# ACCIDENT RECORD

# MANAGEMENT SYSTEM

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

**Bachelor of Computer Science**

In the faculty of Computer Science of Bharathiar University, Coimbatore

Submitted by

**A.JERLIN JEMELA**

**(Reg.No.2022K0131)**

Under the guidance of

**Mrs.N.SUGANTHI MCA.,M.Phil.,(Ph.D)**

Guest lecturer , Department of Computer Science

**Department of Computer Science**

## LRG GOVERNMENT ARTS COLLEGE FORWOMEN

**(Affiliated To Bharathiar University)**

**TIRUPUR-4**

**APRIL-2023**

# *CERTIFICATE*

# CERTIFICATE

This is to certify that the project work entitled **" ACCIDENT RECORD MANAGEMENT SYSTEM"** Submitted to Bharathiar University in partial fulfilled of the requirement for the award of the Degree of Bachelor of computer science is a record of the original work done by **Ms.A.JERLIN JEMELA (Reg.No.2022K0131)** Under my supervisor and that project work has not formed the basis for the any Degree /Diploma /Association /Fellowship or similar title to any candidate of any university.

**Internal Guide** **Head of the Department**

**Mrs.N.SUGANTHI MCA.,M.Phil.,(Ph.D)** **Dr.R.PARIMALAM.Sc.,M.Phil.,Ph.D.,**

Viva-voce examination is held on _____L.R.G Government Arts College for Women, Tirupur-641604.

**Internal Examiner** **External Examiner**

# *DECLARATION*

# DECLARATION

I hereby declare that the project work submitted to the **UG Department of the Computer Science, L.R.G. Government Arts College for Women, Tirupur** , affiliated to Bharathiar University, Coimbatore in the partial fulfillment of the required for the award of Bachelor of Computer Science is an original work done by me during the sixth semester.

**Place:**

**Date:**

**Signature of the Candidate**

**(A.JERLIN JEMELA)**

**(Reg.No:2022K0131)**

# *ACKNOWLEDGEMENT*

# ACKNOWLEDGEMENT

As first and foremost I would like to external my thankfulness to the almighty for blessing the work of my hands, I am grateful to my parents for continued encouragement that they had given to me.

I would like to external my profound gratitude and sincere thanks to **Dr.M.R.YEZHILI MA.,M.Phil.,Ph.D.,** Principal, L.R.G Government Arts College For Women , for the encouragement rendered to me during this project.

It's my privilege to thank **DR.R.PARIMALA MSc.,M.Phil.,Ph.D.,**Incharge Head of the department of computer science for her valuable guidance and support throughout the project development work.

I express my deep sense of gratitude and sincere thanks to my guide **Mrs.N.SUGANTHI MCA.,M.Phil.,(Ph.D)**Lecturer, Department of computer science, for providing all sorts of facilities to complete my project work successfully.

I also extend my sincere thanks to all the other faculty member of the department and technical assistance for their co-operation and valuable guidance.

I thank our college library for providing me with many informative books that help me to enrich my knowledge to bring out the project successfully.

# *SYNOPSIS*

# SYNOPSIS

The Accident Record Management System (ARMS) was produced utilizing Adobe Dreamweaver as the Integrated Development Environments JAVA were utilized for the frontend, was utilized as the scripting dialect, and SQL filled in as the database server. Most of the languages and tools utilized were open source which guaranteed that the application would be robust, reusable, cheap and highly scalable. The result of the developed system demonstrates that road safety officers, clients, policy producers and every other stakeholder can have the capacity to register, login, submit reports and run questions on data that has been already gone into the system, for example, the accident that happened on a particular route or the accident in which a recognized casualty was included. Policy creators can run these inquiries keeping in mind the end goal to make appropriate steps in minimizing road traffic accident occurrences. In conclusion, this system will help make a paperless other option to the present strategy for ARMS data and consequently make data dissemination faster and furthermore enhance first aid to accident occurrences.

*CONTENTS*

## TABLE OF CONTENTS

# 1.INTRODUCTION

## 1.1 OVERVIEW OF THE PROJECT

The accident record management system has been to handle the accident entire details. Which should be only handle by the traffic police inspector. When the accident was happened at any where they should arrive and register the accident details. Before that they should have user login credential to access this application. They can login and register accident details. This application also can be track the accident person previous history details.

These all the records are stored in the database when the inspector register the details via this application. When the inspector wants to search the person previous record can able to track and find the exising accident details as well using their Aadhar or any other ID proof.

The Accident Record Management System (ARMS) was produced utilizing Adobe Dreamweaver as the Integrated Development Environments JAVA were utilized for the frontend, was utilized as the scripting dialect, and MySQL filled in as the database server. Most of the languages and tools utilized were open source which guaranteed that the application would be robust, reusable, cheap and highly scalable.

## 1.2 SYSTEM SPECIFICATION

System Requirements Specification also known as Software Requirements Specification, is a document or set of documentation that describes the features and behavior of a software application.

**WINDOWS OS**

Windows is a graphical operating system developed by Microsoft. It allows users to view and store files, run the software, play games, watch videos, and provides a way to connect to the internet. It was released for both home computing and professional works.

Microsoft introduced the first version as 1.0

It was released for both home computing and professional functions of Windows on 10 November 1983. Later, it was released on many versions of Windows as well as the current version, Windows 10.

In 1993, the first business-oriented version of Windows was released, which is known as Windows NT 3.1. Then it introduced the next versions, Windows 3.5, 4/0, and Windows 2000. When the XP Windows was released by Microsoft in 2001, the company designed its various versions for a personal and business environment. It was designed based on standard x86 hardware, like Intel and AMD processor. Accordingly, it can run on different brands of hardware, such as HP, Dell, and Sony computers, including home-built PCs.
Play Video

Editions of Windows
Microsoft has produced several editions of Windows, starting with Windows XP. These versions have the same core operating system, but some versions included advance features with an additional cost. There are two most common editions of Windows:

- Windows Home
- Windows Professional

Windows Home is basic edition of Windows. It offers all the fundamental functions of Windows, such as browsing the web, connecting to the Internet, playing video games, using

office software, watching videos. Furthermore, it is less expensive and comes pre-installed with many new computers.

**JAVA**

Java is a high-level programming language developed by Sun Microsystems. It was originally designed for developing programs for set-top boxes and handheld devices, but later became a popular choice for creating web applications.

The Java syntax is similar to C++, but is strictly an object-oriented programming language. For example, most Java programs contain classes, which are used to define objects, and methods, which are assigned to individual classes. Java is also known for being stricter than C++, meaning variables and functions must be explicitly defined. This means Java source code may produce errors or "exceptions" more easily than other languages, but it also limits other types of errors that may be caused by undefined variables or unassigned types.

Unlike Windows executable (.EXE files) or Macintosh applications (.APP files), Java programs are not run directly by the operating system. Instead, Java programs are interpreted by the Java Virtual Machine, or JVM, which runs on multiple platforms. This means all Java programs are multiplatform and can run on different platforms, including Macintosh, Windows, and Unix computers. However, the JVM must be installed for Java applications or applets to run at all. Fortunately, the JVM is included as part of the Java Runtime Environment (JRE).

**SQL**

Structured query language (SQL) is a programming language for storing and processing information in a relational database. A relational database stores information in tabular form, with rows and columns representing different data attributes and the various relationships between the data values. You can use SQL statements to store, update, remove, search, and retrieve information from the database. You can also use SQL to maintain and optimize database performance.

Relational database management systems use structured query language (SQL) to store and manage data. The system stores multiple database tables that relate to each other. MS SQL

Server, MySQL, or MS Access are examples of relational database management systems. The following are the components of such a system.

A SQL table is the basic element of a relational database. The SQL database table consists of rows and columns. Database engineers create relationships between multiple database tables to optimize data storage space.

SQL statements, or SQL queries, are valid instructions that relational database management systems understand. Software developers build SQL statements by using different SQL language elements. SQL language elements are components such as identifiers, variables, and search conditions that form a correct SQL statement.

### 1.2.1 HARDWARE SPECIFICATION

- Processor : P 4 700 GHz.
- RAM : 4 GB RAM
- Hard Disk Drive : 180 GB

### 1.2.2 SOFTWARE SPECIFICATION

- Operating System : Windows 7/8/10
- Front End : JAVA
- Back End : SQL

# 2.SYSTEM STUDY

## 2.1 EXISTING SYSTEM

This system is very hard to find accident which may cause some issues. It can't be searching the accident list if we want to search the particular criminal very difficult to tracking. accident details are stored as document wise so we can't search the criminals.

### 2.1.1. DRAWBACKS OF EXISTING SYSTEM

- There is not possible to find one particular accident detail.
- Lot of paper to be wasted
- It takes too much time to register the accident details

## 2.2 PROPOSED SYSTEM

In this accident monitoring system easily find the person has any case or not. Using an area id based we can find the accident details which could be easy to find the criminal details. Any time anywhere we can see about the accident detail, we no need to take the file anywhere.

### 2.2.1 FEATURES OF PROPOSED SYSTEM

- Easy to find accident details
- Fully systematic process
- No one will be escaped

# 3.SYSTEM DESIGN AND DEVELOPMENT

## 3.1 FILE DESIGN

The selection of the file system design approach is done according to the needs of the developers what are the needed requirements and specifications for the new design. It allowed us to identify where our proposal fitted in with relation to current and past file system development. Our experience with file system development is limited so the research served to identify the different techniques that can be used. The variety of file systems encountered show what an active area of research file system development is. The file systems may be from one of the two fundamental categories. In one category, the file system is developed in user space and runs as a user process. Another file system may be developed in the kernel space and runs as a privileged process. Another one is the mixed approach in which we can take the advantages of both aforesaid approaches. Each development option has its own pros and cons. In this article, these design approaches are discussed.

A file system is the data structure designed to support the abstraction of the data blocks as an archive and collection of files. This data structure is unique because it is stored on secondary storage (usually the disk), which is a very slow device.

The file system structure is the most basic level of organization in an operating system. Almost all of the ways an operating system interacts with its users, applications, and security model are dependent upon the way it organizes files on storage devices.

File Design Information systems in business are file and database oriented. Data are accumulated into files that are processed or maintained by the system. The systems analyst is responsible for designing files, determining their contents and selecting a method for organizing the data.

The most important purpose of a file system is to manage user data. This includes storing, retrieving and updating data. Some file systems accept data for storage as a stream of bytes which are collected and stored in a manner efficient for the media.

## 3.2 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

**OBJECTIVES**

- Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
- It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
- When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user
- will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

## 3.3 OUTPUT DESIGN

The design of output is the most important task of any system. During output design, developers identify the type of outputs needed, and consider the necessary output controls and prototype report layouts.

**External Outputs**

Manufacturers create and design external outputs for printers. External outputs enable the system to leave the trigger actions on the part of their recipients or confirm actions to their recipients.

Some of the external outputs are designed as turnaround outputs, which are implemented as a form and re-enter the system as an input.

**Internal outputs**

Internal outputs are present inside the system, and used by end-users and managers. They support the management in decision making and reporting.

**Output Integrity Controls**

Output integrity controls include routing codes to identify the receiving system, and verification messages to confirm successful receipt of messages that are handled by network protocol.

Printed or screen-format reports should include a date/time for report printing and the data. Multipage reports contain report title or description, and pagination. Pre-printed forms usually include a version number and effective date.

## 3.4 DATABASE DESIGN

Today's businesses depend on their databases to provide information essential for day-to-day operations, especially in case of electronic commerce businesses who has a definite advantage with up-to-date database access. Good design forms the foundation of any database, and experienced hands are required in the automation process to design for optimum and stable performance.

Software Solutions have been constantly working on these platforms and have attained a level of expertise. We apply proven methodologies to design, develop, integrate and implement database systems to attain its optimum level of performance and maximize security to meet the client's business model.

**Business needs addressed:**

- Determine the basic objects about which the information is stored
- Determine the relationships between these groups of information and the objects
- Effectively manage data and create intelligent information
- Remote database administration or on site administrative support
- Database creation, management, and maintenance
- Information retrieval efficiency, remove data redundancy and ensure data security

**Data Integrity**

Data integrity means storing all data in one place only and how each application accesses it. This approach results in more consistent information, one update being sufficient to achieve a new record status for all applications. This leads to less data redundancy that is data items need not be duplicated.

**Data Independence**

Data in dependence is the insulation of application programs from changing aspects of physical data organization. This objective seeks to allow changes in the content and organization of physical data without reprogramming of application and allow modifications to application programs without reorganizing the physical data.

## 3.5 SYSTEM DEVELOPMENT

Systems development is the process of defining, designing, testing, and implementing a new software application or program. It could include the internal development of customized systems, the creation of database systems, or the acquisition of third party developed software.

Systems development life cycle phases include planning, system analysis, system design, development, implementation, integration and testing, and operations and maintenance.

### 3.5.1 DESCRIPTION OF MODULES

1.Admin
2.Accident details
3.Case Registration
4.Accident History

**MODULES DESCRIPTION**

### 1.Admin Login

Here admin is an only one login to follow this software, if there's an lot of admin's are there they can also normally use it. Admin the main role is an do after the login.

### 2.Accident details entry

If anywhere the accident has been happened the admin collect all the information and store this module. These values are stored into the database table. Once the details have been captured it will automatically close.

### 3.Case Registration

This module will be handling the process of register the case to the respective members. If once case has been scheduled police department will take care for the investigation.

### 4.Accident History

We can track the accident history by the accident id, collect the all the information about the accident and display in front of the screen.

# 4. SYSTEM TESTING AND IMPLEMENTATION

## 4.1 TESTING METHODOLOGIES

System testing is state of implementation, which is aimed at ensuring that the system works accurately and efficiently as expect before live operation commences. It certifies that the whole set of programs hang together.

System testing requires a test plan that consists of several key activities and step for run program, string, system and user acceptance testing. The implementation of newly designed package is important in adopting a successful new system

Testing is the important stage in software development. the system test in implementation stage in software development process. The system testing implementation should be confirmation that all is correct and an opportunity to show the users that the system works as expected. It accounts the largest percentage of technical effort in the software development process.

Testing phase in the development cycle validates the code against the functional specification testing is vital to achievement of the system goals. The objective of the testing is to discover errors to fulfill this objective a series of test step unit, integration. Validation and system tests were planned and executed the test steps are:

## SYSTEM TESTING

Testing is an integral part of any system development life cycle. Insufficient and untested applications may tend to crash and the result is loss of economic and manpower investment besides user's dissatisfaction and downfall of reputation. Software testing can be looked upon as one among many processes, an organization performs, and that provides the lost opportunity to correct any flaws in the developed system. Software testing includes selecting test data that have more probability of giving errors.

The first step in system testing is to develop a plan that tests all aspects of the system. Completeness, correctness, reliability and maintainability of the software are to be tested for the best quality assurance that the system meets the specification and requirements for its intended use and performance. System testing is the most useful practical process of executing a program with the implicit intention of finding errors that make the program fails. System testing is done in three phases.

- Unit Testing
- Integration Testing
- Validation Testing

**UNIT TESTING**

Unit testing focuses verification effort on the smallest unit of software the module. Using the detailed design and the process specification testing is done to registration by the user with in the boundary of the Login module. The login form receives the username and password details and validates the value with the database. If valid, the home page is displayed.

**INTEGRATION TESTING**

Integration Testing is the process of this activity can be considered as testing the design and hence module interaction. The primary objective of integration testing is to discover errors in the interfaces between the components. Login form and registration form are integrated and tested together. If the user is newly registered, the received details will be stored in the registration table. While logging in, the application will check for valid user name and password in the registration table and if valid the user is prompted for submitting complaints.

Data can be lost across an interface, one module can have adverse effect on another sub function when combined it may not produce the desired major functions. Integration testing is a systematic testing for constructing test to uncover errors associated within an interface.

The objectives taken from unit tested modules and a program structure is built for integrated testing. All the modules are combined and the test is made.

A correction made in this testing is difficult because the vast expenses of the entire program complicated the isolation of causes. In this integration testing step, all the errors are corrected for next testing process.

## VALIDATION TESTING

Validation are independent procedures that are used together for checking that a product, service, or system meets requirements and specifications and that it fulfills its in purpose the actual result from the expected result for the complaint process. Select the complaint category of the complaint by user. The input given to various forms fields are validated effectively. Each module is tested independently. It is tested that the complaint module fields receive the correct input for the necessary details such as complaint category, complaint id, reference name, complaint description, and email for further process.

After the completion of the integrated testing, software is completely assembled as a package; interfacing error has been uncovered and corrected and a final series of software test validation begins.

Validation testing can be defined in many ways but a simple definition is that validation succeeds when the software function in a manner that can be reasonably expected by the customer. After validation test has been conducted, one of two possible conditions exists.

## OUTPUT TESTING

The next process of validation testing, is output testing of the proposed system, since no system could be successful if it does not produce the required output in the specified format. Asking the user about the format required, list the output to be generated or displayed by the system under considerations.

Output testing is a different test whose primary purpose is to fully exercise the computer based system although each test has a different purpose all the work should verify that all system elements have been properly integrated and perform allocated functions.

The output format on the screen is found to be corrected as the format was designed in the system design phase according to the user needs for the hard copy also; the output testing has not resulted in any correction in the system.

## 4.2 SYSTEM IMPLEMENTATION

When the initial design was done for the system, the client was consulted for the acceptance of the design so that further proceedings of the system development can be carried on. After the development of the system a demonstration was given to them about the working of the system. The aim of the system illustration was to identify any malfunction of the system.

After the management of the system was approved the system implemented in the concern, initially the system was run parallel with existing manual system. The system has been tested with live data and has proved to be error free and user friendly.

Implementation is the process of converting a new or revised system design into an operational one when the initial design was done by the system; a demonstration was given to the end user about the working system.

This process is uses to verify and identify any logical mess working of the system by feeding various combinations of test data. After the approval of the system by both end user and management the system was implemented.

System implementation is made up of many activities. The six major activities are as follows.

### CODING

Coding is the process of whereby the physical design specifications created by the analysis team turned into working computer code by the programming team. A design code may be a tool which helps ensure that the aspiration for quality and quantity for customers and their requirements, particularly for large scale projects, sought by the water agency Design pattern are documented tried and tested solutions for recurring problems in a given context. So basically you have a problem context and the proposed solution for the same.

### INSTALLATION

Installation is the process during which the current system is replaced by the new system. This includes conversion of existing data, software, and documentation and work procedures to those consistent with the new system.

## DOCUMENTATION

Documentation is descriptive information that describes the use and operation of the system. The user guide is provided to the end user as the student and administrator. The documentation part contains the details as follows,

User requirement and water agency details administration has been made online. Any customer can request their water requirement details through online and also use of documentation, they can view the purpose of each purpose, The admin could verify the authentication of the users, users requirements and need to take delivery process, thus the documentation is made of full view of project thus it gives the guideline to study the project and how to execute also.

## USER TRAINING AND SUPPORT

The software is installed at the deployment environment, the developer will give training to the end user of the regional transport officer and police admin officer in that software. The goal of an end user training program is to produce a motivated user who has the skills needed to apply what has been to apply what has been learned to perform the job related task. The following are the instruction which is specified the handling and un-handling events in the application,

- The authenticated user of admin and office workers only login in the application with authorized username and password.
- Don’t make user waste their time to come straight to the water agency or make a phone call.
- It can easily track through online by the user.
- Very user friendliness software

## 4.3 IMPLEMENTATION PROCEDURES

Implementation includes all the activities that take place to convert the old system to the new one. Proper implementation is essential to provide a reliable system to meet the organization requirements. Implementation is the stage in the project where the theoretical design is turned into a working system. The most crucial stage is achieving a successful new system & giving the user confidence in that the new system will work efficiently & effectively in the implementation state.

### 4.3.1 IMPLEMENTATION PROCEDURES

**PILOT RUNNING**

Processing the current data by only one user at a time called the pilot running process. When one user is accessing the data at one system, the system is sets to be engaged and connected in network. This process is useful only in system where more than one user is restricted.

**PARALLEL RUNNING:**

Processing the current data by more than one user at a time simultaneously is said to be parallel running process. This same system can be viewed and accessed by more than one user at the time. Hence the implementation method used in the system is a pilot type of implementation.

Implementation is the stage in the project where the theoretical design is turned into a working system. The most crucial stage is achieving a successful new system & giving the user confidence in that the new system will work efficiently & effectively in the implementation state.

The stage consists of,

- Testing the developed program with sample data.
- Detection's and correction of error.
- Creating whether the system meets user requirements.
- Making necessary changes as desired by the user.
- Training user personnel.

## 4.3.2 USER MANUAL

**USER TRAINING**

User Training is designed to prepare the user for testing &consenting the system. .

- User Manual.
- Help Screens.
- Training Demonstration.

**USER MANUAL**

The summary of important functions about the system and software can be provided as a document to the user.

**HELP SCREENS**

This features now available in every software package, especially when it is used with a menu. The user selects the "Help" option from the menu. The system accesses the necessary description or information for user reference.

**TRAINING DEMONSTRATION:**

Another User Training element is a Training Demonstration. Live demonstrations with personal contact are extremely effective for Training Users.

## 4.4 SYSTEM MAINTENANCE

Maintenance is actually the implementation of the review plan. As important as it is, many programmers and analysts are to perform or identify themselves with the maintenance effort. There are psychological, personality and professional reasons for this. Analysts and programmers spend far more time maintaining programs than they do writing them. Maintenance accounts for 50-80 percent of total system development

Maintenance is expensive. One way to reduce the maintenance costs are through maintenance management and software modification audits**.**

- Maintenance is not as rewarding as exciting as developing systems. It is perceived as requiring neither skill not experience.
- Users are not fully cognizant of the maintenance problem or its high cost.
- Few tools and techniques are available for maintenance.
- A good test plan is lacking.
- Standards, procedures, and guidelines are poorly defined and enforced.
- Programs are often maintained without care for structure and documentation.
- There are minimal standards for maintenance.
- Programmers expect that they will not be in their current commitment by time their programs go into the maintenance cycle.

**Corrective Maintenance**

It means repairing, processing or performance failure or making changes because of previously uncovered problems or false assumptions. Task performed to identify, isolate, and rectify a fault so that the failed equipment, machine, or system can be restored to an operational condition within the tolerances or limits established for in-service operations.

Corrective maintenance can be subdivided into "immediate corrective maintenance" (in which work starts immediately after a failure) and "deferred corrective maintenance" (in which work is delayed in conformance to a given set of maintenance rules).

**Perfective Maintenance**

It means changes made to a system to add new features or to improve performance. Preventive maintenance is predetermined work performed to a schedule with the aim of preventing the wear and tear or sudden failure of equipment components. process or control equipment failure can have adverse results in both human and economic terms. In addition to down time and the costs involved to repair and/or replace equipment parts or components, there is the risk of injury to operators, and of acute exposures to chemical and/or physical agents.

Time-based or run-based Periodically inspecting, servicing, cleaning, or replacing parts to prevent sudden failure .On-line monitoring of equipment in order to use important/expensive parts to the limit of their serviceable life. Preventive maintenance involves changes made to a system to reduce the chance of future system failure.

An example of preventive maintenance might be to increase the number of records that a system can process far beyond what is currently needed or to generalize how a system sends report information to a printer so that so that the system can adapt to changes in printer technology.

**Preventive Maintenance**

Changes made to a system to avoid possible future problems Perfective maintenance involves making enhancements to improve processing performance, interface usability, or to add desired, but not necessarily required, system features. The objective of perfective maintenance is to improve response time, system efficiency, reliability, or maintainability.

During system operation, changes in user activity or data pattern can cause a decline in efficiency, and perfective maintenance might be needed to restore performance. Usually, the perfective maintenance work is initiated by the IT department, while the maintenance working.

# 5. CONCLUSION

An **“ACCIDENT RECORD MANAGEMENT SYSTEM”** is a crucial tool for organizations to maintain records of accidents that occur in the workplace. The system allows for the prompt reporting of accidents and helps to identify trends and patterns, which can be used to improve safety and prevent future accidents.

Through the use of an accident record management system, organizations can reduce the risk of workplace accidents and improve safety for employees. It also enables them to comply with legal requirements and reduces the potential for litigation in case of accidents.

In conclusion, implementing an accident record management system is essential for any organization that prioritizes safety in the workplace. It is an effective tool for managing accidents and preventing future incidents. By maintaining accurate records of accidents, organizations can analyze trends and patterns, identify areas for improvement, and implement measures to enhance safety and protect their employees.

# BIBLIOGRAPHY

**Book Reference**:


- Bloch, Joshua. Effective Java: Programming Language Guide. Addison-Wesley, 2017.
- Eckel, Bruce. Thinking in Java. Prentice Hall, 2006.
- Freeman, Eric, and Elisabeth Robson. Head First Java. O'Reilly Media, 2005.
- Horstmann, Cay S. Core Java Volume I--Fundamentals. Prentice Hall, 2018.
- Lea, Doug. Concurrent Programming in Java: Design Principles and Patterns. Addison-Wesley, 2000.


**Websites:**

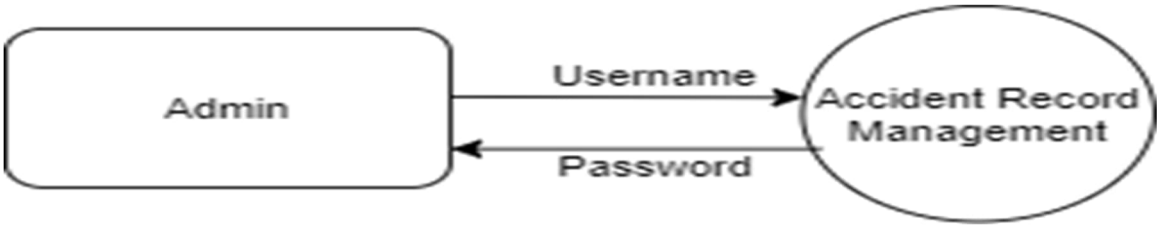

- Baeldung. "Java Tutorials and Articles." Baeldung, 2023, https://www.baeldung.com/java-tutorials.
- GeeksforGeeks. "Java Programming Language." GeeksforGeeks, 2023, https://www.geeksforgeeks.org/java/.
- Stack Overflow. "Questions tagged [java]." Stack Overflow, https://stackoverflow.com/questions/tagged/java.
- Tutorials Point. "Java Tutorial." Tutorials Point, 2023, https://www.tutorialspoint.com/java/index.htm.
- Vogella. "Java Tutorials." Vogella, 2023, https://www.vogella.com/tutorials/java.html.

# APPENDICES

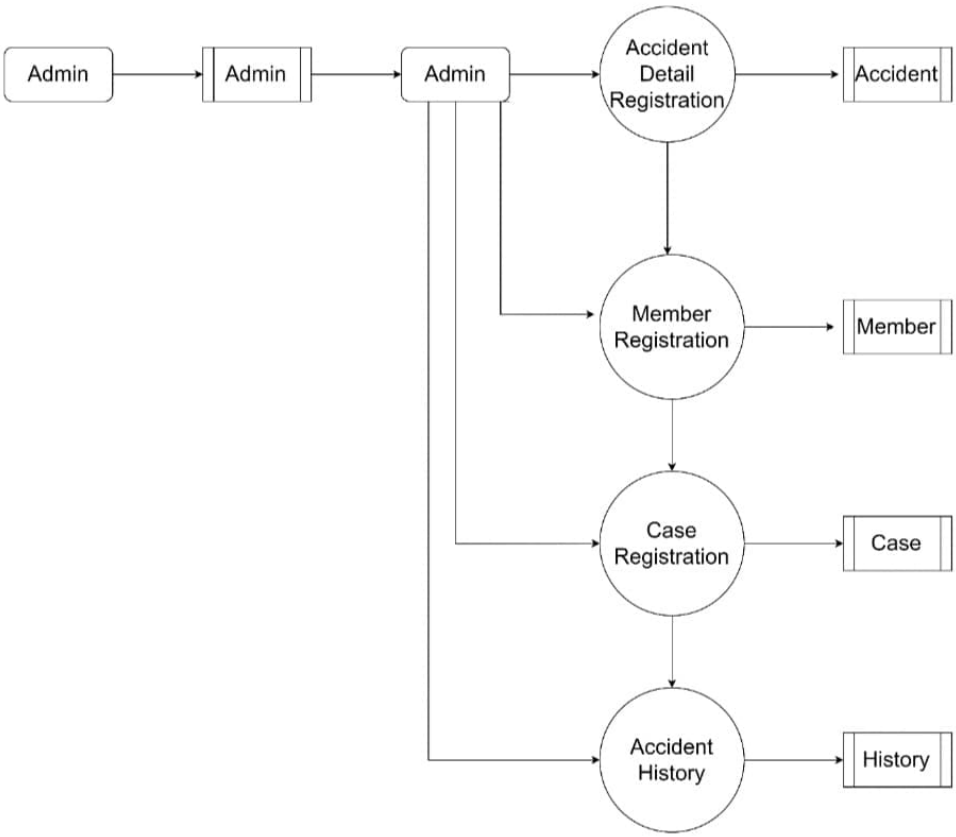
## A. DATA FLOW DIAGRAM

A data-flow diagram (DFD)is a way of representing a flow of a data of a process or system. The DFD also provides information about the outputs and inputs of each entity and process itself. A data-flow diagram is a part of structured-analysis modeling tools.

### LEVEL 0:




### LEVEL 1:



## B. TABLE STRUCTURE

The table needed for each module was designed and the specification of each and every column was given based on the records and details collected during record specification of the system study.

### TABLE NAME: ADMIN REGISTRATION

| FIELD | DATA TYPE | SIZE | CONSTRAINT |
|---|---|---|---|
| Id | Int | 10 | Primary key |
| Username | Varchar | 20 | Not null |
| Password | Varchar | 20 | Not null |

### TABLE NAME: MEMBER REGISTRATION

| FIELD | DATA TYPE | SIZE | CONSTRAINT |
|---|---|---|---|
| Member id | Int | 10 | Primary key |
| First name | Varchar | 20 | Not null |
| Last name | Varchar | 20 | Not null |
| Mobile | Varchar | 20 | Not null |
| Aadhar | Varchar | 15 | Not null |
| Address | Varchar | 15 | Not null |

**TABLE NAME: CASE REGISTRATION**

| FIELD NAME | FIELD TYPE | SIZE | CONSTRAINT |
|---|---|---|---|
| Case id | Int | 10 | Primary key |
| Case name | Varchar | 15 | Not null |
| Description | Varchar | 50 | Not null |

**TABLE NAME: ACCIDENT DETAILS REGISTRATION**

| FIELD | DATA TYPE | SIZE | CONSTRAINT |
|---|---|---|---|
| **Accident id** | **Int** | **10** | **Primary key** |
| **Member id** | **Int** | **10** | **Foreign key** |
| **Case id** | **Int** | **10** | **Foreign key** |
| **Person name** | **Varchar** | **20** | **Not null** |
| **Vehicle no** | **Varchar** | **20** | **Not null** |
| **Mobile** | **Varchar** | **10** | **Not null** |
| **Address** | **Varchar** | **30** | **Not null** |
| **Description** | **Varchar** | **50** | **Not null** |
| **Date** | **Varchar** | **10** | **Not null** |

**TABLE NAME: HISTORY DETAILS**

| FIELD | DATA TYPE | SIZE | CONSTRAINT |
|---|---|---|---|
| **History Id** | **Int** | **10** | **Primary key** |
| **Accident id** | **Int** | **20** | **Foreign key** |
| **Member id** | **Int** | **20** | **Foreign key** |

# C.SAMPLE CODING

```
package com.example.demo.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.demo.response.ViewAllocateHistoryResponse;
import com.example.demo.response.ViewCaseResponse;
import com.example.demo.response.ViewCriminalResponse;
import com.example.demo.response.ViewPoliceResponse;
import cm.example.demo.service. A pi Service;

@RestController
@RequestMapping(value = { "/api" })
public class ApiController {

	@Autowired
	ApiService service;

	@GetMapping("/login/{username}/{password}")
	public Integer login(@PathVariable String username, @PathVariable String password) {
		return service.login(username, password);
	}

	@PostMapping("/add_police/{firstname}/{lastname}/{department}/{profession}" +
"/{username}/{password}/{address}")
	public String add_police(@PathVariable String firstname, @PathVariable String lastname,
			@PathVariable String department, @PathVariable String profession,
@PathVariable String username,
			@PathVariable String password, @PathVariable String address) {
		service.add_police(firstname, lastname, department, profession, username, password,
address);
```

```
		return "Police Saved Sucessfully";
	}

	@GetMapping("/view_police")
	public ResponseEntity<List<ViewPoliceResponse>> view_police()
{

		return ResponseEntity.ok().body(service.view_police());
	}

	@PostMapping("/add_criminal/{firstname}/{lastname}/{mobile}/{aadhar}" + "/{address}")
	public String add_criminal(@PathVariable String firstname, @PathVariable String lastname,
			@PathVariable String mobile, @PathVariable String aadhar, @PathVariable
String address)
 {
		service.add_criminal(firstname, lastname, mobile, aadhar, address);
		return "Criminal Saved Sucessfully";
	}

	@GetMapping("/view_criminal")
	public ResponseEntity<List<ViewCriminalResponse>> view_criminal()
 {
		return ResponseEntity.ok().body(service.view_criminal());
	}

	@PostMapping("/add_case/{casename}/{description}")
	public String add_case(@PathVariable String casename, @PathVariable String description)
 {
		service.add_case(casename, description);
		return "Case name Saved Sucessfully";
	}

	@GetMapping("/view_case")
	public ResponseEntity<List<ViewCaseResponse>> view_case()
{
		return ResponseEntity.ok().body(service.view_case());
	}
	@GetMapping("/get_criminal/{aadhar}")
	public ResponseEntity<List<ViewCriminalResponse>> get_criminal(@PathVariable String
aadhar)
```

```
{
		return ResponseEntity.ok().body(service.get_criminal(aadhar));
	}

	@PostMapping("/allocate_case/{criminal_id}/{case_id}/{personname}/{vehicleno}"
			+ "/{mobile}/{address}/{description}")
	public String allocate_case(@PathVariable Integer criminal_id, @PathVariable Integer case_id,
			@PathVariable String personname, @PathVariable String vehicleno, @PathVariable String mobile,
			@PathVariable String address, @PathVariable String description)
 {
		service.allocate_case(criminal_id, case_id, personname, vehicleno, mobile, address, description);
		return "Case Allocated Sucessfully";
	}

	@GetMapping("/allocate_history/{aadhar}")
	public ResponseEntity<List<ViewAllocateHistoryResponse>> allocate_history(@PathVariable String aadhar)
{
		return ResponseEntity.ok().body(service.allocate_history(aadhar));
	}
}

package com.example.demo.dao;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.query.NativeQuery;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import com.example.demo.response.ViewCriminalResponse;
```

```
@Repository
public class ApiDao
{
	@Autowired
	SessionFactory sf;

	public void add_police(String firstname, String lastname, String department, String profession,
String username,
				String password, String address)
 {
		// TODO Auto-generated method stub
		Session session = sf.getCurrentSession();
		String sql = "INSERT INTO `police` (`id`, `firstname`, `lastname`, `department`,
`profession`, `username`, `password`, `fulladdress`) "
				+ "VALUES (NULL, '" + firstname + "', '" + lastname + "', '" +
department + "', '" + profession + "', '"
				+ username + "', '" + password + "', '" + address + "');";
		session.createSQLQuery(sql).executeUpdate();
	}

	public List<Object[]> view_police()
{
		// TODO Auto-generated method stub
		Session session = sf.getCurrentSession();
		String sql = "Select * from police";
		NativeQuery nq = session.createNativeQuery(sql);
		return nq.list();
	}

	public void add_criminal(String firstname, String lastname, String mobile, String aadhar, String
address) {
		// TODO Auto-generated method stub
		Session session = sf.getCurrentSession();
		String sql = "INSERT INTO `criminal` (`id`, `firstname`, `lastname`, `mobile`,
`aadhar`, `address`) VALUES "
				+ "(NULL, '" + firstname + "', '" + lastname + "', '" + mobile + "', '" +
aadhar + "', '" + address+ "');";
		session.createSQLQuery(sql).executeUpdate();
	}
```

```
	public List<Object[]> view_criminal()
{
		// TODO Auto-generated method stub
		Session session = sf.getCurrentSession();
		String sql = "Select * from criminal";
		NativeQuery nq = session.createNativeQuery(sql);
		return nq.list();
	}

	public void add_case(String casename, String description)
{
		// TODO Auto-generated method stub
		Session session = sf.getCurrentSession();
		String sql = "INSERT INTO `caset` (`id`, `casename`, `description`) VALUES " +
"(NULL, '" + casename + "', '"+ description + "');";
		session.createSQLQuery(sql).executeUpdate();
	}

	public List<Object[]> view_case()
{
		// TODO Auto-generated method stub
		Session session = sf.getCurrentSession();
		String sql = "Select * from caset";
		NativeQuery nq = session.createNativeQuery(sql);
		return nq.list();
	}

	public List<Object[]> get_criminal(String aadhar)
{
		// TODO Auto-generated method stub
		Session session = sf.getCurrentSession();
		String sql = "Select * from criminal where aadhar='" + aadhar + "'";
		NativeQuery nq = session.createNativeQuery(sql);
		return nq.list();
	}

	public void allocate_case(Integer criminal_id, Integer case_id, String personname, String
vehicleno, String mobile,
			String address, String description)
{
```

```
		// TODO Auto-generated method stub
		Session session = sf.getCurrentSession();
		String sql = "INSERT INTO `allocate` (`id`, `criminalid`, `caseid`, `personname`, `vehicleno`, `mobile`, `address`, `description`) VALUES "+ "(NULL, '" + criminal_id + "', '" + case_id + "', '" + personname + "', '" + vehicleno + "', '+ mobile + "', '" + address + "', '" + description + "');";
		session.createSQLQuery(sql).executeUpdate();
	}

	public List<Object[]> allocate_history(String aadhar)
{
		// TODO Auto-generated method stub

		Session session = sf.getCurrentSession();

Stringsql="SELECTallocate.personname,allocate.vehicleno,allocate.mobile,allocate.description,allocate.datee,caset.casename FROM `allocate` JOIN criminal ON(criminal.id=allocate.criminalid) \r\n"+ "
	joincaset ON(caset.id=allocate.caseid)\r\n" + "where criminal.aadhar='" + aadhar + "'";
		NativeQuery nq = session.createNativeQuery(sql);
		return nq.list();
	}

	public Integer login(String username, String password)
 {
		// TODO Auto-generated method stub
		System.out.println("Login");
		Session session = sf.getCurrentSession();
		String sql = "Select * from admin where username='" + username + "' and password='" + password + "'";
		NativeQuery nq = session.createNativeQuery(sql);
		if (nq.list().size() == 0) {
			String sql1 = "Select * from police where username='" + username + "' and password='" + password + "'";
			NativeQuery nq1 = session.createNativeQuery(sql1);
			List<Object[]> a = nq1.list();

			if (nq1.list().size() == 0)
 {
				return -2;
			}
 Else
```

```
{
                                        return (Integer) a.get(0)[0];
                                }
                        }
Else
{
                                return -1;
                        }

        }
}
```

```
package com.example.demo.configuration;

import java.util.Properties;

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.jdbc.datasource.DriverManagerDataSource;
import org.springframework.orm.hibernate5.HibernateTransactionManager;
import org.springframework.orm.hibernate5.LocalSessionFactoryBean;
import org.springframework.transaction.annotation.EnableTransactionManagement;

@Configuration
@EnableTransactionManagement
public class HibernateConfiguration
{

   @Value("${db.driver}")
   private String DB_DRIVER;

   @Value("${db.password}")
   private String DB_PASSWORD;

   @Value("${db.url}")
   private String DB_URL;
```

```
    @Value("${db.username}")
    private String DB_USERNAME;

    @Value("${hibernate.dialect}")
    private String HIBERNATE_DIALECT;

    @Value("${hibernate.show_sql}")
    private String HIBERNATE_SHOW_SQL;

    @Value("${hibernate.hbm2ddl.auto}")
    private String HIBERNATE_HBM2DDL_AUTO;

    @Value("${entitymanager.packagesToScan}")
    private String ENTITYMANAGER_PACKAGES_TO_SCAN;

    @Bean
    public LocalSessionFactoryBean sessionFactory()
  {
        LocalSessionFactoryBean sessionFactory = new LocalSessionFactoryBean();
        sessionFactory.setDataSource(dataSource());
        sessionFactory.setPackagesToScan(ENTITYMANAGER_PACKAGES_TO_SCAN);
        Properties hibernateProperties = new Properties();
        hibernateProperties.put("hibernate.dialect", HIBERNATE_DIALECT);
        hibernateProperties.put("hibernate.show_sql", HIBERNATE_SHOW_SQL);
        hibernateProperties.put("hibernate.hbm2ddl.auto", HIBERNATE_HBM2DDL_AUTO);
        sessionFactory.setHibernateProperties(hibernateProperties);
        return sessionFactory;
    }

    @Bean
    public DataSource dataSource()
  {
        DriverManagerDataSource dataSource = new DriverManagerDataSource();
        dataSource.setDriverClassName(DB_DRIVER);
        dataSource.setUrl(DB_URL);
        dataSource.setUsername(DB_USERNAME);
        dataSource.setPassword(DB_PASSWORD);
        return dataSource;
    }
```

```
    @Bean
    public HibernateTransactionManager transactionManager()
{
        HibernateTransactionManager txManager = new HibernateTransactionManager();
        txManager.setSessionFactory(sessionFactory().getObject());
        return txManager;
    }
}
```

```
package com.example.demo.configuration;

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;

@Configuration
@EnableWebMvc
public class WebConfig extends WebMvcConfigurerAdapter
 {

    @Override
    public void addCorsMappings(CorsRegistry registry)
{
        registry.addMapping("/**");
    }
}
```

```
package com.example.demo.response;

import java.sql.Timestamp;

import com.fasterxml.jackson.annotation.JsonFormat;

public class ViewAllocateHistoryResponse
{

            private String personname;
            private String vehicle;
            private String mobile;
```

```
	@JsonFormat(pattern = "dd-MM-yyyy hh:mm a")
	private Timestamp date;
	private String description;
	private String casename;
	public String getPersonname()
 {
		return personname;
	}
	public void setPersonname(String personname)
 {
		this.personname = personname;
	}
	public String getVehicle()
 {
		return vehicle;
	}
	public void setVehicle(String vehicle)
 {
		this.vehicle = vehicle;
	}
	public String getMobile()
{
		return mobile;
	}
	public void setMobile(String mobile)
 {
		this.mobile = mobile;
	}
	public Timestamp getDate()
{
		return date;
	}
	public void setDate(Timestamp date)
 {
		this.date = date;
	}
	public String getDescription()
{
		return description;
	}
```

```
	public void setDescription(String description)
{
		this.description = description;
	}
	public String getCasename()
 {
		return casename;
	}
	public void setCasename(String casename)
 {
		this.casename = casename;
	}

}

package com.example.demo.response;

public class ViewCaseResponse
 {
private Integer id;
private String casename;
private String description;
public Integer getId()
{
	return id;
}
public void setId(Integer id)
 {
	this.id = id;
}
public String getCasename()
{
	return casename;
}
public void setCasename(String casename)
{
	this.casename = casename;
}
public String getDescription()
{
```

```
        return description;
}
public void setDescription(String description)
{
        this.description = description;
}



}

package com.example.demo.response;

public class ViewPoliceResponse
{
        private String firstname;
        private String lastname;
        private String department;
        private String profession;
        public String getFirstname()
{
                return firstname;
        }
        public void setFirstname(String firstname)
{
                this.firstname = firstname;
        }
        public String getLastname()
 {
                return lastname;
        }
        public void setLastname(String lastname)
{
                this.lastname = lastname;
        }
        public String getDepartment()
{
                return department;
        }
        public void setDepartment(String department)
{
```

```
		this.department = department;
	}
	public String getProfession()
{
		return profession;
	}
	public void setProfession(String profession)
{
		this.profession = profession;
	}

}
package com.example.demo.service;

import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.List;

import javax.transaction.Transactional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.example.demo.dao.ApiDao;
import com.example.demo.response.ViewAllocateHistoryResponse;
import com.example.demo.response.ViewCaseResponse;
import com.example.demo.response.ViewCriminalResponse;
import com.example.demo.response.ViewPoliceResponse;

@Service
@Transactional
public class ApiService
{

	@Autowired
	ApiDao dao;

	public void add_police(String firstname, String lastname, String department, String profession,
String username,String password, String address)
 {
```

```
		// TODO Auto-generated method stub
		dao.add_police(firstname,lastname,department,profession,username,password
				,address);
	}


	public List<ViewPoliceResponse> view_police()
{
	// TODO Auto-generated method stub
	List<Object[]> result =dao.view_police();

	List<ViewPoliceResponse> response = new ArrayList<ViewPoliceResponse>();
	for(int i=0;i<result.size();i++)
{
		Object[] row = result.get(i);
		ViewPoliceResponse obj = new ViewPoliceResponse();

		obj.setFirstname((String) row[1]);
		obj.setLastname((String) row[2]);
		obj.setDepartment((String) row[3]);
		obj.setProfession((String) row[4]);

		response.add(obj);
	}
	return response;
}


	public void add_criminal(String firstname, String lastname, String mobile, String aadhar, String
address) {
		// TODO Auto-generated method stub
		dao.add_criminal(firstname,lastname,mobile,aadhar,address);
	}


	public List<ViewCriminalResponse> view_criminal()
{
		// TODO Auto-generated method stub
		List<Object[]> result =dao.view_criminal();
```

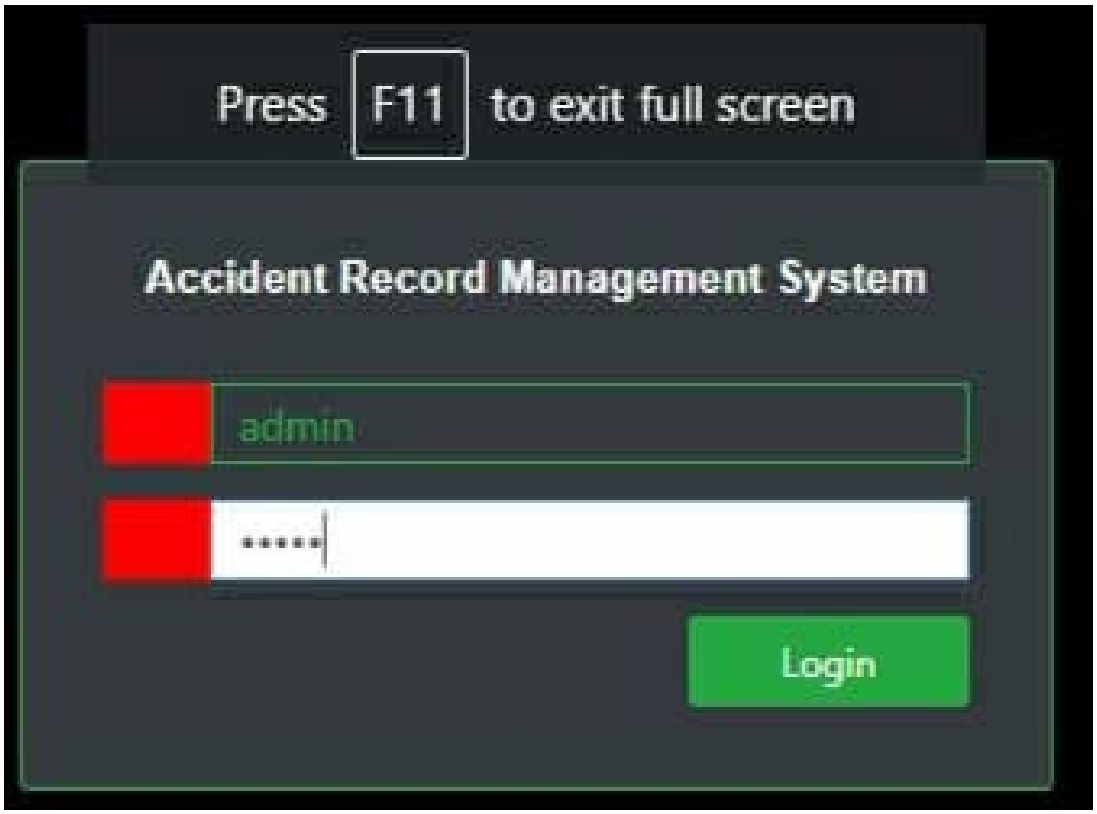
```
		List<ViewCriminalResponse> response = new ArrayList<ViewCriminalResponse>();
		for(int i=0;i<result.size();i++)
{
			Object[] row = result.get(i);
			ViewCriminalResponse obj = new ViewCriminalResponse();
			obj.setId((Integer)row[0]);
			obj.setFirstname((String) row[1]);
			obj.setLastname((String) row[2]);
			obj.setMobile((String) row[3]);
			obj.setAadhar((String) row[4]);
			obj.setAddress((String) row[5]);

			response.add(obj);
		}
		return response;
	}


	public void add_case(String casename, String description)
{
		// TODO Auto-generated method stub
		dao.add_case(casename,description);
	}


	public List<ViewCaseResponse> view_case()
{
		// TODO Auto-generated method stub
		List<Object[]> result =dao.view_case();

		List<ViewCaseResponse> response = new ArrayList<ViewCaseResponse>();
		for(int i=0;i<result.size();i++) {
			Object[] row = result.get(i);
			ViewCaseResponse obj = new ViewCaseResponse();
			obj.setId((Integer)row[0]);
			obj.setCasename((String) row[1]);
			obj.setDescription((String) row[2]);


			response.add(obj);
```

```
		}
		return response;
	}


	public List<ViewCriminalResponse> get_criminal(String aadhar)
{
		// TODO Auto-generated method stub
List<Object[]> result =dao.get_criminal(aadhar);

		List<ViewCriminalResponse> response = new ArrayList<ViewCriminalResponse>();
		for(int i=0;i<result.size();i++)
{
			Object[] row = result.get(i);
			ViewCriminalResponse obj = new ViewCriminalResponse();
			obj.setId((Integer)row[0]);
			obj.setFirstname((String) row[1]);
			obj.setLastname((String) row[2]);
			obj.setMobile((String) row[3]);
			obj.setAadhar((String) row[4]);
			obj.setAddress((String) row[5]);

			response.add(obj);
		}
		return response;
	}


	public void allocate_case(Integer criminal_id, Integer case_id, String personname, String vehicleno, String mobile,
			String address, String description)
{
		// TODO Auto-generated method stub

	dao.allocate_case(criminal_id,case_id,personname,vehicleno,mobile,address,description);
	}


	public List<ViewAllocateHistoryResponse> allocate_history(String aadhar)
{
```

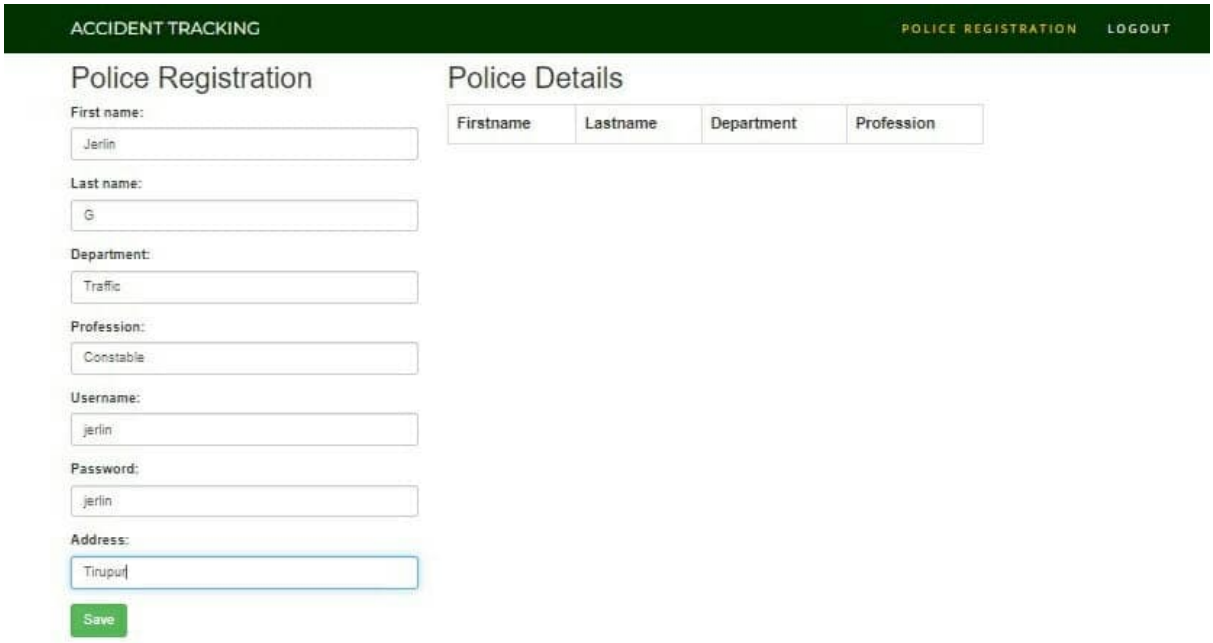
```
		// TODO Auto-generated method stub
List<Object[]> result =dao.allocate_history(aadhar);

		List<ViewAllocateHistoryResponse>response=new
ArrayList<ViewAllocateHistoryResponse>();
		for(int i=0;i<result.size();i++)
{
			Object[] row = result.get(i);
			ViewAllocateHistoryResponse obj = new ViewAllocateHistoryResponse();
			//obj.setId((Integer)row[0]);
			obj.setPersonname((String) row[0]);
			obj.setVehicle((String) row[1]);
			obj.setMobile((String) row[2]);
			obj.setDescription((String) row[3]);
			obj.setDate((Timestamp) row[4]);
			obj.setCasename((String) row[5]);

			response.add(obj);
		}
		return response;
	}


	public Integer login(String username, String password)
{
		// TODO Auto-generated method stub
		return dao.login(username,password);
	}


}
```

## D.SAMPLE INPUT & OUTPUT DESIGN

## ADMIN LOGIN PAGE



# POLICE REGISTRATION



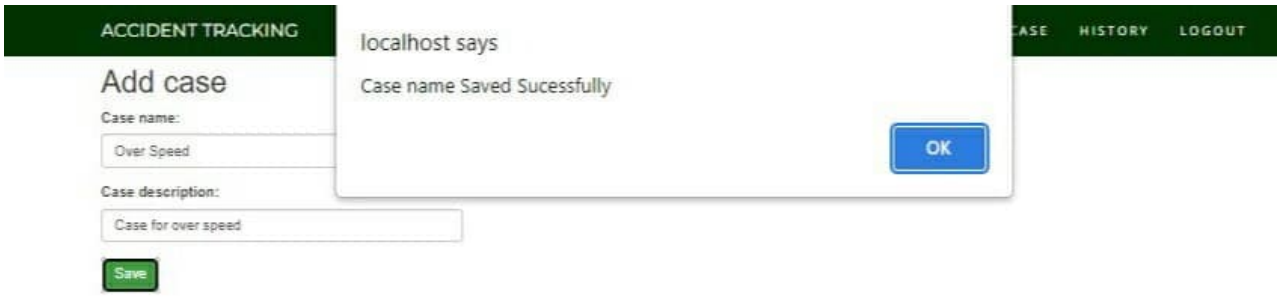
## POLICE DETAIL REGISTERED

**POLICE LOGIN CASE PAGE**

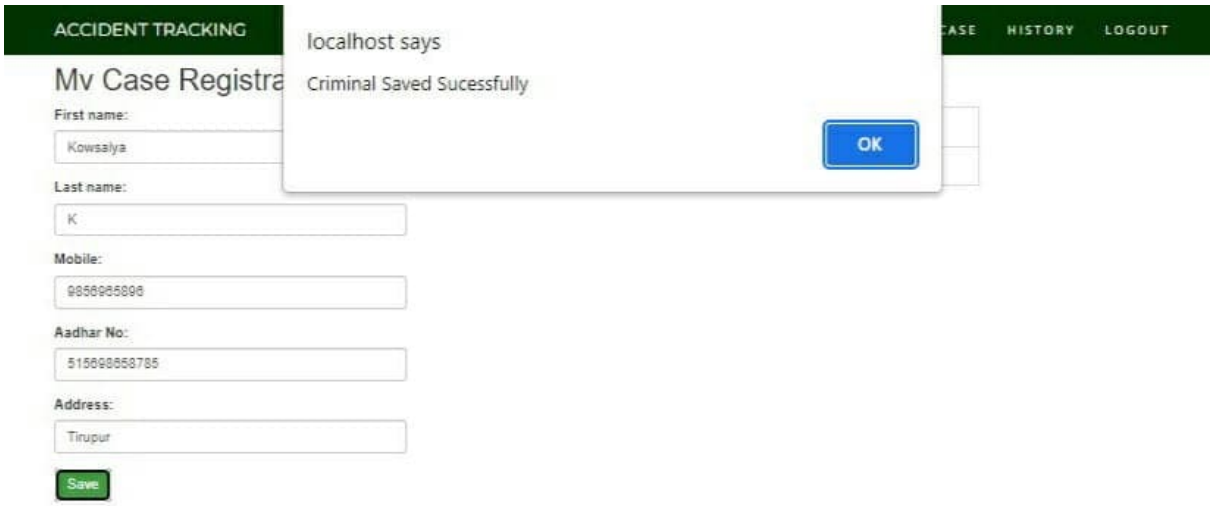
## ADD CASE


ACCIDENT TRACKING
MV CASE REGISTRATION
ADD CASE
ALLOCATE CASE
HISTORY
LOGOUT
Add case
Case name:
Over Speed
Case description:
Case for over speed
Save

## CASE REGISTERED

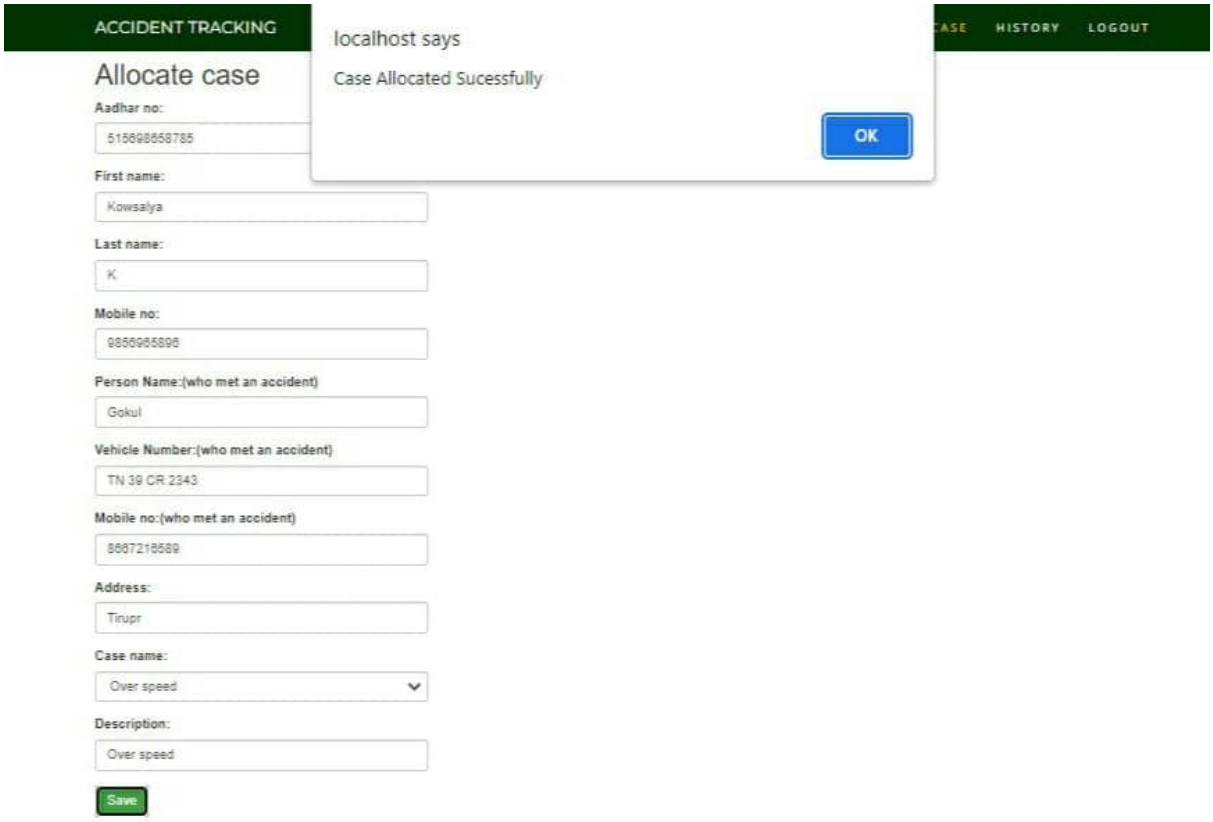


## CASE REGISTRATION DETAILS


ACCIDENT TRACKING
HISTORY
LOGOUT
Mv Case Registra
First name:
Kowsalya
Last name:
K
Mobile:
9856965896
Aadhar No:
515698558785
Address:
Tirupur
Save
localhost says
Criminal Saved Sucessfully
OK


ACCIDENT TRACKING
MV CASE REGISTRATION
ADD CASE
ALLOCATE CASE
HISTORY
LOGOUT
Mv Case Registration
First name:
Last name:
Mobile:
Aadhar No:
Address:
Save
Mv case Details
Firstname
Lastname
Mobile no
Aadhar no
Vivek
B
987654321
1234567890
Kowsalya
K
9856965896
515698658785

# CASE ALLOCATED



# ACCIDENT DETAILS HISTORY