A NEW APPROACH IN FORENSIC BLOCKCHAIN

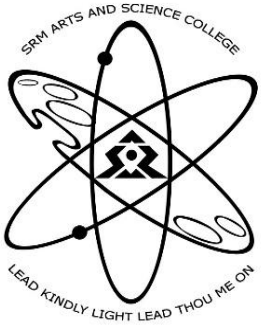
A Project submitted to the University of Madras in partial fulfillment of requirements for the award of the Degree of Master of Computer Applications

Submitted by

ADHULYA F
(Register number: 812200208)

Under the Guidance of

MR S SIVAKUMAR M.C.A., M.Phil., SET.
Assistant Professor
Department of MCA



SRM ARTS AND SCIENCE COLLEGE
KATTANKULATHUR – 603 203

UNIVERSITY OF MADRAS
APRIL 2024

BONAFIDE CERTIFICATE

This is to certify that the Group project report entitled

" A NEW APPROACH IN FORENSIC BLOCKCHAIN"

being submitted to the University of Madras, Chennai - 600 005

by

ADHULYA F

(Register number: 812200208)

for the partial fulfillment for the award of Degree of

MASTER OF COMPUTER APPLICATIONS

is the bonafide record work carried by him/her,

under my guidance and supervision

Signature of the Guide **Head of the Department**

Submitted for the Viva-Voce Examination held on ……..……..at SRM Arts and Science College, Kattankulathur – 603 203.

Date: **Examiners:**

1.

2.

Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

CIN: U80902TN2021PTC141464

www.pantechelearning.com

01/05/2024

TO WHOM IT MAY CONCERN

This is to certify that ADHULYA F (812200208) student of final year MCA from SRM Arts and Science College, Kattankulathur, has successfully completed project work entitled as **"A NEW APPROACH IN FORENSIC BLOCKCHAIN"** using **BLOCKCHAIN** in our organization from December 22, 2023 to April 30, 2024.

This certificate is awarded as recognition of the dedication, hard work, and successful completion of the project.

Thanks & Regards,
Pantech e learning
Chennai-17

Authorized Signatory

ACKNOWLEDGEMENT

I would like to express my thanks to **Dr. T. R. PAARIVENDHAR, B.Sc., M.I.E,** Chairman and **Dr. R. VASUDEVARAJ, M.Com., M.Phil., B.Ed., MBA, Ph.D.,** Principal of our college, for their scholarly guidance and motivation throughout the project work as well as the curriculum.

My sincere thanks to **Dr. K. SARAVANANATHAN M.Sc., M.Phil., Ph.D.** Head of the Department of MCA for the useful guidance and helpful mind to complete this project.

My sincere thanks goes to my guide **MR S SIVAKUMAR M.C.A., M.Phil., SET.** Assistant Professor, Department of MCA, for suggesting the problem, offering inspiring guidance and fruitful discussion throughout the course of the work.

My special thanks to all the staff members, Department of MCA for their support and encouragement for the successful completion of the project.

I acknowledge my profound thanks to my parents for their encouragement and for social and economical support for completing this project work.

ADHULYA

Abstract

A New Approach in Forensic Blockchain introduces a groundbreaking framework designed to safeguard forensic reports using blockchain technology. This approach revolutionizes the storage of investigated data, ensuring its security against unauthorized access. By leveraging blockchain technology, the history of victim records is securely stored in a digital format on distributed networks encrypted to minimize the risk of intrusion. The framework comprises several modules, including Main Page, User Main page, Login, Signup, Forensic Records, Case Details, Investigation To-Do List, Panel Main page, Report Incident, Success Confirmation, and Search functionality. These modules are seamlessly integrated into a user-friendly interface, enhancing operational efficiency. Through the fusion of blockchain technology and web development, this approach optimizes forensic record management by capitalizing on their synergistic relationship. It underscores how the integration of these technologies augments existing methodologies, thereby advancing the field of forensic investigation and data management

1. INTRODUCTION

1.1 About the Project

A New Approach in Forensic Blockchain is an innovative paradigm in forensic blockchain, focusing on enhancing investigative capabilities. Forensic blockchain in a specialized area in the field of blockchain technology that aims to facilate digital forensic investigations by using blockchain technology. It involves the use of Blockchain to create a tamper-proof and secure leger in digital investigation, makes it easier to track and secure way of storing forensic reports in digital form. This integrated system serves as a tool for maintaining and representing the chain of custody crucial throughout an investigation. Forensic blockchain involves storing the history of forensic cases in a digitalized form on blockchains, which are subsequently stored on remote servers. These records are highly encrypted, reducing the probability of unauthorized access. The use of blockchain technology ensures a secure and tamper-proof record of transactions, allowing investigators to trace the movement of funds and assets across distributed ledgers. This approach provides robust support for forensic investigations, facilitating the tracking and verification of critical information related to criminal cases.

1.2 About the Company

Pantech Solutions, established in 2004 by Jeevarajan MK, has swiftly emerged as a prominent figure in the realm of technical training, products, and research initiatives. Originating from humble beginnings with a team of engineering students, the company's focus on providing innovative technological solutions, particularly in electronics and IT, has been unwavering. Offering a comprehensive range of products and services including engineering lab equipment, electronic hobby kits, and technical training resources, Pantech Solutions caters to a diverse clientele across various industries and educational institutions globally. Guided by core values of integrity, candor, service, kindness, competence, and growth, the company strives to achieve global leadership through sustained innovation. Pantech Solutions' dedication to innovation, quality, and customer satisfaction underscores its position as a leading player in the technical solutions landscape. More information about their offerings can be found on their website Pantech Solutions.

1.3 Module Description

There are 11 Different Modules

- Mainpage Module
- User Mainpage Module
- Login Module
- Signup Module
- Forensic Records Module
- Case Details Module
- Investigation To Do Module
- Panel Mainpage Module
- Report Incident Module
- Success Module
- Search Module

1.3.1 Main page Module

This Main page module is built by using html and css. and it is connected with 2 Modules that are user and panel. The Main page Module serves as the central interface, facilitating navigation and access to other modules.

1.3.2 User Mainpage Module

The Mainpage Module serves as the central interface, facilitating navigation and access to other user modules. And it is connected with 3 Modules that are Add Forensic Details Module, Case Details Module and Investigation To-Do Module .

1.3.3 Login Module

The Login Module ensures secure user authentication and authorization, safeguarding system integrity. User needs to provide a correct username and a password, which user enters while registering, in order to login into the site. If information provided by the user matches with the data in the database table then user successfully login into the app else message of login failed is displayed and user need to re-enter correct information. A link to the register activity is also provided for registration of new users.

1.3.4 Sign Up Module

The SignUp module manages user registration, collecting and validating information to create new accounts securely. A new user who wants to access the site needs to register first before login. By clicking on register button in login activity, the register activity gets open. A new user registers by entering full name, username, email, and password. When user enters the information in all textboxes, on the click of register button, the data is transferred to database and user is directed to login activity again.

1.3.5 Add Forensic Details Module

The Forensic Records Module is dedicated to store the Forensic data. Here the Forensic Details are filled such as record ID, Case number, Victim's Name , Age, Gender, Physique, Unique Features, External Injury, Internal Injury, Cause of Death, Evidence, Examination result and these things can be viewed in User Case Details and also in Panel Search it can be Searched using Case Number.

1.3.6 Case Details Module

The Case Details Module offers comprehensive insights providing detailed information. Here the Added Forensic Details can be viewed in Case Detail Module. The record ID, Case number, Victim's Name , Age, Gender, Physique, Unique Features, External Injury, Internal Injury, Cause of Death, Evidence, Examination result are displayed in table format

1.3.7 Investigation To Do Module

The Investigation To Do Module helps investigators manage their tasks and priorities efficiently, ensuring smooth progress in ongoing investigations.

1.3.8 Panel Main Page Module

The Panel Mainpage Module serves as the central hub facilitating navigation and access to another panel module. It is connected with 2 Modules that are Report Incident Module and Search Module

1.3.9 Report Incident Module

The Report Incident Module enables the panel memebers to report incidents promptly and accurately. Reporting Incident will help the Forensic Members to investigate the case and store those Report.

1.3.10 Success Module

The Success Module shows when the Incident is reported it Displays “Submitted Sucessfully” with an reference Number. With this Reference number you can search the case and check the status of the case

1.3.11 Search Module

The Search Module facilitates quick and efficient retrieval of information. It helps the Panel Members to view the case Details

2. SYSTEM STUDY

2.1 EXISTING SYSTEM

In the Past System, the data is stored in Ledger and here the data can go missing or can be changed such issues is being arised.

In all the Existing application runs on single centralized server and if this server hack or crash due to request overload then the services will not be available. As technology evolves, the imperative to enhance data management practices becomes increasingly evident.

To over- come from this problem we can apply decentralized Blockchain technology which will maintain data at multiple nodes or server and if one node down then users can get service from other working nodes.

Some of them are Drawbacks of the Existing System

- Highly susceptible to security risks
- Exposing valuable data to potential breaches
- Unauthorized access and malicious manipulation.
- Costly maintenance, and inefficient retrieval
- Slow performance and limited collaboration
- Scalability poses a significant challenge, as traditional storage systems often struggle to accommodate the exponential growth of data in contemporary digital landscapes.

2.2 PROPOSED SYSTEM

Blockchain store data as blocks/transactions and associate each block with unique hash code and before storing any new block then Blockchain verify hash code of all existing blocks and if blocks are not attack or alter then it will return same hash code and verifcation will get successful and then only Blockchain will store new block. If verification failed then Blockchain will not store any new data so Blockchain consider as immutable which cannot be attack or alter its data from backend. If alter then verifcation get failed.

Advantages of the Proposed System

- Improved analytical capabilities through the integration of web development technologies in forensic blockchain.
- Enhanced interpretability of complex blockchain transactions.
- Streamlined forensic investigations with advanced algorithms.
- Data stored securely

2.3 FEASIBILITY REPORT

Feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort and the time that spend on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources. Thus when a new application is proposed it normally goes through a feasibility study before it is approved for development. The document provide the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as Technical, Economic, Legal, Scheduling, Resource, Market and Operational feasibilities. The following are its features.

2.3.1 Technical Feasibility

- Assessment of the scalability of HTML, CSS, JS, Python, and Flask for handling large volumes of blockchain data.
- Compatibility testing with various blockchain platforms and protocols to ensure seamless integration.
- Evaluation of the availability of necessary libraries, frameworks, and APIs to facilitate integration.
- Consideration of the complexity of implementing features like data encryption, smart contracts, and decentralized storage.

2.3.2 Economic Feasibility

- Detailed estimation of development costs, including software licenses, developer salaries, and infrastructure expenses.

- Analysis of potential cost savings or revenue generation opportunities resulting from improved forensic analysis and streamlined investigations.
- Calculation of the return on investment (ROI) based on anticipated benefits and projected costs over time.
- Exploration of potential funding sources, such as grants, investors, or government initiatives supporting innovation in forensic technologies.

2.3.3 Legal Feasibility

- Review of existing laws and regulations governing data privacy, security, and intellectual property rights in the jurisdictions where the system will be deployed.
- Collaboration with legal experts to ensure compliance with industry-specific regulations, such as GDPR for handling personal data.
- Identification of any legal risks associated with integrating web development technologies into forensic block chain, such as liability for data breaches or non-compliance with regulatory requirements.

2.3.4 Operational Feasibility

- Assessment of the impact on existing forensic investigation processes and workflows.
- Identification of potential training needs for forensic investigators to adapt to the new technology.
- Consideration of user acceptance and stakeholder buy-in for the integration of new tools and methodologies.
- Evaluation of the system's reliability, availability, and performance under real-world operational conditions.

2.3.5 Scheduling Feasibility

- Development of a detailed project plan outlining milestones, deliverables, and dependencies.
- Identification of potential risks and mitigation strategies to ensure timely completion of the integration.
- Consideration of external factors, such as regulatory changes or technology updates, that may impact the project timeline.
- Regular monitoring and adjustment of the schedule to accommodate unforeseen challenges or delays.

2.3.6 Resourse Feasibility

- Assessment of the availability of skilled personnel with expertise in both web development and blockchain technologies.
- Identification of potential training or recruitment needs to fill any skill gaps within the development team.
- Evaluation of hardware and software requirements, including server infrastructure, database systems, and development tools.
- Consideration of long-term resource sustainability and scalability to support future growth and expansion of the system.

2.3.7 Market Feasibility

- Market research to assess the demand for advanced forensic analysis tools and techniques among law enforcement agencies, regulatory bodies, and private sector organizations.
- Analysis of competitive landscape, including existing solutions and emerging technologies in the forensic block chain space.
- Identification of potential partnerships or collaborations with industry stakeholders to drive market adoption and penetration.
- Consideration of user feedback and market trends to inform product development and marketing strategies.

3.SOFTWARE PROJECT PLAN

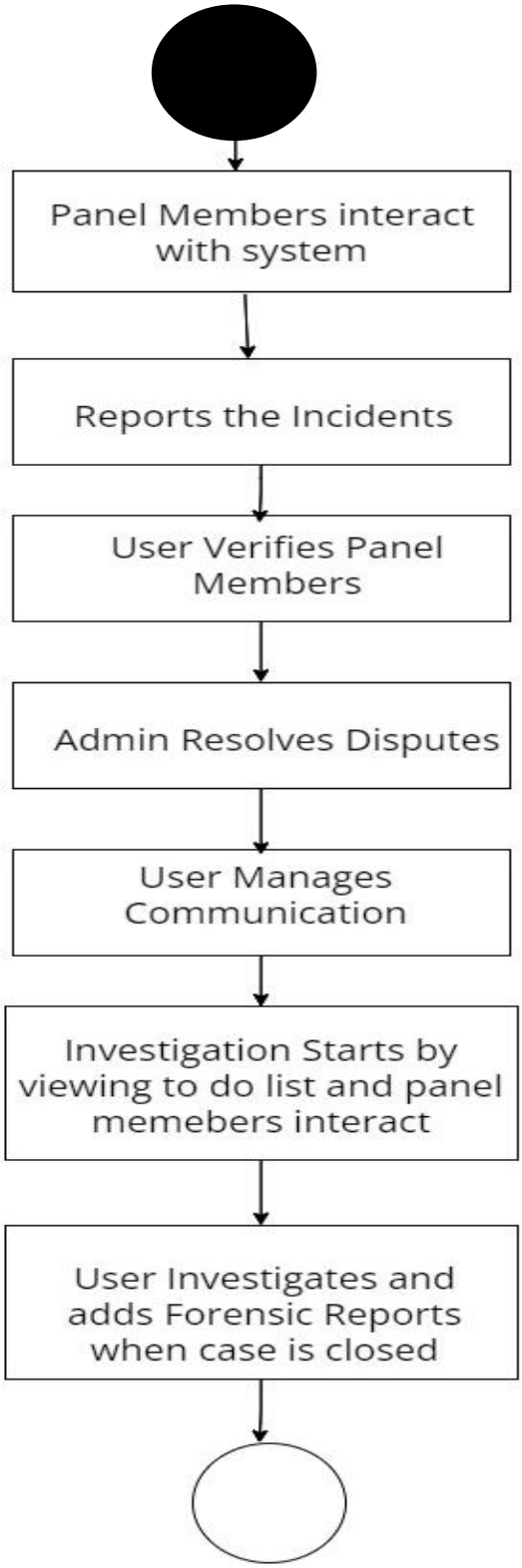


Figure 3.1: Business Diagram

4. SYSTEM ANALYSIS

4.1 DFD Diagram

4.1.1 Definition

DFD is the abbreviation for **Data Flow Diagram**. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart. It is a graphical tool, useful for communicating with users, managers and other personnel. It is useful for analysing existing as well as proposed system.

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyse an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually "say" things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

It provides an overview of

- What data is system processes.
- What transformation are performed.
- What data are stored?

What results are produced, etc.

Data Flow Diagram can be represented in several ways. The DFD belongs to structured-analysis modelling tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.

4.1.2 Features

Data Flow Diagrams (DFDs) are a graphical tool used to represent the flow of data within a system or a process. DFDs are known for their simplicity and effectiveness in illustrating the interactions and transformations of data within a system. Here are some key features of DFDs:

- **Processes:** DFDs represent processes or activities that manipulate data. Each process is labelled with a descriptive verb, indicating the action being performed. These processes are the "workhorses" of the diagram, showing how data is transformed.
- **Data Flows:** Data flows represent the paths along which data travels. They connect processes, data stores, and external entities. Data flows are labelled to describe the type of data being transmitted, such as "User Details" or "Panel Details."
- **Data Stores:** Data stores represent repositories where data is stored within the system. This can include databases, files, or other storage mechanisms. Data stores are labelled with their names, such as "Bookdata Database" or "Panel Records."
- **External Entities:** External entities represent sources or destinations of data that interact with the system but are not considered part of the system itself. These can include users, customers, or other external systems. External entities are labelled with their names, such as "Panel" or "User."
- **Data Flow Labels:** Data flows are often labelled to specify the content of the data being transferred. Labels help in clearly defining what is being transmitted between elements in the diagram.
- **Levels:** DFDs can be organized into multiple levels to provide a hierarchical view of a system. The highest level, known as the Context Diagram (Level 0), provides an overview of the entire system and its interactions with external entities. Subsequent levels (Level 1, Level 2, etc.) offer more detailed views of specific processes and data flows within the system.
- **Simplicity:** DFDs are known for their simplicity. They remove unnecessary details and focus on the essential aspects of data flow and processing within a system, making them easy to understand for a wide range of stakeholders.
- **Visual Representation:** DFDs use visual symbols, such as rectangles, arrows, and labels, to represent system components and their relationships.
- **Communication Tool:** DFDs serve as a common language for communication between stakeholders, including business analysts, system designers, and developers.

They help ensure that all parties have a shared understanding of how data moves within a system.

- **Documentation Tool:** DFDs are valuable for documenting system requirements, design specifications, and system behaviour. They provide a structured way to capture and convey information about data flow and processing.
- **Analytical Tool:** DFDs are used in system analysis and design to identify potential issues, bottlenecks, or inefficiencies in data flow and processing. They aid in problem-solving and optimization.
- **Hierarchy:** DFDs can be organized in a hierarchy with different levels to provide both an overview of the entire system and detailed views of specific aspects of the system.
- **Clarity:** DFDs promote clarity and understanding by simplifying complex systems into a structured and easy-to-follow visual representation.

Data Flow Diagrams are a valuable tool in system analysis, design, and communication, helping stakeholders understand how data is processed within a system and how different system components interact.

4.1.3 Symbols used

1. Entity

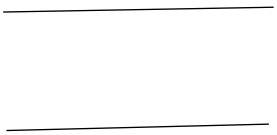
Also known as eternal entities, they are often placed at the source or the terminal ends of the diagram. They provide a real-world representation of things in the diagram and are responsible for sending or receiving data. For instance, in the above data flow diagram, User and Panel are entities.

2. Start and End

It is one of the most important data flow diagram symbols that depict the processing of data at any stage. A process would have incoming data, a management operation, and an output. It

usually provides logical management or change in data. In the diagram, you can see that payment collection or order placement are some of the vital processes.

3. Data Store

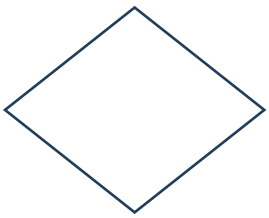


It simply depicts any location where data is stored in the system. It can be a document or an entire database that is used as a repository to store and handle data. In Data Flow Diagrams, Data Stores can have an input to collect data and an optional output to provide the stored data to any other entity or process. In our diagram, customers, userdata ,panel , Record , Bookdata are some of its data storage units.

4. Data Flow

Lastly, any data flow diagram would be incomplete without representing the flow of data from one entity/process to another. As you can see in the diagram, we use directional lines to depict how the data is moving from one place to another. At times, we also add captions on data flow arrows to make others understand these data flow diagram symbols and their meanings.

5. Inspection



An inspection data flow diagram serves as a visual tool for understanding how the inspection process operates, from data input to quality assessments and reporting. It helps stakeholders, including quality control personnel, management, and system designers, gain a clear understanding of how inspection data is managed and processed within the context of a specific inspection or quality control system.

4.1.4 DFD

4.1.5 Level diagram

Level 0:

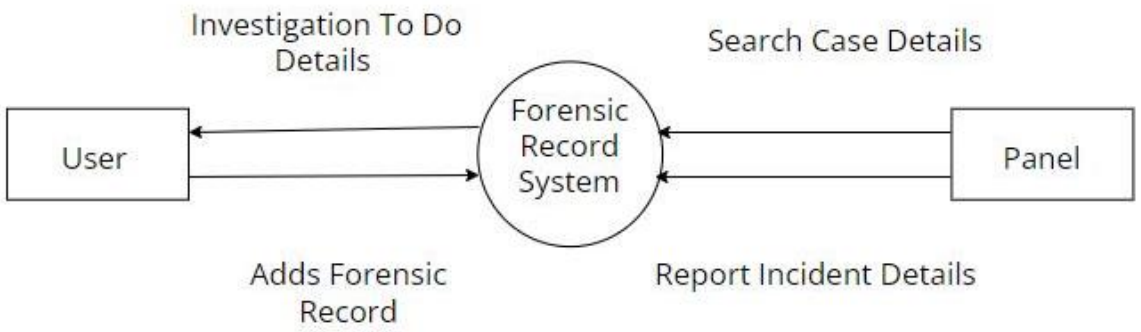


Figure 4.1.4 : Data Flow Diagram (Level 0)

Level 1:

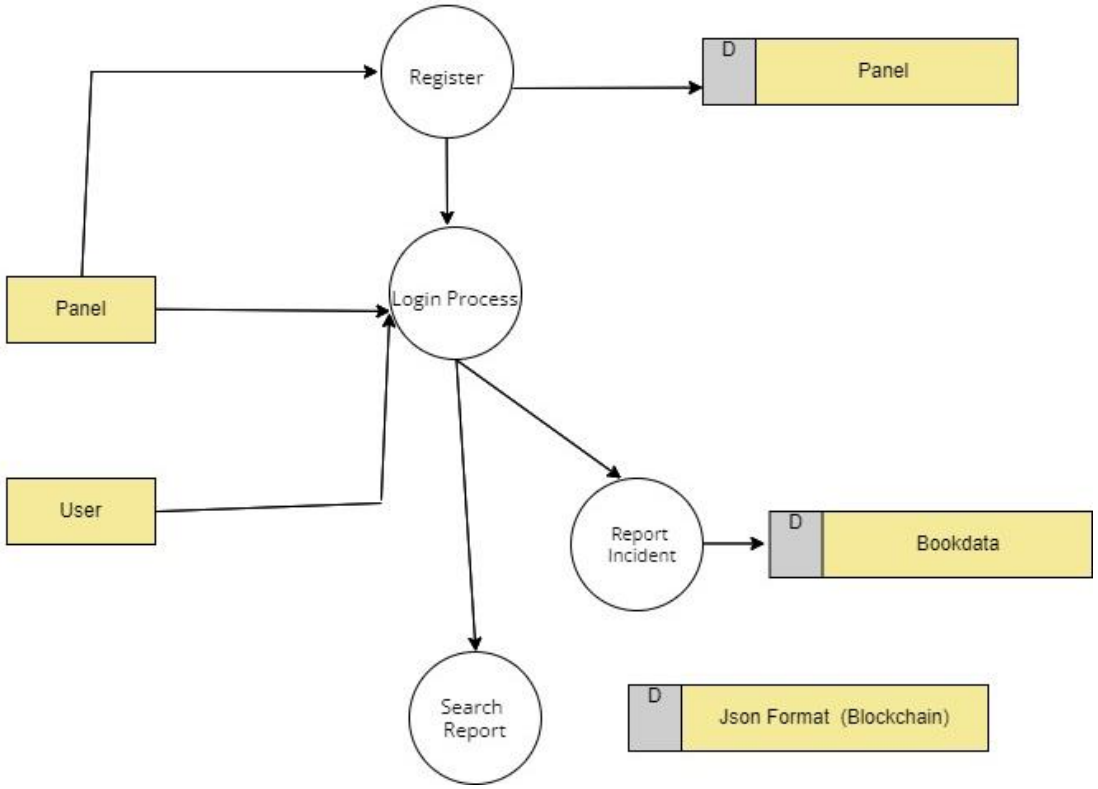


Figure 4.1.5: Data Flow Diagram (Level 1)

Level 2:

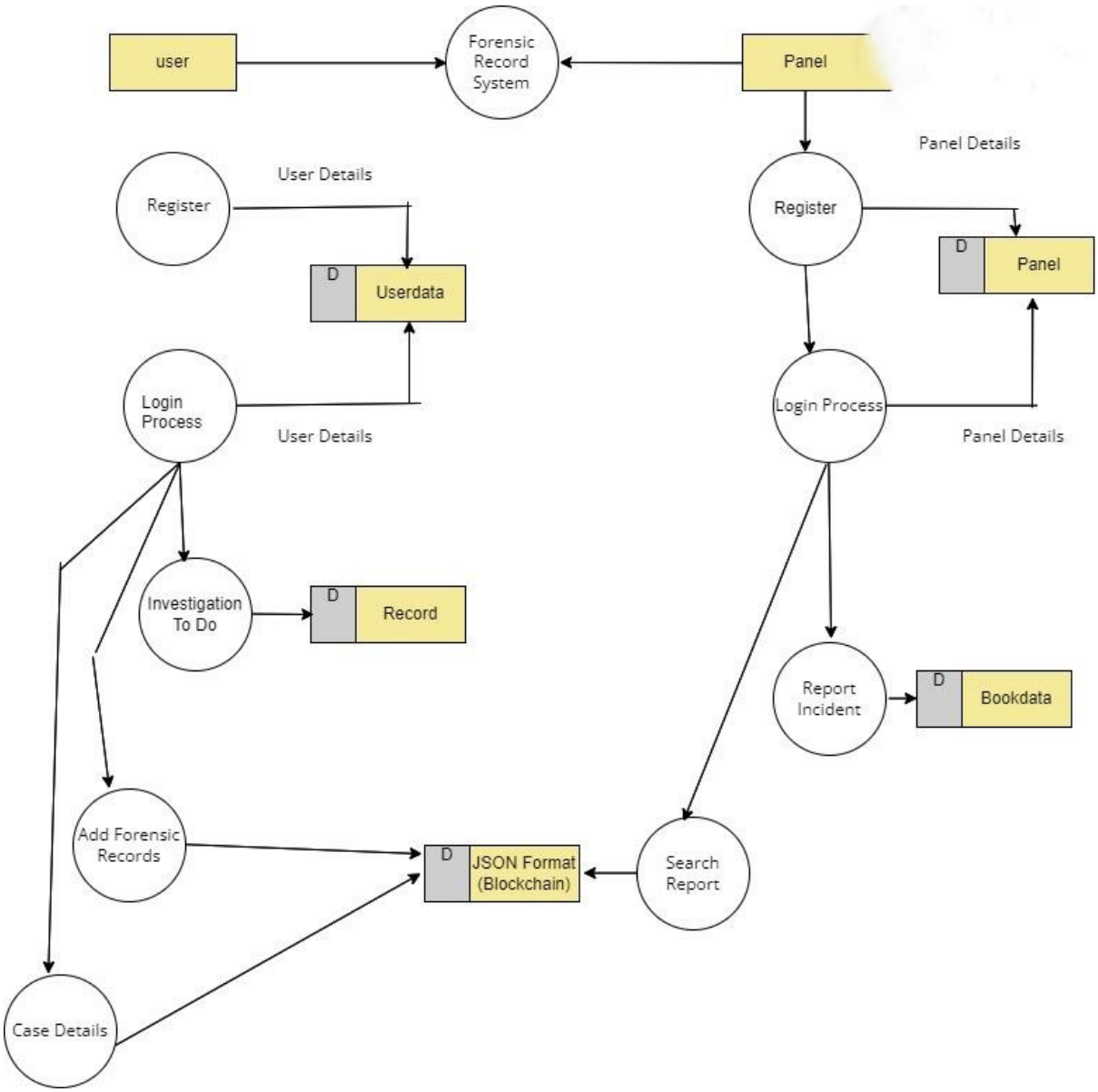


Figure 4.1.6 :Data Flow Diagram

4.2 ER Diagram

4.2.1 Definition

The Entity Relational Model is a model for identifying entities to be represented in the database and representation of how those entities are related. The ER data model specifies enterprise schema that represents the overall logical structure of a database graphically. The Entity Relationship Diagram explains the relationship among the entities present in the

database. ER models are used to model real-world objects like a person, a car, or a company and the relation between these real-world objects.

In short, the ER Diagram is the structural format of the database. Entity Relationship Diagram, also known as ERD, ER Diagram or ER model, is a type of structural diagram for use in database design. An ERD contains different symbols and connectors that visualize two important information.

The major entities within the system scope, and the inter-relationships among these entities. And that's why it's called "Entity" "Relationship" diagram (ERD)! When we talk about entities in ERD, very often we are referring to business objects such as people/roles (e.g., Student), tangible business objects (e.g., Product), intangible business objects (e.g., Log), etc. "Relationship" is about how these entities relate to each other within the system. ER Model stands for Entity Relationship Model is a high-level conceptual data model diagram. ER model helps to systematically analyse data requirements to produce a well- designed database.

The ER Model represents real-world entities and the relationships between them. Creating an ER Model in DBMS is considered as a best practice before implementing your database. ER Model helps you to analyse data requirements systematically to produce a well-designed database.

Why do we use ER diagram?

Helps you to define terms related to entity relationship modelling provide a preview of how all your tables should connect, what fields are going to be on each table Helps to describe entities, attributes, relationships ER diagrams are translatable into relational tables which allows you to build databases quickly ER diagrams can be used by database designers as a blueprint. Entity Relationship Diagram, also known as ERD, ER Diagram or ER model, is a type of structural diagram for use in database design.

An ERD contains different symbols and connectors that visualize two important information: The major entities within the system scope, and the inter-relationships among these entities.

4.2.2 ERD Features

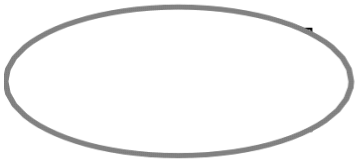
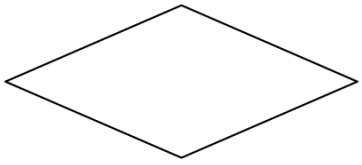
An Entity-Relationship Diagram (ERD) is a detailed visual representation used in database design and system analysis to model the entities, attributes, and relationships within

a database. ERDs provide a structured way to describe and understand how data is organized and interconnected in a database system.

Entity-Relationship Diagrams (ERDs) are a valuable tool in database design and system analysis, providing a visual representation of the structure and relationships of data within a database. Here are the key features and characteristics of ERDs:

- **Entities:** Entities represent objects, concepts, or things that are relevant to the database. Each entity is typically depicted as a rectangle in the diagram. Common examples of entities include case details, Report Incidents.
- **Attributes:** Attributes are characteristics or properties of entities that provide more detailed information about them. Attributes are shown within the entity rectangles and describe the specific data elements associated with each entity. For example, an "Case Details" entity might have attributes like "Record Id", "Name", "Age", and "Physique."
- **Relationships:** Relationships define how entities are related to each other. These are illustrated using diamond shapes connecting the related entities. Relationships are labelled to describe the nature of the association. For instance, a relationship label might be "owns," "is employed by," or "places."

4.2.3 ERD symbols:

Symbols	Symbol Description
	Entity: An Entity is represented by a rectangle which contains theentity's name
	Links & lines: Used to connect entities to their attributes and to connect entities in relationships
	Relationship: A relationship whereentity is existence Independent of other entities, and PK a child doesn't Contain PK component of Parent Entity. Represented By a rhombus

4.2.4 ER Diagram

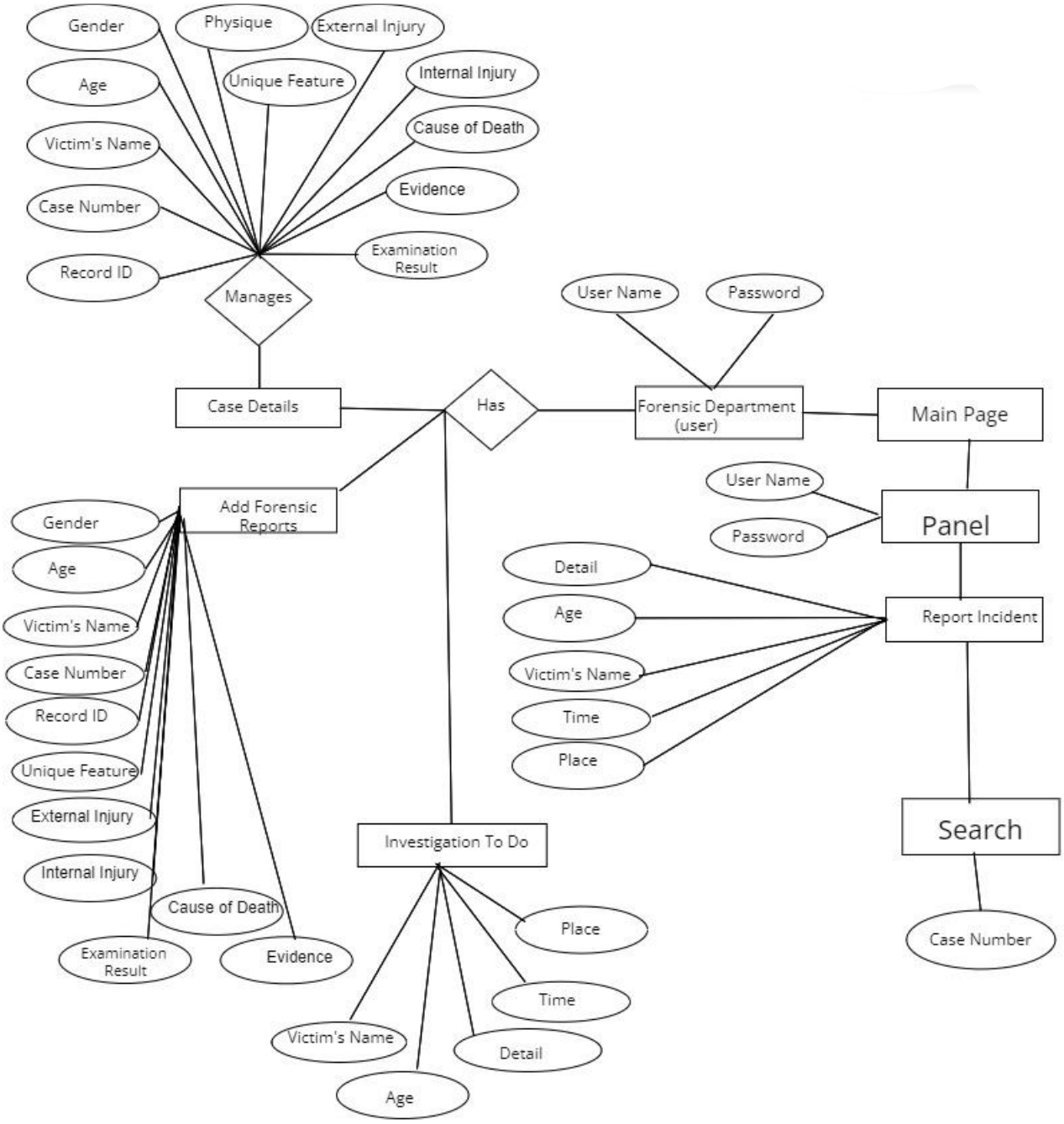


Figure 4.2.5 :Entity Relationship Diagram

5. SYSTEM REQUIREMENT SPECIFICATION

5.1 HARDWARE REQUIREMENT.

Hardware is a set of physical components, which performs the functions of applying appropriate, predefined instructions. In other words, one can say that electronic and mechanical parts of computer constitute hardware.

- **Processors and memory:** The best system to start with is one based on Pentium II with a minimum 32 MB of RAM. Adequate performance requires at least 64 MB of RAM. But for a database server at least 64 to 128 MB of RAM is required.
- **Hardware Processor:** Pentium 2.4 GHz or above
- **Memory:** 256 MB RAM or above

5.2 SOFTWARE REQUIREMENT.

The software is a set of procedures of coded information or a program which when fed into the computer hardware enables the computer to perform the various tasks. Software is like a current inside the wire, which cannot be seen but its effect can be felt.

- **Operating system**: Windows XP(professional)
- **Tool**: VB Visual Studio
- **Front-End**: HTML, CSS, JavaScript
- **Back-End**: Python, Flask , SQL

5.3 SOFTWARE SPECIFICATION

5.3.1 Front End Specification

HTML

The **Hyper Text Markup Language** or **HTML** is the standard markup language for documents designed to be displayed in a web browser. It defines the meaning and structure of web content. It is often assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a webpage semantically and originally included cues for its appearance.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes, and other items. HTML elements are delineated by *tags*, written using angle brackets. Tags such as <**img**> and <**input**> directly introduce content into the page. Browsers do not display the HTML tags but use them to interpret the content of the page.

CSS

Cascading Style Sheets (**CSS**) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript

CSS is designed to enable the separation of content and presentation, including layout, colors, and fonts. This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

JAVASCRIPT

JavaScript, often abbreviated as **JS**, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. As of 2023, 98.7% of websites use JavaScript on the client side for webpage behaviour, often incorporating third-party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on users' devices.

JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard it as dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM).

The ECMAScript standard does not include any input/output (I/O), such as networking, storage, or graphics facilities. In practice, the web browser or other runtime system provides JavaScript APIs for I/O.

JavaScript engines were originally used only in web browsers, but are now core components of some servers and a variety of applications. The most popular runtime system for this usage is Node.js. Although Java and JavaScript are similar in name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in design.

5.3.2 Back End Specification

SQL

Structured Query Language sometimes "sequel" for historical reasons is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream

management system (RDSMS). It is particularly useful in handling structured data, i.e., data incorporating relations among entities and variables.

Introduced in the 1970s, SQL offered two main advantages over older read–write APIs such as ISAM or VSAM. Firstly, it introduced the concept of accessing many records with one single command. Secondly, it eliminates the need to specify how to reach a record, i.e., with or without an index.

PYTHON

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.[32][33]

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000. Python 3.0, released in 2008, was a major revision not completely backward-compatible with earlier versions. Python 2.7.18, released in 2020, was the last release of Python 2

Python consistently ranks as one of the most popular programming languages, and has gained widespread use in the machine learning community.

FLASK

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

Applications that use the Flask framework include Pinterest and LinkedIn.

6. SYSTEM DESIGN

6.1 Table Design with Input

Database Name: Frs

6.1.1 Userdata Table

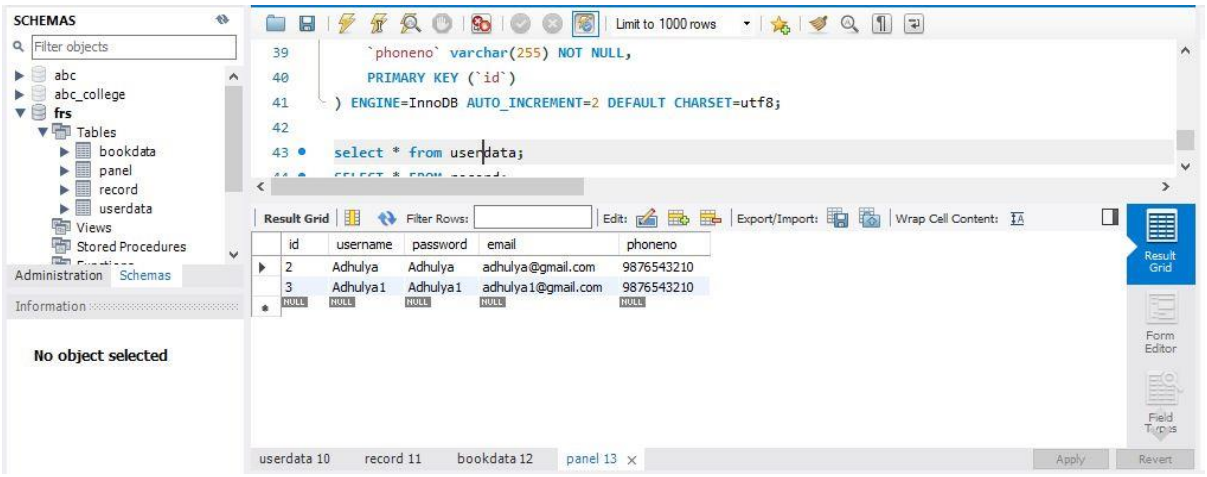


Figure 6.1.1 User Data

6.1.2 Record Table

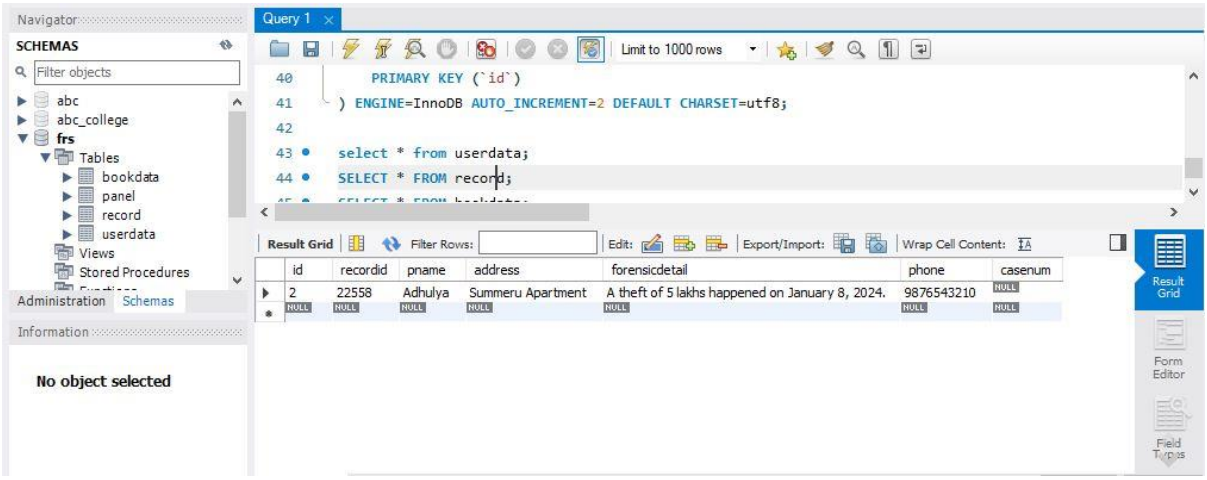


Figure 6.1.2. Record

6.1.3 Bookdata Table

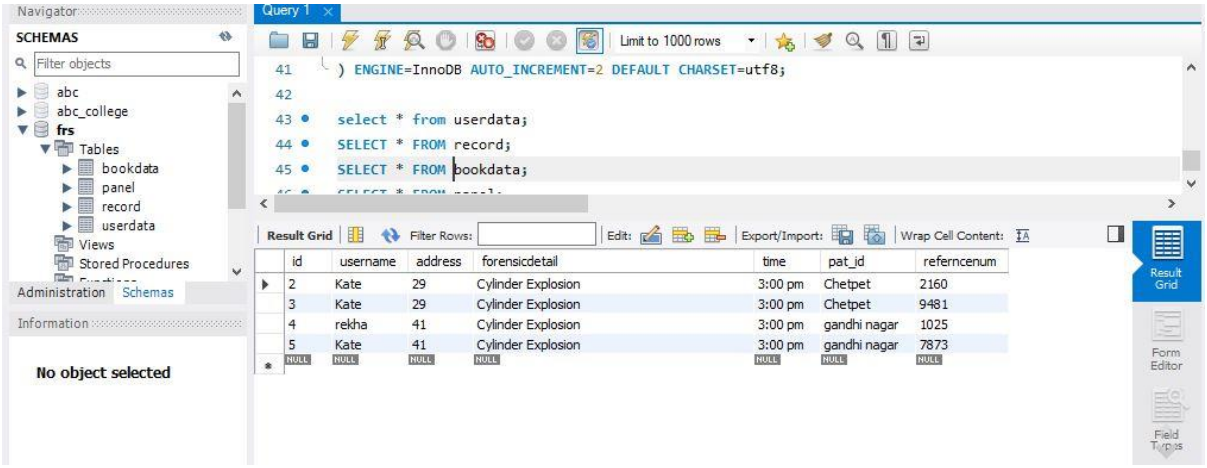


Figure 6.1.3. Book data

6.1.4 Panel Table

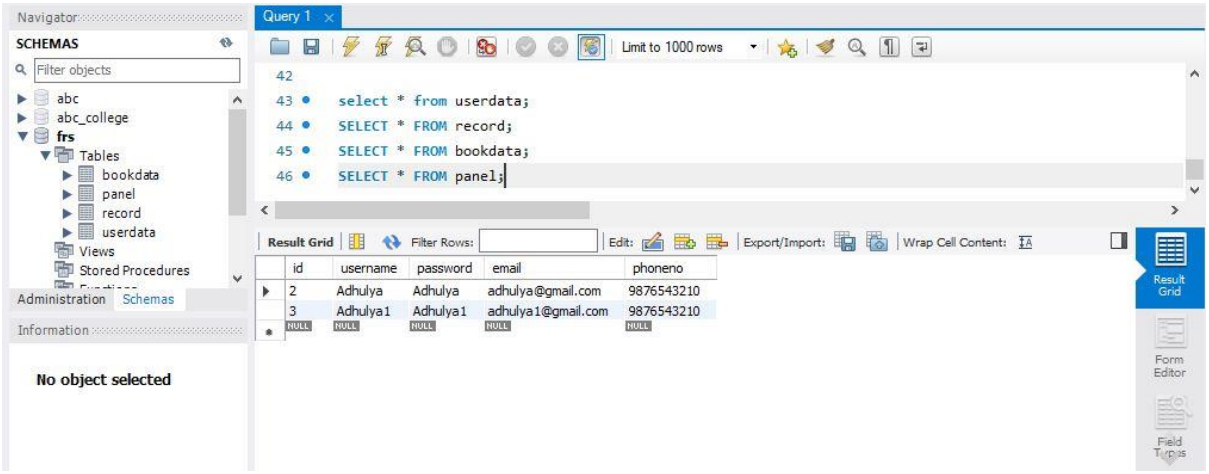


Figure 6.1.4 Panel

6.2 FORM DESIGN

6.2.1 Mainpage Form

Figure 6.2.1 Mainpage

6.2.2 User Registration Form

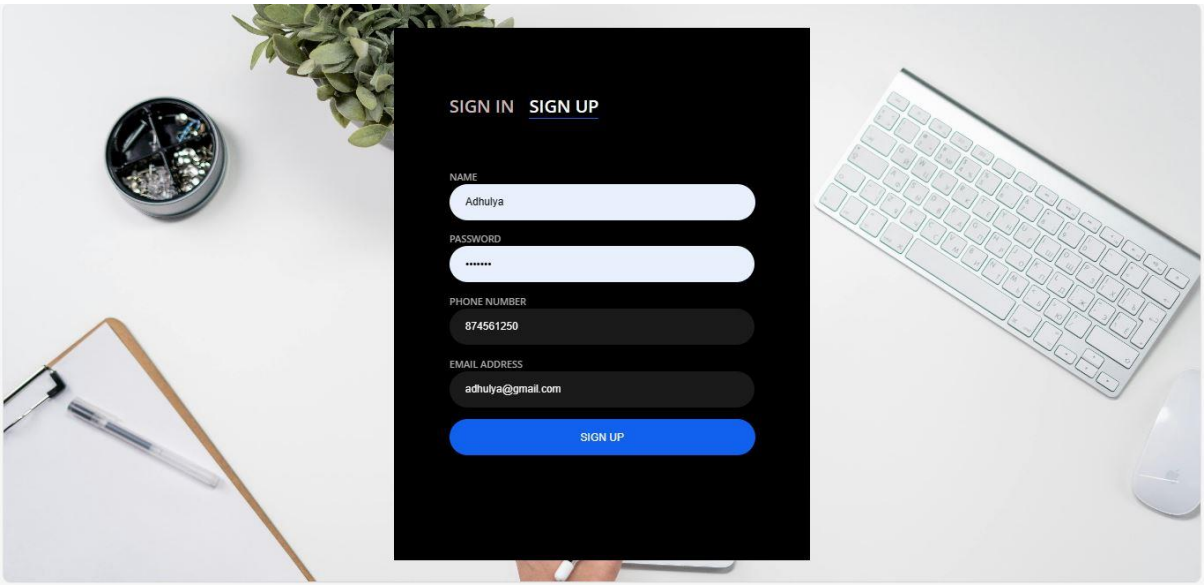


Figure 6.2.2. Registration module

6.2.3 User Login Form

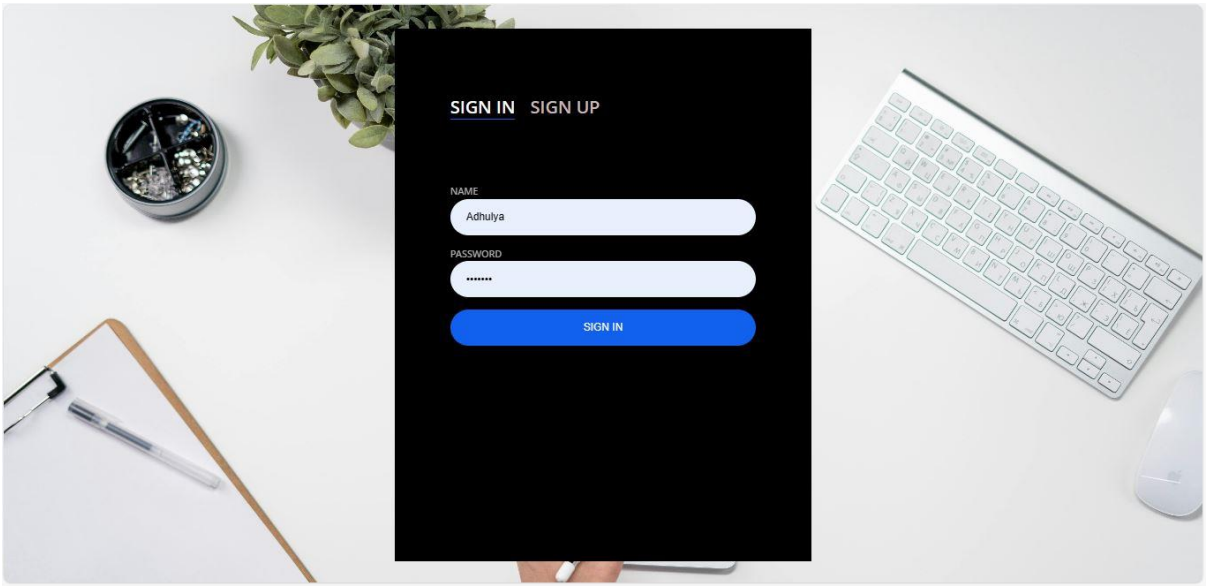


Fig 6.2.3. login form

6.2.4 User Main page

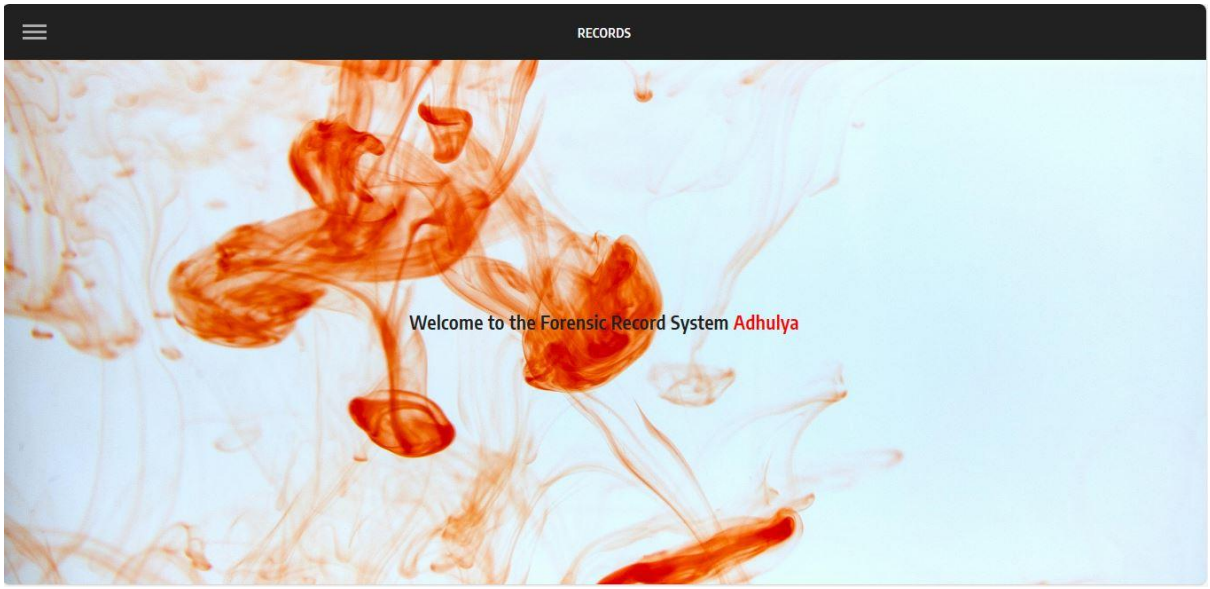


Fig 6.2.4. User Main Page

6.2.5 Add Forensic Record Form

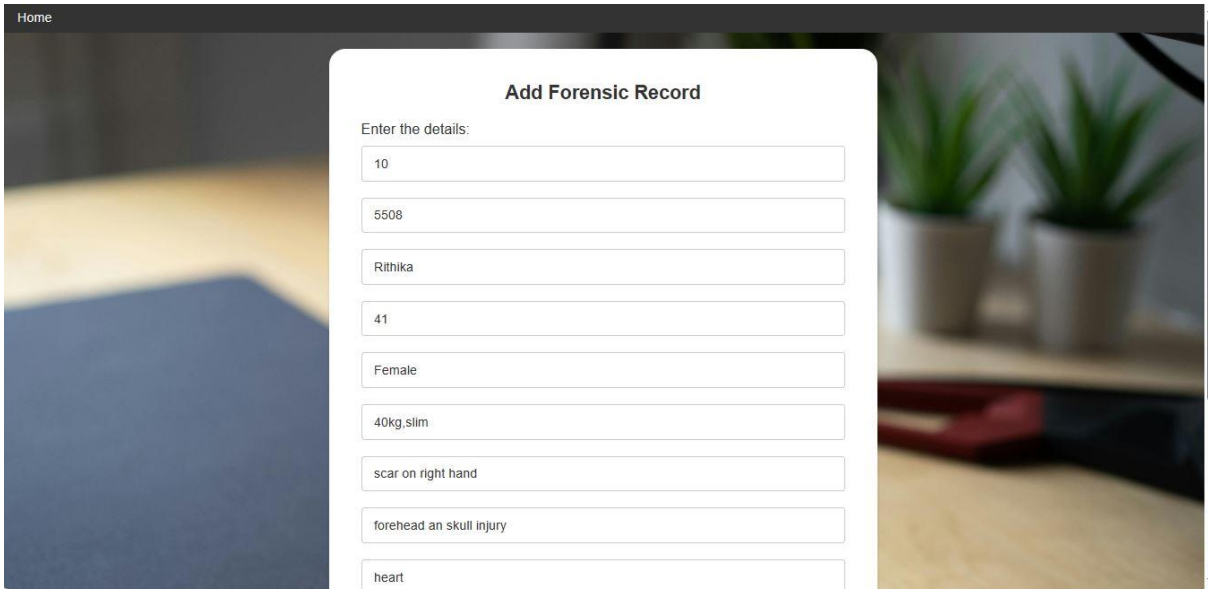


Fig 6.2.5 Destination Form

6.2.6 Investigation To Do Form

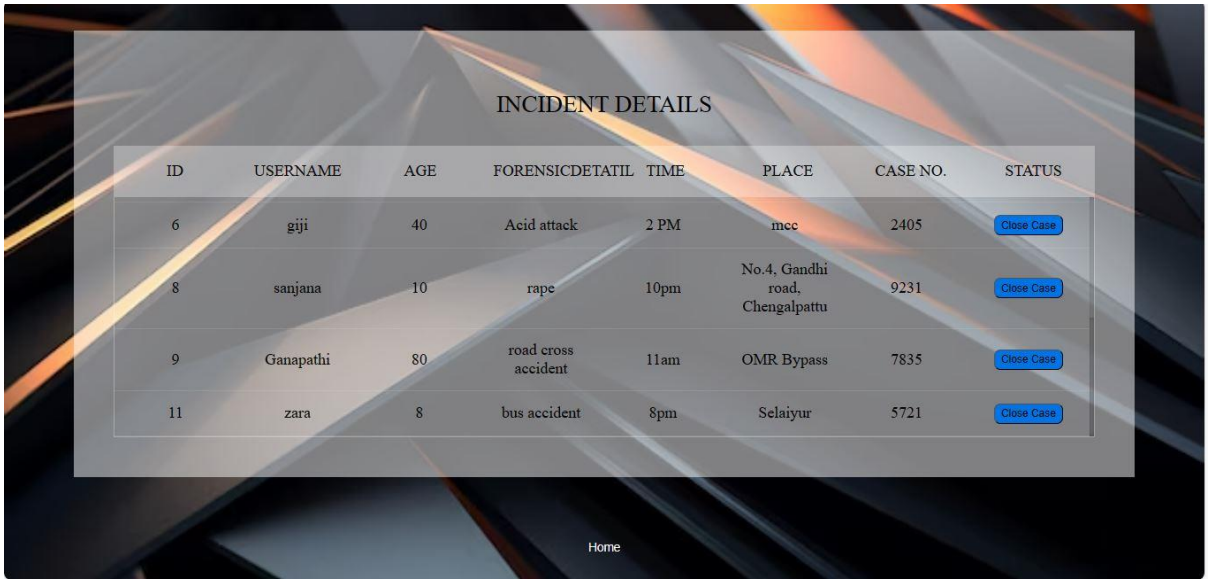


Fig 6.2.6 Investigation To Do Form

6.2.7 Case Detail Form

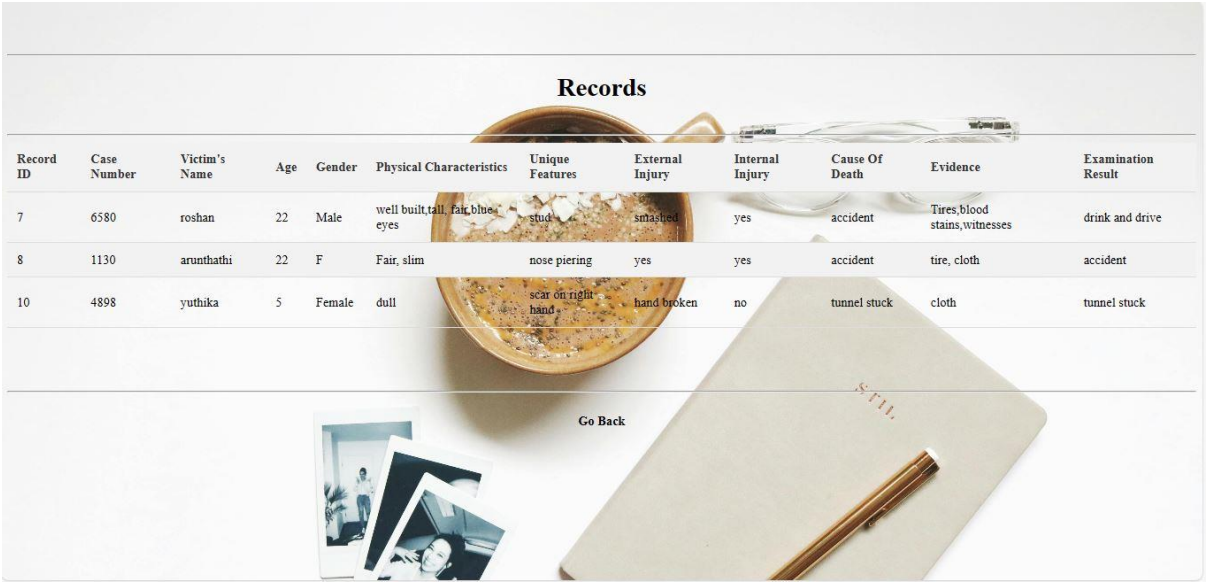


Fig 6.2.7 Case Detail Form

6.2.8 Panel Login Form

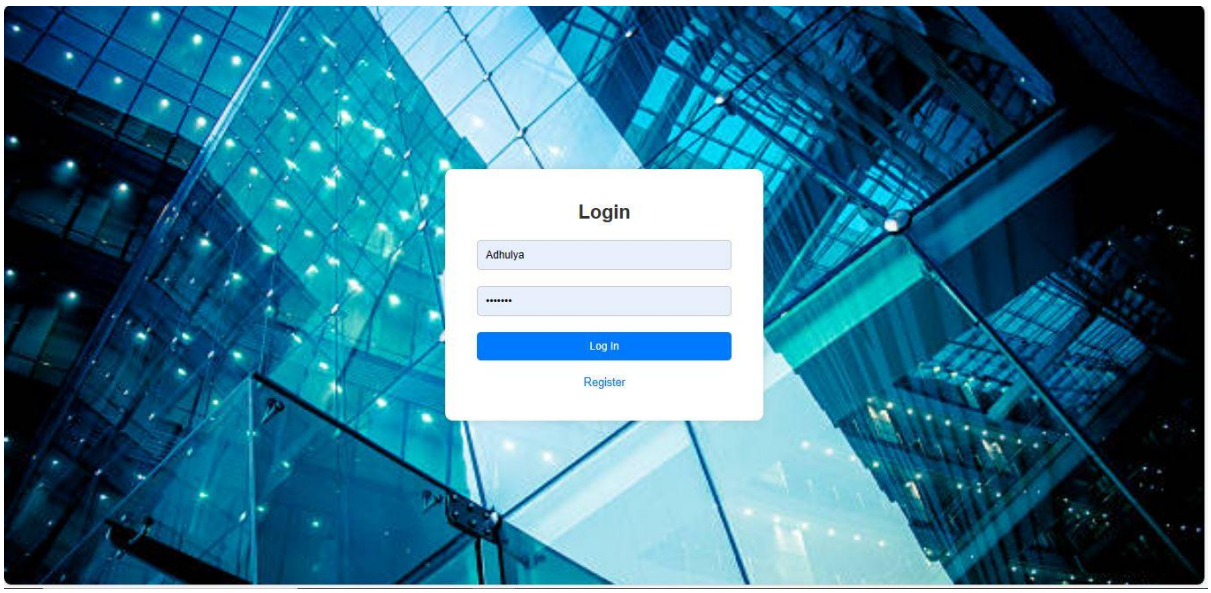


Fig 6.2.8 Panel Login

6.2.9 Panel Registration Form

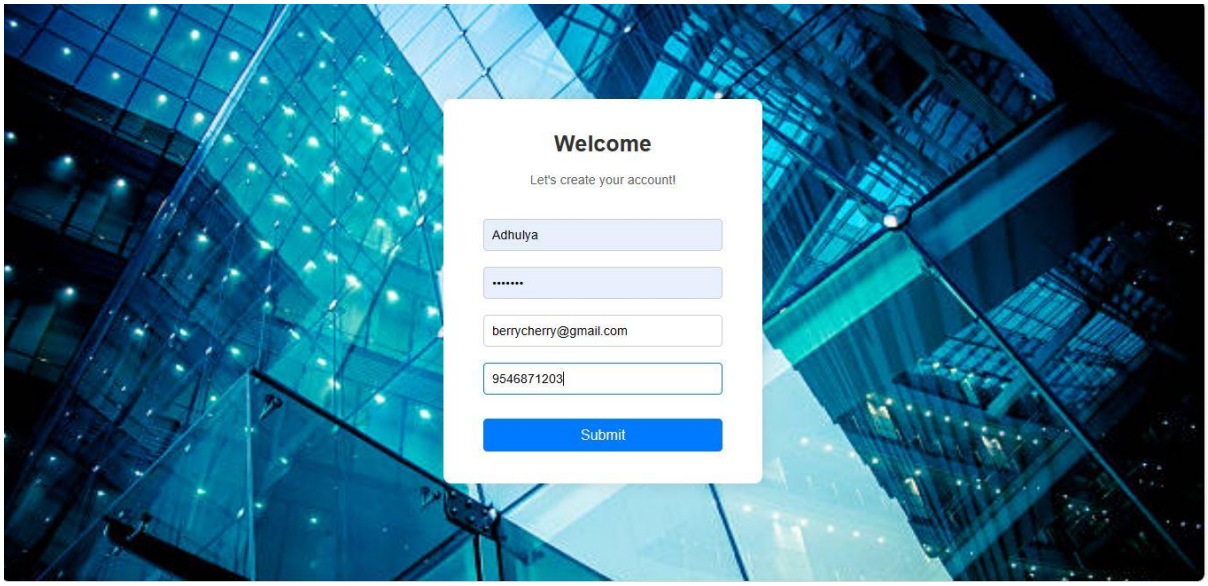


Fig 6.2.9 Registration

6.2.10. Report Incident Form

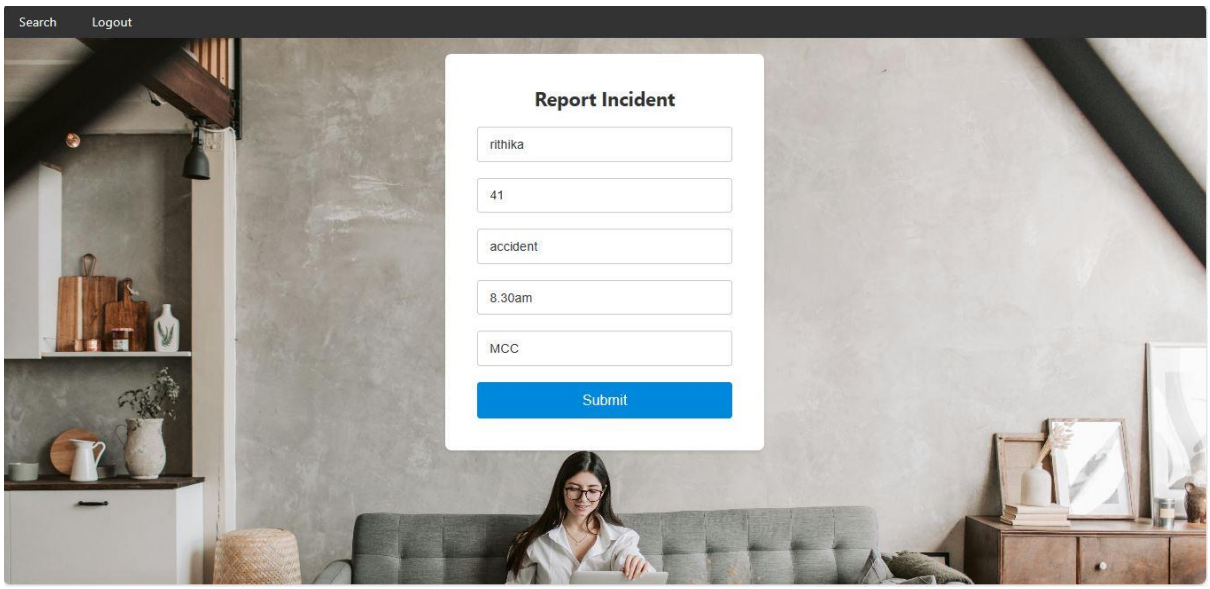


Fig 6.2.10 Report Incident

6.2.11 Search Case Form

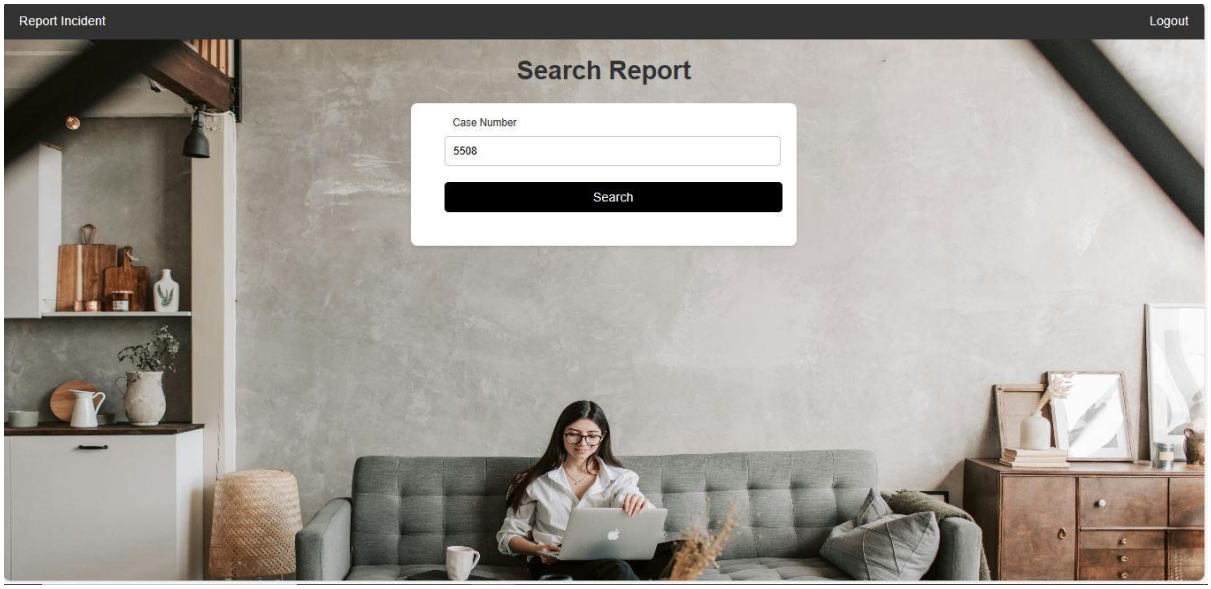


Fig 6.2.11 Search Case

7.SYSTEM IMPLEMENTATION

7.1 Coding of All Modules

MAIN PAGE

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Welcome Page</title>
    <style>
      body {
        margin: 0;
        padding: 0;
        font-family: Arial, sans-serif;
        background-image: url("/static/img6.jpg");
        background-size: cover;
        background-repeat: no-repeat;
      }
      .container {
        display: flex;
      }

      .side-panel {
        background-color: rgba(255, 255, 255, 0.4);
        width: 2500px;
        height: 80px;
        padding-left: 30px;
      }

      .side-panel ul {
        list-style-type: none;
        padding: 0px;
        display: flex;
```

```
    flex-direction: row;
    align-items: center;
  }
  .side-panel ul li {
    margin-right: 50px;
  }

  .side-panel ul li:last-child {
    margin-right: 0;
  }

  .side-panel ul li a {
    color: #000000;
    text-decoration: none;
    font-weight: bold;
  }

  .side-panel ul li a:hover {
    color: #ffffff;
  }

  .main-content .main2 {
    font-size: 120px;
    color: #fff;
  }
  .main-content {
    color: #ffffff;
    font-size: 40px;
    padding-left: 60px;
  }
  .side-panel {
    text-align: center;
  }
</style>
```

```
  </head>
  <body>
    <div class="container">
      <div class="side-panel">
        <ul>
          <li>
            <img src="/static/logo.JPG" alt="" height="50px" width="100px" />
          </li>
          <li><a href="index.html" class="asideAnchor">Home</a></li>
          <li><a href="users.html" class="asideAnchor">Users</a></li>
          <li><a href="panel.html" class="asideAnchor">Panel</a></li>
        </ul>
      </div>
    </div>
    <div class="main-content">
      <h1>A New Approach In Forensic <br />Blockchain</h1>
    </div>
  </body>
</html>
```

USER MAIN PAGE

Users_home.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Users Home Page</title>
    <style>
      @import
url("https://fonts.googleapis.com/css?family=Encode+Sans+Condensed:400,600");

      * {
        outline: none;
```

```
}

strong {
  font-weight: 600;
}

.page {
  width: 100%;
  height: 100vh;
  background: #fdfdfd;
  font-family: "Encode Sans Condensed", sans-serif;
  font-weight: 600;
  letter-spacing: 0.03em;
  color: #212121;
}

header {
  display: flex;
  position: fixed;
  width: 100%;
  height: 70px;
  background: #212121;
  color: #fff;
  justify-content: center;
  align-items: center;
  -webkit-tap-highlight-color: rgba(0, 0, 0, 0);
}

main {
  padding: 70px 20px 0;
  display: flex;
  flex-direction: column;
  height: 100%;
}
```

```
main > div {
  margin: auto;
  max-width: 600px;
}

main h2 span {
  color: #bf7497;
}

main p {
  line-height: 1.5;
  font-weight: 200;
  margin: 20px 0;
}

main small {
  font-weight: 300;
  color: #888;
}

#nav-container {
  position: fixed;
  height: 100vh;
  width: 100%;
  pointer-events: none;
}
#nav-container .bg {
  position: absolute;
  top: 70px;
  left: 0;
  width: 100%;
  height: calc(100% - 70px);
  visibility: hidden;
```

```
  opacity: 0;
  transition: 0.3s;
  background: #000;
}
#nav-container:focus-within .bg {
  visibility: visible;
  opacity: 0.6;
}
#nav-container * {
  visibility: visible;
}

.button {
  position: relative;
  display: flex;
  flex-direction: column;
  justify-content: center;
  z-index: 1;
  -webkit-appearance: none;
  border: 0;
  background: transparent;
  border-radius: 0;
  height: 70px;
  width: 30px;
  cursor: pointer;
  pointer-events: auto;
  margin-left: 25px;
  touch-action: manipulation;
  -webkit-tap-highlight-color: rgba(0, 0, 0, 0);
}
.icon-bar {
  display: block;
  width: 100%;
  height: 3px;
```

```
  background: #aaa;
  transition: 0.3s;
}
.icon-bar + .icon-bar {
  margin-top: 5px;
}

#nav-container:focus-within .button {
  pointer-events: none;
}
#nav-container:focus-within .icon-bar:nth-of-type(1) {
  transform: translate3d(0, 8px, 0) rotate(45deg);
}
#nav-container:focus-within .icon-bar:nth-of-type(2) {
  opacity: 0;
}
#nav-container:focus-within .icon-bar:nth-of-type(3) {
  transform: translate3d(0, -8px, 0) rotate(-45deg);
}

#nav-content {
  margin-top: 70px;
  padding: 20px;
  width: 90%;
  max-width: 300px;
  position: absolute;
  top: 0;
  left: 0;
  height: calc(100% - 70px);
  background: #ececec;
  pointer-events: auto;
  -webkit-tap-highlight-color: rgba(0, 0, 0, 0);
  transform: translateX(-100%);
  transition: transform 0.3s;
```

```
  will-change: transform;
  contain: paint;
}

#nav-content ul {
  height: 100%;
  display: flex;
  flex-direction: column;
}

#nav-content li a {
  padding: 10px 5px;
  display: block;
  text-transform: uppercase;
  transition: color 0.1s;
}

#nav-content li a:hover {
  color: #bf7497;
}

#nav-content li:not(.small) + .small {
  margin-top: auto;
}

.small {
  display: flex;
  align-self: center;
}

.small a {
  font-size: 12px;
  font-weight: 400;
  color: #888;
```

```
}
.small a + a {
  margin-left: 15px;
}

#nav-container:focus-within #nav-content {
  transform: none;
}

* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

html,
body {
  height: 100%;
}

main {
  background-image: url("/static/red.jpg");
  background-size: cover;
  background-repeat: no-repeat;
}

a,
a:visited,
a:focus,
a:active,
a:link {
  text-decoration: none;
  outline: 0;
}
```

```
    a {
      color: currentColor;
      transition: 0.2s ease-in-out;
    }

    h1,
    h2,
    h3,
    h4 {
      margin: 0;
    }

    ul {
      padding: 0;
      list-style: none;
    }

    #nav-content {
      background-color: #f0defd;
    }

    img {
      vertical-align: middle;
      height: auto;
      width: 100%;
    }

    main h2 span {
      color: rgb(255, 0, 0);
    }
  </style>
</head>
<body>
```

```
<div class="page">
  <header tabindex="0">RECORDS</header>
  <div id="nav-container">
    <div class="bg"></div>
    <div class="button" tabindex="0">
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
    </div>
    <div id="nav-content" tabindex="0">
      <ul>
        <li><a href="addFRS.html">Add Forensic Record</a></li>
        <li><a href="panelhome.html">Investigation Todo</a></li>
        <li>
          <a
            href="#"
            onclick="document.getElementById('caseDetail').submit(); return false;"
          >
            Case Detail
          </a>
          <form
            id="caseDetail"
            action="{{ url_for('generate_html_from_json_folder') }}"
            method="GET"
            style="display: none"
          ></form>
        </li>
        <li><a href="logout">Logout</a></li>
      </ul>
    </div>
  </div>

  <main>
    <div class="content">
```

```
        <h2>
          Welcome to the Forensic Record System <span>{{username}}</span>
        </h2>
       </div>
     </main>
   </div>
  </body>
</html>
```

ADD FORENSIC REPORT

addFRS.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Add Records</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-image: url("/static/mouse.jpg");
      background-size: cover;
      ;
      margin: 0;
      padding: 0;
    }

    .navbar {
      background-color: #333;
      padding: 10px 0;
    }

    .navbar a {
```

```
  color: #fff;
  text-decoration: none;
  padding: 10px 20px;
}

.navbar a:hover {
  background-color: #555;
}

.container {
  background-color: #fff;
  border-radius: 20px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  padding: 40px;
  width: 80%;
  max-width: 600px;
  margin: 20px auto;
}

.title {
  color: #333;
  font-size: 24px;
  font-weight: bold;
  margin-bottom: 20px;
  text-align: center;
}

.subtitle {
  color: #333;
  font-size: 18px;
  margin-bottom: 10px;
}

.input-container {
```

```
      margin-bottom: 20px;
    }

    .input {
      border: 1px solid #ccc;
      border-radius: 4px;
      color: #333;
      font-size: 16px;
      padding: 12px 15px;
      width: 100%;
      box-sizing: border-box;
    }

    .submit {
      background-color: #08d;
      border: none;
      border-radius: 4px;
      color: #fff;
      cursor: pointer;
      font-size: 18px;
      padding: 12px 0;
      width: 100%;
    }

    .submit:hover {
      background-color: #0077cc;
    }
  </style>
</head>
<body>
  <div class="navbar">
    <a href="users_home.html">Home</a>
  </div>
```

```
 <div class="container">
   <div class="title">Add Forensic Record</div>
   <form action="/adddata" method="post">
     <div class="subtitle">Enter the details:</div>

     <div class="input-container">
       <input id="rid" class="input" type="text" placeholder="Record ID" name="rid"
required>
     </div>

     <div class="input-container">
       <input id="cnum" class="input" type="text" placeholder="Case Number"
name="cnum">
     </div>

     <div class="input-container">
       <input id="name" class="input" type="text" placeholder="Victim's Name"
name="name" required>
     </div>

     <div class="input-container">
       <input id="age" class="input" type="text" placeholder="Age" name="age" required>
     </div>

     <div class="input-container">
       <input id="gender" class="input" type="text" placeholder="Gender" name="gender"
required>
     </div>

     <div class="input-container">
       <input id="phy" class="input" type="text" placeholder="Physical Characteristics"
name="phy" required>
     </div>
```

```
    <div class="input-container">
      <input id="uni" class="input" type="text" placeholder="Unique Features" name="uni"
required>
    </div>

    <div class="input-container">
      <input id="ex" class="input" type="text" placeholder="External Injury" name="ex"
required>
    </div>

    <div class="input-container">
      <input id="int" class="input" type="text" placeholder="Internal Injury" name="int"
required>
    </div>

    <div class="input-container">
      <input id="death" class="input" type="text" placeholder="Cause of Death"
name="death" required>
    </div>

    <div class="input-container">
      <input id="evidence" class="input" type="text" placeholder="Evidence"
name="evidence" required>
    </div>

    <div class="input-container">
      <input id="result" class="input" type="text" placeholder="Examination Result"
name="result" required>
    </div>
    <button type="submit" class="submit">Submit</button>
   </form>
  </div>
</body>
</html>
```

CASE DETAILS

Output.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Records</title>
    <style>
      body{
        background-image: url("../static/records.jpg");
        color: rgb(0, 0, 0);
        background-size:cover;
      }
      a {
          text-decoration: none;
          color: #000000;
          font-weight: bold;
        }

        a:hover {
          text-decoration: underline;
        }

        hr {
          margin-top: 40px;
        }
        table {
          width: 100%;
          border-collapse: collapse;
          margin-bottom: 20px; /* Added margin for spacing */
```

```
    }

    th, td {
        padding: 12px; /* Increased padding for better spacing */
        text-align: left;
        border-bottom: 1px solid #ddd;
    }

    th {
        background-color: #f2f2f2; /* Added background color for table header */
        color: #333; /* Adjusted color for better contrast */
        font-weight: bold;
    }

    tbody tr:hover {
        background-color: #f5f5f5; /* Added hover effect for better interaction */
    }

    tbody tr:nth-child(even) {
        background-color: #f2f2f2; /* Added alternate row background color */
    }

  </style>
</head>
<body>
  {% block content %}
  <br />
  <hr />
  {% endblock %} {% block check %}

  <center>
    <h1>Records</h1>
  </center>
```

```
<hr />
<table>
  <thead>
    <tr>
      <th>Record ID</th>
      <th>Case Number</th>
      <th>Victim's Name</th>
      <th>Age</th>
      <th>Gender</th>
      <th>Physical Characteristics</th>
      <th>Unique Features</th>
      <th>External Injury</th>
      <th>Internal Injury</th>
      <th>Cause Of Death</th>
      <th>Evidence</th>
      <th>Examination Result</th>
    </tr>
  </thead>
  <tbody>
    {% for result in results %}
    <tr>
      <td>{{result['record_id'] }}</td>
      <td>{{result['cnum'] }}</td>
      <td>{{result['name'] }}</td>
      <td>{{result['age'] }}</td>
      <td>{{result['gender'] }}</td>
      <td>{{result['phy'] }}</td>
      <td>{{result['uni'] }}</td>
      <td>{{result['ex'] }}</td>
      <td>{{result['int'] }}</td>
      <td>{{result['death'] }}</td>
      <td>{{result['evidence'] }}</td>
      <td>{{result['vresult'] }}</td>
    </tr>
```

```
      {% endfor %}
    </tbody>
  </table>
  <br />
  {% endblock %}
  <hr />
  <br />
  <div>
    <center><a href="users_home.html">Go Back</a></center>
  </div>
 </body>
</html>
```

INVESTIGATION TO DO

Panelhome.html

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <title>Panel Main Page</title>
  <style>
   h1 {
  font-size: 30px;
  color: black;
  text-transform: uppercase;
  font-weight: 300;
  text-align: center;
  margin-bottom: 15px;
 }
 table {
  width: 100%;
  table-layout: fixed;
```

```
}
.tbl-header {
  background-color: rgba(255, 255, 255, 0.3);
}
.tbl-content {
  height: 300px;
  overflow-x: auto;
  margin-top: 0px;
  border: 1px solid rgba(255, 255, 255, 0.3);
}
th {
  padding: 20px 15px;
  text-align: center; /* Center align the text */
  font-weight: 500;
  font-size: 20px;
  color: black;
  text-transform: uppercase;
  white-space: nowrap; /* Prevent text wrapping */
}
td {
  padding: 15px;
  text-align: center;
  vertical-align: middle;
  font-weight: 300;
  font-size: 20px;
  color: black;
  border-bottom: solid 1px rgba(255, 255, 255, 0.1);
}

    /* demo styles */

    @import url(https://fonts.googleapis.com/css?family=Roboto:400,500,300,700);
    body {
      background-image: url("/static/img2.avif");
```

```
  background-size: cover;
  background-position: center;
  background-repeat: no-repeat;
}

section {
  padding: 50px;
  background-color: rgba(255, 255, 255, 0.5);
}

.container {
  padding: 30px;
  padding-left: 80px;
  padding-right: 80px;
}

h1 {
  padding-bottom: 20px;
}

/* follow me template */
.made-with-love {
  margin-top: 40px;
  padding: 10px;
  clear: left;
  text-align: center;
  font-size: 15px;
  font-family: arial;
  color: white;
}
.made-with-love i {
  font-style: normal;
  color: #f50057;
  font-size: 14px;
```

```
  position: relative;
  top: 2px;
}
.made-with-love a {
  color: white;
  text-decoration: none;
}
.made-with-love a:hover {
  text-decoration: underline;
}

.done-btn {
  display: inline-block;
  cursor: pointer;
  text-align: center;
  border-radius: 8px;
  color: black;
  background-color: #0074e4;
  padding-top: 4px;
  padding-bottom: 4px;
  padding-left: 8px;
  padding-right: 8px;
}
/* for custom scrollbar for webkit browser*/

::-webkit-scrollbar {
  width: 6px;
}
::-webkit-scrollbar-track {
  -webkit-box-shadow: inset 0 0 6px rgba(0, 0, 0, 0.3);
}
::-webkit-scrollbar-thumb {
  -webkit-box-shadow: inset 0 0 6px rgba(0, 0, 0, 0.3);
}
```

```
</style>
<script>
  // '.tbl-content' consumed little space for vertical scrollbar, scrollbar width depend on browser/os/platfrom. Here calculate the scollbar width .
  $(window)
    .on("load resize ", function () {
      var scrollWidth =
        $(".tbl-content").width() - $(".tbl-content table").width();
      $(".tbl-header").css({ "padding-right": scrollWidth });
    })
    .resize();
</script>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script>
  $(document).ready(function () {
    // Attach click event to the "Done" button
    $("button.done-btn").on("click", function () {
      var row = $(this).closest("tr");
      var id = row.find("td:first").text(); // Assuming the ID is in the first column
      removeRow(id);
      row.remove();
    });

    function removeRow(id) {
      // Send a request to the server to remove the row from the database
      $.ajax({
        url: "/remove_row/" + id,
        method: "POST",
        success: function (data) {
          console.log(data);
        },
        error: function (error) {
          console.error("Error:", error);
        },
```

```
      });
    }
  });
 </script>
</head>
<body>
 <div class="container">
  <section>
   <h1>Incident Details</h1>
   <div class="tbl-header">
    <table cellpadding="0" cellspacing="0" border="0">
     <thead>
      <tr>
       <th>ID</th>
       <th>USERNAME</th>
       <th>AGE</th>
       <th>FORENSICDETATIL</th>
       <th>TIME</th>
       <th>PLACE</th>
       <th>CASE NO.</th>
       <th>STATUS</th>
      </tr>
     </thead>
    </table>
   </div>

   <div class="tbl-content">
    <table cellpadding="0" cellspacing="0" border="0">
     <tbody>
      {% for item in data %}
      <tr>
       <td>{{item.id}}</td>
       <td>{{item.username}}</td>
       <td>{{item.address}}</td>
```

```
            <td>{{item.forensicdetail}}</td>
            <td>{{item.time}}</td>
            <td>{{item.pat_id}}</td>
            <td>{{item.referncenum}}</td>
            <td><button class="done-btn">Close Case</button></td>
          </tr>
          {% endfor %}
        </tbody>
      </table>
    </div>
  </section>
 </div>
 <!-- follow me template -->
 <div class="made-with-love">
  <a href="index.html">Home</a>
 </div>
</body>
</html>
```

REPORT INCIDENT

Book.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Book Appointment</title>
 <style>
  body {
   font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
   margin: 0;
   padding: 0;
   background-image: url("/static/search.jpg");
```

```
  background-size: cover;
}

.navbar {
  background-color: #333;
  padding: 10px 0;
}

.navbar a {
  color: #fff;
  text-decoration: none;
  padding: 10px 20px;
}

.navbar a:hover {
  background-color: #555;
}

.container {
  background-color: #fff;
  border-radius: 8px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  padding: 40px;
  width: 320px;
  text-align: center;
  margin: 20px auto;
}

.title {
  color: #333;
  font-size: 24px;
  font-weight: bold;
  margin-bottom: 20px;
}
```

```
.input-container {
  margin-bottom: 20px;
}

.input {
  border: 1px solid #ccc;
  border-radius: 4px;
  color: #333;
  font-size: 16px;
  padding: 12px 15px;
  width: 100%;
  box-sizing: border-box;
  transition: border-color 0.3s ease;
}

.input:focus {
  outline: none;
  border-color: #08d;
}

.submit {
  background-color: #08d;
  border: none;
  border-radius: 4px;
  color: #fff;
  cursor: pointer;
  font-size: 18px;
  padding: 12px 0;
  width: 100%;
  transition: background-color 0.3s ease;
}

.submit:hover {
```

```
    background-color: #0077cc;
   }
  </style>
</head>
<body>
  <div class="navbar">
   <a href="search.html">Search</a>
   <a href="index.html">Logout</a>
  </div>

  <div class="container">
   <div class="title">Report Incident</div>
   <form action="/panelbook" method="post">
    <div class="input-container">
     <input id="name" class="input" type="text" name="name" placeholder="Name"
required>
    </div>

    <div class="input-container">
     <input id="age" class="input" type="text" name="age" placeholder="Age" required>
    </div>

    <div class="input-container">
     <input id="forensicdetail" class="input" type="text" name="forensicdetail"
placeholder="Forensic Detail" required>
    </div>

    <div class="input-container">
     <input id="time" class="input" type="text" name="time" placeholder="Time" required>
    </div>

    <div class="input-container">
     <input id="patid" class="input" type="text" name="patid" placeholder="Place"
required>
```

```
    </div>

    <button type="submit" class="submit">Submit</button>
   </form>
  </div>
</body>
</html>
```

SUCCESS

Message.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Message</title>
  <style>
   body {
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
    margin: 0;
    padding: 0;
   }
   .container {
    max-width: 600px;
    margin: 50px auto;
    background-color: #fff;
    border-radius: 10px;
    padding: 20px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
   }
   .center {
```

```
      text-align: center;
    }
    h1 {
      font-size: 32px;
      color: #333;
    }
    .reference {
      font-size: 24px;
      color: #555;
      margin-top: 10px;
    }
    hr {
      border: none;
      border-top: 2px solid #ddd;
      margin: 20px 0;
    }
    a {
      text-decoration: none;
      color: #007bff;
      font-weight: bold;
    }
    a:hover {
      text-decoration: underline;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="center">
      <h1>Submitted Successfully!</h1>
      <div class="reference">Reference Number: <span id="ref">{{ref}}</span></div>
    </div>
    <hr />
    <div class="center">
```

```
      <a href="index.html">Go Back</a>
    </div>
  </div>
</body>
</html>
```

SEARCH

Search.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Search Report</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-image: url("/static/search.jpg");
      background-size: 1510px 900px;
      margin: 0;
      padding: 0;
    }
    .header {
      background-color: #333;
      padding: 10px;
      color: #fff;
      display: flex;
      justify-content: space-between;
      align-items: center;
    }
    .header a {
      color: #fff;
```

```
  text-decoration: none;
  padding: 5px 10px;
}
.header a:hover {
  background-color: #555;
}
h1 {
  text-align: center;
  color: #333;
}
form {
  max-width: 400px;
  margin: 20px auto;
  padding: 42px;
  background-color: #fff;
  border-radius: 8px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}
.input-container {
  position: relative;
  margin-bottom: 20px;
}
.input-container label {
  position: absolute;
  left: 10px;
  top: -17px;
  color: #888;
  transition: all 0.25s ease-in-out;
}
.input-container input {
  width: 100%;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
```

```
    outline: none;
   }
   .input-container input:focus + label,
   .input-container input:valid + label {
    top: -25px;
    font-size: 13px;
    color: #333;
   }
   .submit {
    display: block;
    width: 106%;
    padding: 10px;
    border: none;
    border-radius: 5px;
    background-color: #000000;
    color: #fff;
    font-size: 16px;
    cursor: pointer;
   }
   .submit:hover {
    background-color:#242323c5;
   }
   .message {
    text-align: center;
    margin-top: 20px;
    color: #333;
   }
  </style>
</head>
<body>
  <div class="header">
   <a href="book.html">Report Incident</a>
   <a href="index.html">Logout</a>
  </div>
```

```
 <div>
  <h1>Search Report</h1>
  <form action="/panelsearch" method="post">
   <div class="form">
    <div class="input-container ic2">
     <input
      id="caseno"
      class="input"
      type="text"
      placeholder=" "
      name="caseno"
      required
     />
     <label for="caseno" class="placeholder">Case Number</label>
    </div>
    <button type="submit" class="submit">Search</button>
   </div>
  </form>
 </div>
 <div class="message">{{res}}</div>
</body>
</html>
```

APP.PY

```
from flask import Flask, render_template, request, session, redirect, url_for,flash
from flask_mysqldb import MySQL
import MySQLdb.cursors
import re
import blockChain
import os
import json
import random
```

```
app = Flask(__name__)

app.secret_key = 'your secret key'
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = 'Adhulya'
app.config['MYSQL_DB'] = 'FRS'

mysql = MySQL(app)
username= ''

@app.route('/')
def main_page():
    return render_template('index.html')

@app.route('/index')
def index():
    return render_template('index.html')


@app.route('/index.html')
def home():
    return render_template("index.html")


@app.route('/panel.html')
def panel():
    return render_template('panel.html')

def generate_number():
    return random.randint(1000, 9999)


@app.route('/panelhome.html',methods=['get'])
```

```
def panelhome():
    #username = request.form['username']
    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cursor.execute("select * from bookdata")
    print("INN")
    data = cursor.fetchall()
    print(data)
    # Redirect to home page
    return render_template('panelhome.html', data=data, username=username)



@app.route('/users.html')
def users():
    return render_template('users.html')

@app.route('/users_home.html', methods=["post", "get"])
def users_home():
    return render_template("users_home.html")


@app.route('/addFRS.html')
def addFRS():
    return render_template('addFRS.html')


@app.route('/book.html')
def book():
    return render_template('book.html')


@app.route('/blockchain.html')
def blockchain():
    return render_template('blockchain.html')
```

```
@app.route('/panelregister.html')
def panelregiter():
    return render_template("panelregister.html")

@app.route('/message.html')
def successMessage():
    return render_template('message.html')


@app.route('/logout')
def logout():
    session.clear()
    return render_template('index.html')


@app.route('/userslogin', methods=["post", "get"])
def userslogin():
    if request.method == "get":
        return f"The URL /data is accessed directly. Try going to '/form' to submit form"
    else:
        msg = ''
        username = request.form["username"]
        password = request.form["password"]
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM userdata WHERE username = %s AND password =
%s', ([username, password]))
        account = cursor.fetchone()
        if account:
            session['logged'] = True
            session['id'] = account['id']
            session['username'] = account['username']
            # Redirect to home page
            return render_template('users_home.html', username=username)
        else:
```

```
            msg = 'Incorrect username/password!'
        return render_template('users.html', msg=msg)


@app.route("/userssregister", methods=["post", "get"])
def usersregister():
    if request.method == "get":
        return f"The URL /data is accessed directly. Try going to '/form' to submit form"
    else:
        msg = ''
        username = request.form['username']
        password = request.form['password']
        email = request.form['email']
        phone = request.form['phone_number']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM userdata WHERE username = %s', ([username],))
        account = cursor.fetchone()
        if account:
            msg = 'Account already exists!'
            return render_template('users.html', msg=msg)
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address!'
            return render_template('users.html', msg=msg)
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'Username must contain only characters and numbers!'
            return render_template('users.html', msg=msg)
        elif not username or not password or not email or not phone:
            msg = 'Please fill out the form!'
            return render_template('users.html', msg=msg)
        else:

            cursor.execute('INSERT INTO userdata VALUES (NULL, %s, %s, %s, %s)',
                           (username, password, email, phone))
            mysql.connection.commit()
```

```
            msg = 'You have successfully registered!'
        return redirect(url_for('users_home', msg=msg, username=username))


@app.route('/adddata', methods=['post', 'get'])
def adddata():
    if request.method == "get":
        return f"The URL /data is accessed directly. Try going to '/form' to submit form"
    else:
        msg = ''
        record_id = request.form['rid']
        vcnum = request.form['cnum']
        vname = request.form['name']
        vage = request.form['age']
        vgender = request.form['gender']
        vphy = request.form['phy']
        vuni = request.form['uni']
        vex = request.form['ex']
        vint = request.form['int']
        vdeath = request.form['death']
        vevidence = request.form['evidence']
        vresult = request.form['result']
        text = record_id  + vcnum + vname + vage + vgender + vphy + vuni + vex + vint +
vdeath + vevidence + vresult
        print(type(text))
        print(text)
        if len(text) < 1:
            return redirect(url_for('index'))
        try:
            make_proof = request.form['make_proof']
        except Exception:
            make_proof = False
        blockChain.write_block(text, record_id, vcnum, vname, vage, vgender, vphy, vuni, vex,
vint, vdeath, vevidence, vresult, make_proof)
```

```
        if not record_id  or not vname or not vage or not vgender or not vphy or not vuni or not
vex or not vint or not vdeath or not vevidence or not vresult:
            msg = 'Please Fill All the Fields'
            return render_template('addFRS.html', msg=msg)
        else:
            cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
            cursor.execute('INSERT INTO record VALUES (NULL, %s, %s, %s, %s ,%s, %s,
%s, %s, %s,%s,%s)',
                           (vname, vage, vgender ,vphy, vuni, vex, vint, vdeath, vevidence, vresult,
vcnum))
            mysql.connection.commit()
            msg = 'Data Successfully stored into Block chain'
        return render_template('blockchain.html', msg=msg)


@app.route('/check', methods=['POST'])
def integrity():
    results = blockChain.check_blocks_integrity()
    if request.method == 'POST':
        return render_template('blockchain.html', results=results)
    return redirect(url_for('users_home'))


@app.route('/mining', methods=['POST'])
def mining():
    if request.method == 'POST':
        max_index = int(blockChain.get_next_block())

        for i in range(2, max_index):
            blockChain.get_POW(i)
        return render_template('blockchain.html', querry=max_index)
    return redirect(url_for('users_home'))
```

```
@app.route('/bookdata', methods=['post', 'get'])
def book_data():
    if request.method == "get":
        return f"The URL /data is accessed directly. Try going to '/form' to submit form"
    else:
        msg = ''
        username = request.form['name']
        address = request.form['age']
        forensicdetail = request.form['forensicdetail']
        time = request.form['time']
        patient_id = request.form['patid']
        if not username or not address or not forensicdetail or not time or not patient_id:
            msg = 'Please Fill All the Fields'
            return render_template('book.html', msg=msg)
        else:
            cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
            cursor.execute('INSERT INTO bookdata VALUES (NULL, %s, %s, %s, %s, %s,
%s)',
                           (username, address, forensicdetail, time, patient_id))
            mysql.connection.commit()
            msg = 'Data Successfully stored into Block chain'
        return render_template('users_home.html', msg=msg)


@app.route('/panellogin', methods=['post', 'get'])
def panellogin():
    if request.method == "get":
        return f"The URL /data is accessed directly. Try going to '/form' to submit form"
    else:
        msg = ''
        username = request.form["username"]
        password = request.form["password"]
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
```

```
        cursor.execute('SELECT * FROM panel WHERE username = %s AND password = %s',
([username, password]))
        account = cursor.fetchone()
        if account:
            session['logged'] = True
            session['id'] = account['id']
            session['username'] = account['username']
            return render_template('book.html')

        else:
            msg = 'Incorrect username/password!'
        return render_template('panel.html', msg=msg)

@app.route('/panelbook', methods=['post', 'get'])
def panelbook():
    if request.method == "get":
        return f"The URL /data is accessed directly. Try going to '/form' to submit form"
    else:
        msg = ''
        ref = ''
        username = request.form['name']
        address = request.form['age']
        forensicdetail = request.form['forensicdetail']
        time = request.form['time']
        patient_id = request.form['patid']

        # Generate a random number
        referncenum = generate_number()
        print(referncenum)

        if not username or not address or not forensicdetail or not time or not patient_id:
            msg = 'Please Fill All the Fields'
            return render_template('book.html', msg=msg)
        else:
```

```
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('INSERT INTO bookdata VALUES (NULL, %s, %s, %s, %s, %s,
%s)',
                       (username, address, forensicdetail, time, patient_id, referncenum))
        mysql.connection.commit()
        msg = 'Data Successfully stored into Block chain'
        print(msg)
    return render_template('message.html', msg=msg, ref=referncenum)

@app.route('/search.html', methods=['GET'])
def search_page():
    return render_template('search.html')

@app.route('/panelsearch', methods=['GET','POST'])
def search_report_page():
    res = {}

    if request.method == 'POST':
        casenum = request.form.get('caseno')

        if casenum:
            cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
            cursor.execute('SELECT * FROM record WHERE casenum = %s', ([casenum],))
            account = cursor.fetchone()

            if account:
                return render_template('details.html', res=account)

            res = 'CASE IN PROGRESS'
    return render_template('search.html', res=res)

@app.route('/pregister', methods=['post', 'get'])
def pregister():
    if request.method == "get":
```

```
        return f"The URL /data is accessed directly. Try going to '/form' to submit form"
    else:
        msg = ''
        username = request.form['username']
        password = request.form['password']
        email = request.form['email']
        phone = request.form['phone']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM panel WHERE username = %s', ([username],))
        account = cursor.fetchone()
        if account:
            msg = 'Account already exists!'
            return render_template('panel.html', msg=msg)
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address!'
            return render_template('panel.html', msg=msg)
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'Username must contain only characters and numbers!'
            return render_template('panel.html', msg=msg)
        elif not username or not password or not email or not phone:
            msg = 'Please fill out the form!'
            return render_template('panel.html', msg=msg)
        else:

            cursor.execute('INSERT INTO panel VALUES (NULL, %s, %s, %s, %s)',
                           (username, password, email, phone))
            mysql.connection.commit()
            msg = 'You have successfully registered!'
        return render_template('panel.html', msg=msg, username=username)
@app.route('/test', methods=['GET'])
def test():
    return render_template('panel.html')

@app.route('/outputTable', methods=['GET'])
```

```
def generate_html_from_json_folder():
    results = blockChain.check_blocks_integrity()
    return render_template('output.html', results=results)

@app.route('/panelhome.html')
def fetch_row():
    # Fetch data from the database and pass it to the template
    data = fetch_data_from_database()
    return render_template('panelhome.html', data=data)

@app.route('/remove_row/<int:row_id>', methods=['POST'])
def remove_row(row_id):
    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cursor.execute('DELETE FROM bookdata WHERE id = %s', (row_id,))
    mysql.connection.commit()

def fetch_data_from_database():
    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cursor.execute('SELECT * FROM bookdata')
    data = cursor.fetchall()
    data = fetch_data_from_database()
    return render_template('panelhome.html', data=data)

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=8000)
```

BLOCKCHAIN

```
import hashlib
import json
import os
from time import time
```

```
BLOCKCHAIN_DIR = os.curdir + '/blocks/'


def check_blocks_integrity():
    result = list()
    cur_proof = - 1
    for i in range(2, int(get_next_block())):
        prev_index = str(i - 1)
        cur_index = str(i)
        tmp = {'block': '', 'result': '', 'proof': '','text':''}
        try:
            file_dict = json.load(open(BLOCKCHAIN_DIR + cur_index + '.json'))
            cur_hash = file_dict['prev_hash']
            cur_proof = file_dict['proof']
            cur_text= file_dict['text']
            cur_record_id = file_dict['record_id'];
            cur_cnum = file_dict['cnum']
            cur_name = file_dict['name']
            cur_age = file_dict['age']
            cur_gender = file_dict['gender']
            cur_phy= file_dict['phy']
            cur_uni = file_dict['uni']
            cur_ex = file_dict['ex']
            cur_int = file_dict['int']
            cur_death = file_dict['death']
            cur_evidence = file_dict['evidence']
            cur_result = file_dict['vresult']

        except Exception as e:
            print(e)
        try:
            prev_hash = hashlib.sha256(open(BLOCKCHAIN_DIR + prev_index + '.json',
'rb').read()).hexdigest()
        except Exception as e:
```

```
            print(e)
        tmp['block'] = prev_index
        tmp['proof'] = cur_proof
        tmp['text'] = cur_text
        tmp['record_id'] = cur_record_id
        tmp['cnum'] = cur_cnum
        tmp['name'] = cur_name
        tmp['age'] = cur_age
        tmp['gender'] = cur_gender
        tmp['phy'] = cur_phy
        tmp['uni'] = cur_uni
        tmp['ex'] = cur_ex
        tmp['int'] = cur_int
        tmp['death'] = cur_death
        tmp['evidence'] = cur_evidence
        tmp['vresult'] = cur_result
        if cur_hash == prev_hash:
            tmp['result'] = 'ok'
        else:
            tmp['result'] = 'error'
        result.append(tmp)
    return result


def check_block(index):
    cur_index = str(index)
    prev_index = str(int(index) - 1)
    cur_proof = - 1
    cur_hash = 0
    prev_hash = 0
    tmp = {'block': '', 'result': '', 'proof': ''}
    try:
        file_dict = json.load(open(BLOCKCHAIN_DIR + cur_index + '.json'))
        cur_hash = file_dict['prev_hash']
```

```
        cur_proof = file_dict['proof']
    except Exception as e:
        print(e)
    try:
        prev_hash = hashlib.sha256(open(BLOCKCHAIN_DIR + prev_index + '.json',
'rb').read()).hexdigest()
    except Exception as e:
        print(e)
    tmp['block'] = prev_index
    tmp['proof'] = cur_proof
    if cur_hash == prev_hash:
        tmp['result'] = 'ok'
    else:
        tmp['result'] = 'error'
    return tmp


def get_hash(file_name):
    file_name = str(file_name)
    if not file_name.endswith('.json'):
        file_name += '.json'
    try:
        with open(BLOCKCHAIN_DIR + file_name, 'rb') as file:
            return hashlib.sha256(file.read()).hexdigest()
    except Exception as e:
        print('File "' + file_name + '" does not exist!n', e)


def get_next_block():
    files = os.listdir(BLOCKCHAIN_DIR)
    index_list = [int(file.split('.')[0]) for file in files]
    cur_index = sorted(index_list)[-1]
    next_index = cur_index + 1
    return str(next_index)
```

```
def is_valid_proof(last_proof, proof, difficulty):
    guess = f'{last_proof}{proof}'.encode()
    guess_hash = hashlib.sha256(guess).hexdigest()
    return guess_hash[:difficulty] == '0' * difficulty


def get_POW(file_name, difficulty=1):
    # POW - proof of work
    file_name = str(file_name)
    if file_name.endswith('.json'):
        file_name = int(file_name.split('.')[0])
    else:
        file_name = int(file_name)

    last_proof = json.load(open(BLOCKCHAIN_DIR + str(file_name - 1) + '.json'))['proof']
    proof = 0
    while is_valid_proof(last_proof, proof, difficulty) is False:
        proof += 1
    cur_block = json.load(open(BLOCKCHAIN_DIR + str(file_name) + '.json'))
    cur_block['proof'] = proof
    cur_block['prev_hash'] = get_hash(str(file_name - 1))
    with open(BLOCKCHAIN_DIR + str(file_name) + '.json', 'w') as file:
        json.dump(cur_block, file, indent=4, ensure_ascii=False)


def write_block(text, record_id, vcnum, vname, vage, vgender, vphy, vuni, vex, vint, vdeath,
vevidence, vresult, make_proof=False):
    cur_index = get_next_block()
    prev_index = str(int(cur_index) - 1)
    prev_block_hash = get_hash(prev_index)
    data = {'text': text,
            'prev_hash': prev_block_hash,
            'timestamp': time(),
```

```
            'proof': -1,
            'index': cur_index,
            'record_id': record_id,
            'cnum': vcnum,
            'name': vname,
            'age': vage,
            'gender': vgender,
            'phy': vphy,
            'uni': vuni,
            'ex': vex,
            'int': vint,
            'death': vdeath,
            'evidence': vevidence,
            'vresult': vresult
        }

    with open(BLOCKCHAIN_DIR + cur_index + '.json', 'w') as file:
        json.dump(data, file, indent=4, ensure_ascii=False)
    if make_proof is True:
        get_POW(str(cur_index))


if __name__ == '__main__':
    # for i in range(10):
    # write_block(str(i),True)
    for i in range(2, 10):
        print(check_block(str(i)))
    print(check_blocks_integrity())
```

8. TESTING

8.1 Objectives

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

8.2 Test Process

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

8.3 Test Cases

A test case is a set of conditions or variables under which a tester will determine whether a system under test satisfies requirements or works correctly. The process of developing test cases can also help find problems in the requirements or design of an application. A test case can have the following elements. Note, however, that a test management tool is normally used by companies and the format is determined by the tool used.

8.4 Testing Types

8.4.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application; it is done after the completion of an individual unit before integration. This is a structural testing that relies on knowledge of its construction and is invasive.

Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

8.4.2 Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation and user manuals. Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

8.4.3 System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System is based on process descriptions and flows, emphasizing pre-driven process links testing and integration points. The process of performing a variety of tests on a system to explore functionality or to identify problems. System testing is usually required before and after a system is put in place. A series of systematic procedures are referred to while testing is being performed. These procedures tell the tester how the system should perform and where common mistakes may be found. Testers usually try to "break the system" by entering data that may cause the system to malfunction or return incorrect information For example, a tester may put in a city in a search engine designed to only accept states, to see how the system will respond to the incorrect input.

8.5 Verification and Validation

8.5.1 Verification

Verification is the process of evaluating work-products of a development phase to determine whether they meet the specified requirements.

- Evaluates the intermediary products to check whether it meets the specific requirements of the particular phase
- Checks whether the product is built as per the specified requirement and design specification
- Checks "Are we building the product right"?
- This is done without executing the software
- Involves all the static testing techniques
- Examples includes reviews, inspection and walkthrough

8.5.2 Validation

The process of evaluating software during the development process or at to determine whether it satisfies specified business of the development process requirements. ValidationTesting ensures that the product actually meets the client's needs. It can also be defined as to demonstrate that the product fulfills its intended end when deployed in an appropriate environment**.**

- Evaluates the final product to check whether it meets the business needs. It determines whether the software is fit for use and satisfies the business need.
- Checks "Are we building the right product"?
- Is done with executing the software
- Example includes all types of testing like smoke, regression, functional, systems and UA

9. REPORT

Input

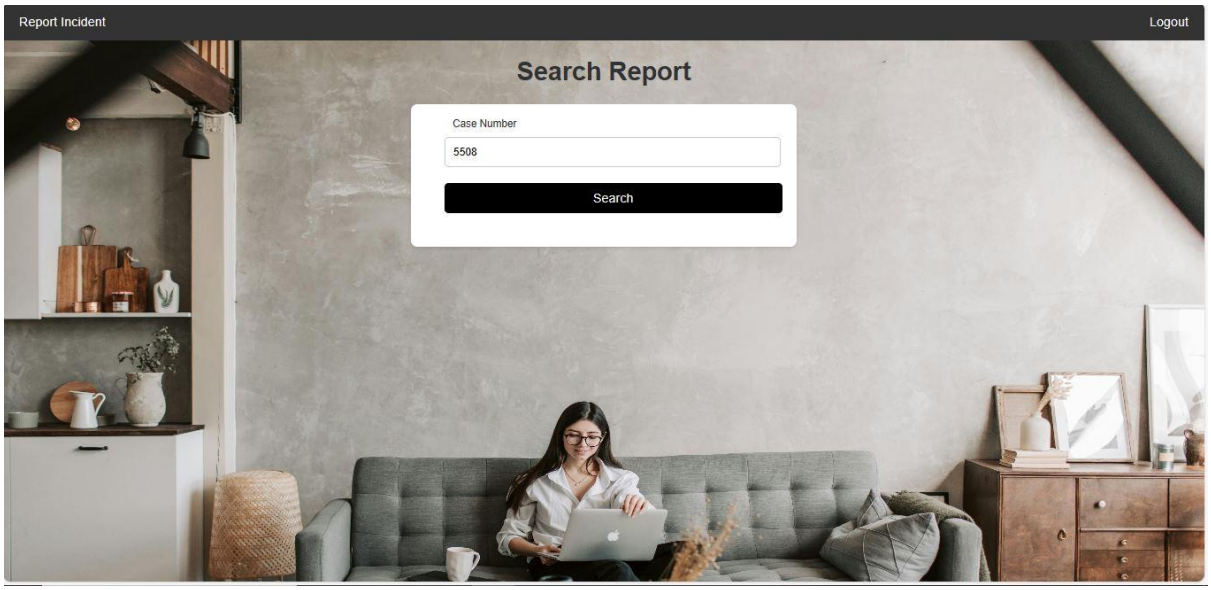


Figure 9.1: Search Case

Output

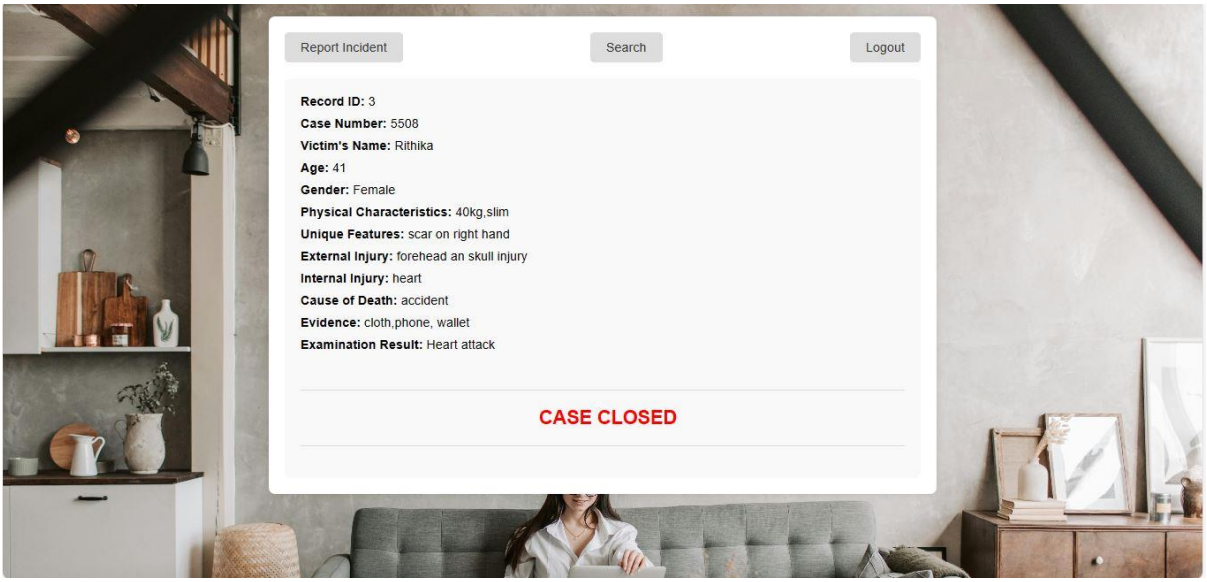


Figure 9.2: Output of Search case

Conclusion

A New Approach in Forensic Blockchain project has been successfully completed with various features and benefits the Officers and the Forensic team. This has significantly simplified the travel planning process for users. Efficiency: The project incorporated efficiency strategies by integrating various tools and algorithms. The use of blockchain technology to enhance Forensic report integrity in the criminal justice system is crucial. It provides an inexpensive and comprehensive solution, allowing agencies to maintain existing forensic reports while minimizing human error and data cannot be modified or changed. Through blockchain, verified evidence tracking across law enforcement agencies becomes possible, with only hashed representations of evidence states registered. This system ensures the case details is stored safely. Forensic blockchain systems can improve investigation accuracy and speed.

FUTURE ENHANCEMENT

The integration of forensic blockchain systems with the Internet of Things (IOT) holds significant promise in revolutionizing the collection and analysis of forensic evidence. By leveraging IOT devices deployed at crime scenes, data can be collected in real-time and transmitted directly to forensic blockchain systems, expediting investigations and enhancing accuracy. Additionally, the integration of forensic blockchain with emerging technologies like virtual and augmented reality offers new dimensions to forensic investigations. For instance, virtual reality can recreate crime scenes, aiding investigators in identifying potential evidence. Moreover, incorporating machine learning algorithms into forensic blockchain systems automates the process of evidence identification and analysis, resulting in faster and more accurate forensic investigations. This convergence of technologies represents a transformative shift in forensic science, promising to optimize investigative processes and improve outcomes

REFERENCES

[1] Fadi, Z. Karim, E. G. Abdellatif and B. Mohammed, "A Survey on Blockchain and Artificial Intelligence Technologies for Enhancing Security and Privacy in Smart Environments" in IEEE Access, vol. 10, pp. 93168-93186, 2022, doi: 10.1109/ACCESS.2022.3203568.

[2] F. Xiaohua, C., et.al., "Artificial Intelligence and Blockchain for Future Cyber Security Application," 2021 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligenee and Computing, Intl Cont on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress .AB, Canada, 2021, pp. 802-805.

[3] Huh, S., Cho, S., and Kim, S. (2021). "Managing devices using block chain St platform in Proceedings of the 22th International Conference on Advanced Communication Technology (Piscataway, NJ: IEE), 464-467.

[4] K. Sharma and N. Jain, "An Integration of Block chain and Artificial intelligenee: A Coneept," 2021 International Conference on Intelligent Computing and Control Systems (ICCS), Madurai, India, 2021, pp. 1487-1490.

[S] K. Salah, M. H. U. Rehman, N. Nizamuddin and A. Al-Fuqaha, "Blockchain for Al: Review and Open Research Challenges" in IEEE Access, vol. 7, pp. 10127-10149, 2020, doi: 10.1109/AC- CESS.2018.2890507.

[6] M. J. J. Gul, A. Paul, S. Rho and M. Kim, "Blockchain based healthcare system with Artificial Intelligence" 2020 International Conference on Computational Science and Computational Intelligence (CSCD, Las Vegas, NV, USA, 2020, pp. 740-741, doi: 10.1109/CSCIS1800.2020.00138.