request.POST, request.FILES, instance=evidence)

        if form.is_valid():
            # Save the form data without committing to the database
            instance = form.save(commit=False)
            instance.crime_idnum = crime_report # Associate the evidence with the
crime report
            instance.save() # Commit to the database

            # Update the status of the crime report to 'Preliminary Investigation
completed'
            crime_report.charge()
            crime_report.save()

            return redirect('view_crime', crime_id=crime_id)
        else:
            form = EvidenceCrimeForm()

            return render(request, 'view.html', {'form': form})
def case(request):
    if request.method == 'POST':
        crime_id = request.POST.get('crime_idnum')
        crime_report = CrimeReport.objects.get(id=crime_id) # Assuming the
crime_id is valid

        try:
```

```
        evidence =
EvidenceCrimeReport.objects.get(crime_idnum=crime_report)
    except EvidenceCrimeReport.DoesNotExist:
        evidence = None
    # Create an instance of the EvidenceCrimeForm
    form = EvidenceCrimeForm(request.POST, request.FILES, instance=evidence)

    if form.is_valid():
        # Save the form data without committing to the database
        instance = form.save(commit=False)
        instance.crime_idnum = crime_report # Associate the evidence with the
crime report
        instance.save() # Commit to the database

        # Update the status of the crime report to 'Preliminary Investigation
completed'
        crime_report.case()
        crime_report.save()

        return redirect('view_crime', crime_id=crime_id)
    else:
        form = EvidenceCrimeForm()

    return render(request, 'view.html', {'form': form})
def final(request):
    if request.method == 'POST':
        crime_id = request.POST.get('crime_idnum')
        crime_report = CrimeReport.objects.get(id=crime_id) # Assuming the
crime_id is valid

        try:
            evidence =
EvidenceCrimeReport.objects.get(crime_idnum=crime_report)
        except EvidenceCrimeReport.DoesNotExist:
            evidence = None
        # Create an instance of the EvidenceCrimeForm
        form = EvidenceCrimeForm(request.POST, request.FILES, instance=evidence)

        if form.is_valid():
            # Save the form data without committing to the database
            instance = form.save(commit=False)
            instance.crime_idnum = crime_report # Associate the evidence with the
crime report
            instance.save() # Commit to the database

            # Update the status of the crime report to 'Preliminary Investigation
completed'
            crime_report.final()
            crime_report.save()
```

```
        return redirect('view_crime', crime_id=crime_id)
    else:
        form = EvidenceCrimeForm()

        return render(request, 'view.html', {'form': form})
# Import your models and forms here

# def up_final(request):
#     if request.method == 'POST':
#         form = EvidenceCrimeForm(request.POST, request.FILES)
#         crime_id = request.POST.get('crime_idnum')

#         try:
#             crime_report = CrimeReport.objects.get(id=int(crime_id))
#         except (CrimeReport.DoesNotExist, ValueError):
#             crime_report = None # Handle the case where the CrimeReport doesn't
# exist or the ID is not valid

#         if crime_report:
#             # Try to get an existing EvidenceCrimeReport object for the given
# crime_id
#             try:
#                 evidence_report =
EvidenceCrimeReport.objects.get(crime_idnum=crime_report)
#                 # Update the existing object's fields with the form data
#                 form = EvidenceCrimeForm(request.POST, request.FILES,
instance=evidence_report)
#                 except EvidenceCrimeReport.DoesNotExist:
#                     evidence_report = None # Handle the case where the
EvidenceCrimeReport doesn't exist

#                 if form.is_valid():
#                     instance = form.save(commit=False)
#                     instance.crime_idnum = crime_report
#                     instance.save()
#                     return redirect(reverse('view_crime', kwargs={'crime_id':
crime_report.id}))

#         else:
#             # Handle the case where the CrimeReport doesn't exist or the ID is
not valid
#             raise Http404("CrimeReport not found")

#     else:
#         form = EvidenceCrimeForm()
#         return render(request, 'view_crime.html', {'form': form})
```

```
def view_doc(request, crime_id):
    try:
        task=DocReport.objects.get(id=crime_id)
        form=DocReportForm(request.POST or None, instance=task)
        return render(request, 'view_doc.html', {'form':form})
    except DocReport.DoesNotExist:
        task=DocReport.objects.get(id=crime_id)
        form=DocReportForm(request.POST or None, instance=task)
        return render(request, 'view_doc.html', {'form':form})

def view_public(request, crime_id):
    try:
        task=PublicReport.objects.get(id=crime_id)
        form=PublicForm(request.POST or None, instance=task)
        return render(request, 'view_public.html', {'form':form})
    except PublicReport.DoesNotExist:
        task=PublicReport.objects.get(id=crime_id)
        form=PublicForm(request.POST or None, instance=task)
        return render(request, 'view_public.html', {'form':form})

def upload_evidence(request):
    if request.method == 'POST' and request.FILES.get('evidence_image_label'):
        uploaded_file = request.FILES['evidence_image_label']

        # Ensure that the CrimeReport with the given ID exists
        crime_report_id = request.POST.get('crime_report_id')
        try:
            crime_report = CrimeReport.objects.get(id=crime_report_id)
        except CrimeReport.DoesNotExist:
            return JsonResponse({'message': 'CrimeReport does not exist'},
                                status=400)

        fir_file = FIRFile(crime_report=crime_report, file=uploaded_file)
        fir_file.save()

        return JsonResponse({'message': 'File uploaded successfully'})

    return JsonResponse({'message': 'File upload failed'}, status=400)

def prisonstaff(request):
    return render(request, 'prisonstaff.html')

def control_page(request):
    crime_reports = CrimeReport.objects.all()
    doc_reports = DocReport.objects.all()
    public_reports = PublicReport.objects.all()
    # Assuming you have models associated with the forms
    data1 = CrimeReport.objects.all()
    data2 = DocReport.objects.all()
```

```
data3 = PublicReport.objects.all()
return render(request, 'control_page.html', {
    'crime_reports': crime_reports,
    'doc_reports': doc_reports,
    'public_reports': public_reports,
    'data_from_model1': data1,
    'data_from_model2': data2,
    'data_from_model3': data3,
})

def control_status(request):
    crime_reports = CrimeReport.objects.all()
    doc_reports = DocReport.objects.all()
    public_reports = PublicReport.objects.all()
    # Assuming you have models associated with the forms
    data1 = CrimeReport.objects.all()
    data2 = DocReport.objects.all()
    data3 = PublicReport.objects.all()
    return render(request, 'control_status.html', {
        'crime_reports': crime_reports,
        'doc_reports': doc_reports,
        'public_reports': public_reports,
        'data_from_model1': data1,
        'data_from_model2': data2,
        'data_from_model3': data3,
    })

def report_prison(request):
    form = PrisonReportForm(request.POST)

    pri = form.save(commit=False)
    print(form.errors)
    # Set the organizer to the currently logged-in user
    pri.org_user = request.user
    pri.save() # Commit the webinar to the database

    # Process speakers
    inmate_name = request.POST.getlist('inmate_name[]')
    inmate_id = request.POST.getlist('inmate_id[]')
    for i in range(len(inmate_name)):
        inmate = Inmate.objects.create(
            inmate_name = inmate_name[i],
            inmate_id = inmate_id[i]
        )
        pri.inmates.add(inmate)
    return render(request, 'report_prison.html')

from django.utils import timezone
from django.core.mail import send_mail
```

```
from django.contrib import messages
from django.shortcuts import render, redirect
from .models import Appointment
from django.contrib import messages

def book_appointment(request):
    inmates = Inmate.objects.all()
    if request.method == 'POST':
        in_id = request.POST.get('in_name')
        time = request.POST.get('timet')
        date = request.POST.get('dated')
        # Check if the selected time slot is available and other validation if
needed
        # If the time slot is available, create an Appointment instance and save it
        appointment = Appointment(ap_name=in_id, time_slot=time, date=date)
        print(appointment)
        appointment.save()

        # You can also add additional logic, such as sending confirmation emails

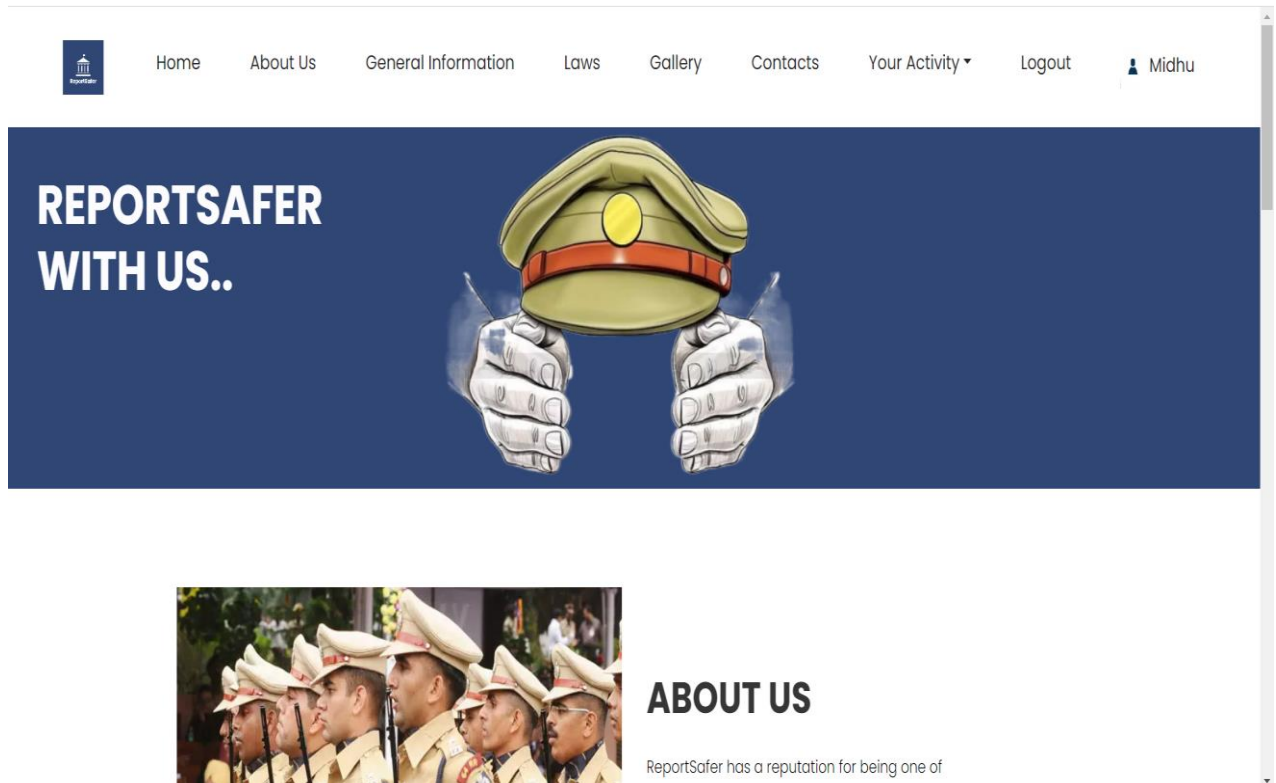
        # Add a success message
        messages.success(request, 'Appointment booked successfully.')

        return redirect('book_appointment') # Redirect to the same page or another
page
    else:
        # Handle GET requests if needed
        return render(request, 'book_appointment.html', {'inmates': inmates})

def appointment_view(request):
    appointments = Appointment.objects.all() # Retrieve all appointments
    return render(request, 'view_appointment.html', {'appointments': appointments})
```

9.2 SCREEN SHOTS


User: Home Page



User: Crime Category Selection Page

The screenshot shows the Crime Category Selection Page. It features the same navigation bar as the home page. The main content area has a light blue background. At the top, it says 'Select the Crime Category:'. Below this is a white dropdown menu with the text 'Select the Category' and a downward arrow. Further down, there is a paragraph of text: 'By using our online crime reporting service, you agree to report accurate and truthful information only. We do not guarantee the accuracy of user-provided content and are not responsible for any consequences resulting from its use. We may moderate and remove inappropriate content. All platform content is protected by copyright, and you may not use it without permission. We reserve the right to modify these terms at any time. I will provide all the known facts and details about the incident as I can while reporting. If you have questions, please contact us. By submitting a report, you acknowledge your agreement with these terms.' Below this text is a checkbox labeled 'I agree to the Terms and Conditions'. At the bottom right of the form is a dark blue button with the word 'Submit' in white.

Law Enforcement: Crime status viewing Page



Home


Laws

Gallery

Contacts

Your Activity

Logout




 Kanjirappally Law Enforcement

MANAGE STATUS OF THE CRIME

CRIME NUMBER	CRIME STATUS	ACTIONS
CNHB/1	Arrest and Detention	<div>View documents</div>
CNHB/26	Case Closed	<div>View documents</div>
CNHB/28	Inquiry and investigation completed	<div>View documents</div>
CNHB/37	Arrest and Detention	<div>View documents</div>
CNHB/38	Inquiry and investigation in progress	<div>View documents</div>
CNHB/39	Preliminary investigation completed	<div>View documents</div>
CNHB/40	Case Closure in progress	<div>View documents</div>
CNHB/41	Crime Reported	<div>View documents</div>

Law Enforcement: Document Upload Page

	Home	Laws	Gallery	Contacts	Your Activity ▾	Logout	Kanjirappally Law Enforcement
UPLOAD DOCUMENTS RELATED TO EACH STAGE OF INVESTIGATION							
<div>FIR: Currently: evidence_fir/ASSIGNMENT_1.pdf <input type="checkbox"/> Clear Change: <input type="button" value="Choose File"/> No file chosen <input type="button" value="Upload"/></div>							
<div>Witness Statement: Currently: evidence_witness/Report_8.pdf <input type="checkbox"/> Clear Change: <input type="button" value="Choose File"/> No file chosen <input type="button" value="Upload"/></div>							
<div>Forensic Report:</div>							


User: Report Offence related to Documents Page

The screenshot shows a web browser window with the URL `127.0.0.1:8000/report_doc/`. The page has a navigation bar with links: Home, About Us, General Information, Laws, Gallery, Contacts, Your Activity, Logout, and a user profile for Midhu. The main content area is a form titled 'Report Offence related to Documents'. The form is divided into two columns. The left column contains: 'Informer Name' (text input with 'Midhu'), 'Your e-mail ID' (text input), 'Please provide a description of the Incident *' (text area with placeholder 'A detailed description of what happened.'), and 'Please provide a description of the Offender' (text area with placeholder 'If known, provide details about the offender, including their name, physical appearance, age, and'). The right column contains: 'Your Aadhar Number *' (text input), 'Any kind of Threat or Violence faced' (radio buttons for 'Yes' and 'No', with a 'Clear Selection' button), 'Please provide a description of the Victim' (text area with placeholder 'Provide information about the victim, including their name, age, gender, and contact details.'), and 'Witness Information' (text area with placeholder 'Include details of any witnesses to the offense, including their names, contact information, and').


User: Report Offence related to Human body Page


The screenshot shows a web browser window with the URL `127.0.0.1:8000/report_crime/`. The page has a navigation bar with links: Home, About Us, General Information, Laws, Gallery, Contacts, Your Activity, Logout, and a user profile for Midhu. The main content area is a form titled 'Offence Against Human Body'. The form is divided into two columns. The left column contains: 'Informer Name' (text input with 'Midhu'), 'Location of Incidence *' (dropdown menu), 'Type of Offence' (dropdown menu with 'Select type'), and 'Please provide a description of the Incident *' (text area with placeholder 'A detailed description of what happened.'). The right column contains: 'Your Aadhar Number *' (text input), 'Your Specified Station' (text input), 'Date and Time of Incidence' (text input with placeholder 'dd-mm-yyyy --:--' and a calendar icon), and 'Please provide a description of the Victim' (text area with placeholder 'If the victim is someone else, provide informations about them including their name, age, gender, and').

Prison Warden: Home Page

[Home](#)[Laws](#)[Gallery](#)[Contacts](#)[Your Activity ▾](#)[Logout](#)[Prison Warden](#)

REPORTSAFER WITH US..





ABOUT US

ReportSafer has a reputation for being one of the best-managed law-enforcement forces in

User: Report Offence related to Public Tranquility Page

You are signed in as mca

Amal Jyothi College of Engineering

Academic Enterprise Solutions

Future Scope: Tech & Safety

ReportSafer | Offence relating to public tranquility

127.0.0.1:8000/report_public/

Gmail

YouTube

Maps

News

Translate

Avast 2020

Jujutsu Kaisen Subti...


fa-star-o: Font Awe...

How To Make Ecom...

Login - VMware IT...

Open-Source UI ele...

All Bookmarks

[Home](#)[About Us](#)[General Information](#)[Laws](#)[Gallery](#)[Contacts](#)[Your Activity ▾](#)[Logout](#)[Midhu](#)

Offence relating to public tranquility

Informer Name	Your Aadhar Number *
<input type="text" value="Midhu"/>	<input type="text"/>
Your e-mail ID	Date and Time of Incidence
<input type="text"/>	<input type="text" value="dd-mm-yyyy --:--"/>
Location of Incidence *	Your Specified Station
<input type="text"/>	<input type="text"/>
Type of Offence	
<input type="text" value="Select type"/>	
Please provide a description of the Incident *	Please provide a description of the Victim