**CHAPTER 1 - SYNOPSIS**

* 1. **Title:** ‘ Crime Record Management System’
  2. **Statement about the problem:**

The rate of crimes around us has grown on a large scale in the past years. These crimes need to be recorded in a very simplified and properly automated manner so that the traditional paper and pen format can be safely replaced.

* 1. **Why this topic :**

Paper has been used as a system of recording complaints relating to crimes since a long time. Although used extensively in the past, this method has no utility in today’s world. This may be due to the dangers relating to information theft, destruction of paper material by various causes. Also, using book-keeping as a tracking method is a really tedious procedure. The new development would hence be efficient enough to get results avoiding time consumption. Details about an already present complainant, officer, crime record, suspect, etc. can be uploaded with ease through the proposed method. The workload on officers is also reduced to a great extent by using an automated system.

* 1. **Objectives:**

1. Recording and preserving complaints relating to crimes in a fast and accessible way.
2. Photos and information about criminals can be gathered on the database, in order to be aware of dangerous criminals.
3. Information about any official, crime report, FIR, etc. can be uploaded and changed with ease.
4. Easy-to-use interface.
5. Automation of work.
6. Avoiding the hassle of preserving data using paper material and hence, reducing the possibility of data loss.
7. Secure system with distinguished authority.
8. Decrease the crime rate in the long run.
   1. **Proposed architecture:**

Database

Complainant

System

Admin

* 1. **Methodology adopted:**

The methodology for the project management will be the incremental model, where the model is designed, implemented and tested incrementally (a little more is added each time) until the product is finished. It involves both development and maintenance. The product is defined as finished when it satisfies all of its requirements. This model combines the elements of the waterfall model with the iterative philosophy of prototyping.

* 1. **Requirements:**

Hardware Requirement:

Hard Disk: Minimum 10GB HDD

CPU Type: Processor Type Intel Core 2 Duo and onwards

Software Requirement:

Windows OS- XP, vista, 7, 8

Microsoft SQL server

* 1. **Testing technology:**

Testing is the major quality control measure used during software development. Its basic function is to detect defects in the software. The goal of testing is to uncover requirement, design, and coding errors in the program.

The Techniques used are:

1. Unit Testing: In this different modules are tested individually. Each and every field on each form is tested during the coding phase to check whether all the validation rules applied are working properly and the system is storing correct data.
2. Integration Testing: In this all the modules are tested together. It is tested to ensure proper functioning of the system such as any added database by administrator.
3. System Testing: After the system is put together, system testing is performed. The whole system is checked for whether the system is giving correct output as per the given expectations.
   1. **Contribution:**

This project will provide important information regarding past, previous as well as potential crimes and criminals, thus protecting the ones who are vulnerable. It will also help to develop the overall criminal records database in order to preserve information and ultimately control and reduce the crime rate.

**CHAPTER 2 – THEORETICAL BACKGROUND**

**2.1 Existing system:**

The current system is one where the work is done manually. It is time consuming and also very costly, because it involves a lot of paper work. To manually handle such a system is a very difficult task. But now-a-days because of computerization this job has become easier.

The following are the reasons why the existing system should be computerized:

1. To increase efficiency with reduced cost.
2. To reduce the burden of paper work.
3. To save time of all the entities in the processes.

**2.1.1 Limitations of existing system:**

Maintaining data in registers or files involves many limitations such as:

1. Poor report generation: The entire information is not integrated together which can lead to improper judgments.
2. Increases the paper work: As all the data has to be maintained in files or registers, the data entry process becomes time consuming.
3. Chances of Information Leakage or Loss of Information increase: As all data is not integrated, it can lead to the loss of some important information.
4. Maintaining records is not integrated: As different registers are maintained for different records there is no integration within that data.
5. Duplication of data: There can be a possibility of the same information being stored in different registers or files.
6. Unorganized data: Searching for information from various files can prove in-efficient.
7. Error prone data**:** As data entry involves a manual process, there is a possibility of missing out on important data.
8. Accuracy: As the system is manual in nature, there are many chances of human errors.
9. Less Reliable: Use of paper for storing valuable data information is not reliable.
10. Storage Requirements: As files and registers are used, the storage space requirement is increased.

**2.2 Proposed system:**

To reduce the inconvenience that was found in the current system, it has been automated so as to provide a user friendly interface that will help data entry.

The proposed system will include the following features:

1. Creating a database for the different functions and entities containing the information present in them.
2. The users will have a distinct set of access rights which will be protected by a distinct id and a password.
3. The developed system will also print reports and maintain crime related information.
4. It will have a comfortable and user friendly interface.
5. The user interface will be designed using MICROSOFT VISUAL STUDIO and database will be implemented in SQL SERVER.
6. Also, the system is intended to take very few inputs from the user.

**2.2.1 Advantages of the proposed system:**

1. User friendly, accurate and robust system.
2. Store information about various processes, i.e. adding, editing and searching crime related information.
3. Report generation.
4. Security of data.
5. Integration of all functions into one system.
6. Removal of redundancy of data.
7. Removal of inconsistency of data.

**CHAPTER 3 - OBJECTIVES & SCOPE**

* 1. **Objectives:**

The main aim of developing this system is to build a user friendly, efficient and robust system. The CRMS i.e. Crime Record Management System will record details of all crime related information. The objectives of the project are as follows:

1. To maintain security by using a specific username and password by the users.
2. To efficiently store all the relevant data about the different entities and processes involved, which would help to organize the data successfully.
3. Automation of the processes of validating and storing information, which helps in avoiding time wastage.
4. To efficiently store the data in the system in a tabular form as it provides easy access, while avoiding redundancy.
5. To prevent leakage of important information, by removing the pen and paper format.
   1. **Scope:**

The system provides adding, searching and editing information as well as generating reports with a user-friendly Graphics User Interface (GUI)

The officer would have an account, secured with his own unique id and password, to enable him to add, edit and search the following -

1. New Complainant Record.
2. New FIR.
3. New Case Record.
4. New Suspect Record.
5. New Case Disposal Record.
6. New Non-Cognizable offence Record.

The admin would have an account secured with his own unique id and password in order to add, Edit and search the following-

1. New Officer Record.
2. New Complainant Record.
3. New FIR.
4. New Case Record.
5. New Suspect Record.
6. New Case disposal Record.
7. New Non-Cognizable offence Record.

**CHAPTER 4 - PROBLEM DEFINITION**

**4.1 Defining the problem:**

Defining a problem is the most important activity of the project. The objective is to design precisely the problem to be solved and thereby determine the scope of the new system. The first task is to review the needs that originally initiated the need for the project. The second task is to identify the expected capabilities of the new system. The objective is to define the scope of the problem in terms of the requirements of the information system that can solve it. Although at first this may not appear to define the problem, it is a necessary part of understanding the total scope of the project. The ‘Crime Record Management System’ has been designed keeping in mind the requirements of the users. The system has been designed with the aspect that it should reach the extent to which it meets the system specifications. The system design is developed with the purpose that it should provide quite ease for the client working on the other end. The system has been designed taking care that it provides proper outputs to the client so that the client can be able to make proper decisions at crucial times thus reducing the consumption of time and even large expenditure.

The following problems were found, according to a survey made:

1. The recording process in the existing system was hectic and time consuming.
2. The data which was recorded had not been validated properly, as the process seemed to be too long for the officers.
3. The officers also complained of lesser time being available for other important processes in the system.
4. Also, there has been a constant danger of loss and theft of information.

**CHAPTER 5 - SYSTEM ANALYSIS & DESIGN**

* 1. **Introduction:**

This phase is a detailed appraisal of the existing system. This appraisal includes how the system works and what it does. It also includes finding out in more detail about the problemswith the system and what the user requires from a new system or any new changes in system. The output of this phase results in the model of the new system. The model describes the system functions as well as the data and system information flow. The phase also contains the detailed set ofuser requirements, which are used to set objectives for a new system.A system analysis should be the first undertaking of a feasibility study as it clearlydefines the project outline and the users’ requirements. Once these questions have beenanswered, the people undertaking the feasibility study will have outlined the projectneeds.

* 1. **Risk analysis:**

Risk Analysis is mostly done at the start of the project. Before collecting all the information we must check whether it is feasible, safe to do the project or to check any risk that may occur during the process. These risks may occur at any point during the process and if proper measures aren’t taken it affects the project in different ways.

There are 3 types of risks:

1. Business risk
2. Project risk
3. Product risk
   1. **Proposed system study:**

It is always necessary to study and recognize the problem of the existing system, which will help in finding out the requirements for the new system. System study helps in finding different alternatives for better solutions. The project study basically deals with different operations and steps:

1. Data Gathering
2. Study of Existing System
3. Analyzing problem
4. Studying various documents
5. Feasibility study for further improvement
   1. **User requirements:**

In order to enhance the existing system the requirements concluded are as follows:

1. Storing information about FIRs, Cases, Suspects, Complaints, Case Disposal.
2. Generating repots as per requirement.
3. Storing information of the Officers and the Administrator.
4. Storing Complainant data.
5. Providing security to data as well as reduce inconsistency of data.
   1. **Feasibility study:**

A feasibility study is the study of positive possibilities of the project. It is also a measure of how beneficial or practical the development of information system would be.

The different types of feasibility are as follows:

1. Economic feasibility
2. Operational feasibility
3. Technical feasibility
   * 1. **Economic Feasibility:**

Higher level of automation often requires more funds. Financial benefits must equal or exceed the costs. To assure this, one must estimate the following:

1. If the Department has adequate cash flow for funding the development.
2. The cost to conduct a full system investigation.
3. The cost of hardware and software for the class of applications being considered.
4. The benefits in the form of reduced costs or fewer costly errors.
5. The cost if nothing changes (i.e. the proposed system is not developed) for a project to be judged feasible, it must pass all these tests.
   * 1. **Operational Feasibility:**

The operational feasibility is obtained by consulting the system user whether it satisfies the user’s requirements. A system with an easy interface will always help the user to use the system. The new system has a completely user friendly interface. It has been designed to be intuitive, so that even an inexperienced person can easily handle the system. Business functions are reengineered to achieve broader scope and higher level of automation. Manual processes too are modified.

The issues to be taken into concern are**:**

1. Level of computer competency.
2. Change of job responsibility.
3. Loss of employment due to increased automation.
4. The nature and level of user involvement in the development and implementation of system.
   * 1. **Technical Feasibility:**

It is essential to check whether the proposed system is technically feasible and to determine the technology and skills necessary to carry out the successful implementation of the project.

The necessary software required for the development of system is:

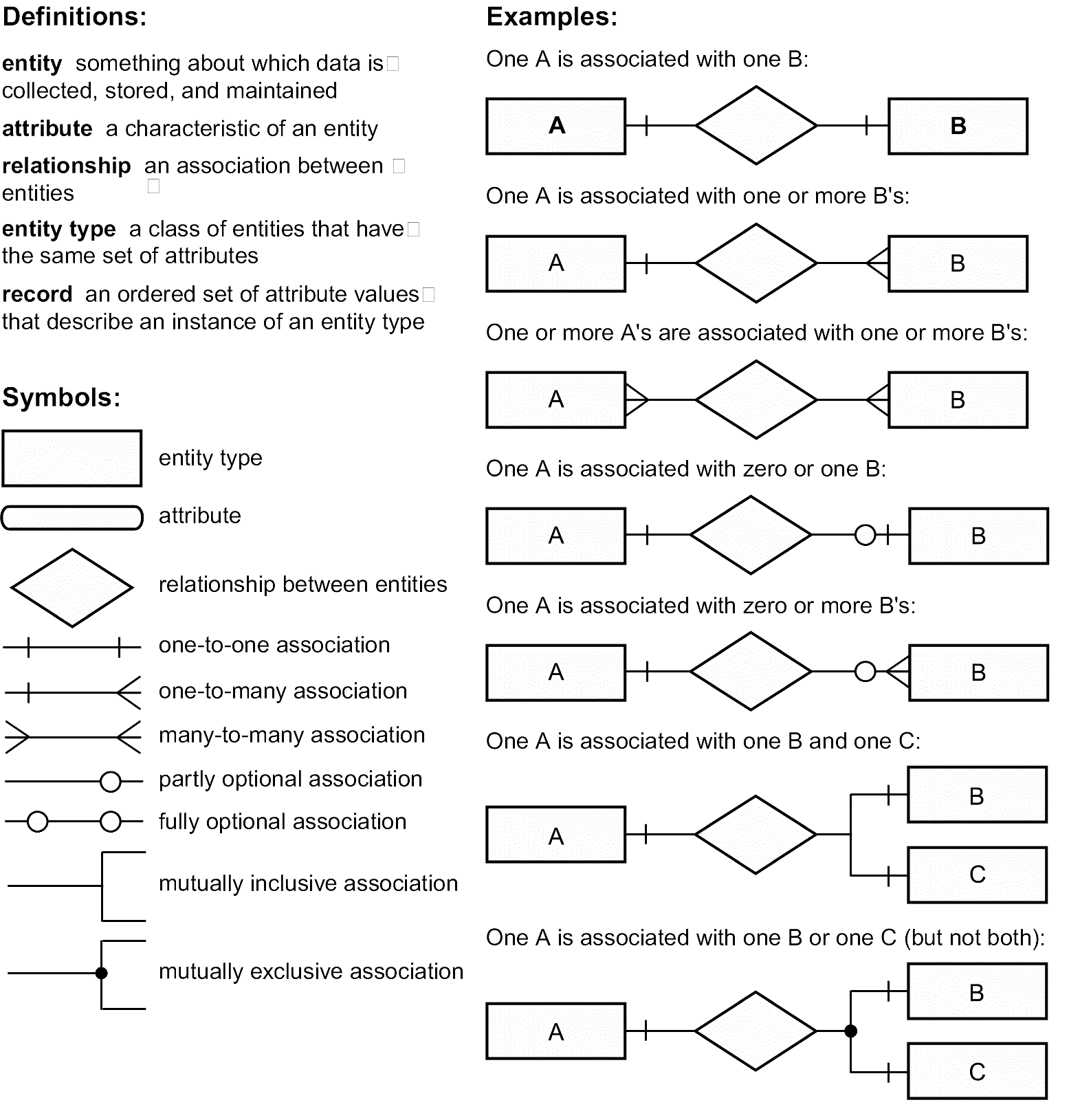
1. Microsoft Visual Studio 2010
2. SQL Server 2008
   1. **Diagrams:**

**5.6.1 ER Diagram:**

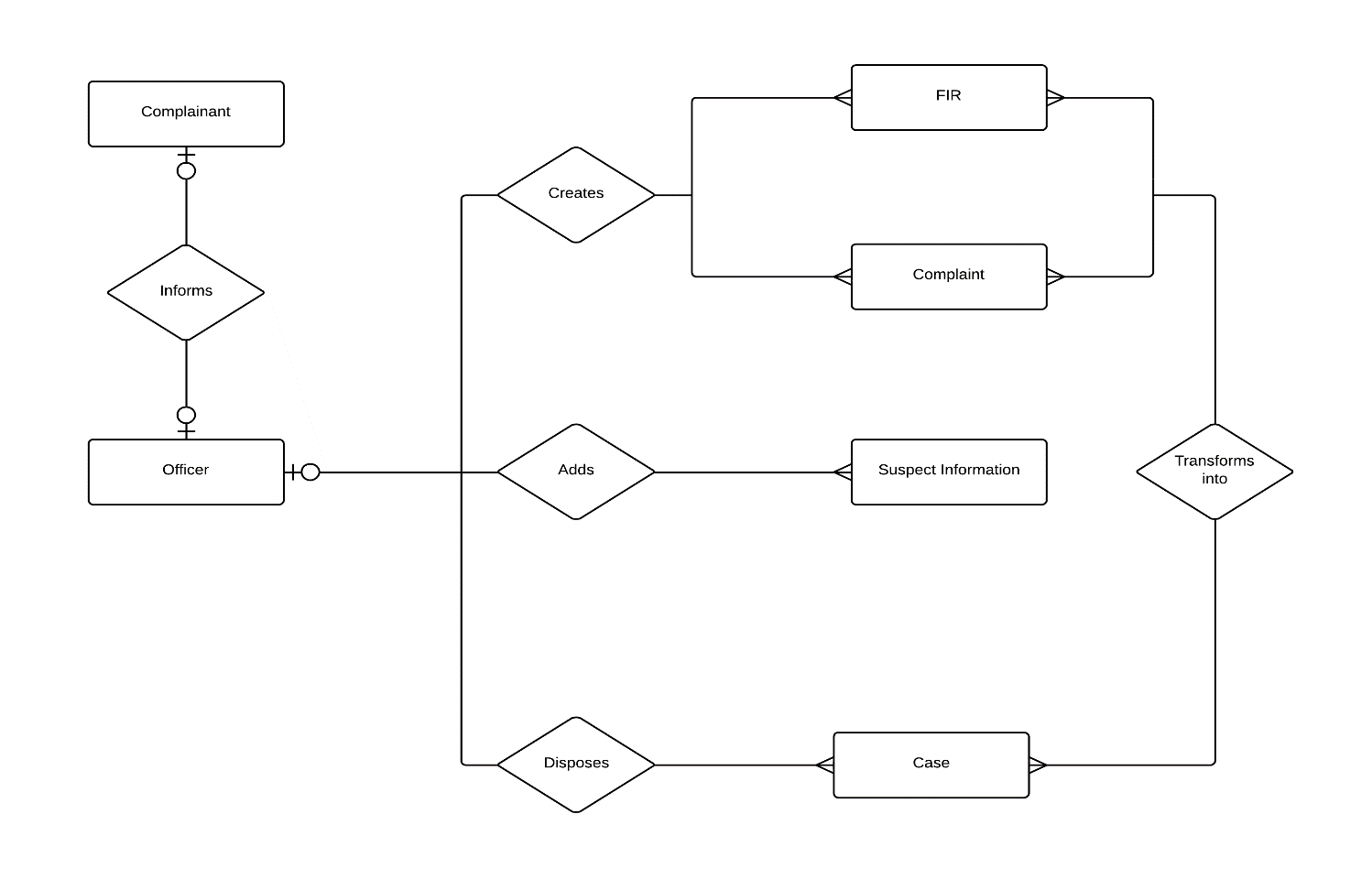
Data models are tools used in diagrams to describe the data requirements and assumptions in the system from a top down perspective. An entity relationship diagram is a graphical representation of an organization’s data storage requirements. Entity relationship diagrams are abstractions of the real world which simplify the problem to be solved while retaining its essential features. Entity relationship diagrams are used to identify the data that must be captured, stored and retrieved in order to support the business activities performed by the organization.

There are three basic elements in ER models:

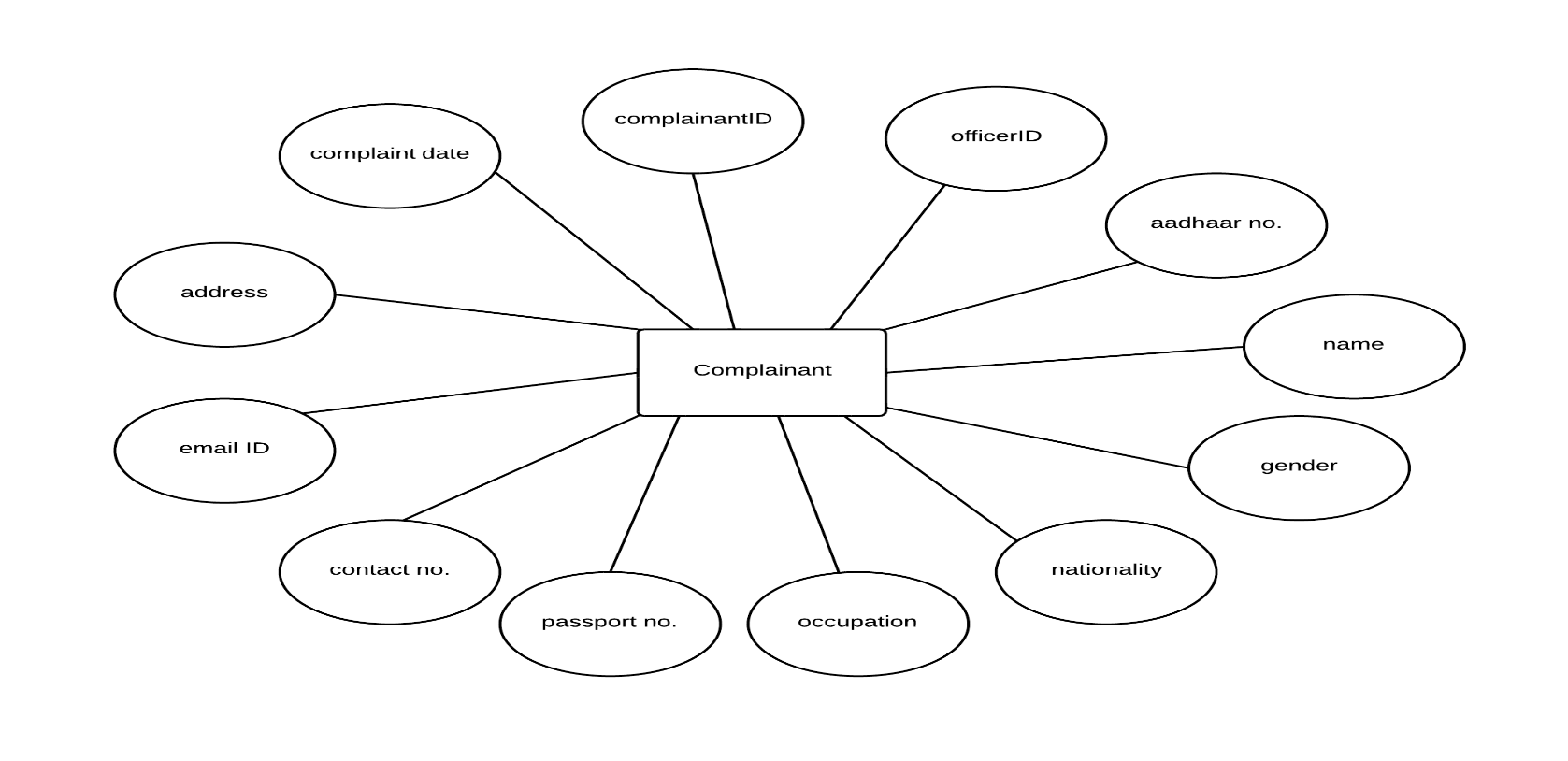
1. Entities: Entities are the “things” about which we seek information.
2. Attributes: Attributes are the data we collect about the entities.
3. Relationships: Relationships provide the structure needed to draw information from multiple entities.

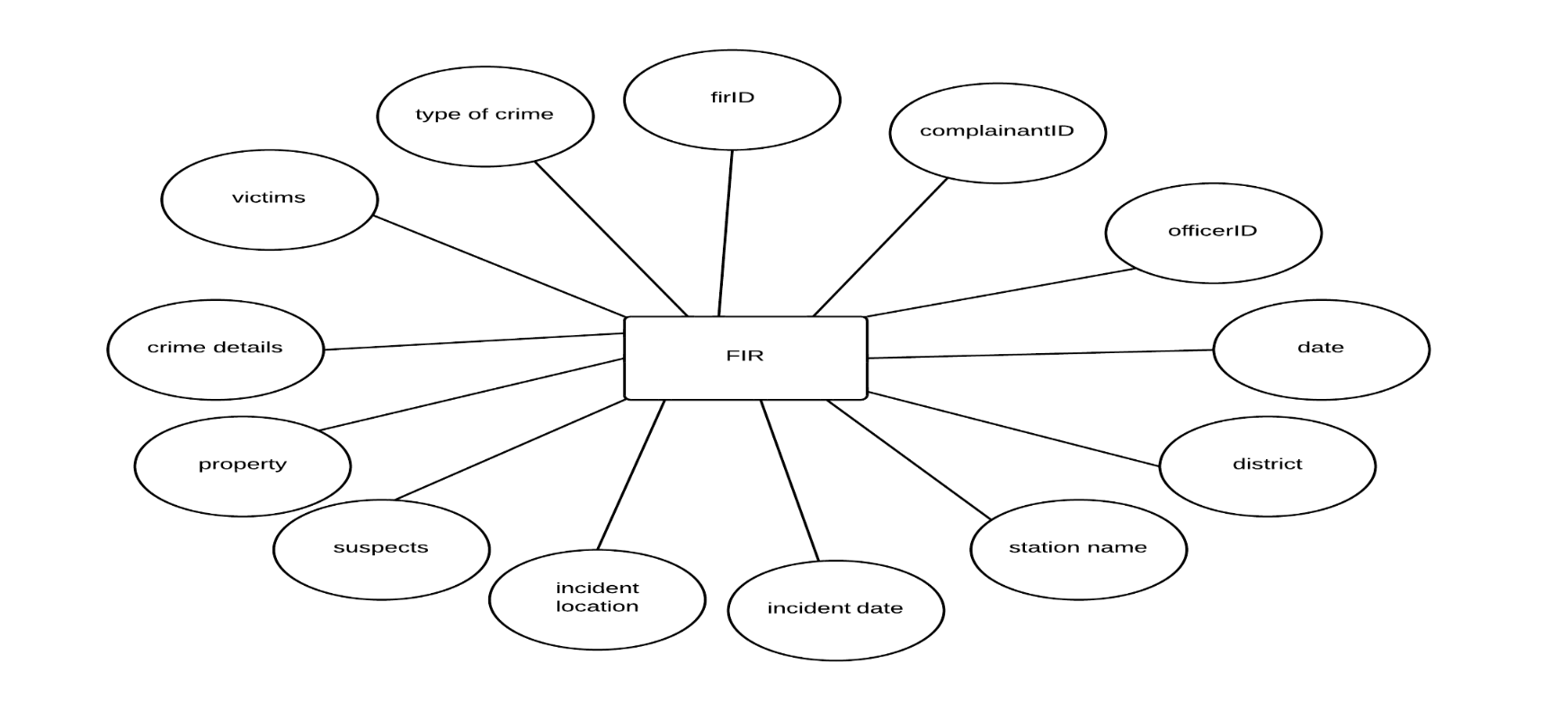
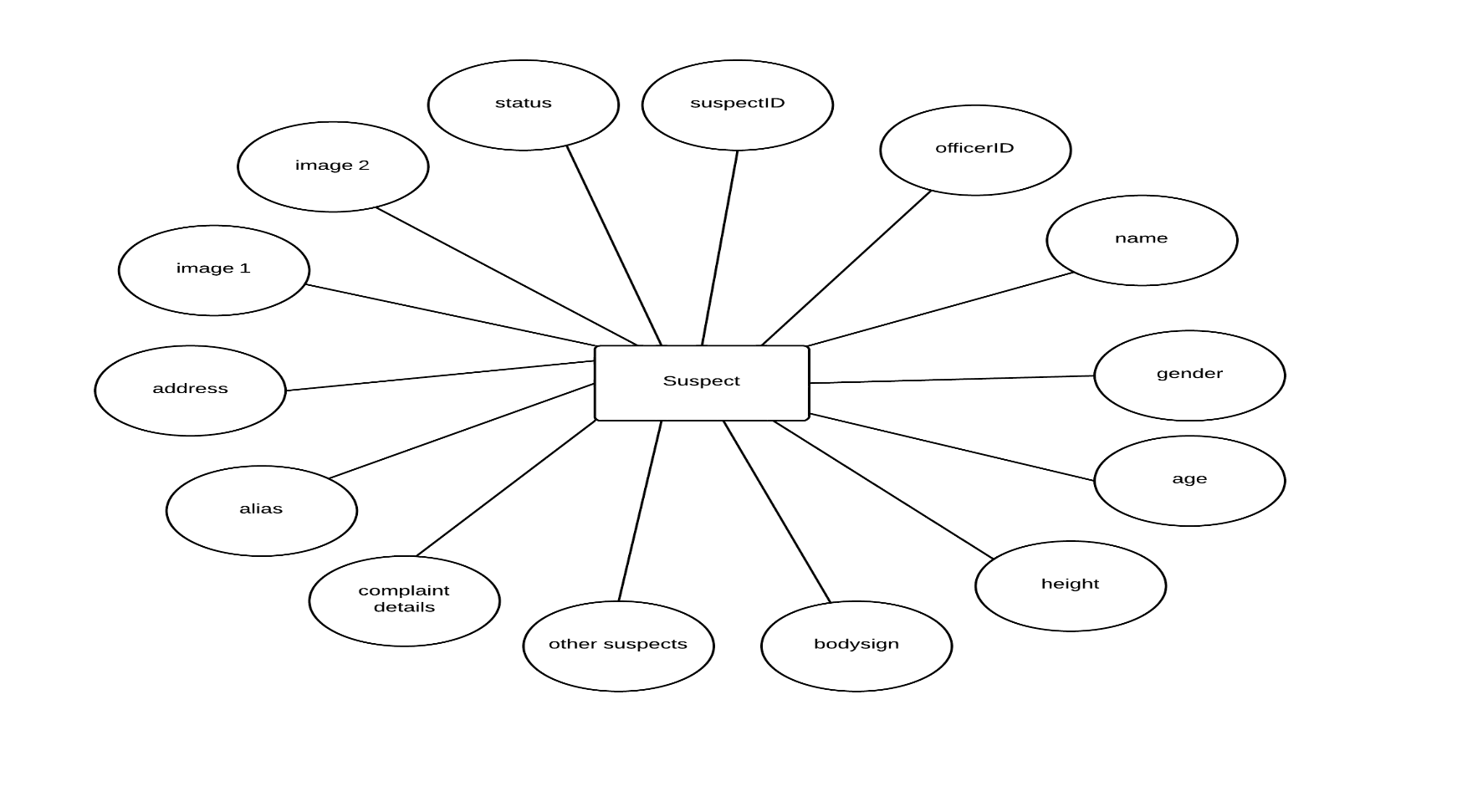
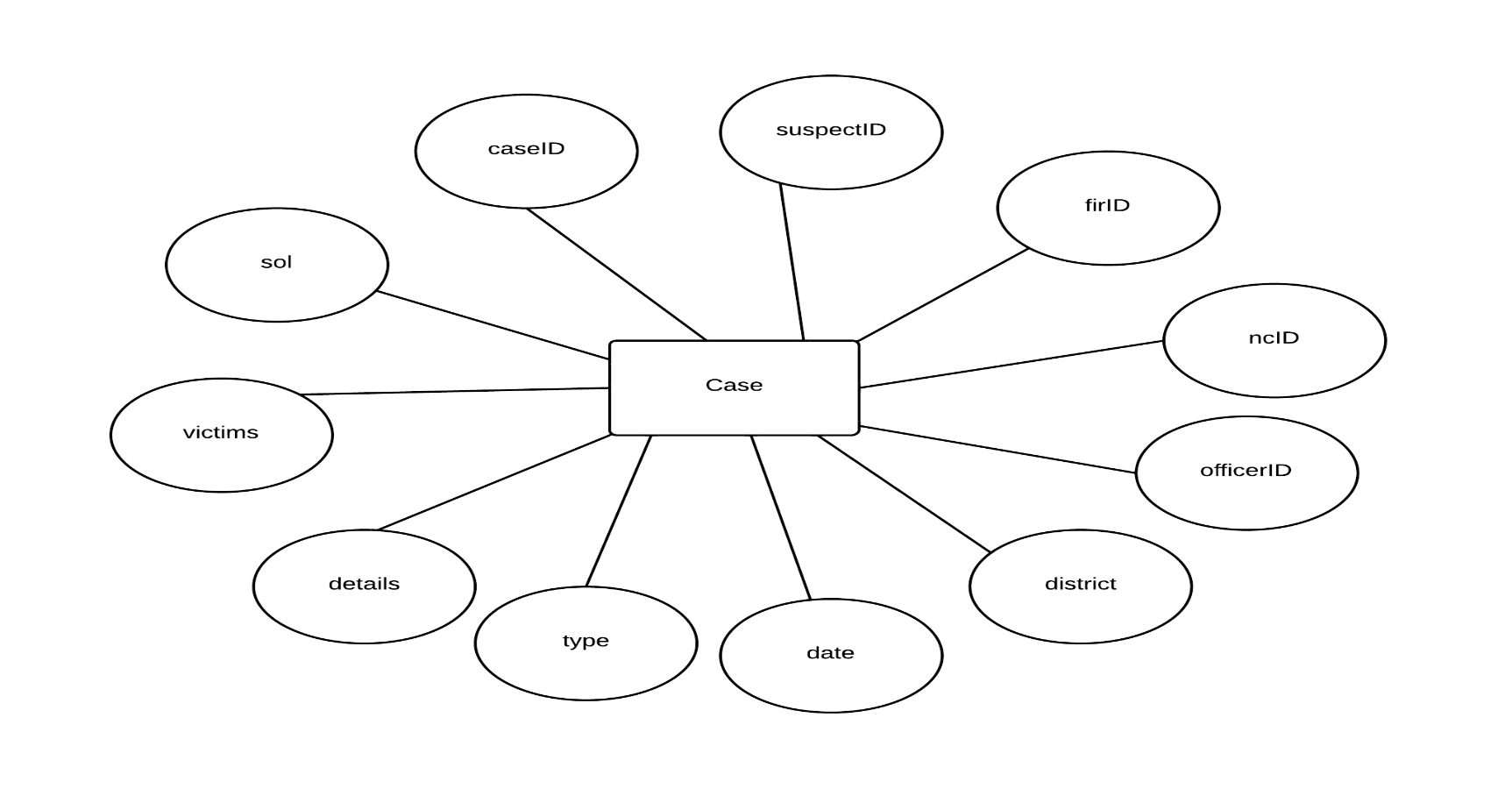
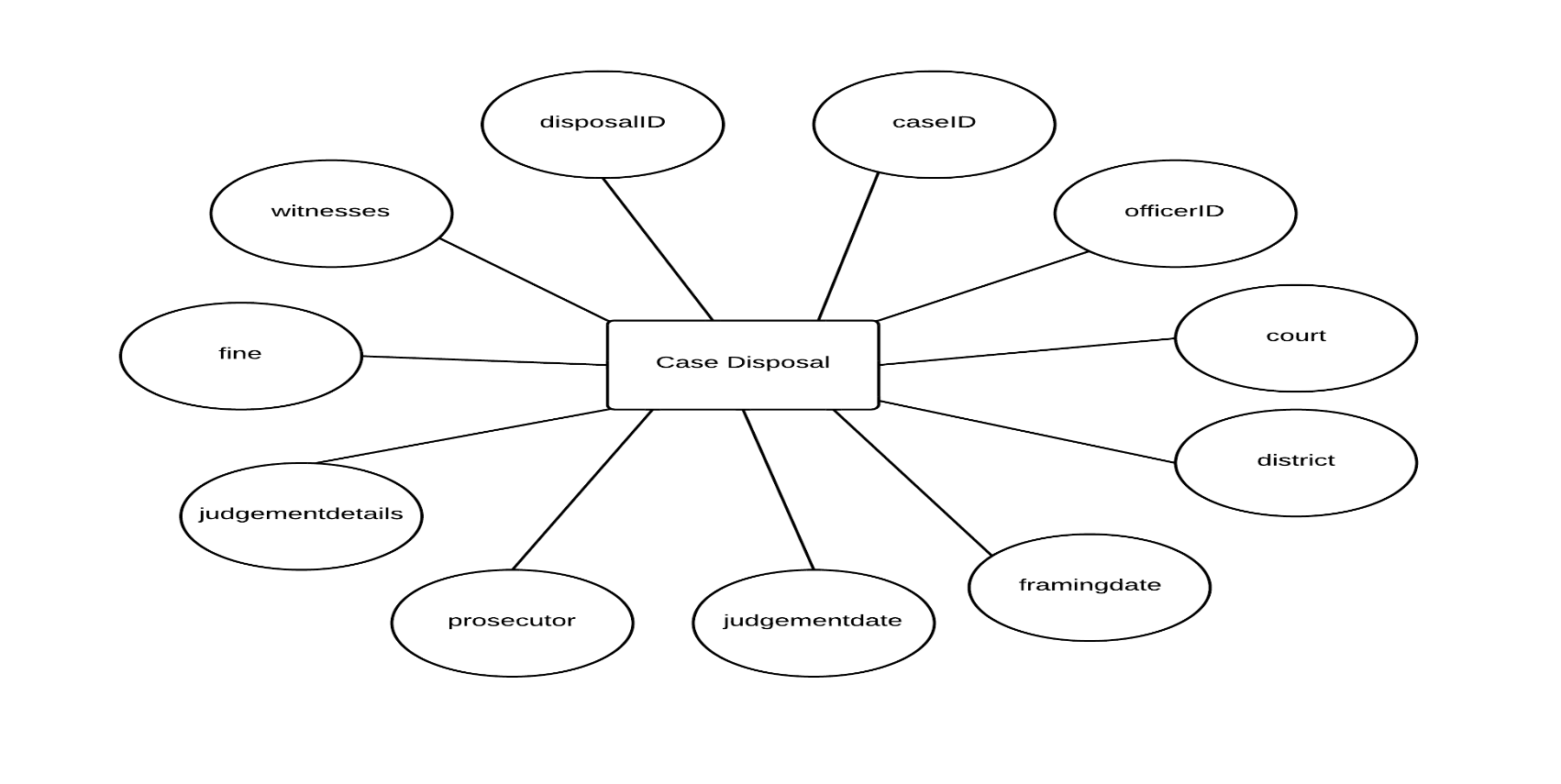
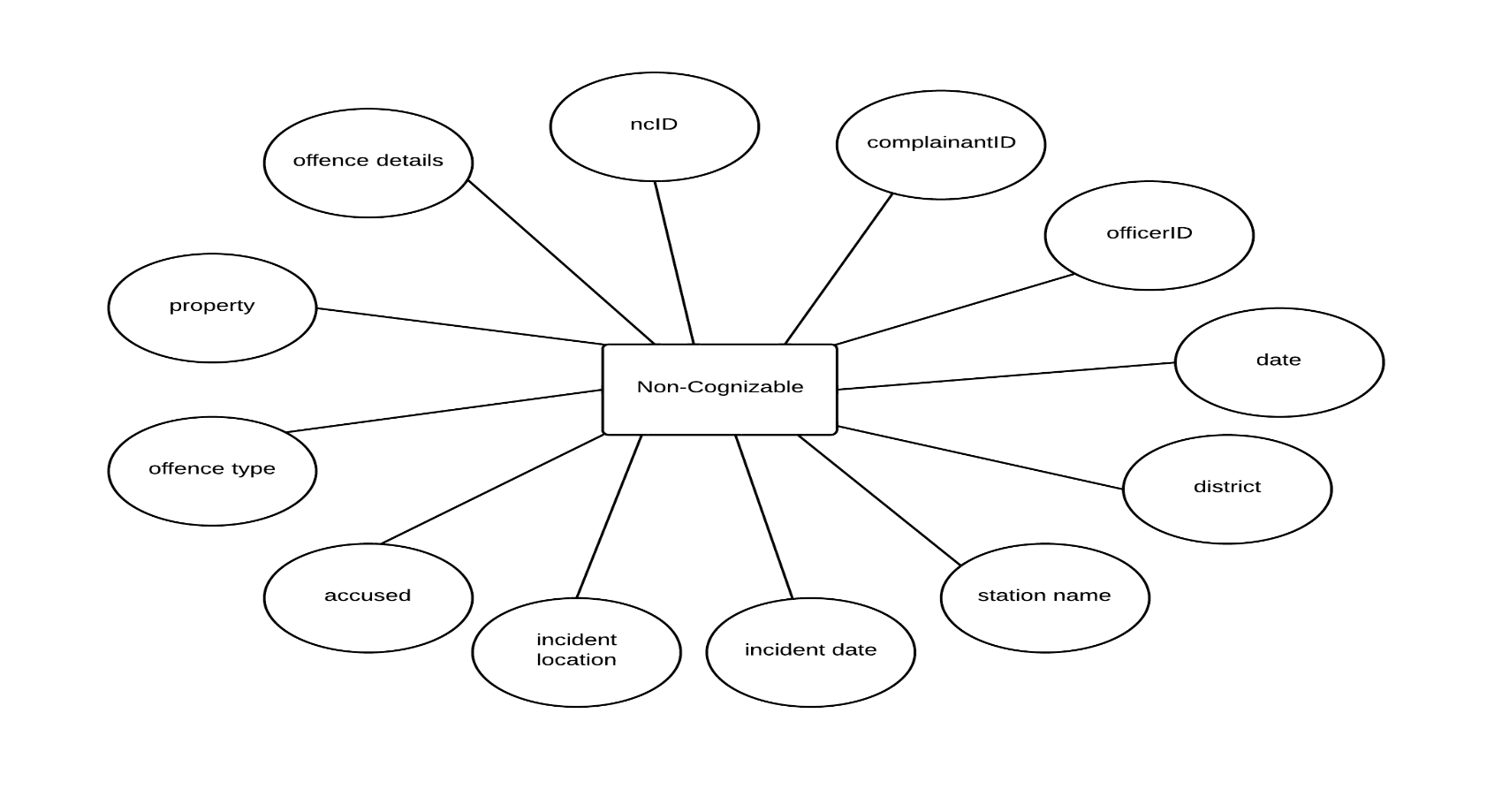


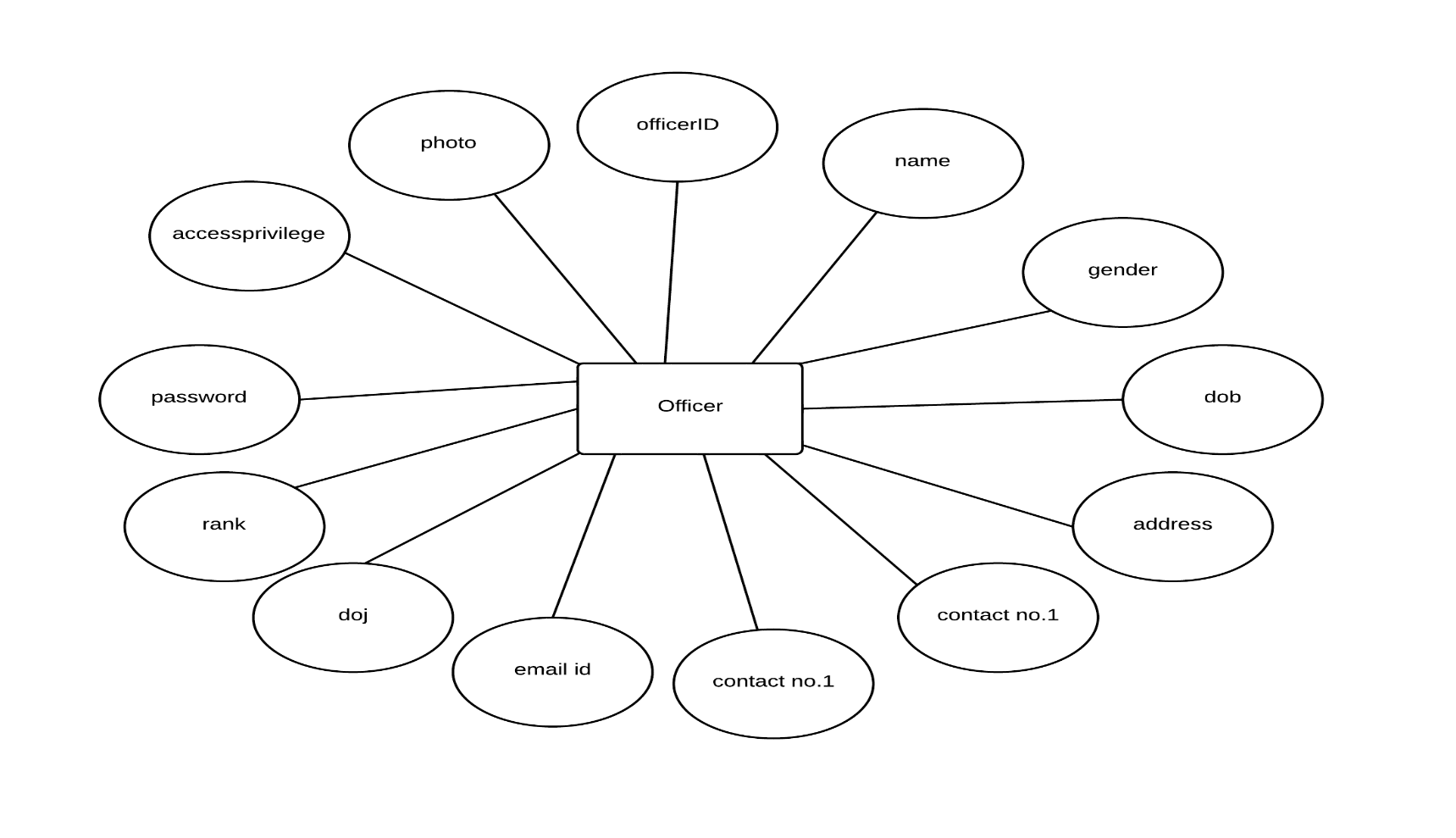
**Table 5.6.1 ER Notations**

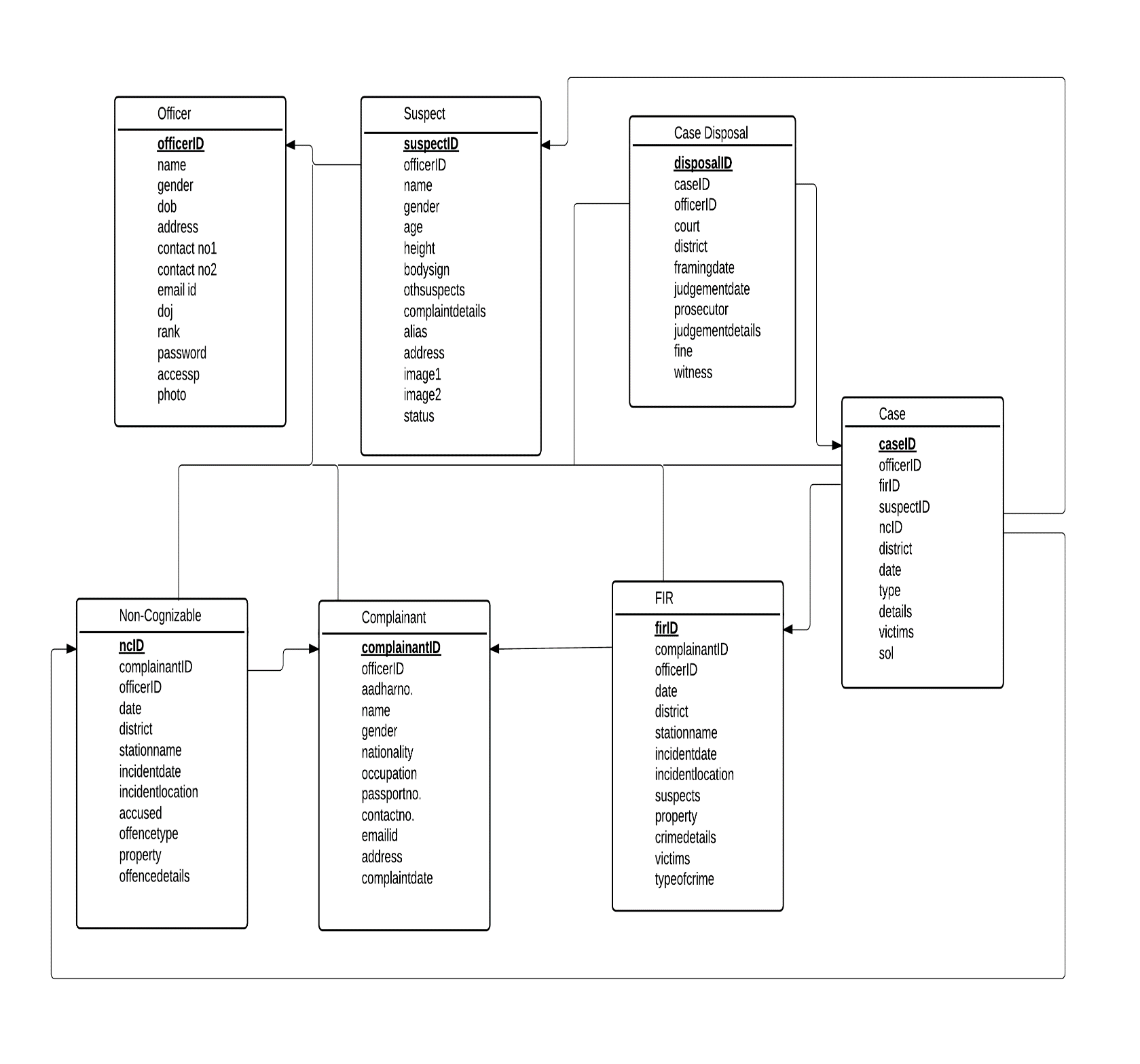
**Fig. 5.6.1(ER Diagram)**

**Entities in Crime Record Management System:**

****

1. Complainant Entity:
2. FIR Entity:
3. Suspect Entity:
4. Case Entity:
5. ****Case Disposal Entity:
6. ****Non-Cognizable Entity:

****Officer Entity:

* + 1. **Database Schema:**

**Fig. 5.6.2**

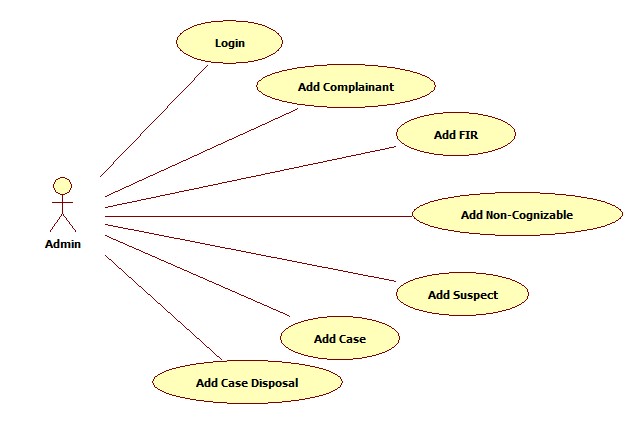
* + 1. **Use Case Diagrams:**

A use case diagram in the [Unified Modeling Language](http://en.wikipedia.org/wiki/Unified_Modeling_Language) (UML) is a type of behavioral diagram defined by and created from a [Use-case analysis](http://en.wikipedia.org/wiki/Use-case_analysis). Its purpose is to present a graphical overview of the functionality provided by a system in terms of [actors](http://en.wikipedia.org/wiki/Actor_(UML)), their goals (represented as [use cases](http://en.wikipedia.org/wiki/Use_case)), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can hence be depicted.

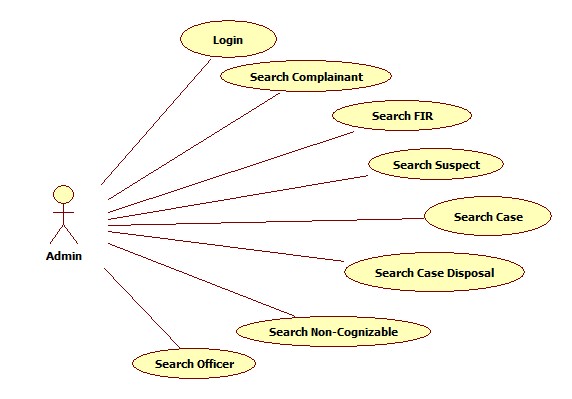
|  |  |  |
| --- | --- | --- |
| **Name** | **Symbol** | **Description** |
| Actor |  | The actor represents a user or another system that will interact with the system |
| Use Case |  | A use case is an external view of the system that represents some action that the user might perform in order to complete a task. |

Use cases are used in almost every project.  They are helpful in exposing requirements and planning the project. During the initial stage of a project most use cases should be defined, but as the project continues more might become visible.

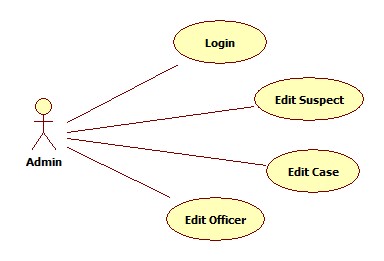
**Use Case Diagrams:**

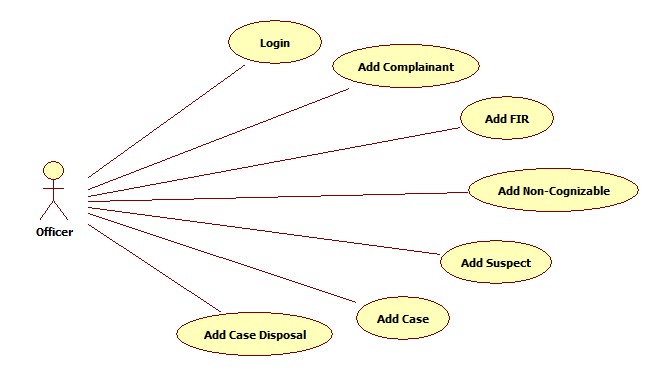


**Fig. 5.6.3.1 (Add Information for Admin)**

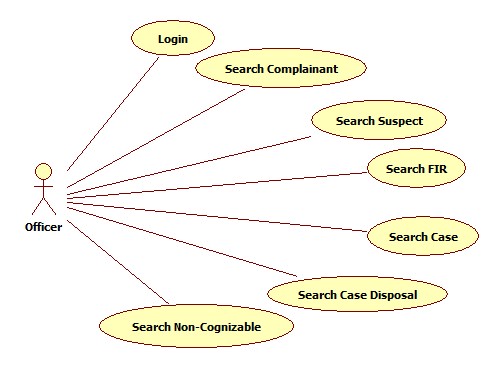


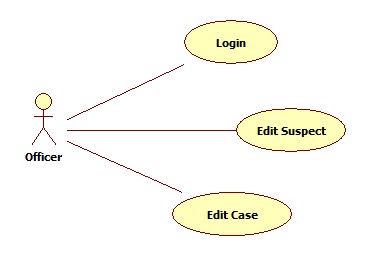
**Fig. 5.6.3.2 (Search Information for Admin)**



**Fig. 5.6.3.3 (Edit Information for Admin)**

**Fig. 5.6.3.4 (Add Information for Officer)**

**Fig. 5.6.3.5 (Search Information for Officer)**



**Fig. 5.6.3.6 (Edit Information for Officer)**

Use Case Name: Login.

Participating Actors: Admin, Officer.

Flow of Events: Select access level, provide username and password.

Entry Conditions: User must know the username and password.

Exit Conditions: User successfully logged into the system.

Quality Requirements: Display proper error messages while logging into the system.

Use Case Name: Add FIR Information.

Participating Actors: Admin, Officer.

Flow of Events: User will enter FIR Information.

Entry Conditions: User must know the details of FIR.

Exit Conditions: FIR Information is successfully added into the system.

Quality Requirements: Display proper error messages while adding information into the system.

Use Case Name: Add Complainant Information.

Participating Actors: Admin, Officer.

Flow of Events: User will enter Complainant Information.

Entry Conditions: User must know the details of Complainant.

Exit Conditions: Complainant Information is successfully added into the system.

Quality Requirements: Display proper error messages while adding information into the system.

Use Case Name: Add Suspect Information.

Participating Actors: Admin, Officer.

Flow of Events: User will enter Suspect Information.

Entry Conditions: User must know the details of Suspect.

Exit Conditions: Suspect Information is successfully added into the system.

Quality Requirements: Display proper error messages while adding information into the system.

Use Case Name: Add Case Information.

Participating Actors: Admin, Officer.

Flow of Events: User will enter Case Information.

Entry Conditions: User must know the details of Case.

Exit Conditions: Case Information is successfully added into the system.

Quality Requirements: Display proper error messages while adding information into the system.

Use Case Name: Add Case Disposal Information.

Participating Actors: Admin, Officer.

Flow of Events: User will enter Case Disposal Information.

Entry Conditions: User must know the details of Case Disposal.

Exit Conditions: Case Disposal Information is successfully added into the system.

Quality Requirements: Display proper error messages while adding information into the system.

Use Case Name: Add Non-Cognizable Information.

Participating Actors: Admin, Officer.

Flow of Events: User will enter Non-Cognizable Information.

Entry Conditions: User must know the details of Non-Cognizable.

Exit Conditions: Non-Cognizable Information is successfully added into the system.

Quality Requirements: Display proper error messages while adding information into the system.

Use Case Name: Add Officer Information.

Participating Actors: Admin.

Flow of Events: User will enter Officer Information.

Entry Conditions: User must know the details of Officer.

Exit Conditions: Officer Information is successfully added into the system.

Quality Requirements: Display proper error messages while adding information into the system.

Use Case Name: Search FIR Information.

Participating Actors: Admin, Officer

Flow of Events: User can view all FIR Information.

Entry Conditions: Display the details of FIR.

Exit Conditions: FIR details successfully displayed.

Quality Requirements: N/A

Use Case Name: Search Complainant Information.

Participating Actors: Admin, Officer

Flow of Events: User can view all Complainant Information.

Entry Conditions: Display the details of Complainant.

Exit Conditions: Complainant details successfully displayed.

Quality Requirements: N/A

Use Case Name: Search Suspect Information.

Participating Actors: Admin, Officer

Flow of Events: User can view all Suspect Information.

Entry Conditions: Display the details of Suspect.

Exit Conditions: Suspect details successfully displayed.

Quality Requirements: N/A

Use Case Name: Search Case Information.

Participating Actors: Admin, Officer

Flow of Events: User can view all Case Information.

Entry Conditions: Display the details of Case.

Exit Conditions: Case details successfully displayed.

Quality Requirements: N/A

Use Case Name: Search Case Disposal Information.

Participating Actors: Admin, Officer

Flow of Events: User can view all Case Disposal Information.

Entry Conditions: Display the details of Case Disposal.

Exit Conditions: Case Disposal details successfully displayed.

Quality Requirements: N/A

Use Case Name: Search Non-Cognizable Information.

Participating Actors: Admin, Officer

Flow of Events: User can view all Non-Cognizable Information.

Entry Conditions: Display the details of Non-Cognizable.

Exit Conditions: Non-Cognizable details successfully displayed.

Quality Requirements: N/A

Use Case Name: Search Officer Information.

Participating Actors: Admin.

Flow of Events: User can view all Officer Information.

Entry Conditions: Display the details of Officer.

Exit Conditions: Officer Details successfully displayed.

Quality Requirements: N/A

Use Case Name: Edit Suspect Information.

Participating Actors: Admin, Officer.

Flow of Events: User can edit Suspect Information.

Entry Conditions: Users should know the information to be updated.

Exit Conditions: Suspect Information is successfully updated.

Quality Requirements: Display proper error messages while editing Information.

Use Case Name: Edit Case Information.

Participating Actors: Admin, Officer.

Flow of Events: User can edit Case Information.

Entry Conditions: User should know the information to be updated.

Exit Conditions: Case Information is successfully updated.

Quality Requirements: Display proper error messages while editing Information.

Use Case Name: Edit Officer Information.

Participating Actors: Admin.

Flow of Events: User can edit Officer Information.

Entry Conditions: User should know the information to be updated.

Exit Conditions: Officer Information is successfully updated.

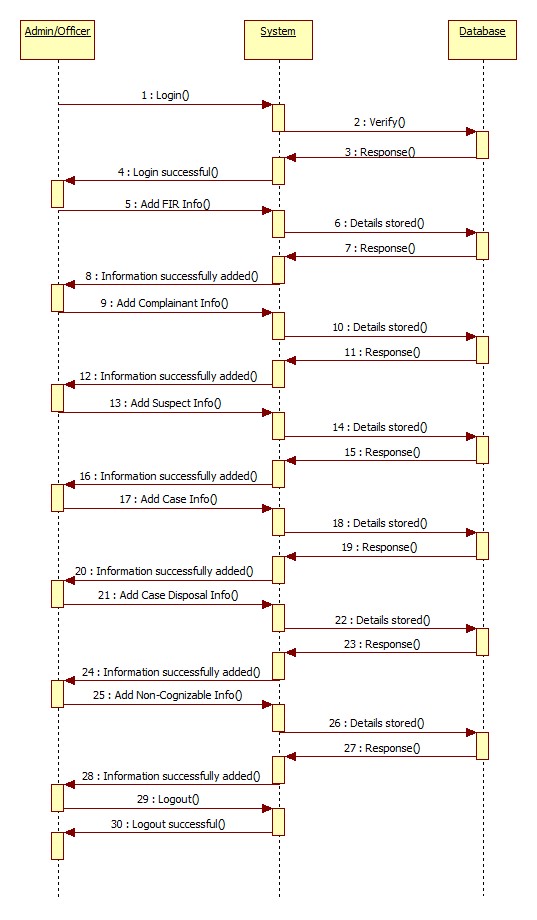
Quality Requirements: Display proper error messages while editing Information.

* + 1. **Sequence Diagrams:**

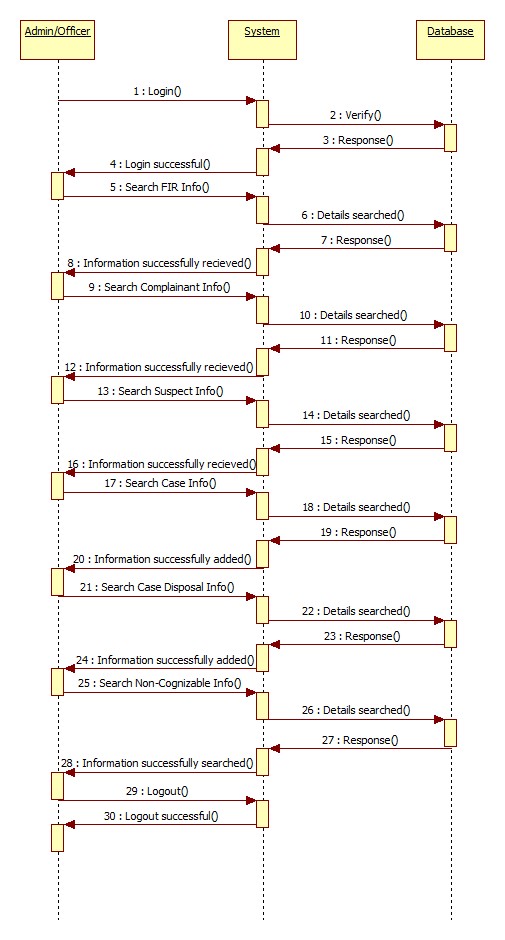
A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called Event-trace diagrams, event scenarios, and timing diagrams. The sequence diagram is used primarily to show the interactions between objects in the sequential order that those interactions occur. One of the primary uses of sequence diagrams is in the transition from requirements expressed as use cases to the next and more formal level of refinement. Use cases are often refined into one or more sequence diagrams. In addition to their use in designing new systems, sequence diagrams can be used to document how objects in an existing (call it "legacy") system currently interact.

Symbols:

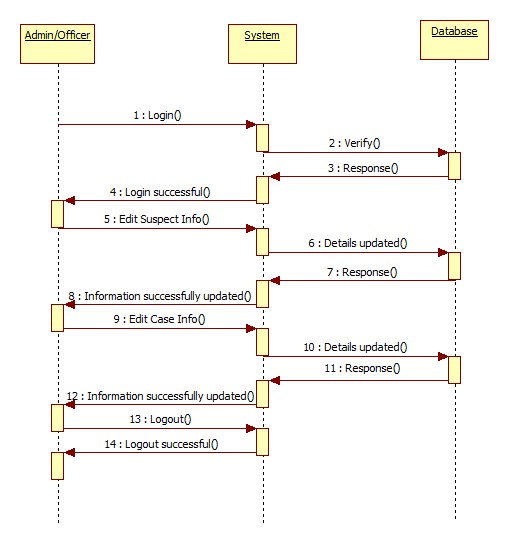
|  |  |  |
| --- | --- | --- |
| Synchronous message |  | An instantaneous communication between objects that conveys information, with the expectation that an action will be initiated as a result. |
| Activation |  | The period during which an object is performing an action |
| Object instance |  | An object that is created, performs actions, and/or is destroyed during the lifeline. |

1. For adding Information

**Fig. 5.6.4.1**

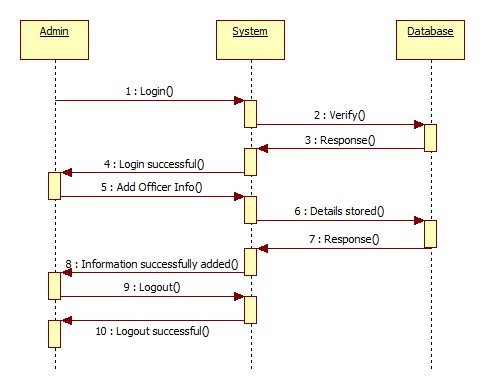
1. For Searching Information

**Fig. 5.6.4.2**

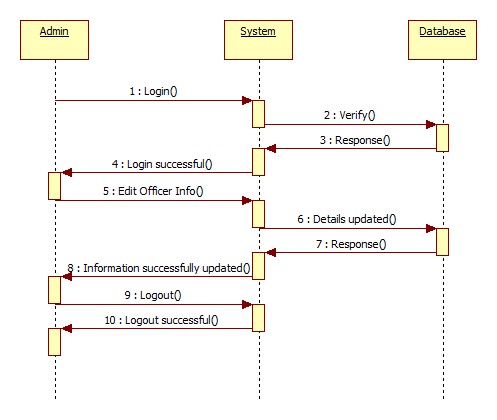
1. For Editing Information

**Fig. 5.6.4.3**

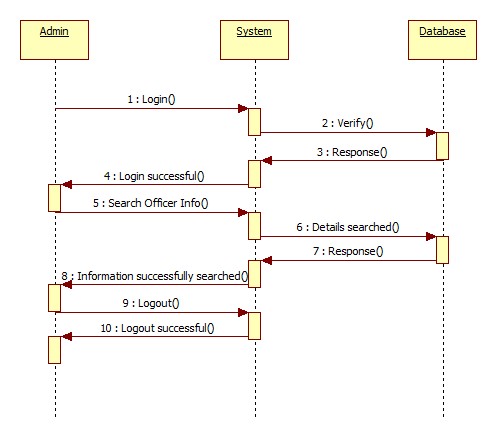
1. For Adding Officer Information



**Fig. 5.6.4.4**

1. For Searching Officer Information

**Fig. 5.6.4.5**

1. For Editing Officer Information

**Fig. 5.6.4.6**

* + 1. **Data Flow Diagrams:**

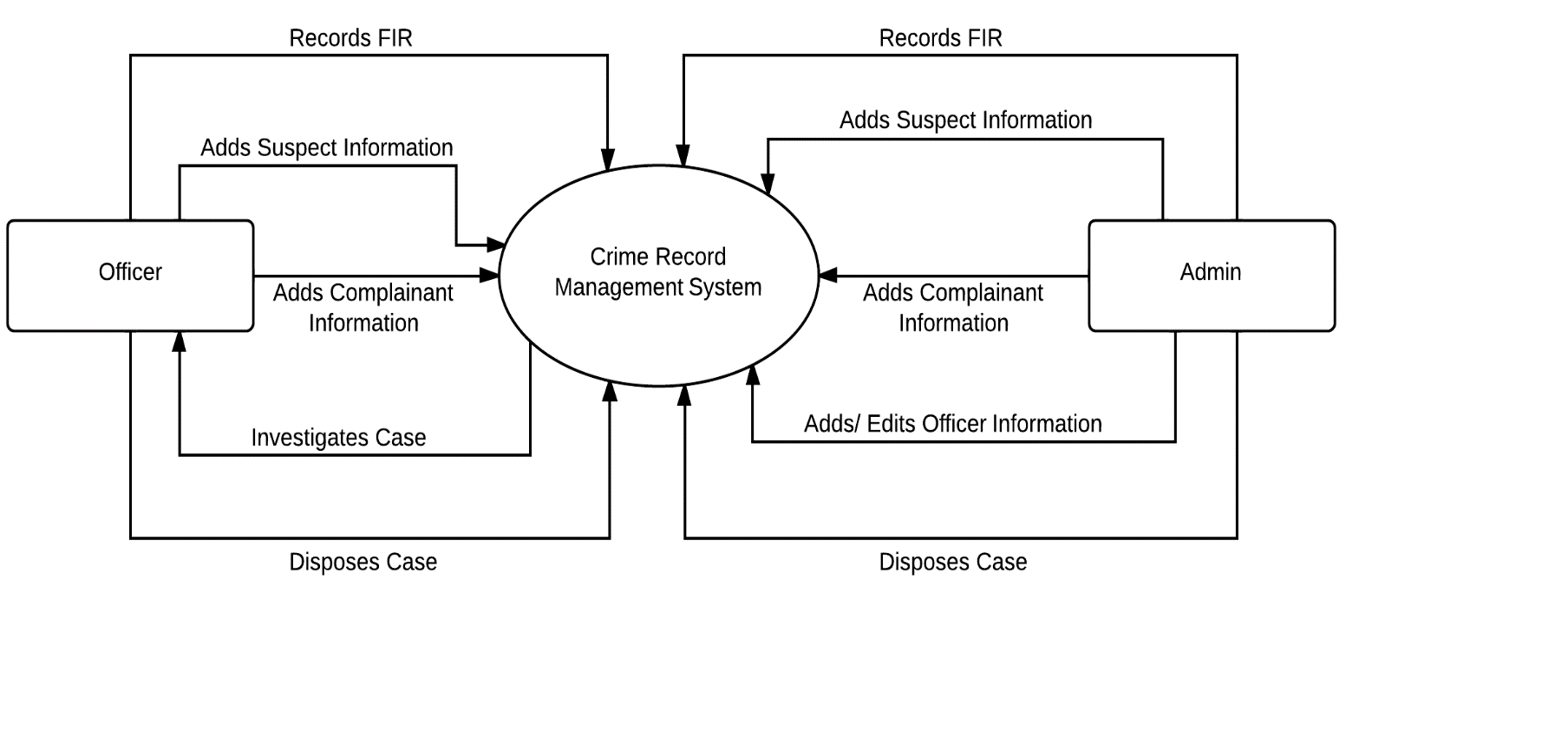
A data flow diagram (DFD) is a graphical system model that shows all of the main requirements for an information system in one diagram: inputs and outputs, processes, and data storage. Everyone working on a development project can see all aspects of the system working together at once with DFD. That is one reason for its popularity. The DFD is also easy to read because it is a graphical model. The DFD is mainly used during problem analysis. End Users, management, and all information systems workers typically can read and interpret the DFD with minimal training.

Notations Used:

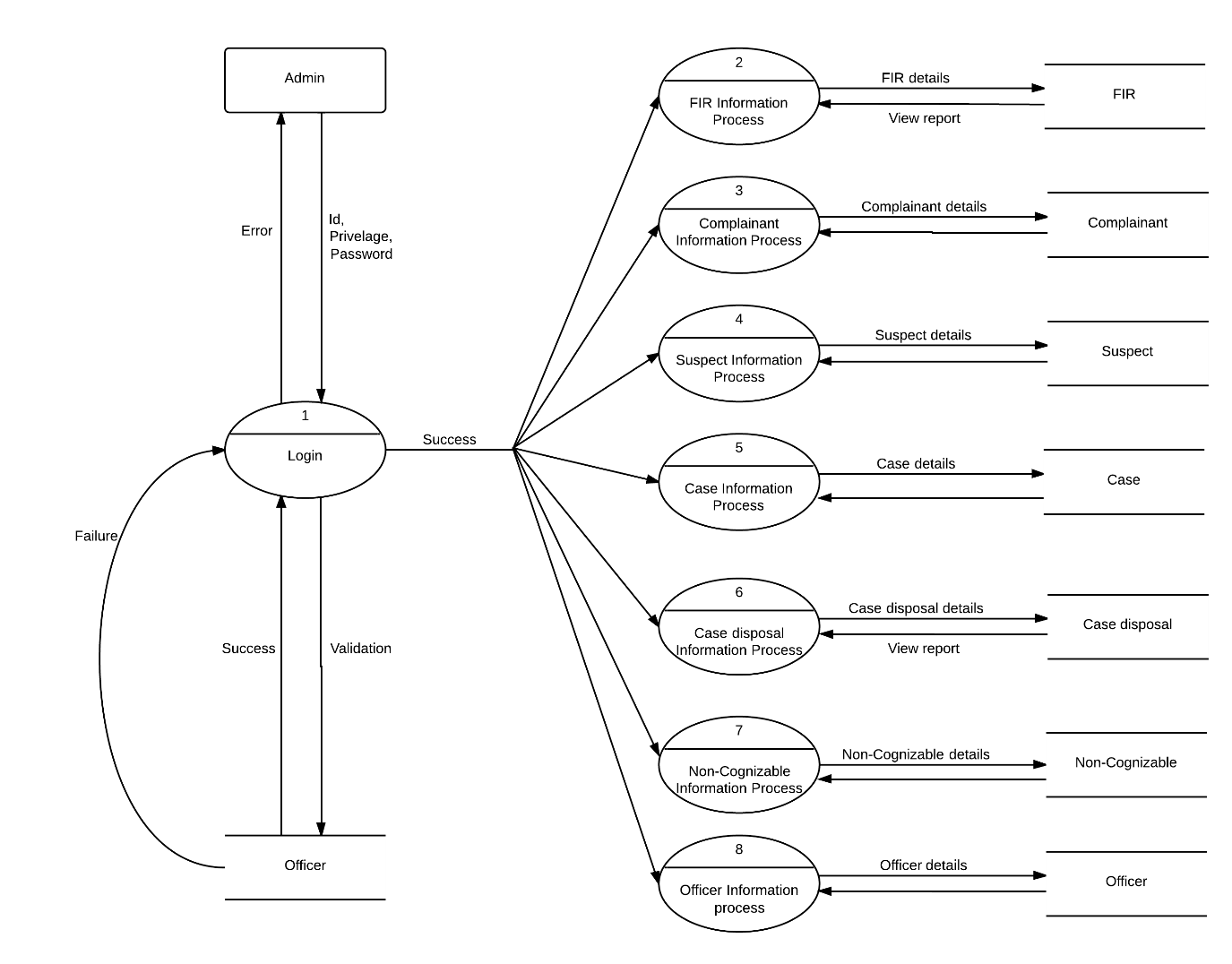
|  |  |  |
| --- | --- | --- |
| Name | Symbol | Description |
| Process |  | A Process transforms incoming data flow into outgoing data flow |
| Data Store |  | Data stores are repositories of data in the system |
| Data Flow |  | Data flows are pipelines through which packets of information flow |
| External Entity |  | External entities are objects outside the system, with which the system communicates |

**Table 5.6.5**

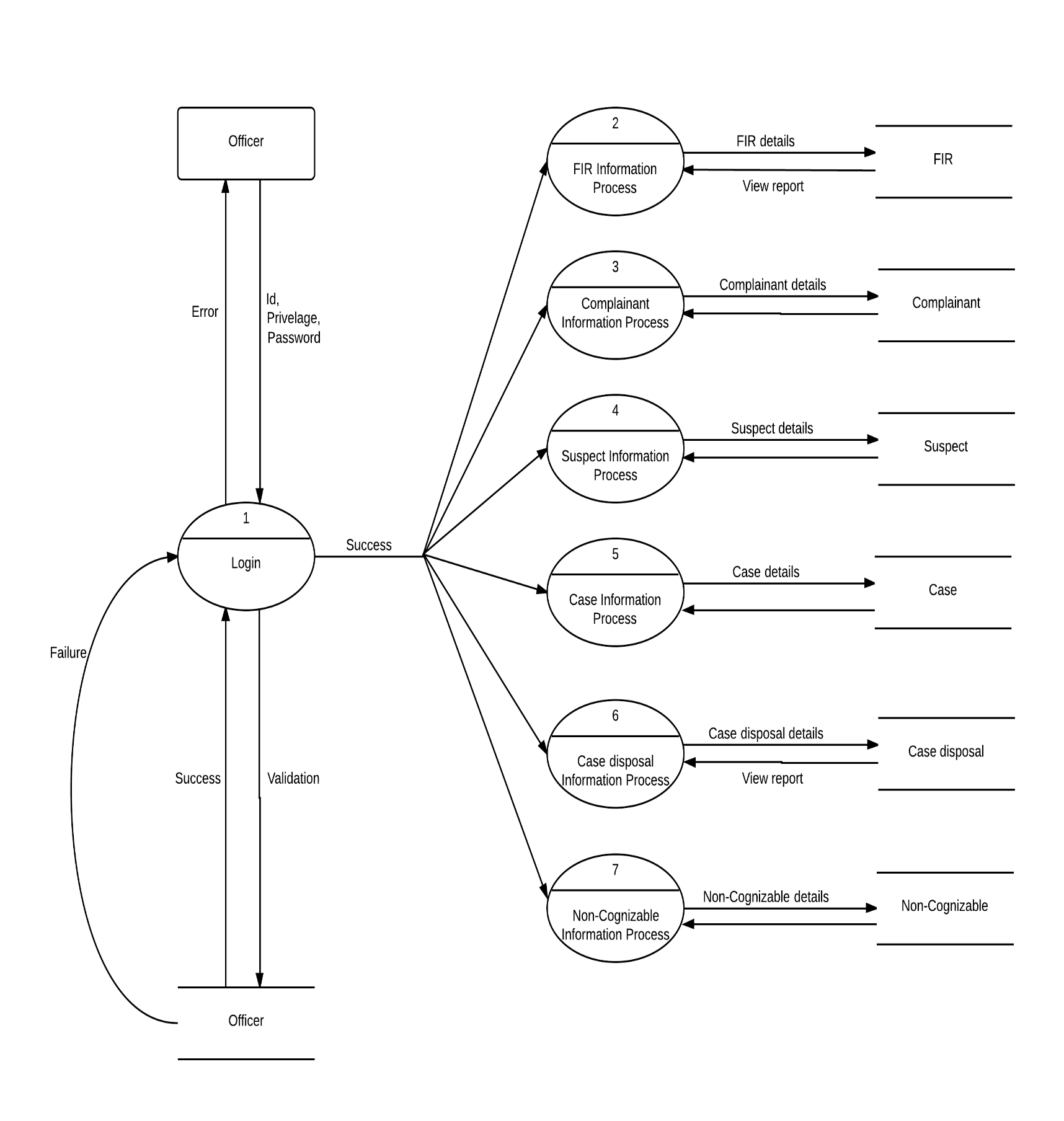
**Context Level DFD**



**Fig. 5.6.5.1**

**First Level DFD (Admin)**

**Fig. 5.6.5.2**

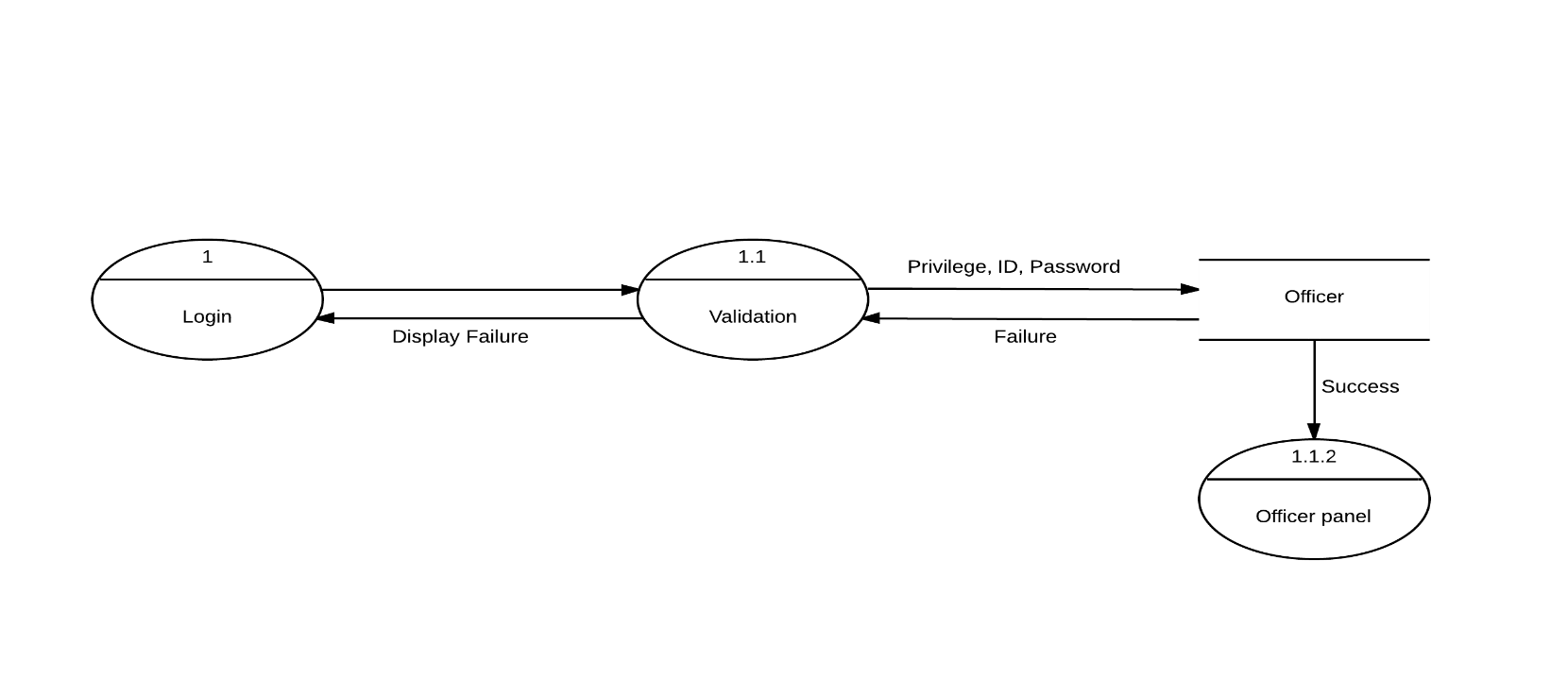
**First Level DFD (Officer)**

**Fig. 5.6.5.3 Level 1 Officer**

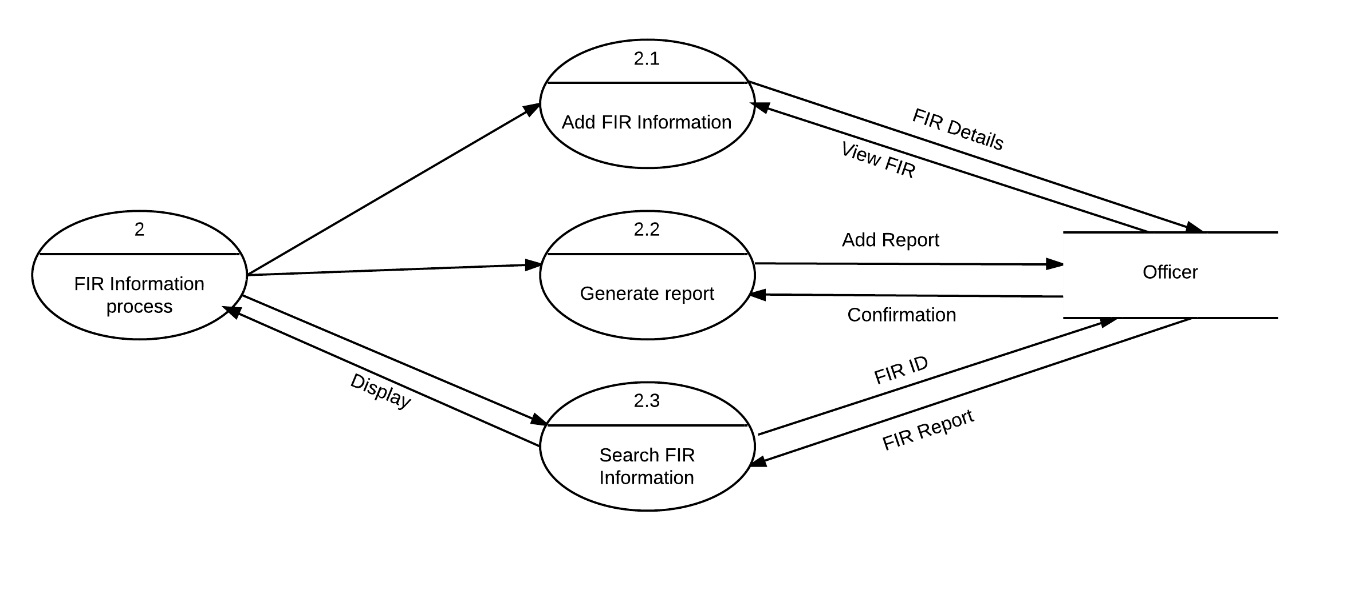
**Second Level DFD for Login (Admin)**



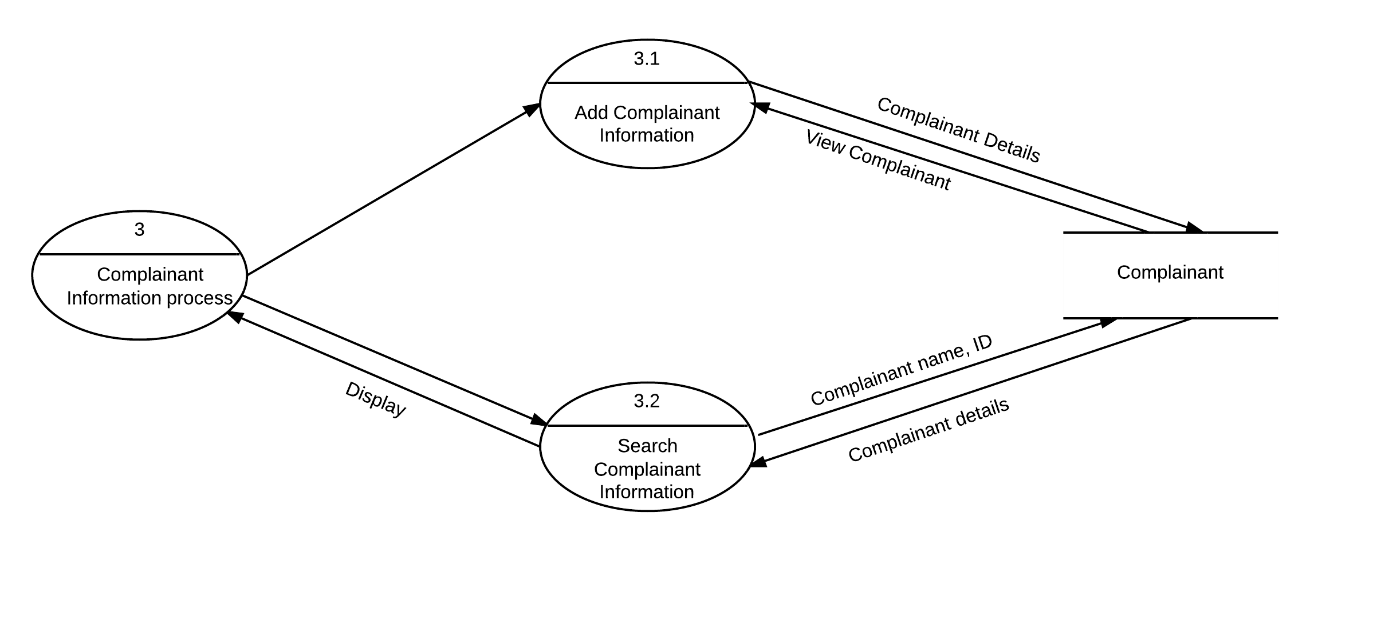
**Fig. 5.6.5.4 Level 2 Login Admin**

**Second Level DFD for Login (Officer)**

**Fig. 5.6.5.5 Level 2 Login Officer**

**Second Level DFD for FIR**

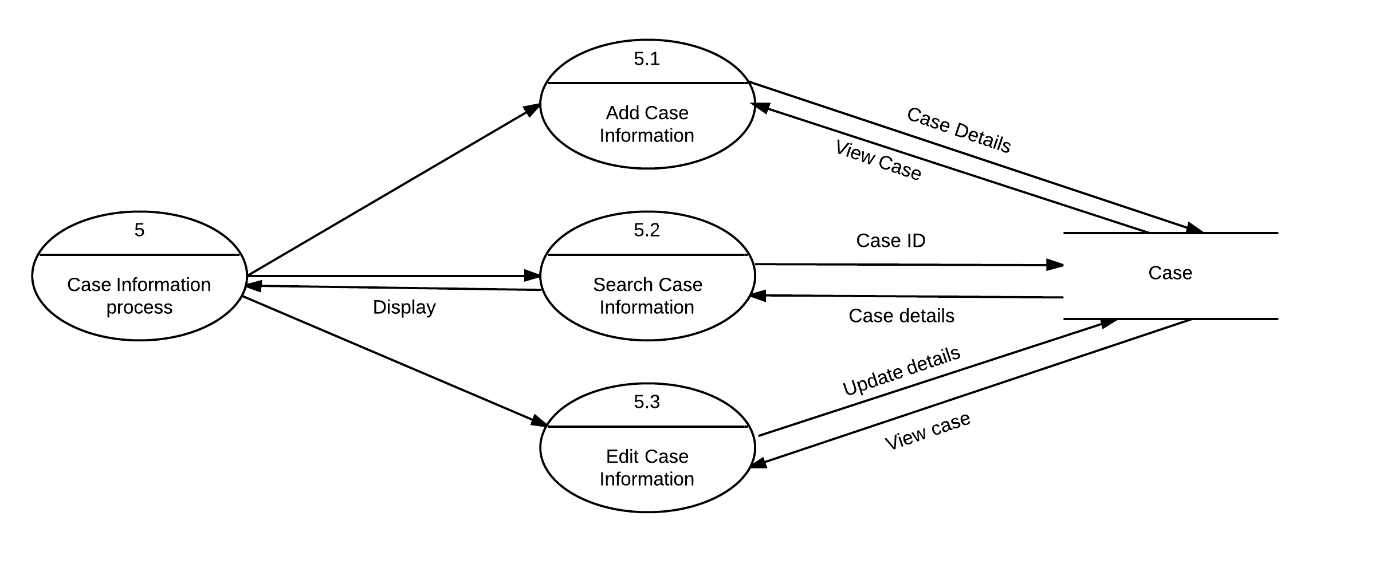
**Fig. 5.6.5.6 Level 2 FIR**

**Second Level DFD for Complainant**

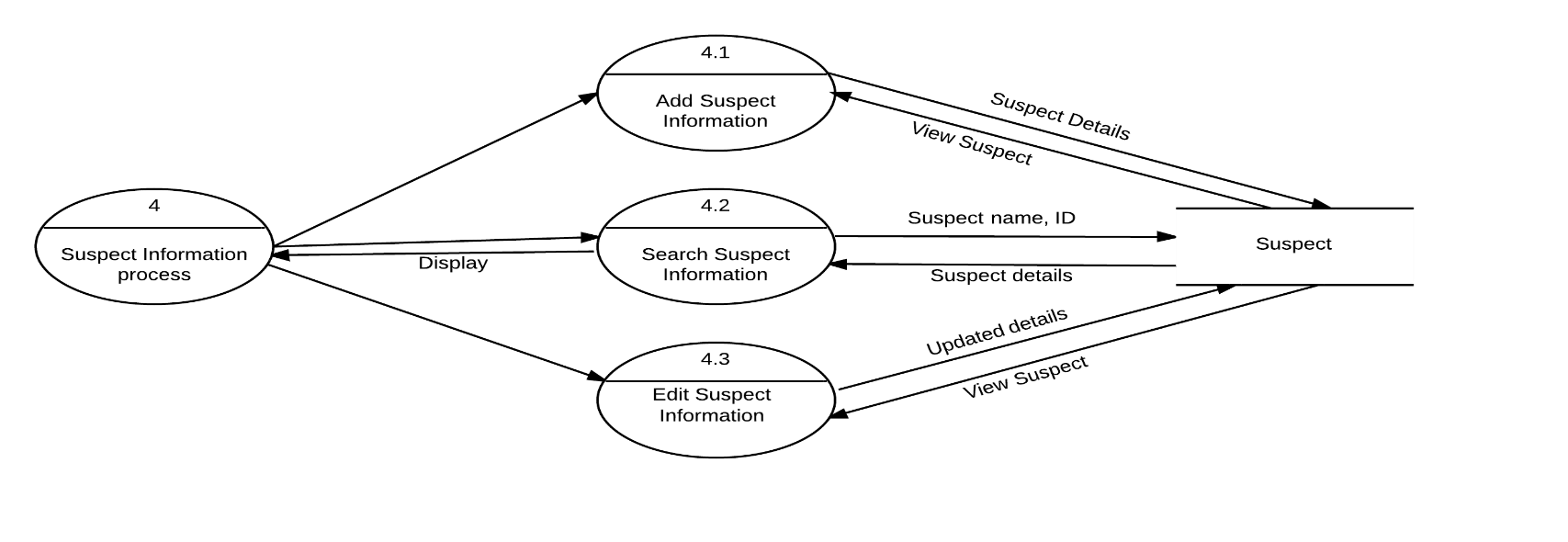
**Fig. 5.6.5.7 Level 2 Complainant**

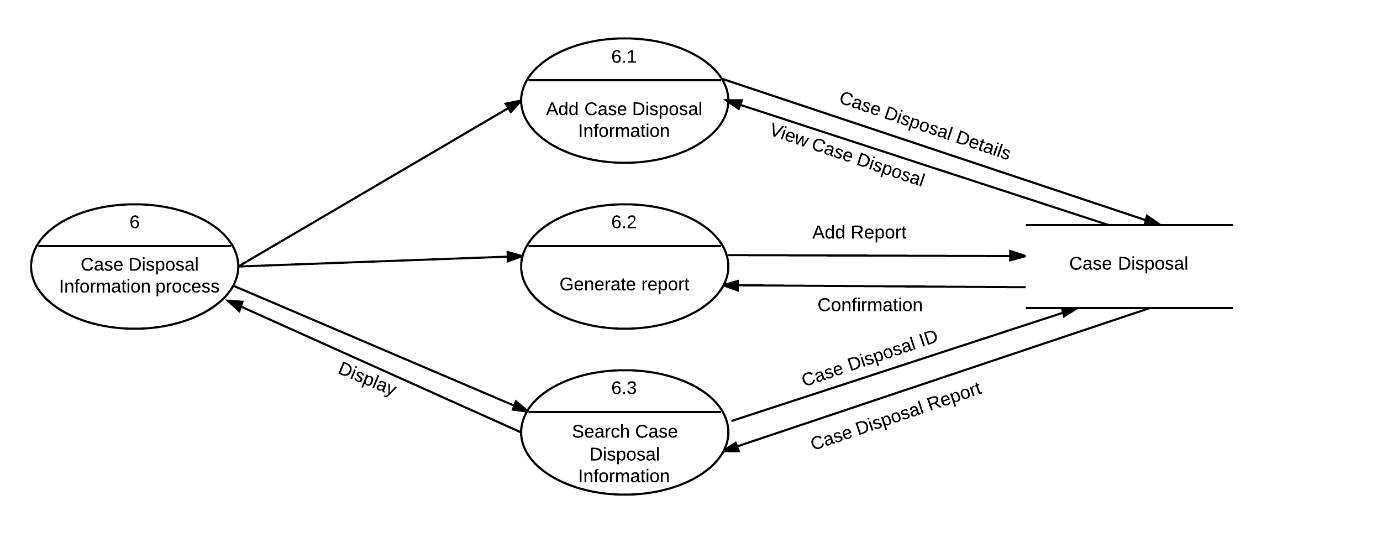
**Second Level DFD for Suspect**

**Fig. 5.6.5.8 Level 2 Suspect**

****

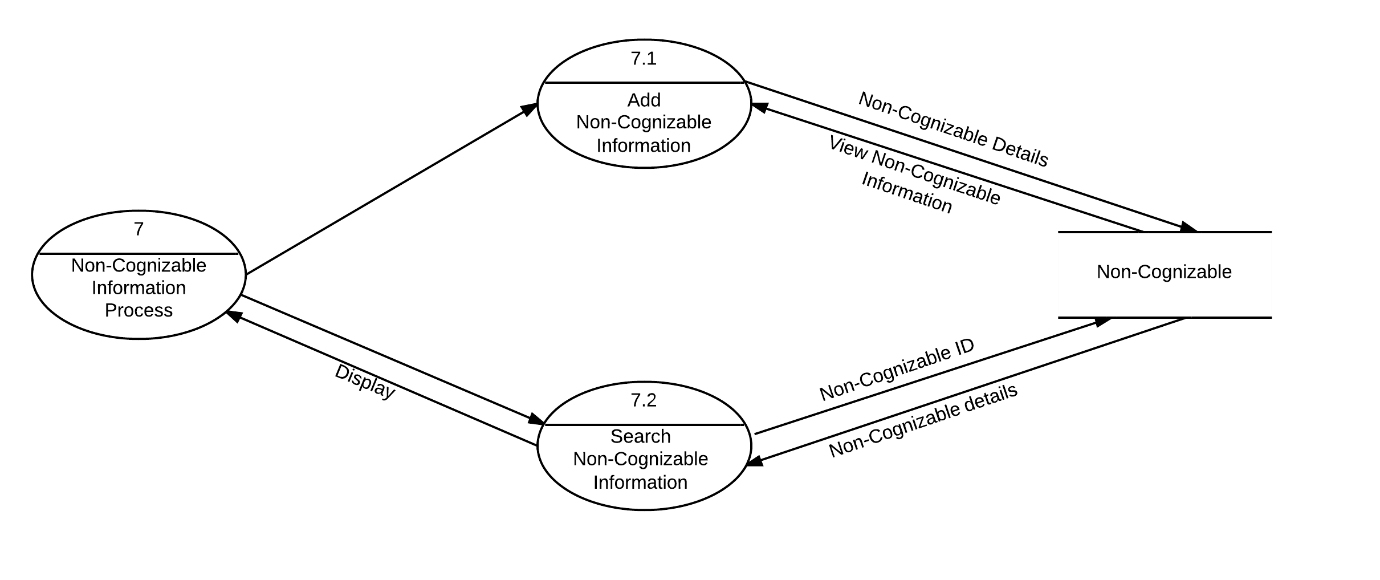
**Second Level DFD for Case**

**Fig. 5.6.5.9 Level 2 Case**

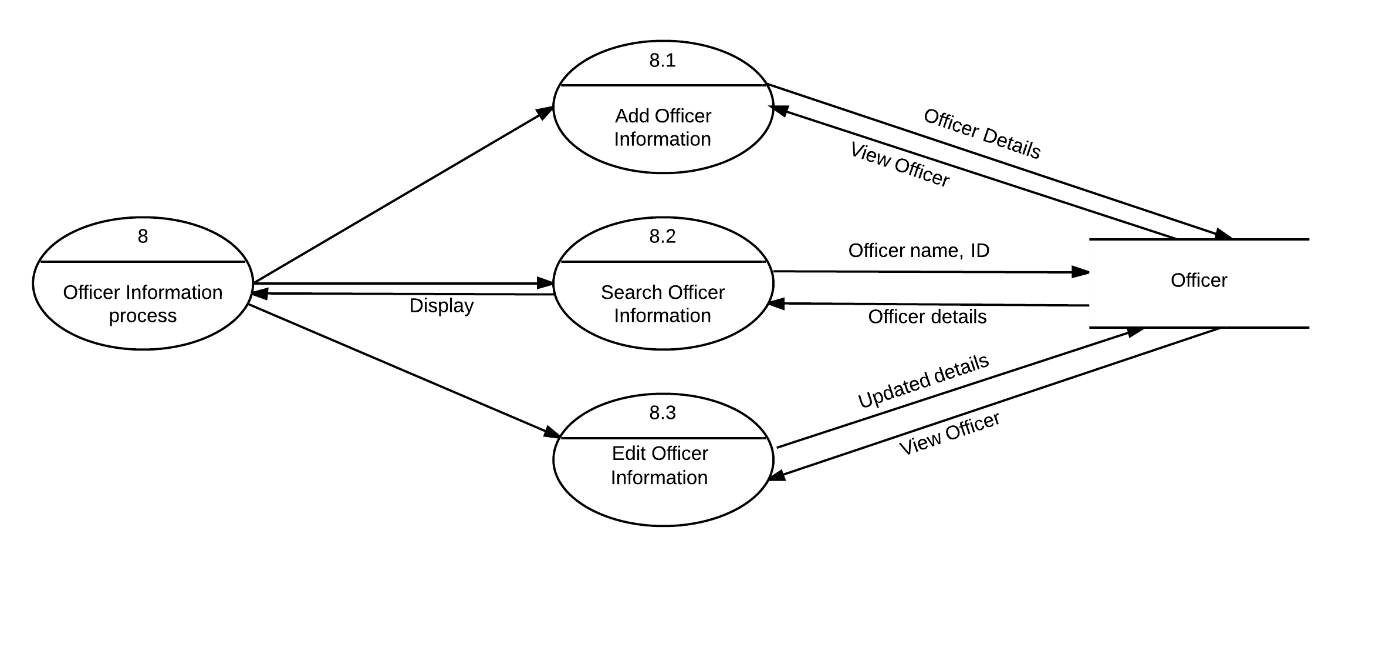
**Second Level DFD for Case Disposal**

**Fig. 5.6.5.10 Level 2 Case Disposal**

**Second Level DFD for Non-Cognizable**

****

**Fig. 5.6.5.11 Level 2 Non-Cognizable**

**Second Level DFD for Officer**

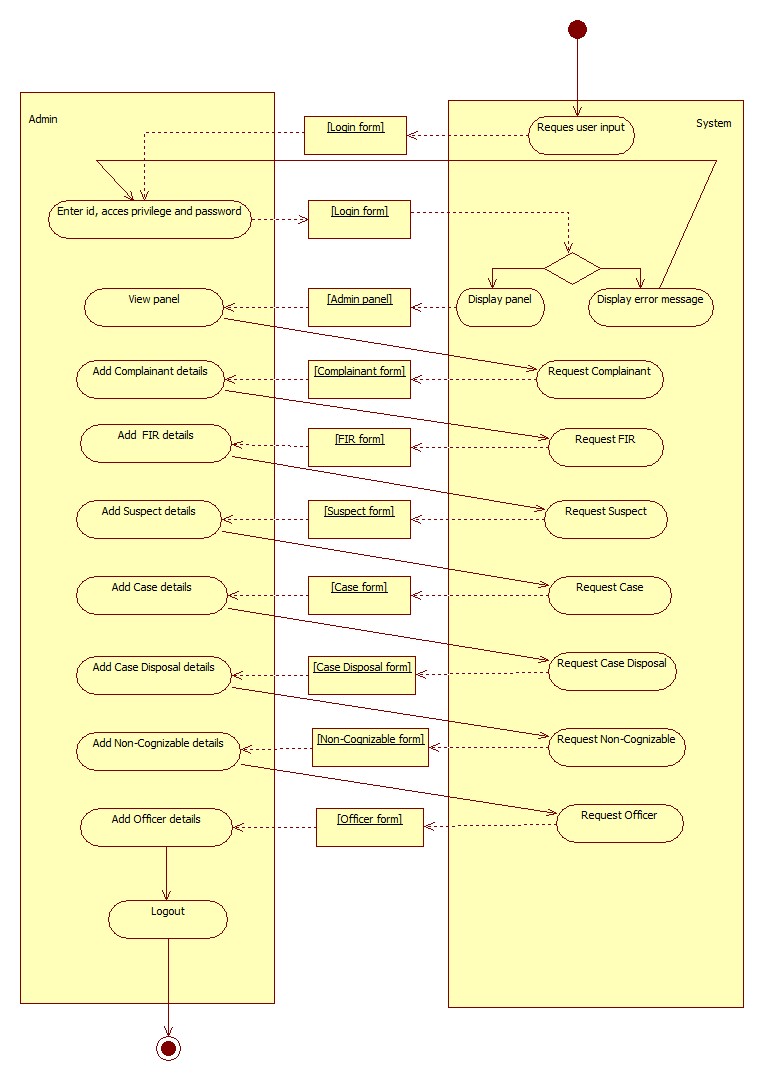
**Fig. 5.6.5.12 Level 2 Officer**

* + 1. **Activity Diagrams:**

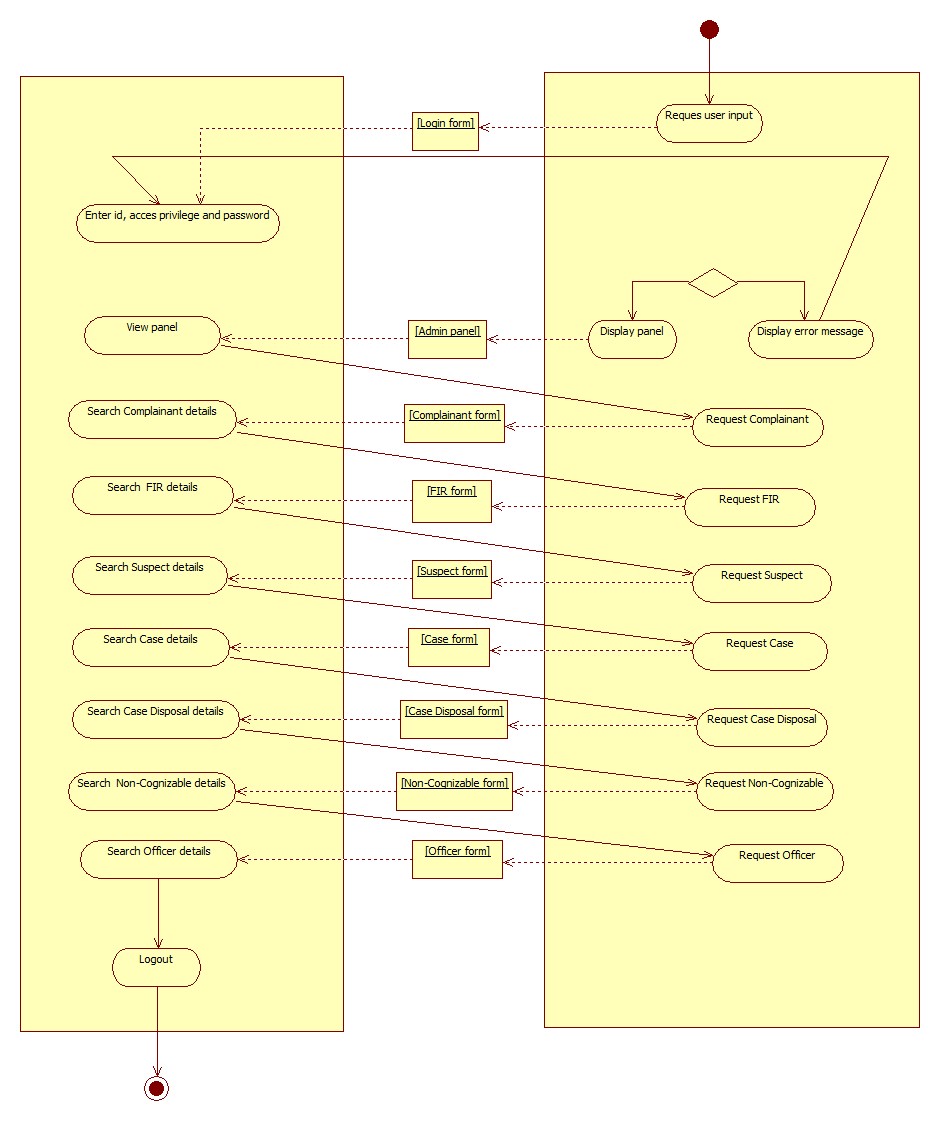
Activity diagrams describe the workflow behavior of a system. Activity diagrams are similar to state diagrams because activities are the state of doing something. The diagrams describe the state of activities by showing the sequence of activities performed. Activity diagrams can show activities that are conditional or parallel. Activity diagrams should be used in conjunction with other modeling techniques such as interaction diagrams and state diagrams. The main reason to use activity diagrams is to model the workflow behind the system being designed.

|  |  |  |
| --- | --- | --- |
| Free-form transition link |  | A transition link represents control flow between nodes |
| Rounded rectangle |  | Represents the activity |
| Diamond |  | A logic where a decision is to be made |
| Initial activity |  | Shows the first activity of the flow |
| Final activity |  | Shows the final activity of the flow |

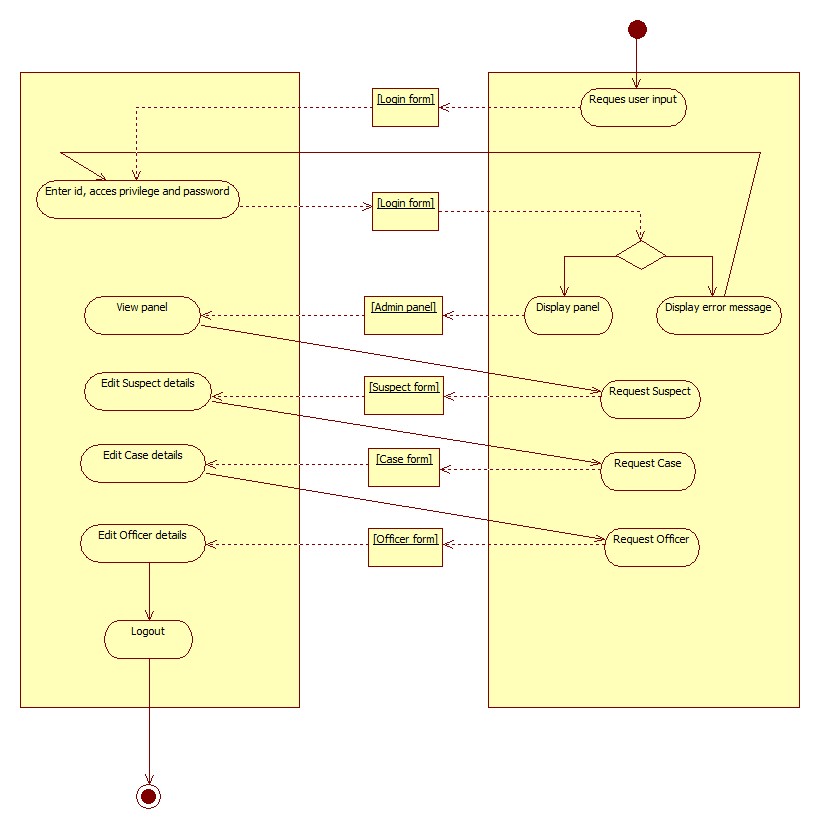
Table 5.6.6

**Activity Diagram for Adding Information (Admin):**

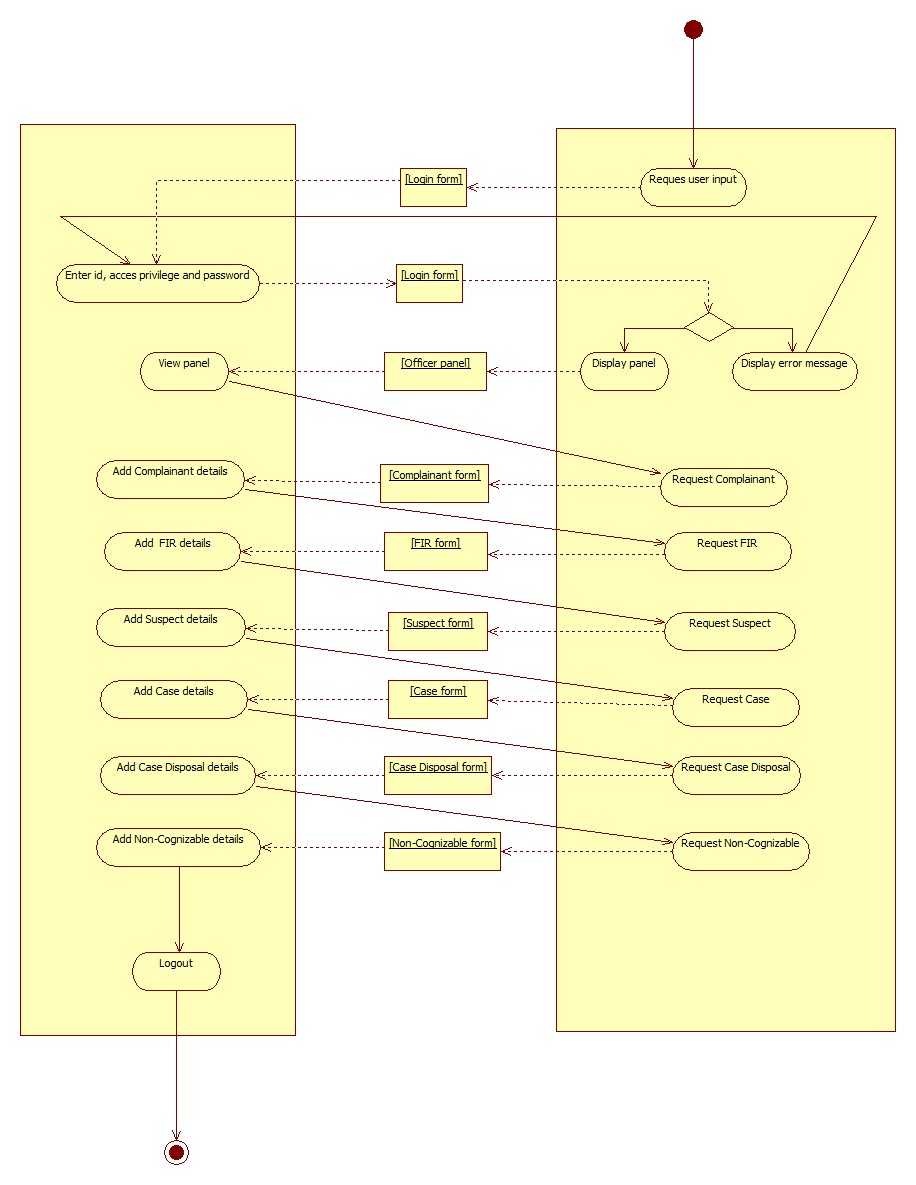
**Fig. 5.6.6.1 Adding information**

**Activity diagram for searching information (Admin):**

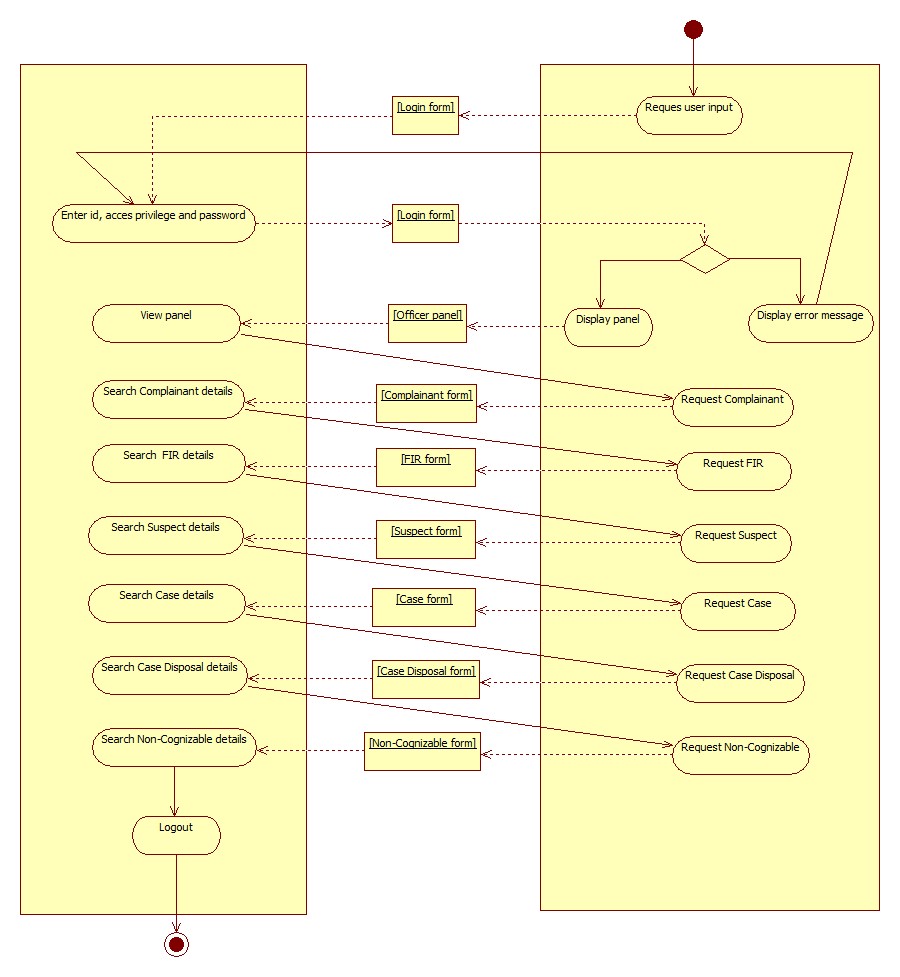
**Fig. 5.6.6.2 Searching information**

**Activity diagram for editing information (Admin):**

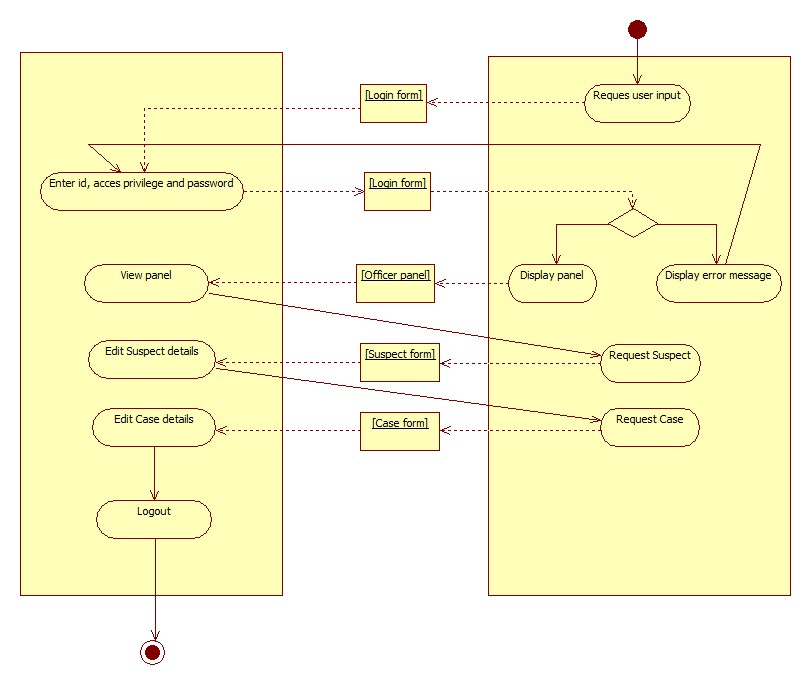
**Fig. 5.6.6.3 Editing information**

**Activity diagram for adding information (Officer):**

**Fig. 5.6.6.4 Adding information**

**Activity diagram for searching information (Officer):**

**Fig. 5.6.6.5 Searching information**

**Activity diagram for editing information (Officer):**

**Fig.5.6.6.6**

* + 1. **Class Diagram:**

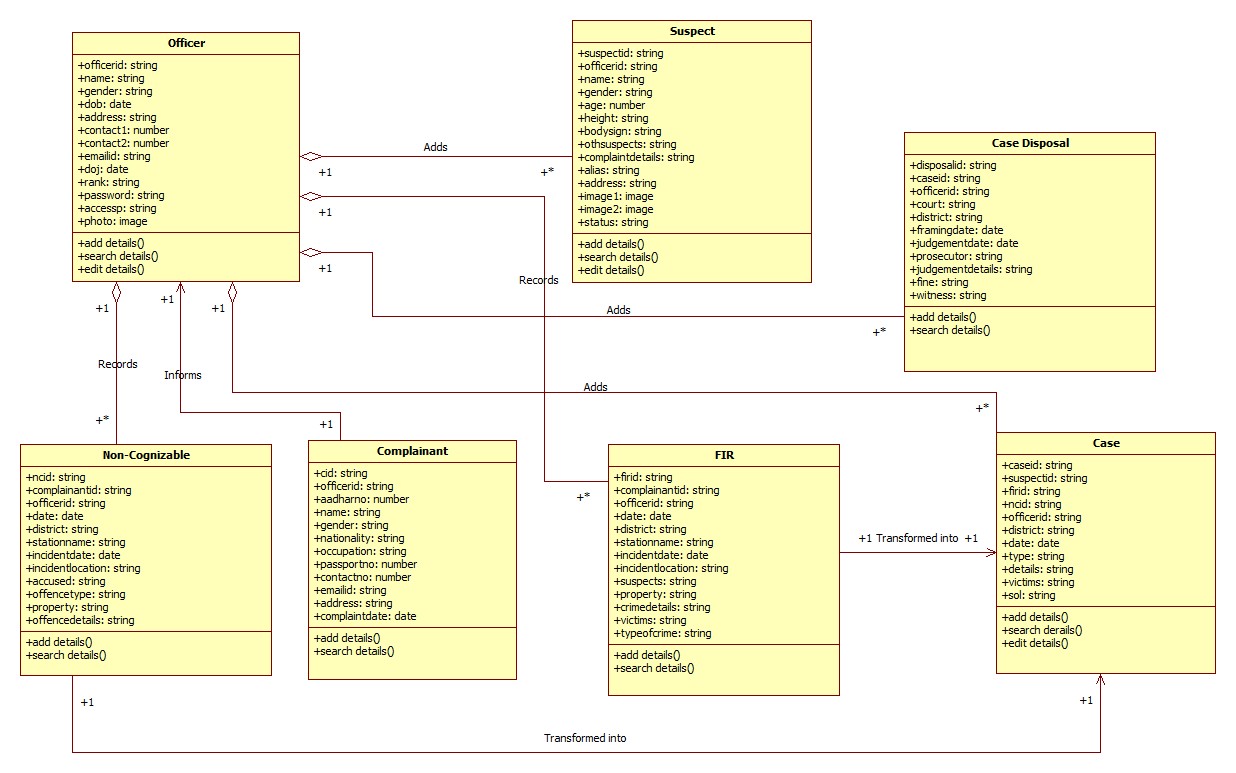
The class diagram is the main building block of object oriented modeling. In this diagram, classes are represented with boxes which contain three parts:

1. The top part contains the name of the class.
2. The middle part contains the attributes of the class.
3. The bottom part gives the methods or operations the class can take or undertake.

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. no. | Symbol | Name | Description |
| 1. |  | Association | Represents the static relationship shared among the objects of two classes |
| 2 |  | Aggregation | It is an association that represents a part-whole or part-of relationship. |
| 3 |  | Generalization | Indicates that one of the two related classes is considered to be a specialized form of the other |
| 4 |  | Dependency | It is a weaker form of bond which indicates that one class depends on another because it uses it at some point in time |

**Multiplicity Table:** Indicates how many of the objects at this end can be linked to each object at the other.

|  |  |
| --- | --- |
| **0..1** | No instances, or one instance |
| **1** | Exactly one instance |
| **0..\*** or **\*** | Zero or more instances |
| **1..\*** | One or more instances |

**Class Diagram:**

**Fig. 5.6.7**

* + 1. **Table list:**

**Officer table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field name** | **Data type** | **Constraint** | **Description** |
| 1 | officer id | nvarchar | Primary key | Stores unique id |
| 2 | name | nvarchar | Not null | Stores name of officer |
| 3 | gender | nvarchar | Not null | Stores gender of officer |
| 4 | dob | nvarchar | Not null | Stores date of birth |
| 5 | address | nvarchar | Not null | Stores address |
| 6 | contact1 | nvarchar | Not null | Stores contact no. |
| 7 | contact2 | nvarchar | Null | Stores contact no. |
| 8 | email id | nvarchar | Not null | Stores email id |
| 9 | doj | nvarchar | Not null | Stores date of joining |
| 10 | rank | nvarchar | Not null | Stores rank of officer |
| 11 | password | nvarchar | Not null | Stores password |
| 12 | accessp | nvarchar | Not null | Select access privilege |
| 13 | photo | image | Not null | Stores photo |

**Table 5.6.8.1**

**Complainant table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field name** | **Data type** | **Constraint** | **Description** |
| 1 | complainant id | nvarchar | Primary key | Stores complainant id |
| 2 | officer id | nvarchar | Foreign key | Stores officer id |
| 3 | aadhar no. | nvarchar | Null | Stores aadhar no. |
| 4 | name | nvarchar | Not null | Stores name of the complainant |
| 5 | gender | nvarchar | Not null | Store gender |
| 6 | nationality | nvarchar | Not null | Stores nationality |
| 7 | occupation | nvarchar | Not null | Stores occupation info |
| 8 | passport no. | nvarchar | Not null | Stores passport no. |
| 9 | contact no. |  | Not null | Stores contact no. |
| 10 | email id | nvarchar | Null | Stores email d of complainant |
| 11 | address | nvarchar | Null | Store address of complainant |
| 12 | complaint date | nvarchar | Not null | Stores the date of recording the complainant |

**Table 5.6.8.2**

**Suspect table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field name** | **Data type** | **Constraint** | **Description** |
| 1 | suspect id | nvarchar | Primary key | Stores suspect id |
| 2 | officer id | nvarchar | Foreign key | Stores officer id |
| 3 | Name | nvarchar | Not null | Stores name of suspect |
| 4 | Gender | nvarchar | Not null | Stores gender |
| 5 | Age | nvarchar | Not null | Stores age |
| 6 | Height | nvarchar | Not null | Stores height |
| 7 | Body sign | nvarchar | Null | Stores body sign of suspect |
| 8 | Other suspects | nvarchar | Null | Stores details about other suspects |
| 9 | Complaint details | nvarchar | Not null | Stores complaint details |
| 10 | Alias | nvarchar | Null | Stores alias details |
| 11 | Address | nvarchar | Not null | Stores address of suspect |
| 12 | Image 1 | image | Not null | Stores suspect image |
| 13 | Image 2 | image | Null | Stores suspect image |
| 14 | status | nvarchar | Not null | Stores suspect status |

**Table 5.6.8.3**

**FIR table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field name** | **Data type** | **Constraint** | **Description** |
| 1 | fir id | nvarchar | Primary key | Stores fir id |
| 2 | complainant id | nvarchar | Foreign key | Stores complainant id |
| 3 | officer id | nvarchar | Foreign key | Stores officer id |
| 4 | date | datetime | Not null | Stores date |
| 5 | district | nvarchar | Not null | Stores district name |
| 6 | station name | nvarchar | Not null | Stores station name |
| 7 | incident date | nvarchar | Not null | Stores date of incident |
| 8 | incident location | nvarchar | Not null | Stores incident location |
| 9 | suspects | nvarchar | Not null | Stores details of suspects |
| 10 | property | nvarchar | Not null | Stores property damage details |
| 11 | crime details | nvarchar | Not null | Stores crime details |
| 12 | victims | nvarchar | Not null | Stores victims |
| 13 | type of crime | nvarchar | Not null | Stores type of crime |

**Table 5.6.8.4**

**Case tables:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field name** | **Data type** | **Constraint** | **Description** |
| 1 | case id | nvarchar | Primary key | Stores case id |
| 2 | suspect id | nvarchar | Foreign key | Stores suspect id |
| 3 | fir id | nvarchar | Foreign key | Stores fir id |
| 4 | nc id | nvarchar | Foreign key | Stores nc id |
| 5 | officer id | nvarchar | Foreign key | Stores officer id |
| 6 | district | nvarchar | Not null | Stores district name |
| 7 | date | nvarchar | Not null | Stores date of case |
| 8 | type | nvarchar | Not null | Stores type of case |
| 9 | details | nvarchar | Not null | Stores details of case |
| 10 | victims | nvarchar | Not null | Stores victims’ name |
| 11 | sol | nvarchar | Not null | Stores section of law |

**Table 5.6.8.5**

**Case Disposal table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field name** | **Data type** | **Constraint** | **Description** |
| 1 | case disposal id | nvarchar | Primary key | Stores case disposal id |
| 2 | case id | nvarchar | Foreign key | Stores case id |
| 3 | officer id | nvarchar | Foreign key | Stores officer id |
| 4 | court | nvarchar | Not null | Stores court name |
| 5 | district | nvarchar | Not null | Stores district name |
| 6 | framing date | nvarchar | Not null | Stores date of framing |
| 7 | judgement date | nvarchar | Not null | Stores date of judgement |
| 8 | prosecutor | nvarchar | Not null | Stores prosecutor name |
| 9 | judgement details | nvarchar | Not null | Stores judgement details |
| 10 | fine | nvarchar | Not null | Stores fine amount |
| 11 | witnesses | nvarchar | Null | Stores witness details |

**Table 5.6.8.6**

**Non-Cognizable table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field name** | **Data type** | **Constraint** | **Description** |
| 1 | nc id | nvarchar | Primary key | Stores nc id |
| 2 | complainant id | nvarchar | Foreign key | Stores complainant id |
| 3 | officer id | nvarchar | Foreign key | Stores officer id |
| 4 | date | nvarchar | Not null | Stores date of recording |
| 5 | district | nvarchar | Not null | Stores district name |
| 6 | station name | nvarchar | Not null | Stores station name |
| 7 | incident date | datetime | Not null | Stores incident date |
| 8 | incident location | nvarchar | Not null | Stores incident location |
| 9 | accused | nvarchar | Not null | Stores names of accused |
| 10 | offence type | nvarchar | Not null | Stores offence type |
| 11 | property | nvarchar | Not null | Stores property damage details |
| 12 | offence details | nvarchar | Not null | Stores offence details |

**Table 5.6.8.7**

**CHATPER 6 – SYSTEM PLANNING**

* 1. **Gantt Chart:**

A Gantt chart is a type of bar chart that illustrates a project schedule. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements comprise the work breakdown structure of the project. Some Gantt charts also show the dependency (i.e., precedence network) relationships between activities. Gantt charts can be used to show current schedule status using percent-complete shadings. In the Gantt chart we show the time spent for each phase of the software development. Gantt charts is a project-planning tool that can be used to represent the timing of tasks required to complete a project. Since Gantt charts are simple to understand and easy to construct, they are used by most project managers for all but the most complex projects. In a Gantt chart, each task takes up one row. Dates run along the top in increments of days, weeks or months, depending on the total length of the project. The expected time for each task is represented by a horizontal bar whose left end marks the expected beginning of the task and whose right end marks the expected completion date. Tasks may run sequentially, in parallel or overlapping.

* + 1. **Project Scheduling:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task name** | **Duration** | **Start** | **Finish** |
| Project overview document | 5 days | 15/11/2013 | 20/11/2013 |
| Initial program analysis | 10 days | 20/11/2013 | 30/11/2013 |
| Software requirement specification | 10 days | 30/11/2013 | 10/12/2013 |
| Implementation plan | 6 days | 10/12/2013 | 16/12/2013 |
| Formal requirement specification | 10 days | 16/12/2013 | 26/12/2013 |
| Database design | 6 days | 26/12/2013 | 1/01/2014 |
| Form design | 9 days | 1/01/2014 | 10/01/2014 |
| Design documentation | 11 days | 10/01/2014 | 21/01/2014 |
| Implementing the project | 46 days | 21/01/2014 | 8/03/2014 |
| Testing | 9 days | 8/03/2014 | 17/03/201 |
| Documentation | 10 days | 17/03/2014 | 27/03/2014 |

* + 1. **Gantt Chart:**

****

**CHATPER 7 – SYSTEM**

**IMPLEMENTATION**

* 1. **Methodology adopted:**

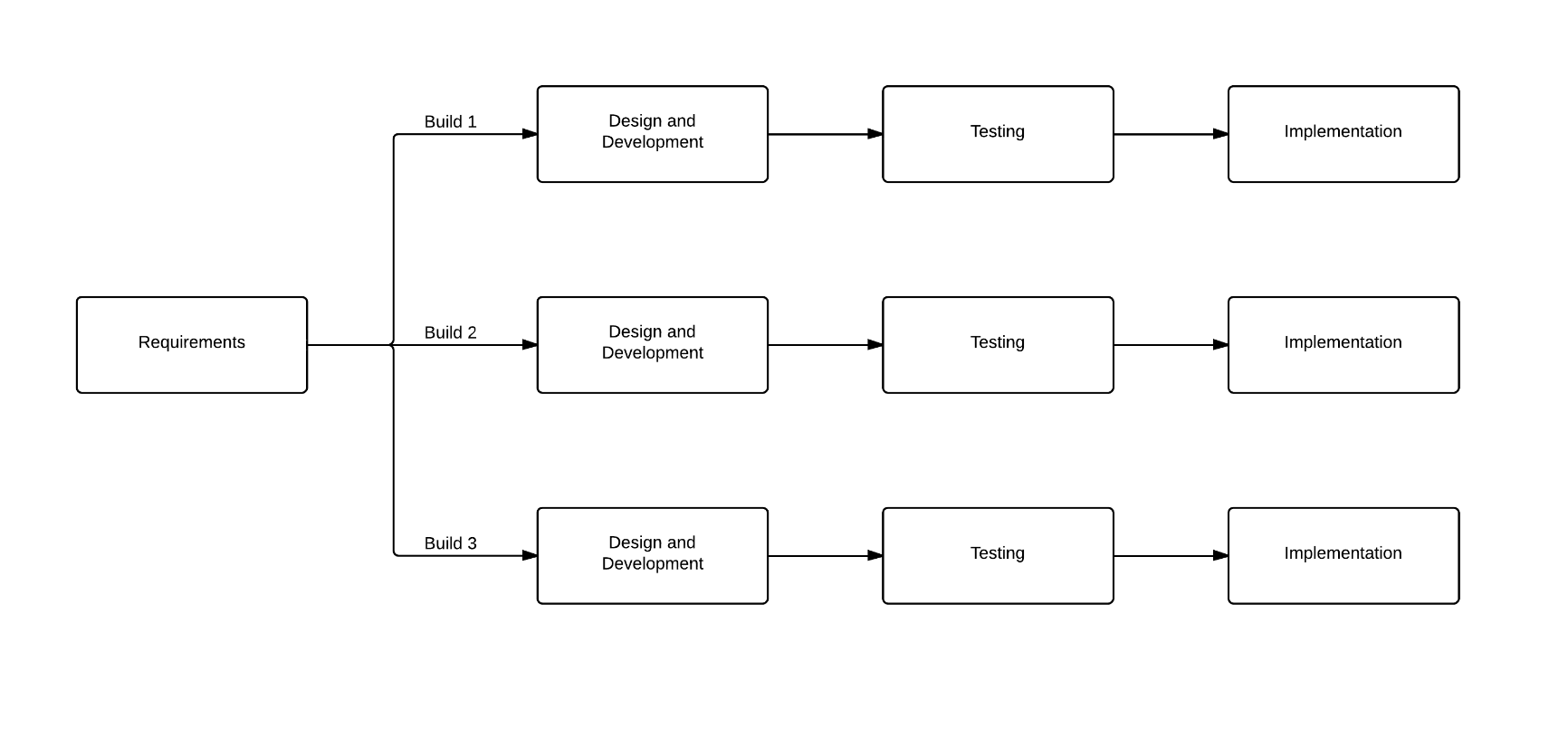
The methodology adopted while developing the system is the Incremental Model. It is a method of [software development](http://en.wikipedia.org/wiki/Software_development) where the model is [designed](http://en.wikipedia.org/wiki/Software_design), implemented and [tested](http://en.wikipedia.org/wiki/Software_testing) incrementally (a little more is added each time) until the product is finished. It involves both development and maintenance. The product is defined as finished when it satisfies all of its requirements.

The product is decomposed into a number of components, each of which is designed and built separately (termed as builds). Each component is delivered to the client when it is complete. This allows partial utilization of the product and avoids a long development time. It also avoids a large initial capital outlay and subsequent long waiting period. This model of development also helps ease the traumatic effect of introducing a completely new system all at once.

The incremental Model is an evolution of the waterfall model, where the waterfall model is incrementally applied.

The series of releases is referred to as “increments”, with each increment providing more functionality to the customers. After the first increment, a core product is delivered, which can already be used by the customer. Based on customer feedback, a plan is developed for the next increments, and modifications are made accordingly. This process continues, with increments being delivered until the complete product is delivered. The incremental philosophy is also used in the agile process model.

**Fig. 7.1.1 (Incremental Model)**

Advantages of this model are:

1. After each iteration, regression testing should be conducted. During this testing, faulty elements of the software can be quickly identified because few changes are made within any single iteration.
2. It is generally easier to test and debug than other methods of software development because relatively smaller changes are made during each iteration. This allows for more targeted and rigorous testing of each element within the overall product.
3. Initial product delivery is faster and costs lower.

Disadvantages of this model are:

1. Resulting cost may exceed the cost of the organization.
2. As additional functionality is added to the product, problems may arise related to system architecture which were not evident in earlier prototypes.

Tasks involved:

1. Communication: helps to understand the objective.
2. Planning: required as many people (software teams) work on the same project but different function at same time.
3. Modeling: involves business modeling, data modeling, and process modeling.
4. Construction: this involves the reuse software components and automatic code.
5. Deployment: integration of all the increments.

**7.2 User Manual:**

**A. For Admin**

Adding data:

Firstly Go to Admin Panel 🡪 Click on the appropriate menu 🡪 Add Record 🡪 Add Complainant, FIR, Non-Cognizable complaint, Suspect, Case, Case Disposal, and Officer.

Searching data:

Go to Admin Panel🡪 Click on the appropriate menu🡪 All Records🡪 Search Complainant, FIR, Non-Cognizable, Suspect, Case, Case Disposal, and Officer.

Editing data:

Go to Admin Panel🡪 Click on the tab menu🡪 Go to the appropriate tab page🡪 Click the refresh button🡪 Double-click a row in the data grid view🡪 Make necessary changes🡪 Click on update button🡪 Update Suspect, Case, and Officer.

**B. For Officer**

Adding data:

Firstly Go to Officer Panel 🡪 Click on the appropriate menu 🡪 Add Record 🡪 Add Complainant, FIR, Non-Cognizable complaint, Suspect, Case, and Case Disposal.

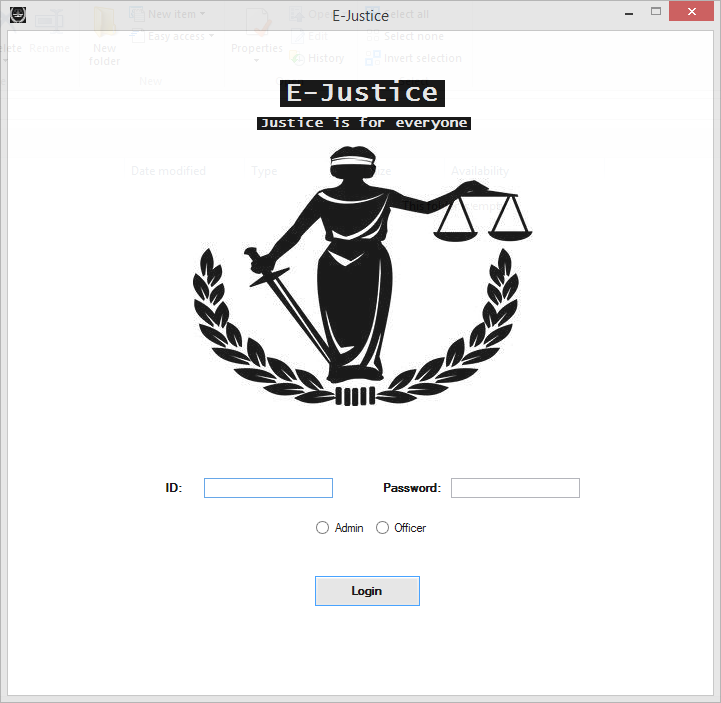
Searching data:

Go to Officer Panel🡪 Click on the appropriate menu🡪 All Records🡪 Search Complainant, FIR, Non-Cognizable, Suspect, Case, and Case Disposal.

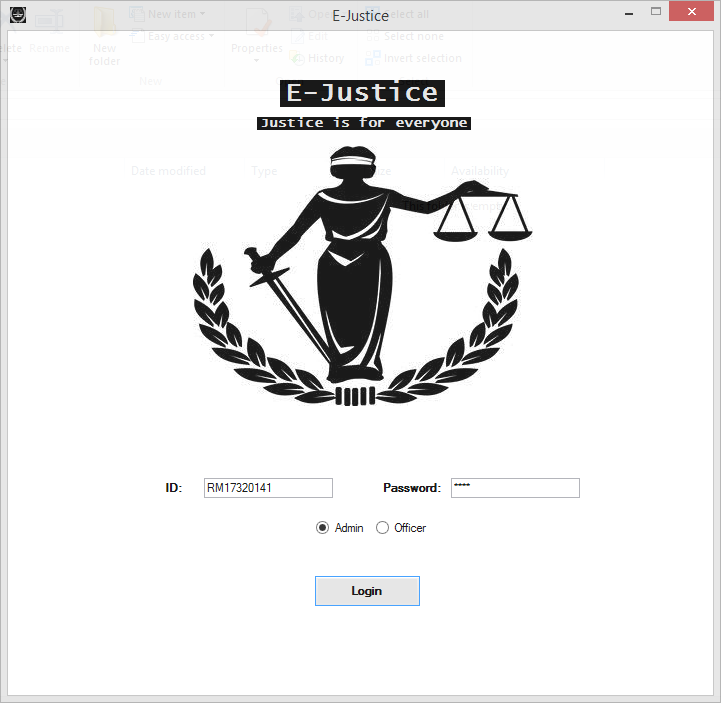
Editing data:

Go to Admin Panel🡪 Click on the tab menu🡪 Go to the appropriate tab page🡪 Click the refresh button🡪 Double-click a row in the data grid view🡪 Make necessary changes🡪 Click on update button🡪 Update Suspect, Case, and Officer.

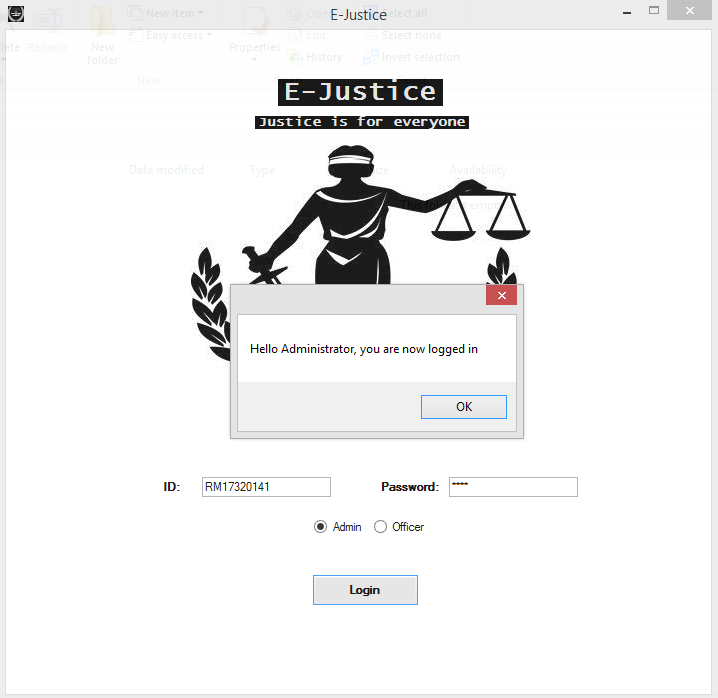
**CHAPTER 8 – SCREENSHOTS**

****

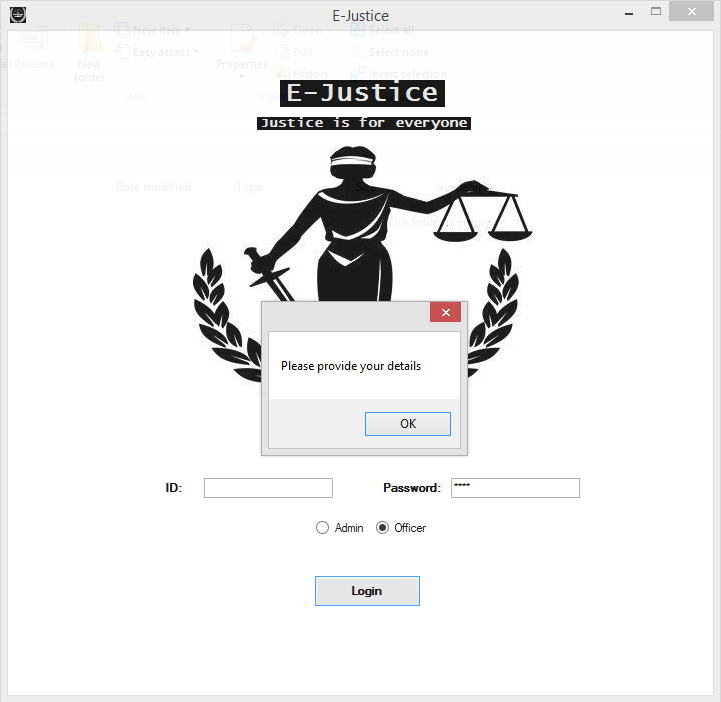
**Fig. 8.1 Login screen**

****

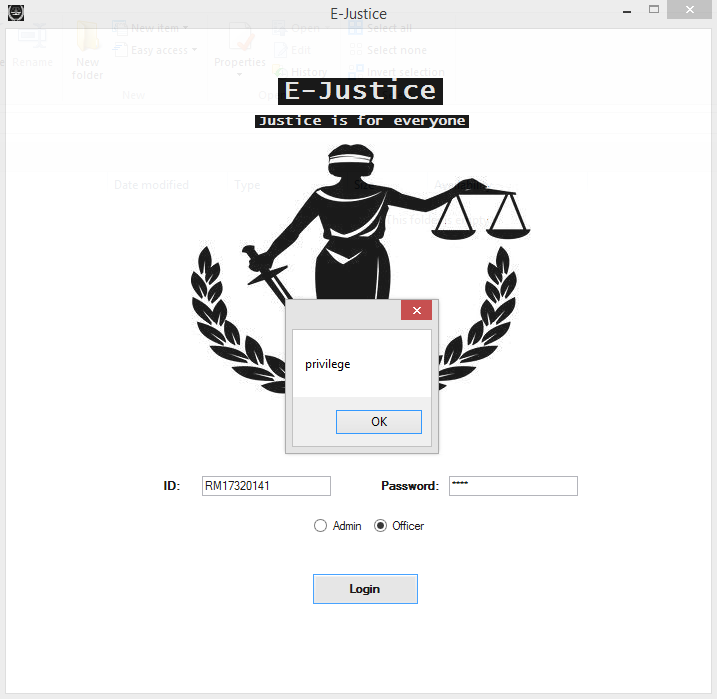
**Fig. 8.2 Login screen with details**

****

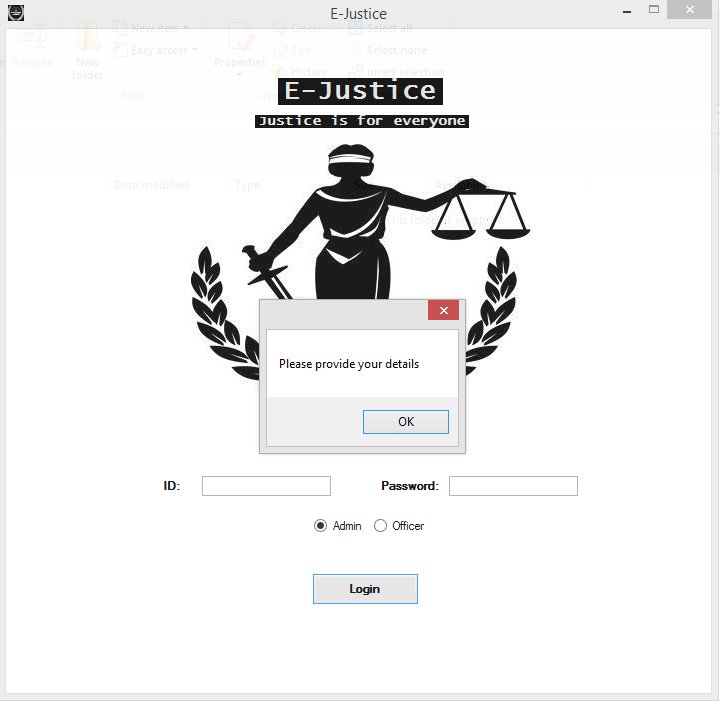
**Fig. 8.3 Message box showing correct login for Administrator**

****

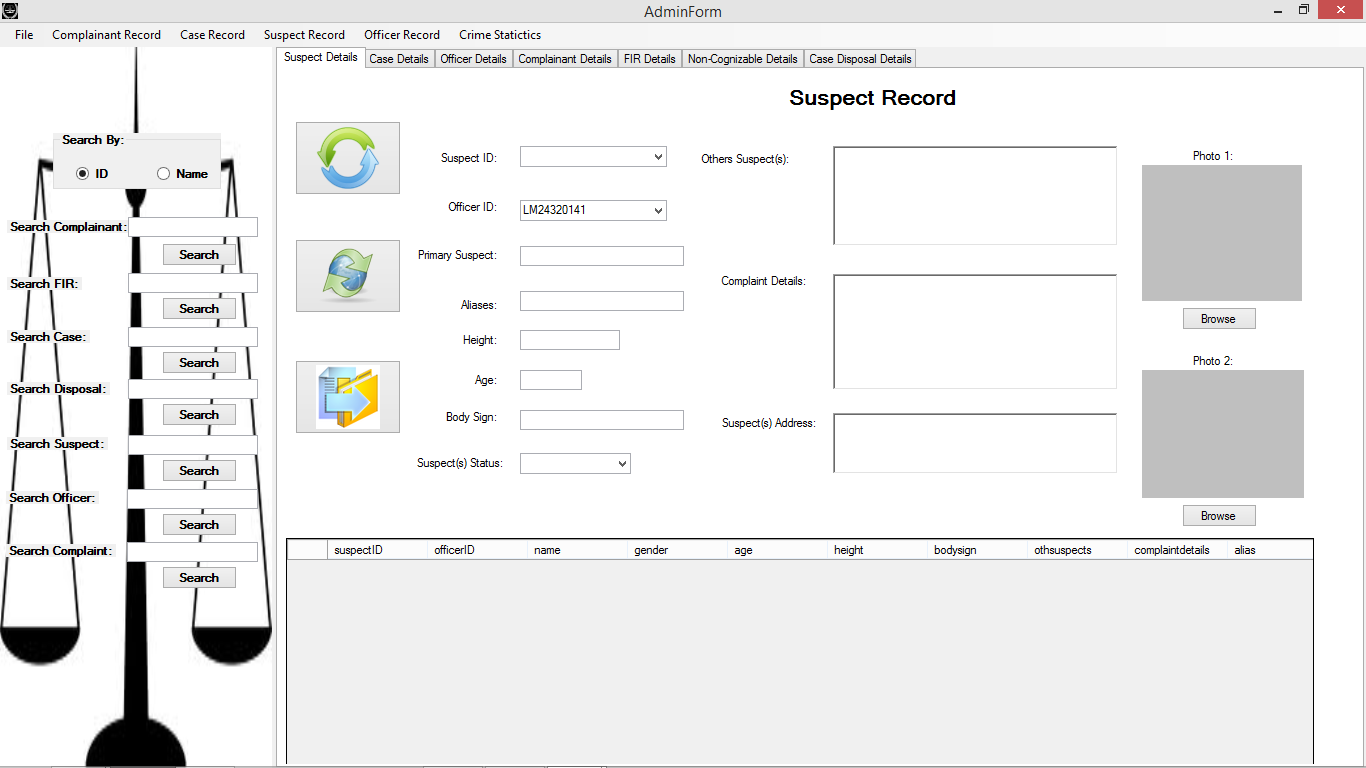
**Fig. 8.4 Message box indicating incomplete details**

****

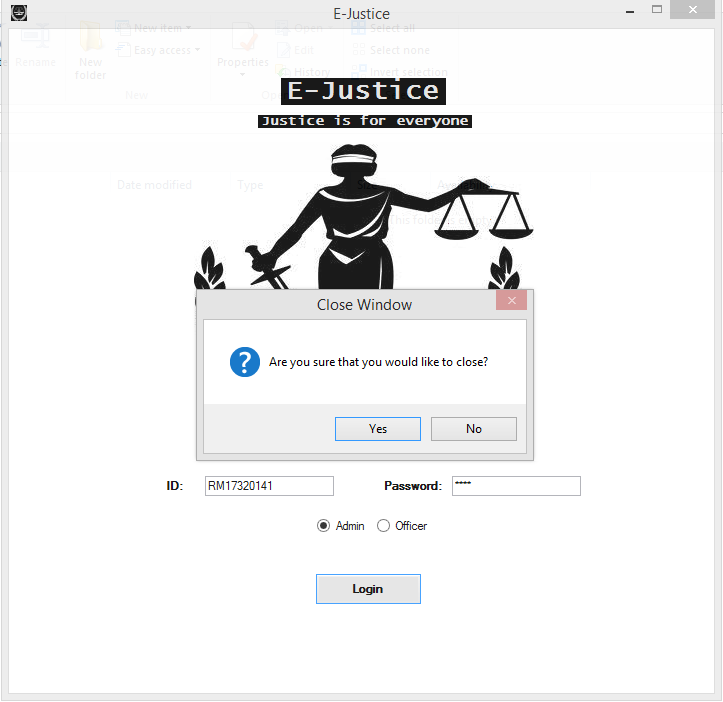
**Fig. 8.5 Message box showing incorrect privilege selection for Admin details**

****

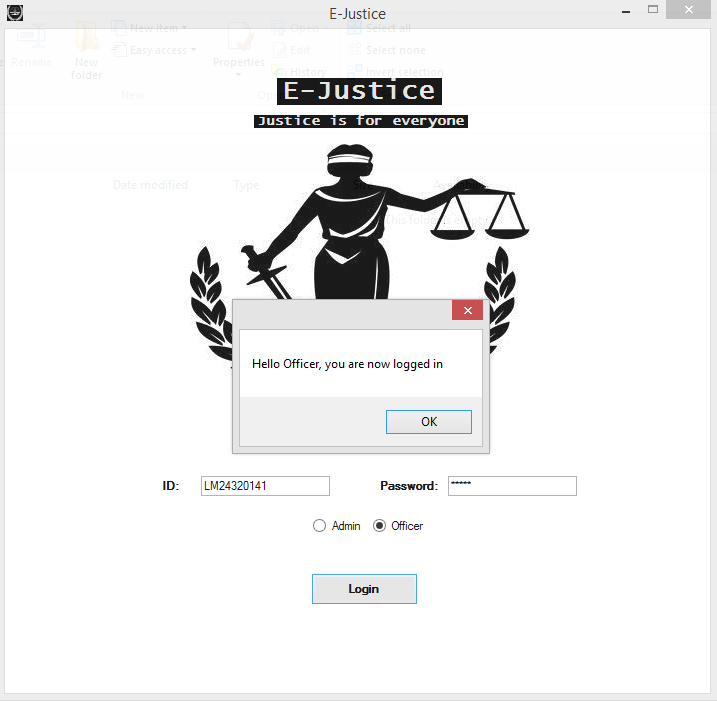
**Fig. 8.6 Message box providing incomplete details (Admin)**

****

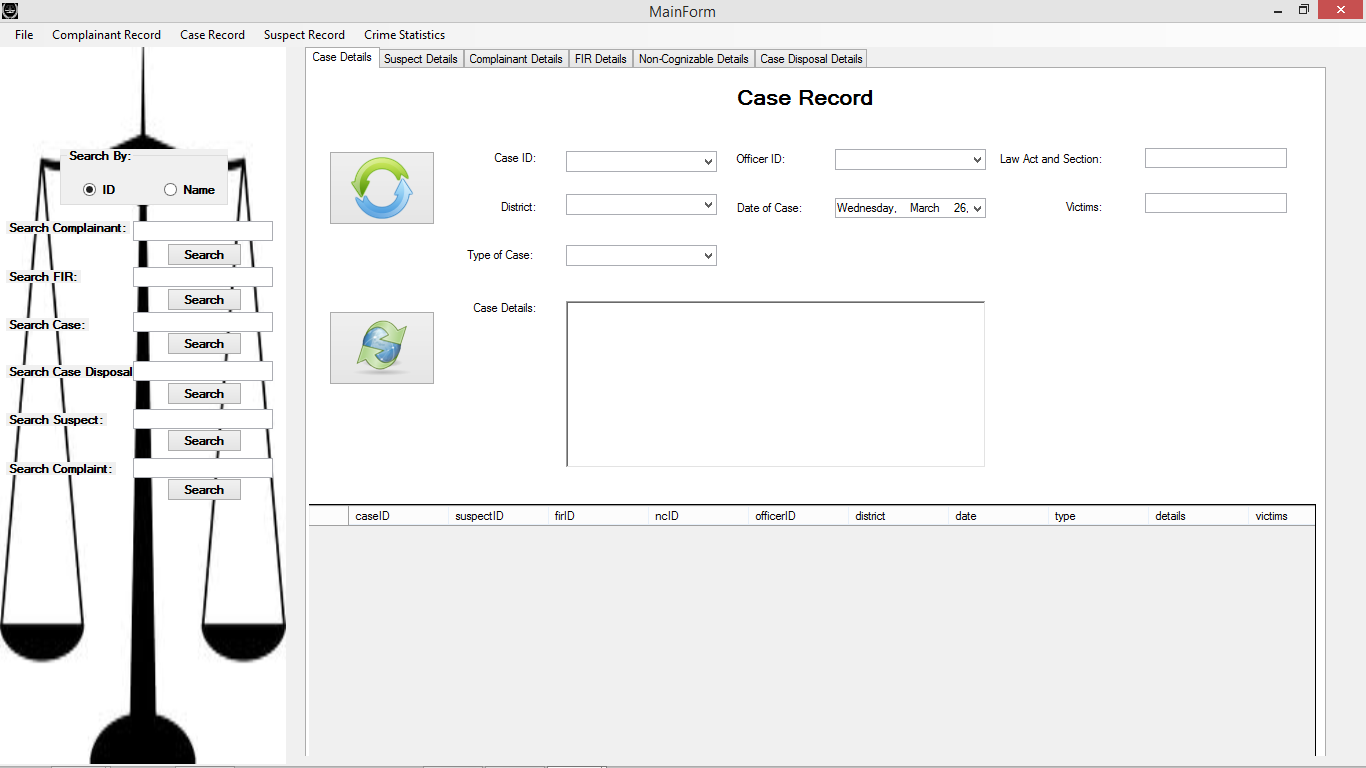
**Fig. 8.7 Show Admin form after login**

****

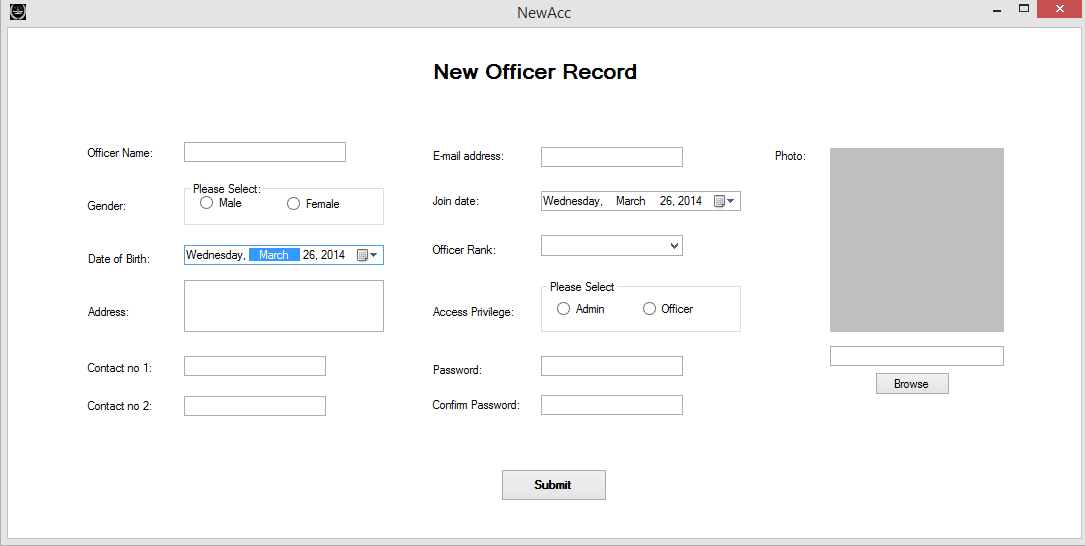
**Fig. 8.8 A verify close message**

****

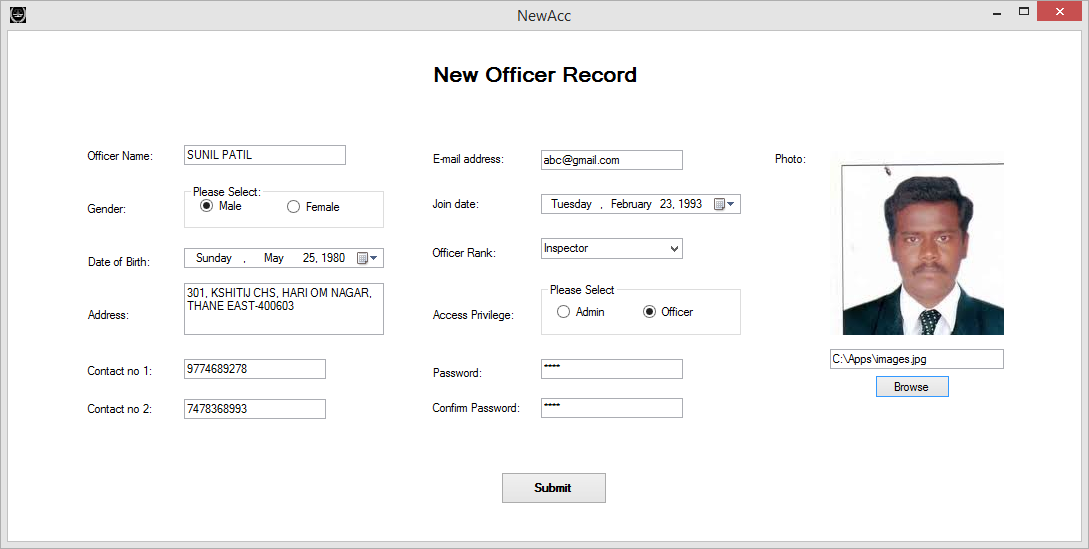
**Fig. 8.9 Message box showing correct login for Officer**

****

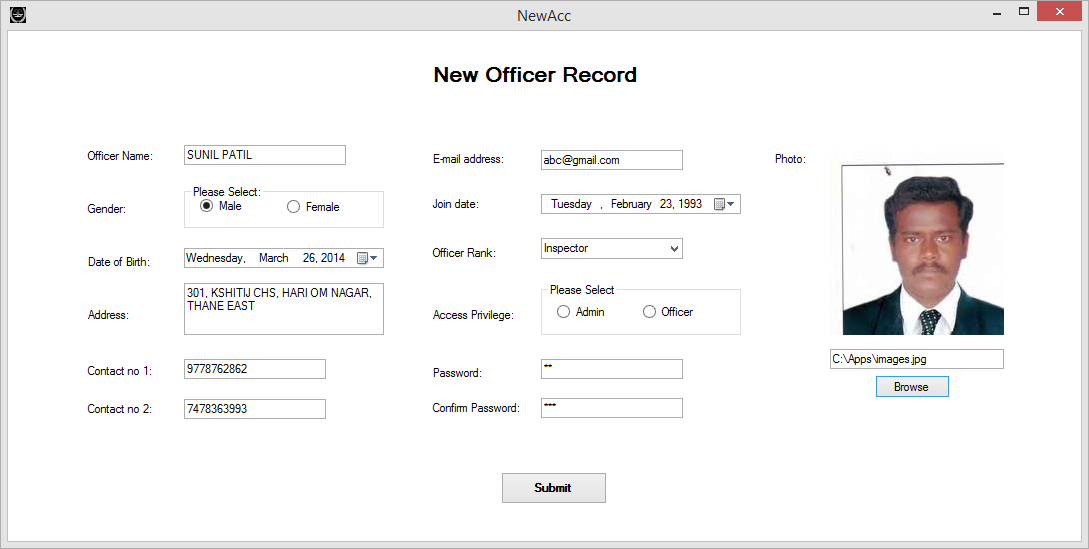
**Fig. 8.10 Show main form after Login**

****

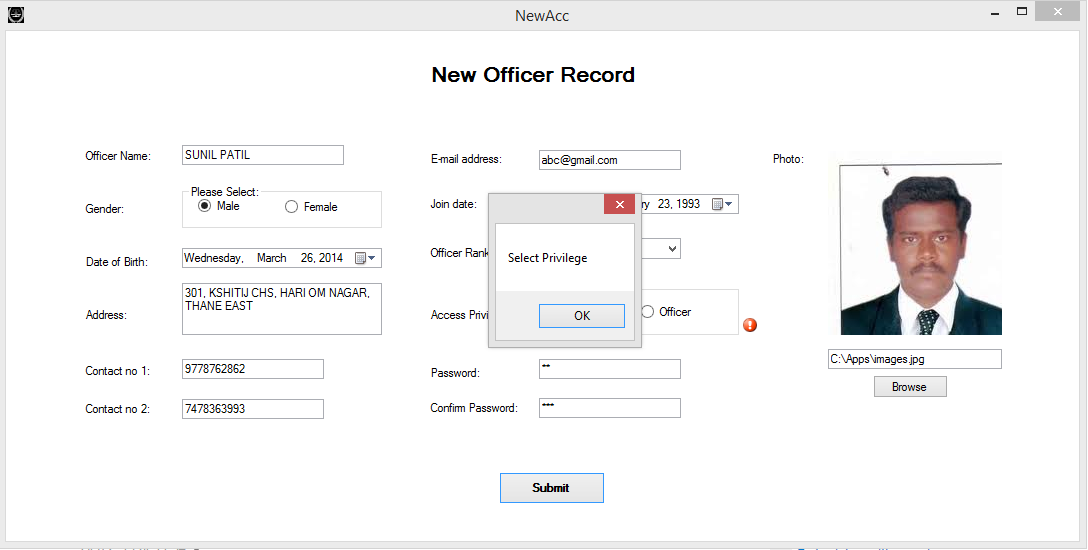
**Fig. 8.11 New Account for Officer:**

****

**Fig. 8.12 Adding data for Officer**

****

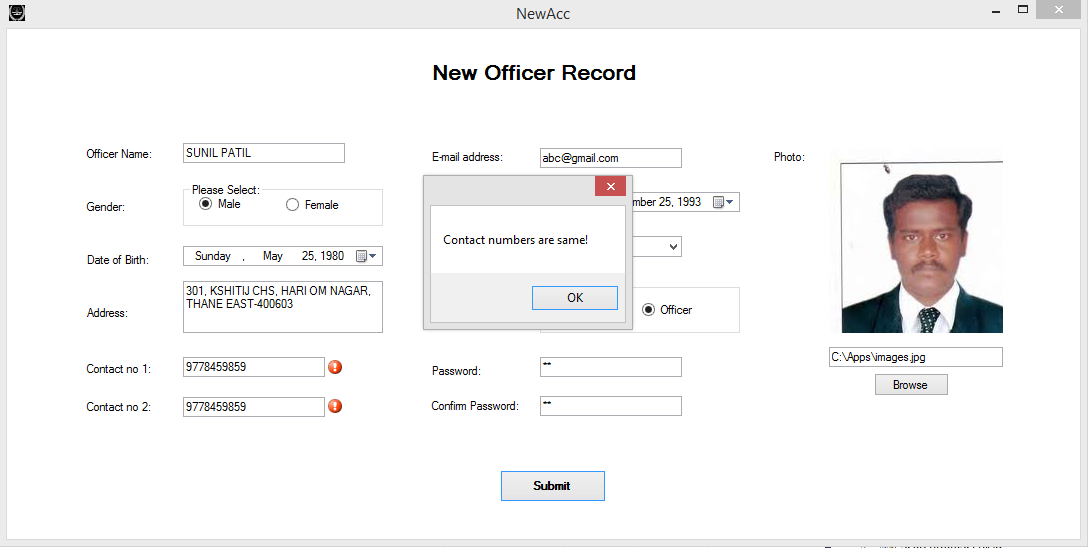
**Fig. 8.13 For providing incorrect information:**

****

**Fig. 8.14 Display of Error message for privilege**

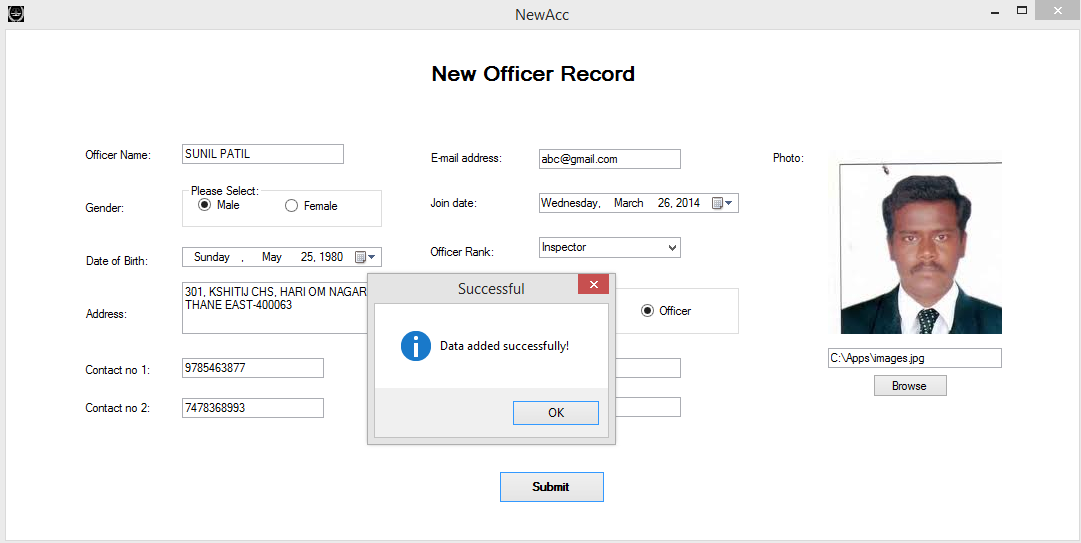
****

**Fig. 8.15 Display of error message for incorrect password**

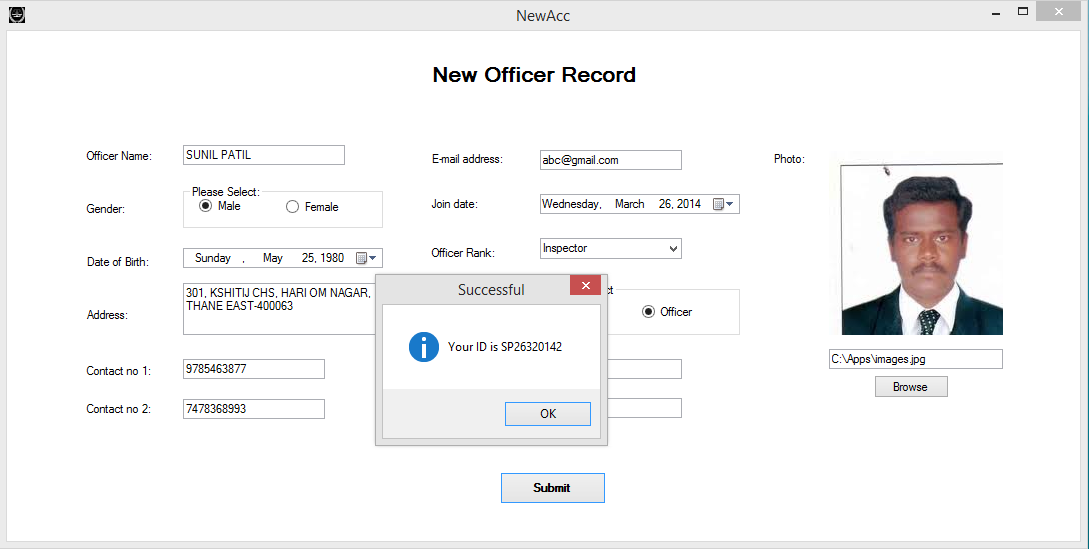
****

**Fig. 8.16 Display of error message for**

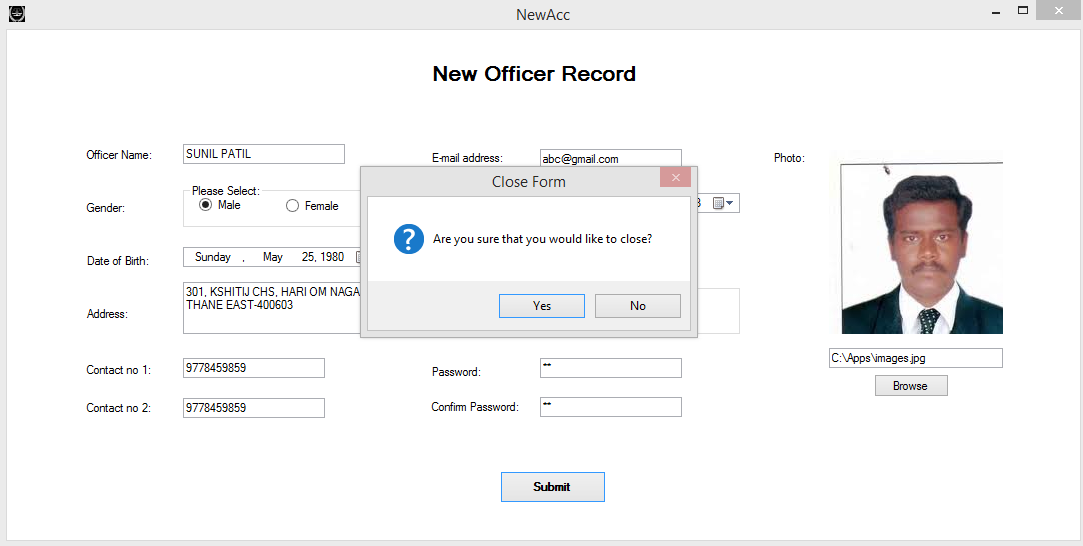
**providing 2 same contact numbers:**

****

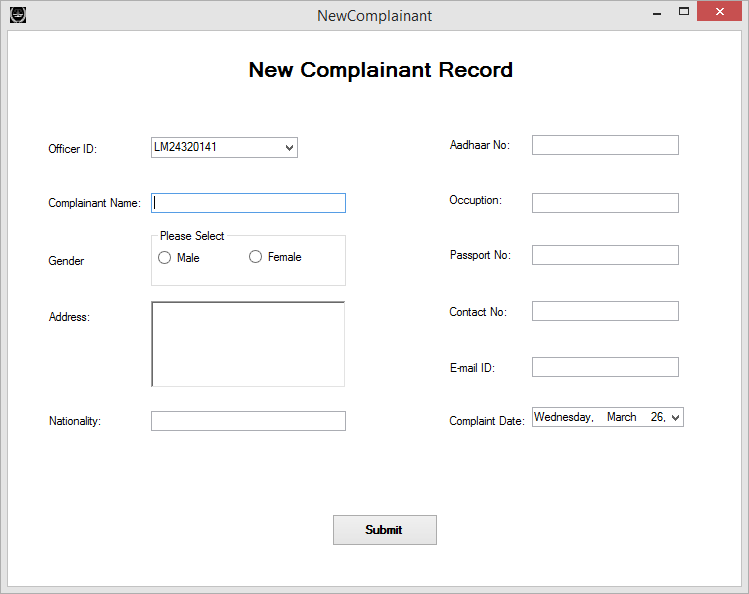
**Fig. 8.17 Confirmation message**

****

**Fig. 8.18 Generation and display of officer’s id**

****

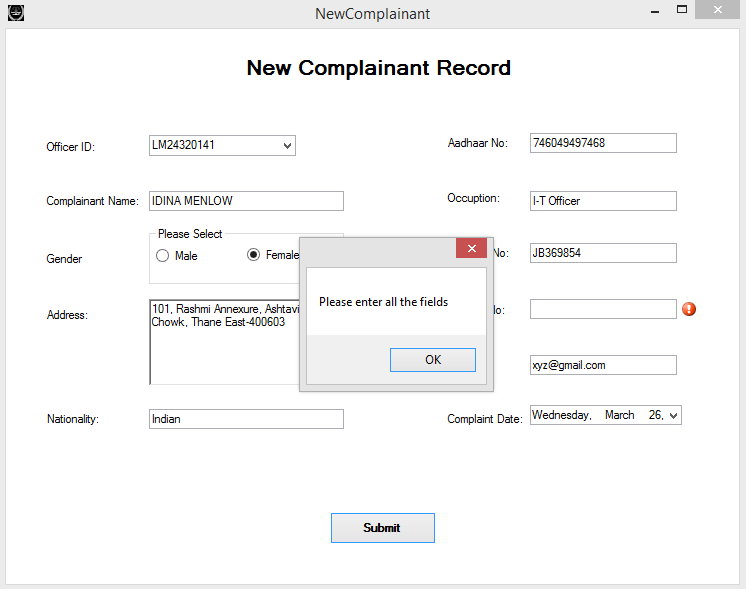
**Fig. 8.19 A verify close message**

****

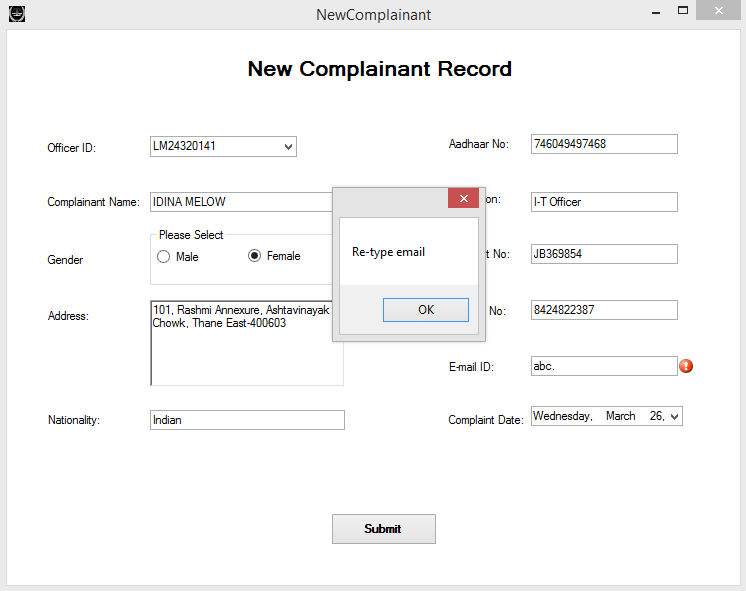
**Fig. 8.20 New Complainant**

****

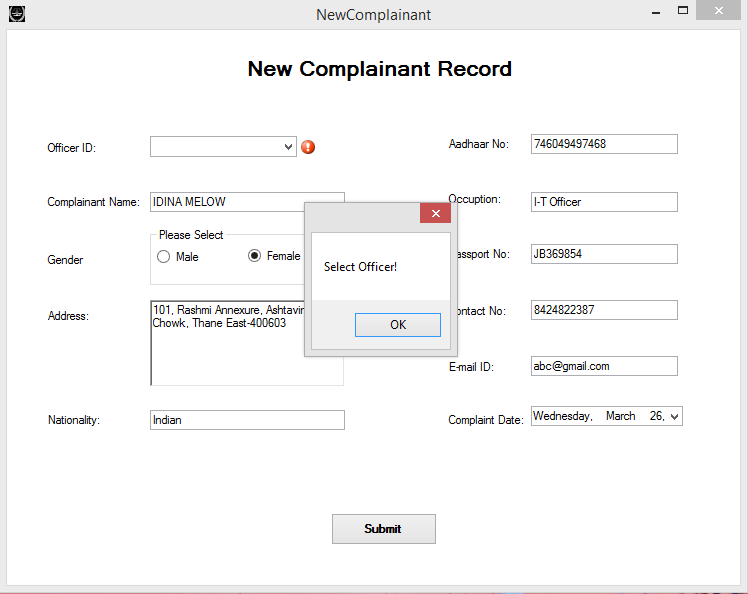
**Fig. 8.21 Providing correct information**

****

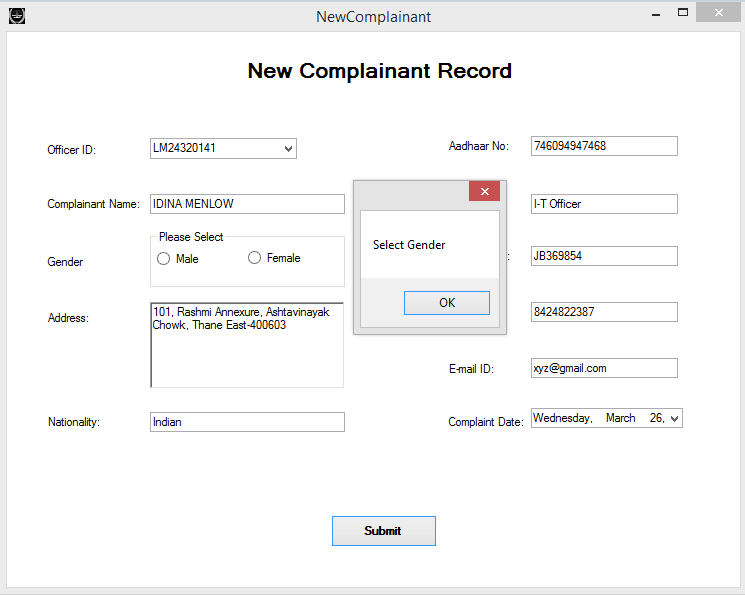
**Fig. 8.22 Providing incomplete information**

****

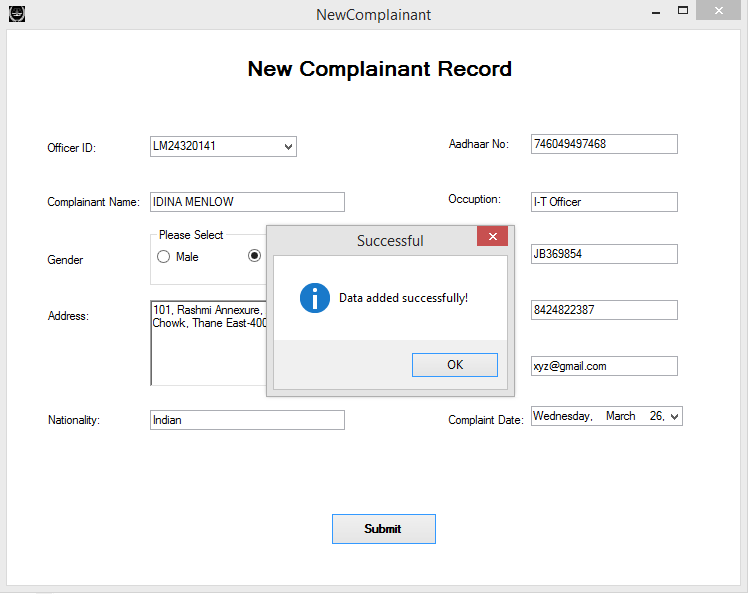
**Fig. 8.23 Incorrect Email format**

****

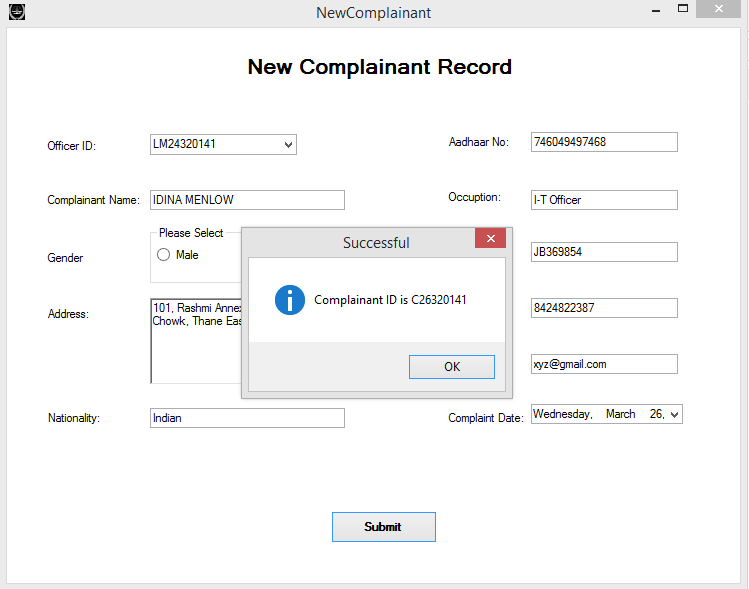
**Fig. 8.23 Non-selection of Officer**

****

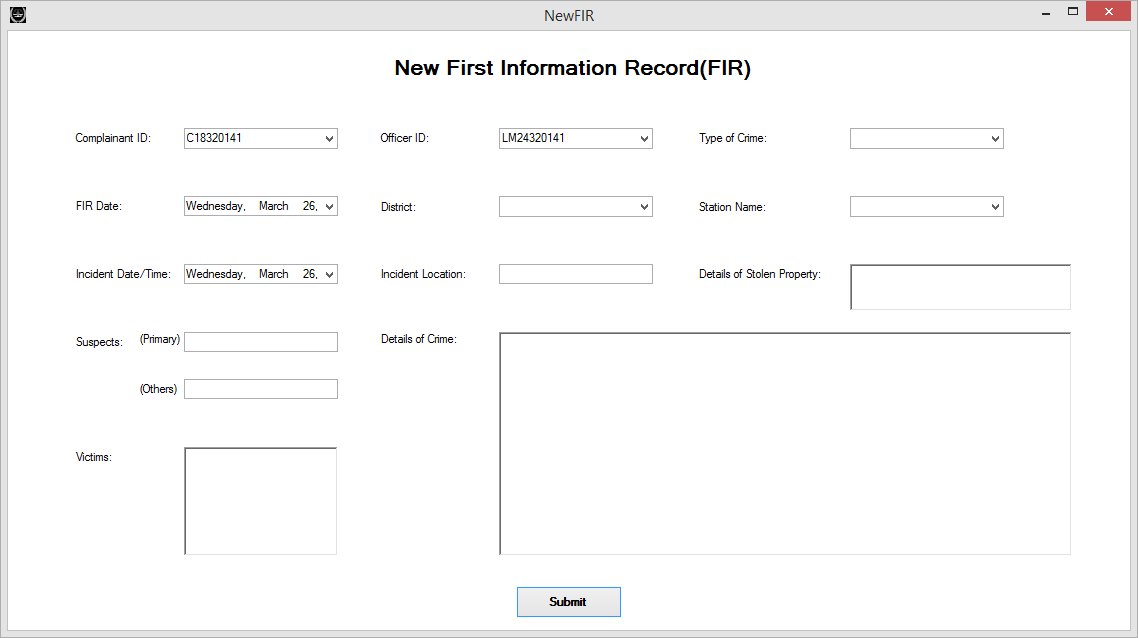
**Fig. 8.24 Non-selection of Gender**

****

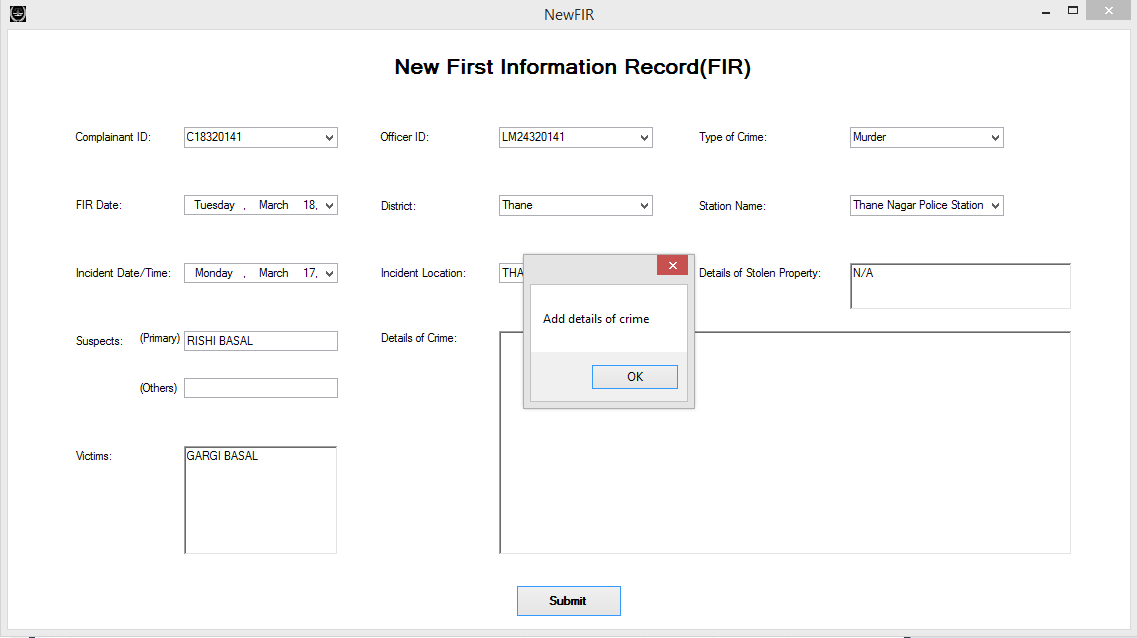
**Fig. 8.25 Confirmation message**

****

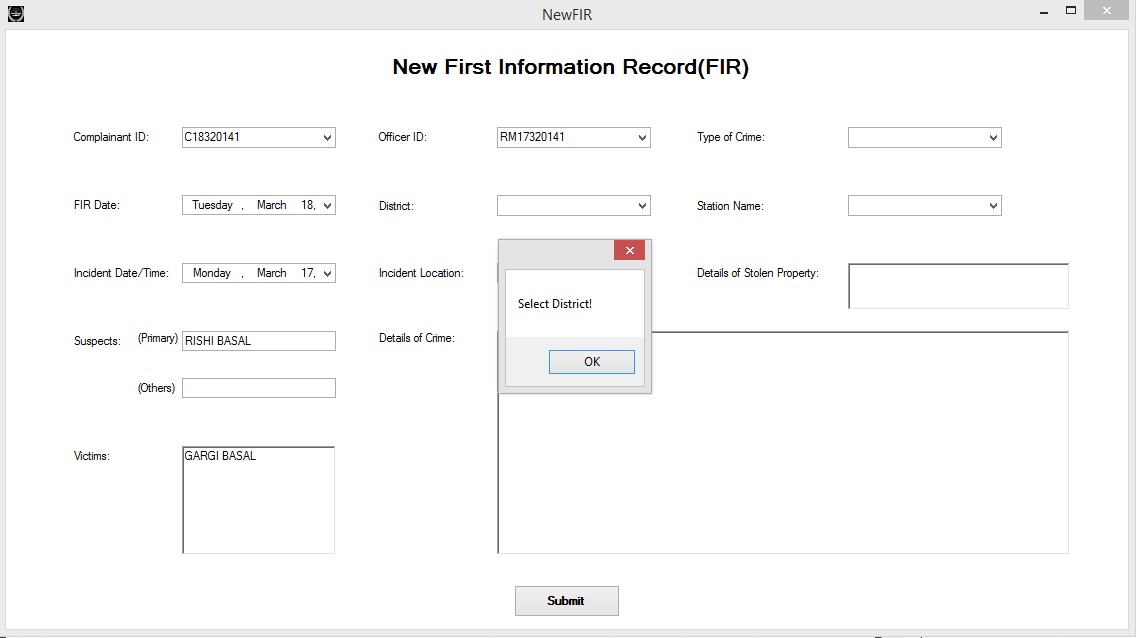
**Fig. 8.26 Generation and display of Complainant’s id**

****

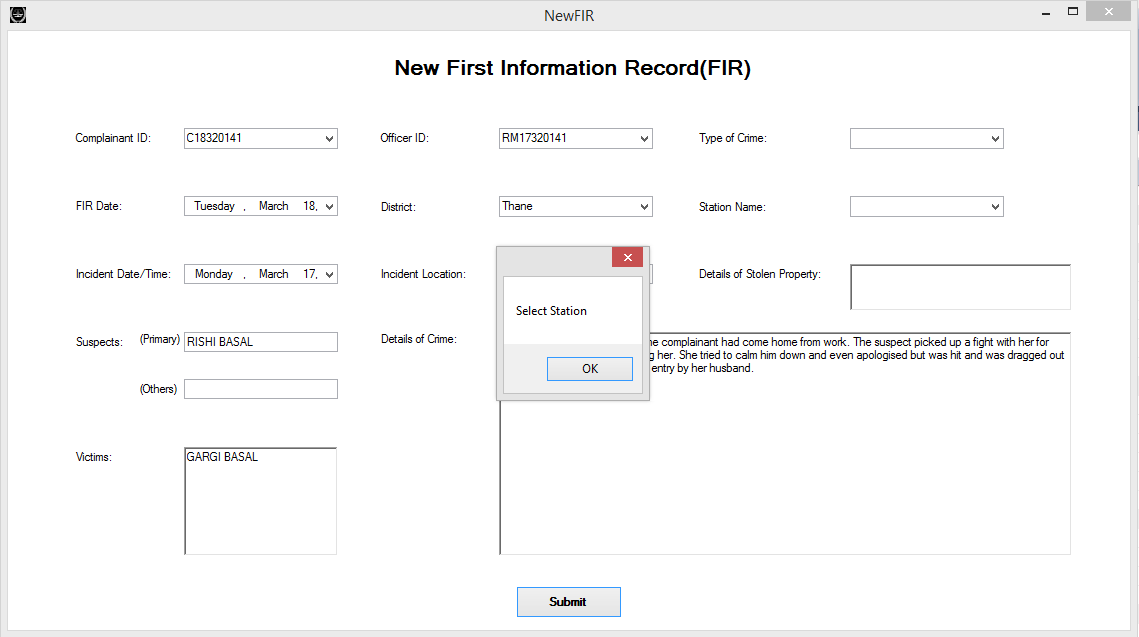
**Fig. 8.27 New FIR**

****

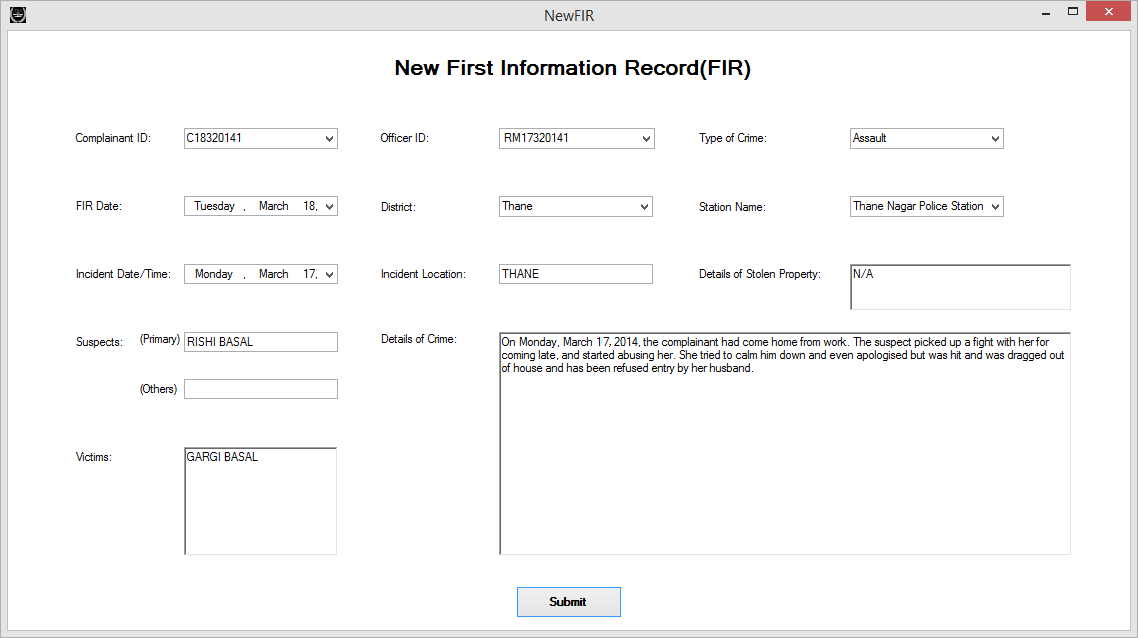
**Fig. 8.27 Details of crime not added**

****

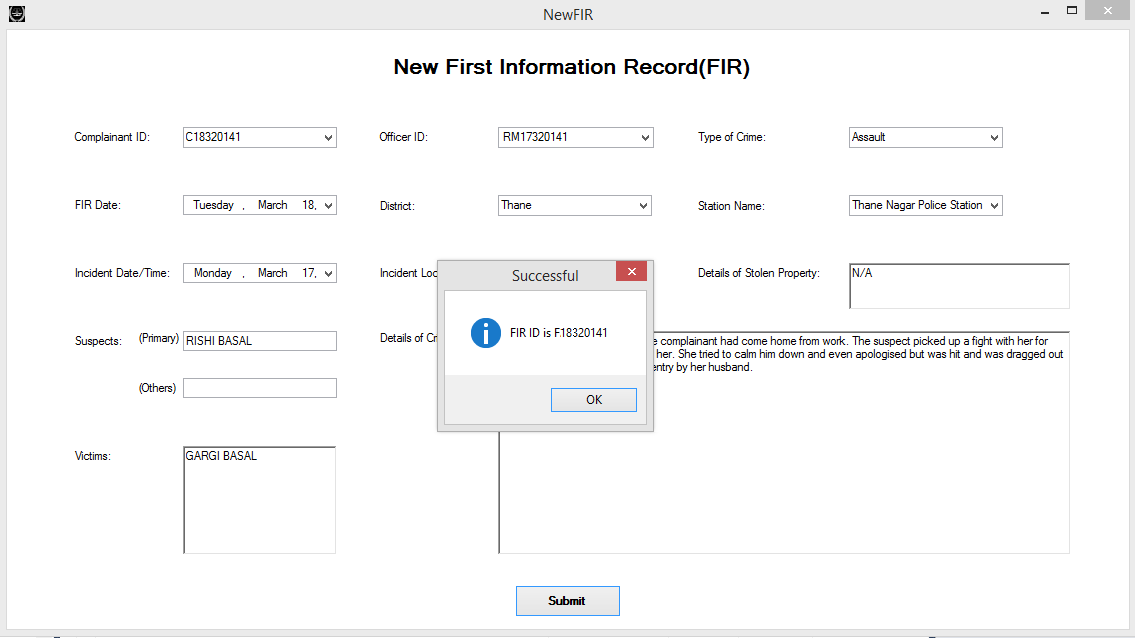
**Fig. 8.28 Select district**

****

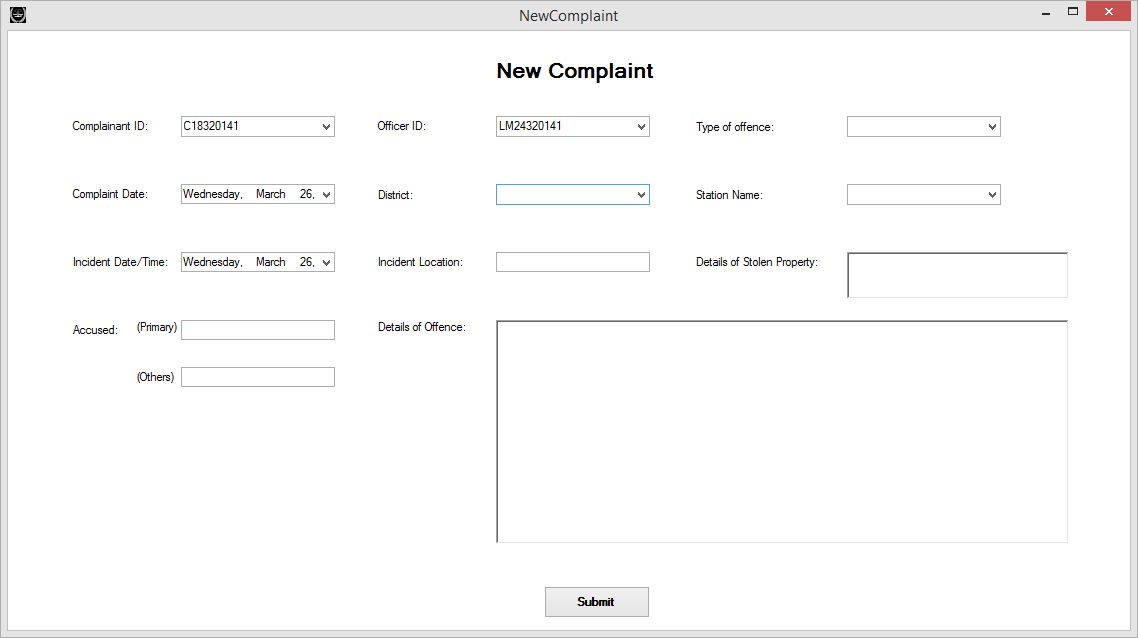
**Fig. 8.29 Select station**



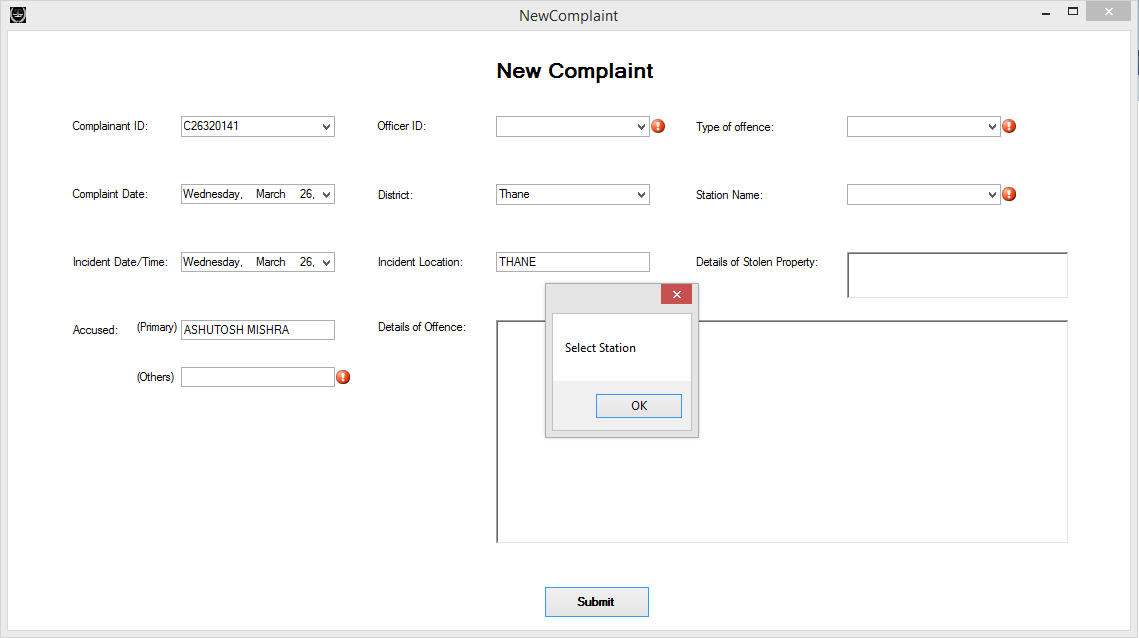
**Fig. 8.30 Correct details filled**

****

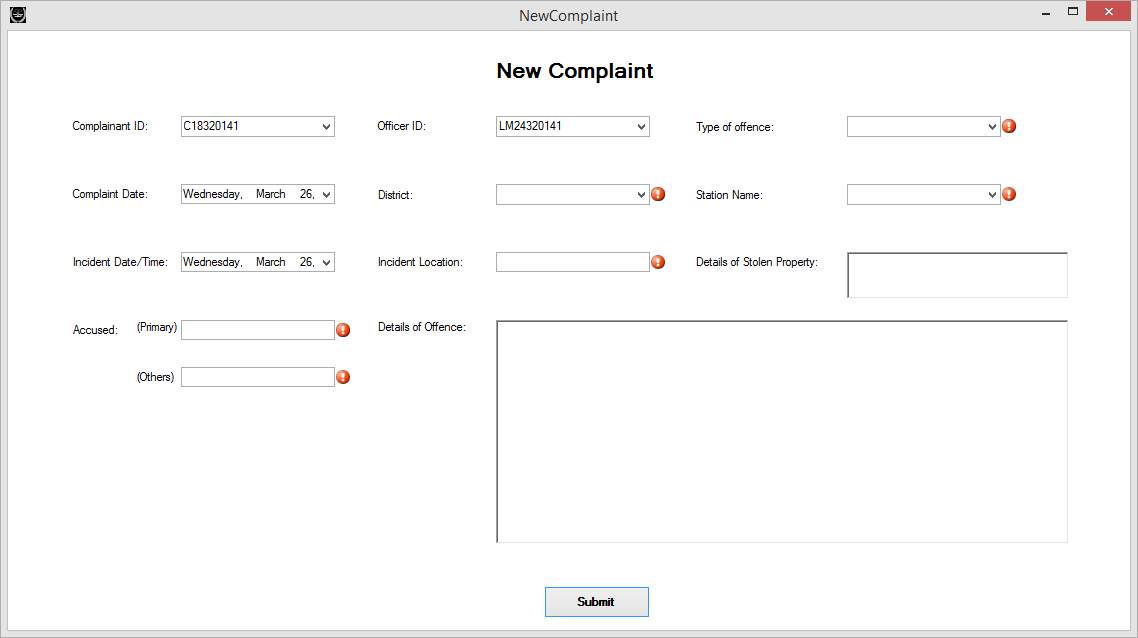
**Fig. 8.31 Display of FIR id**

****

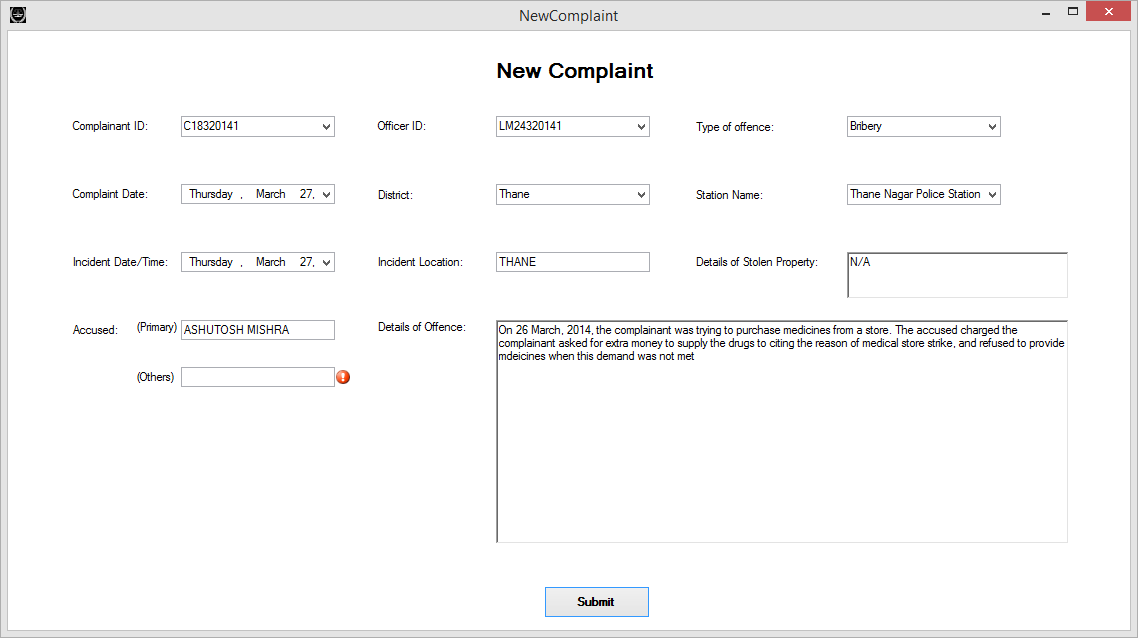
**Fig. 8.32 New Non-Cognizable complaint**

****

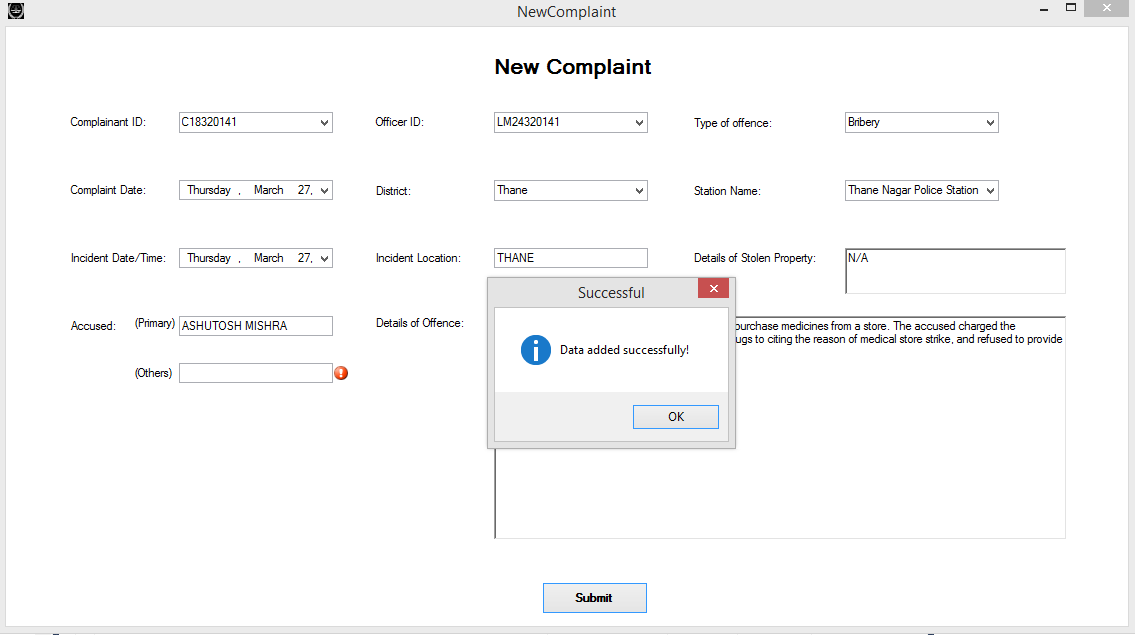
**Fig. 8.33 Select station**

****

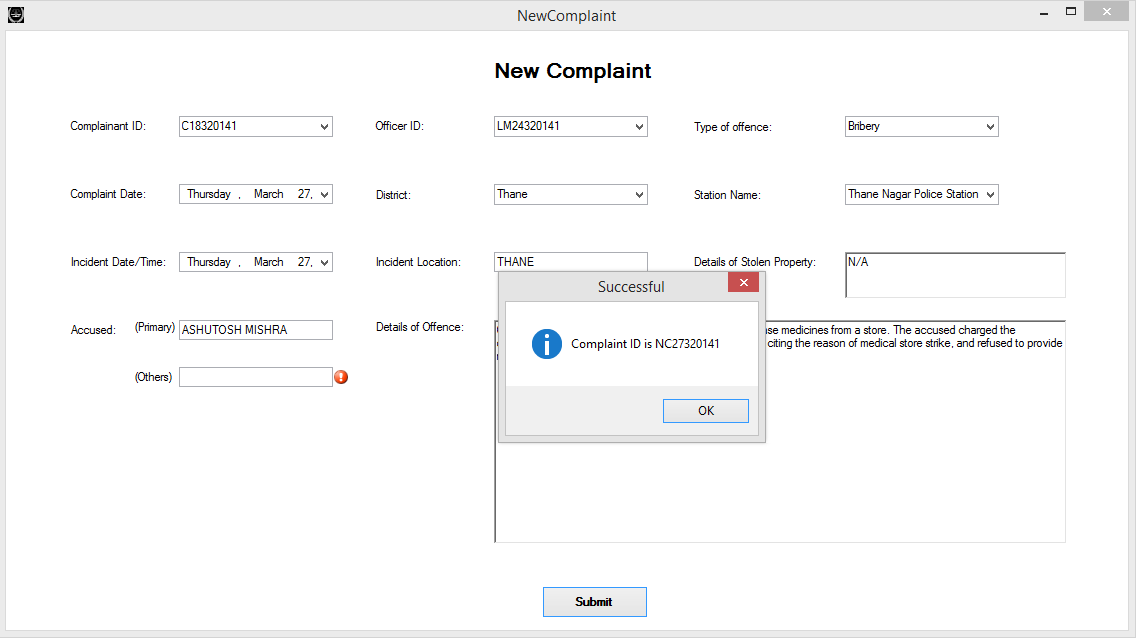
**Fig. 8.34 Empty field errors**

****

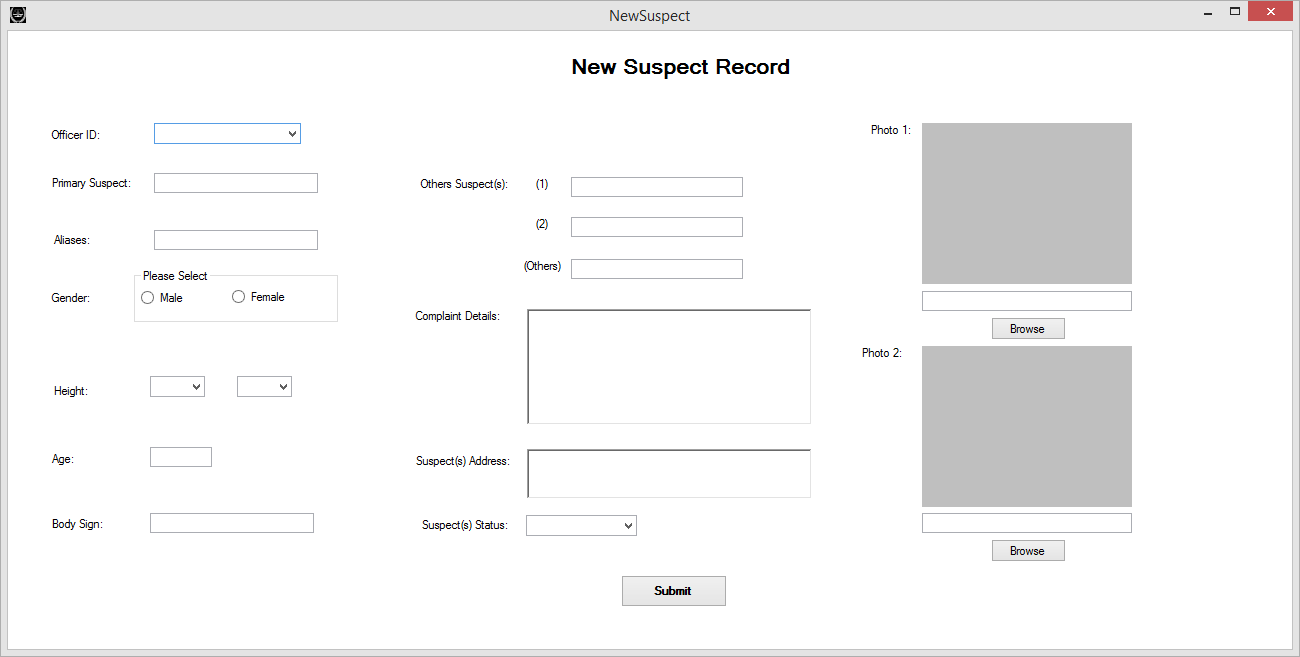
**Fig. 8.35 Correct details filled**

****

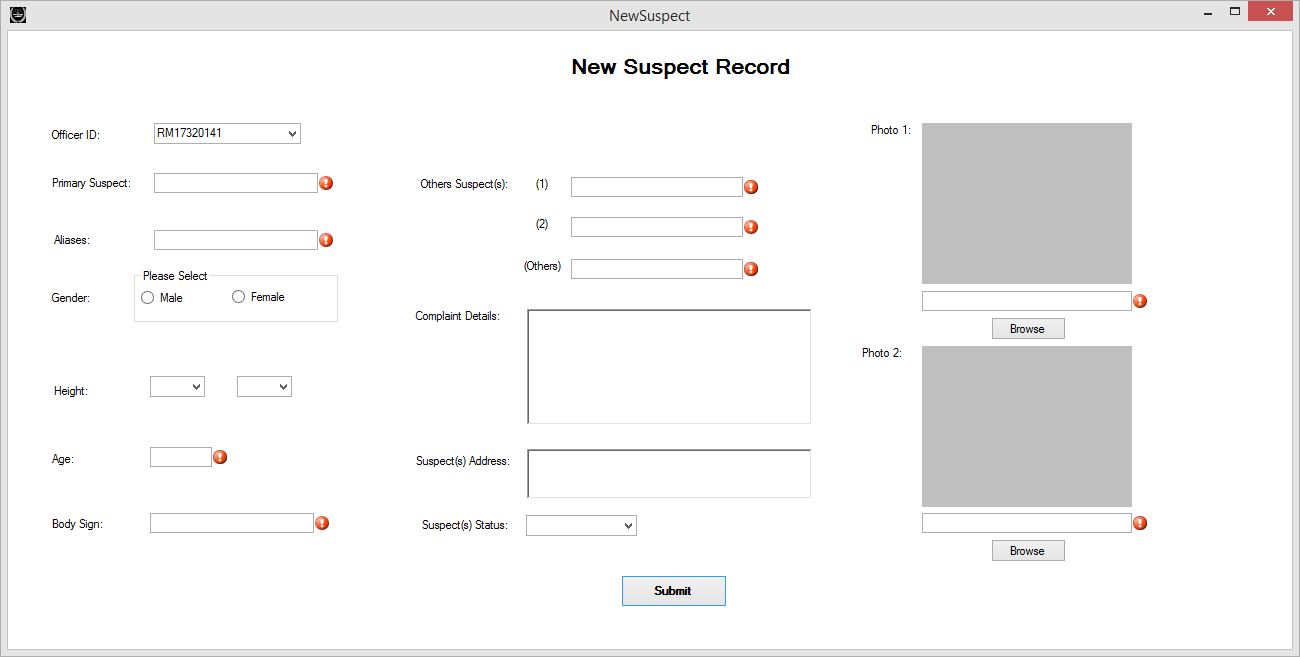
**Fig. 8.36 Data added successfully**

****

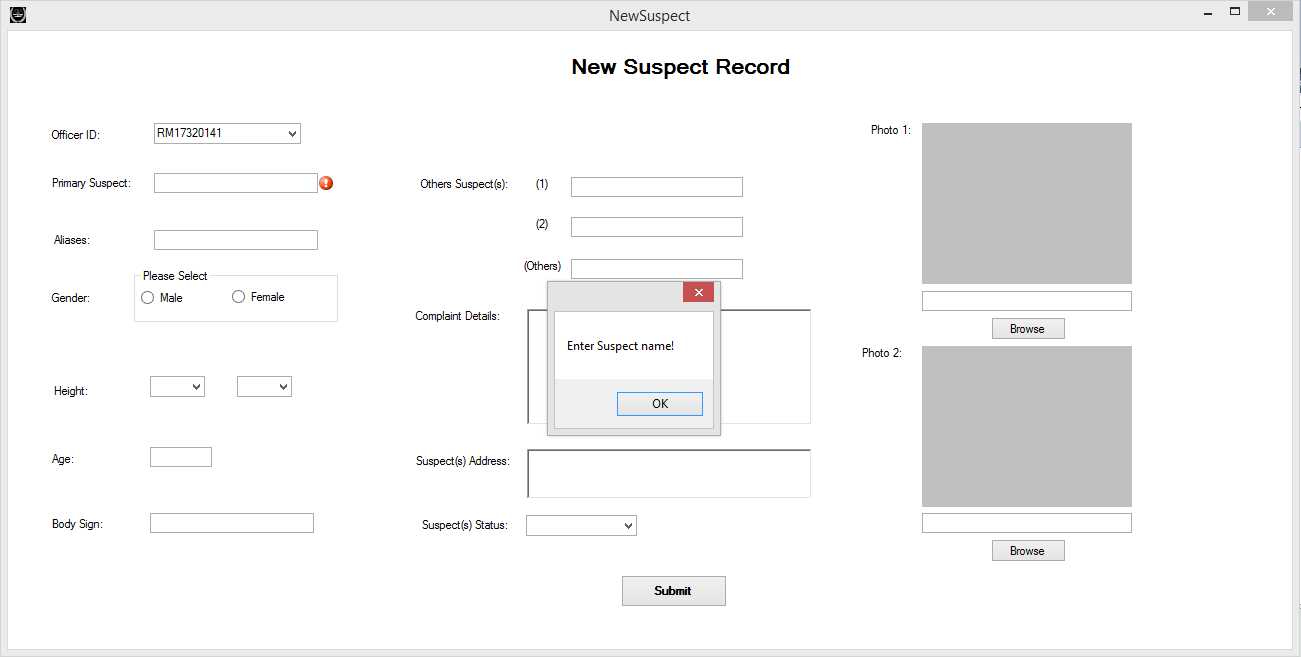
**Fig. 8.37 Display complaint id**

****

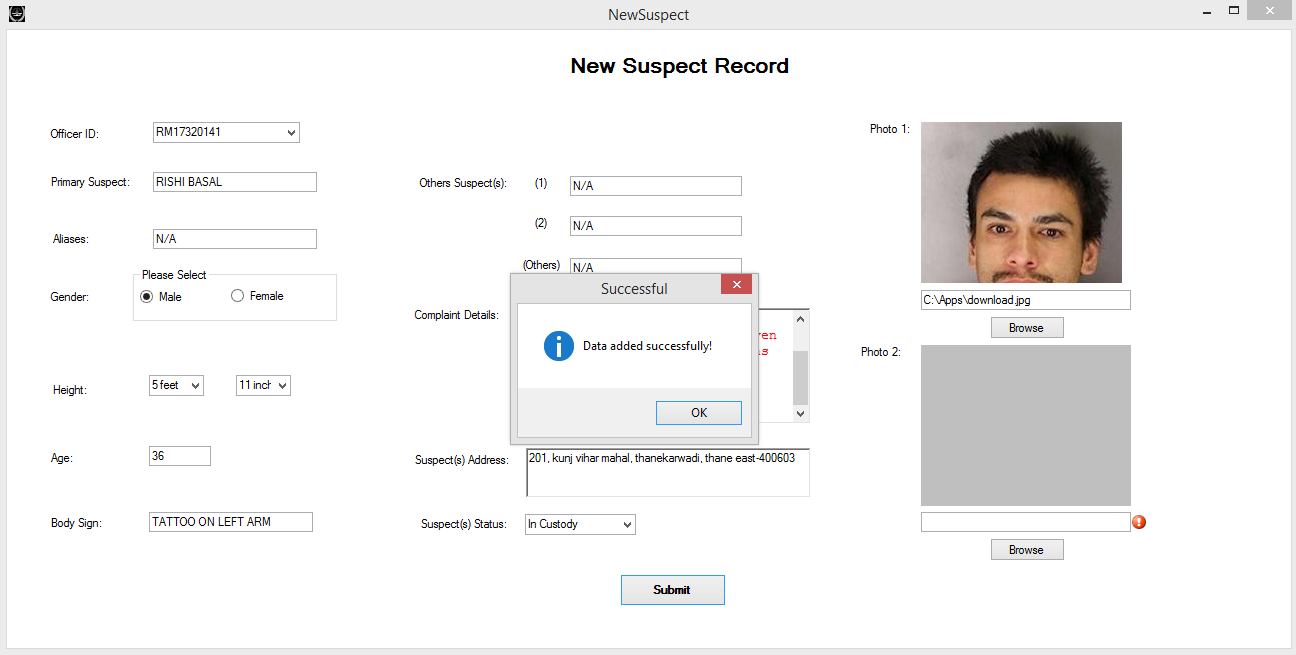
**Fig. 8.37 New Suspect**

****

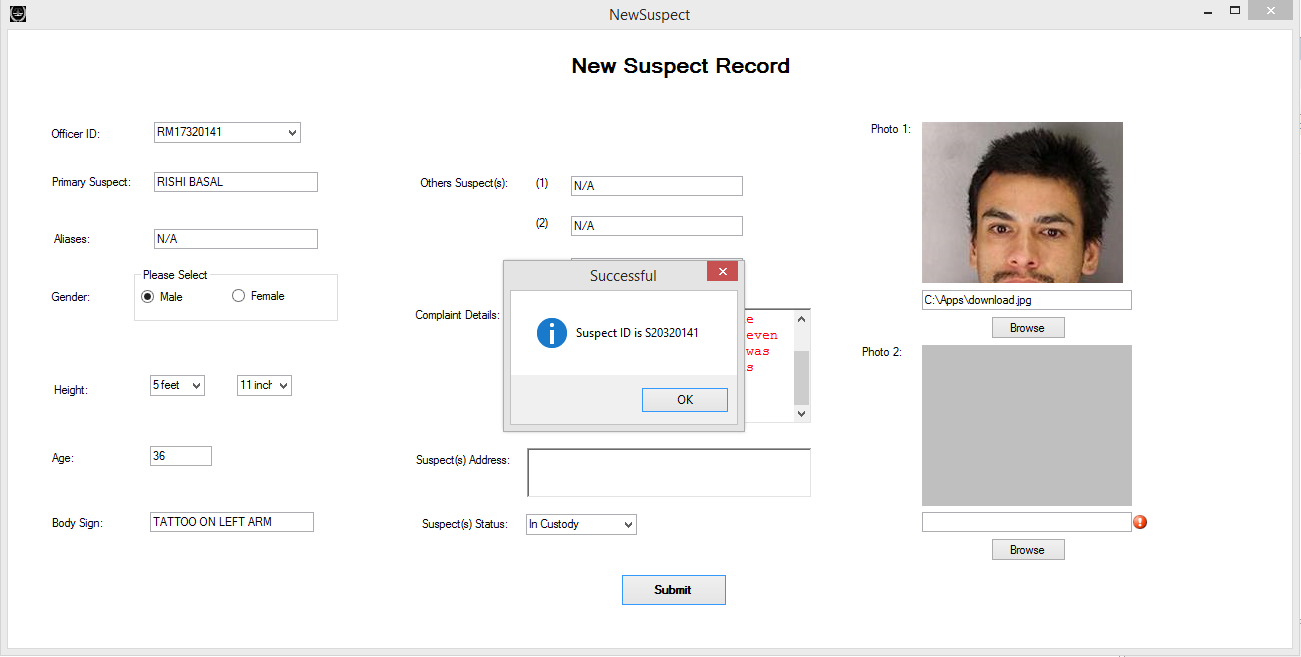
**Fig. 8.38 No information entered**

****

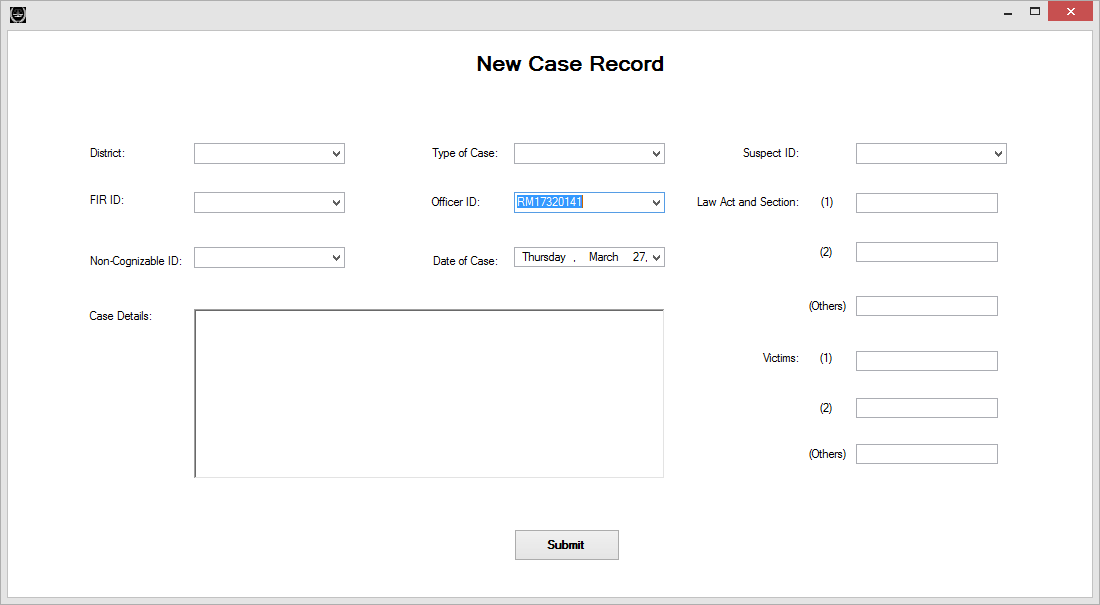
**Fig. 8.39 Suspect name not entered**

****

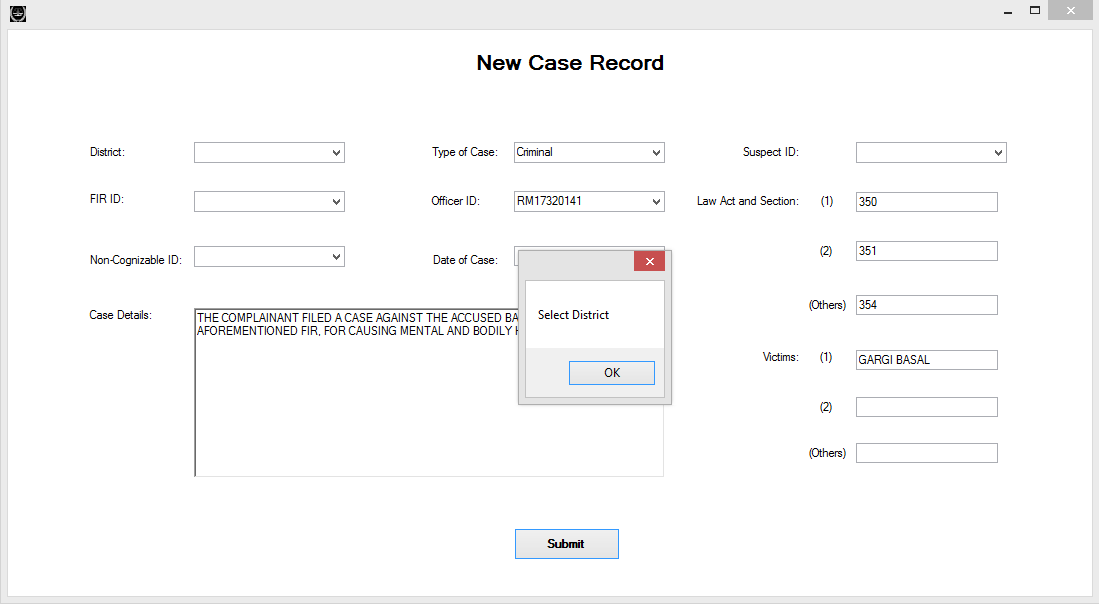
**Fig. 8.40 Suspect details added successfully**

****

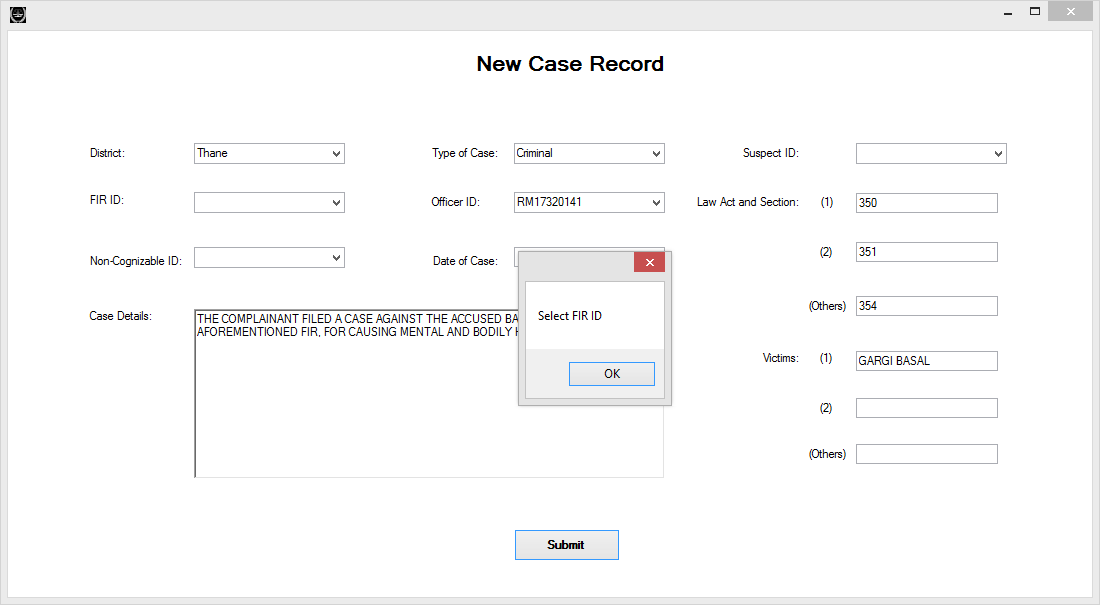
**Fig. 8.41 Suspect ID generated**

****

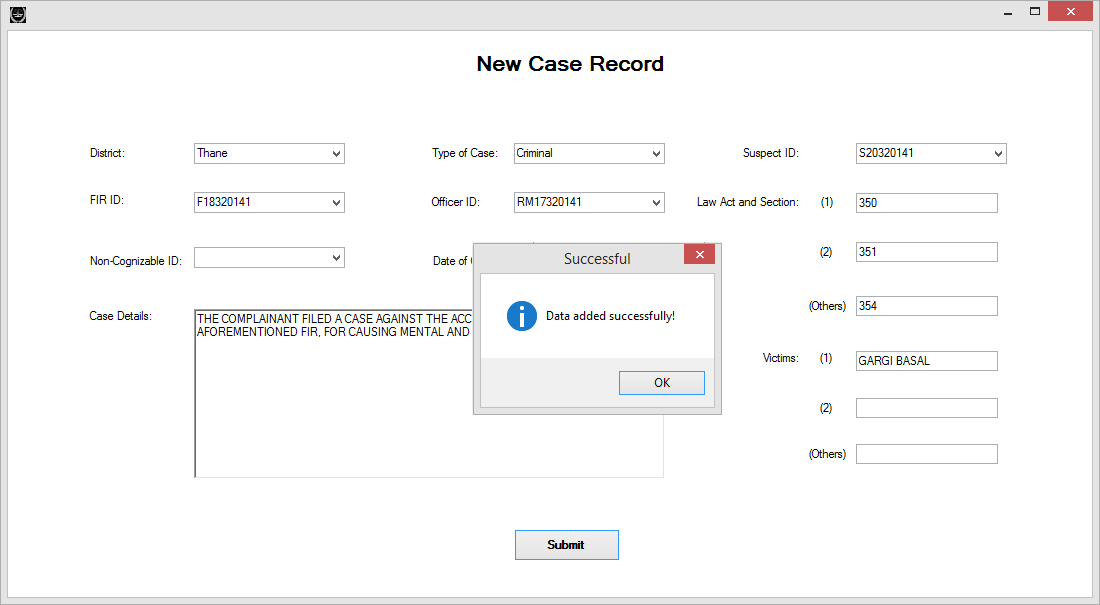
**Fig. 8.42 New Case Record**

****

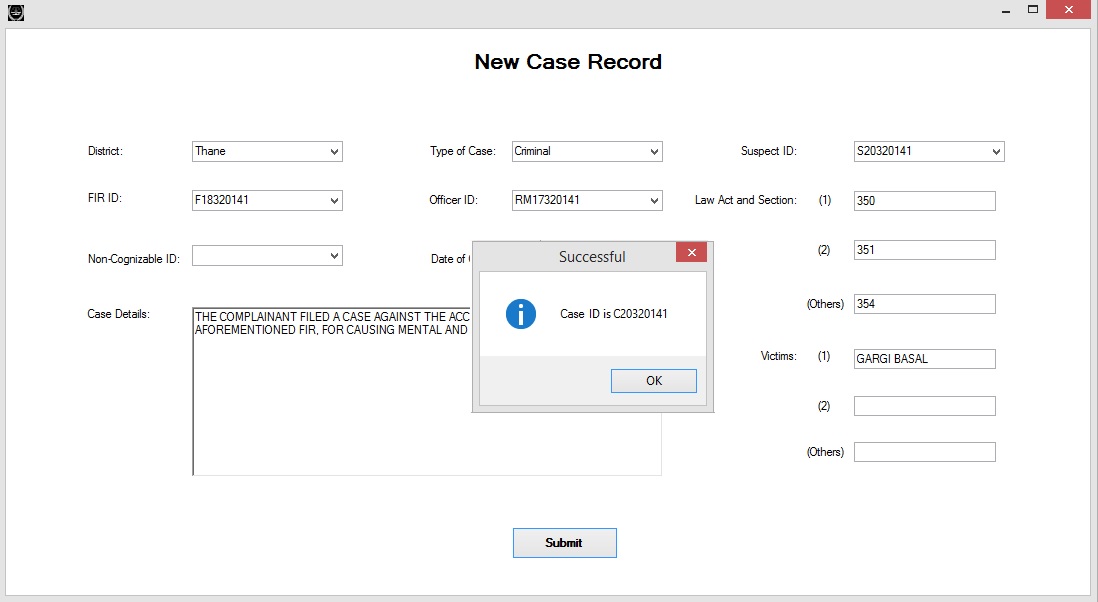
**Fig. 8.43 District not selected**

****

**Fig. 8.44 FIR ID not selected**

****

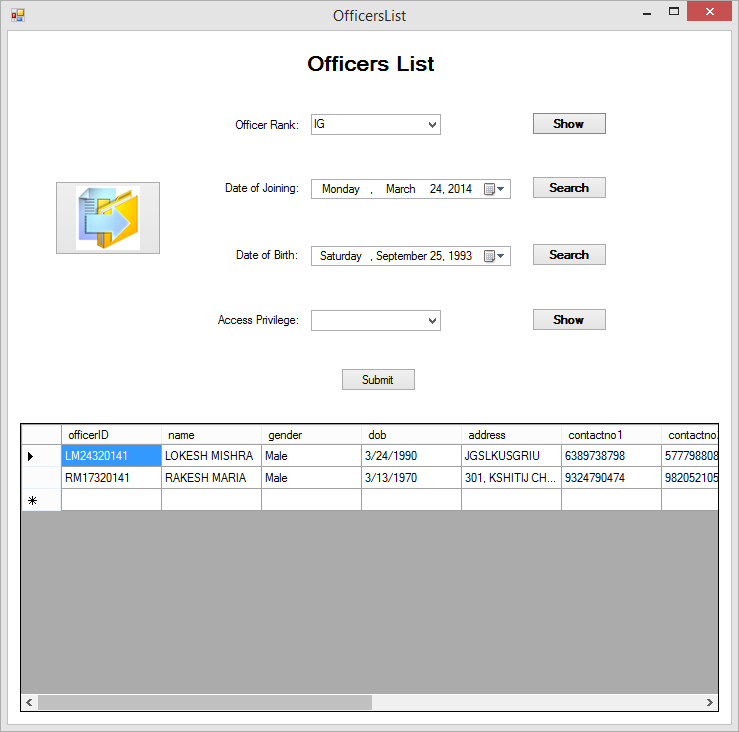
**Fig. 8.45 Case Data successfully added**

****

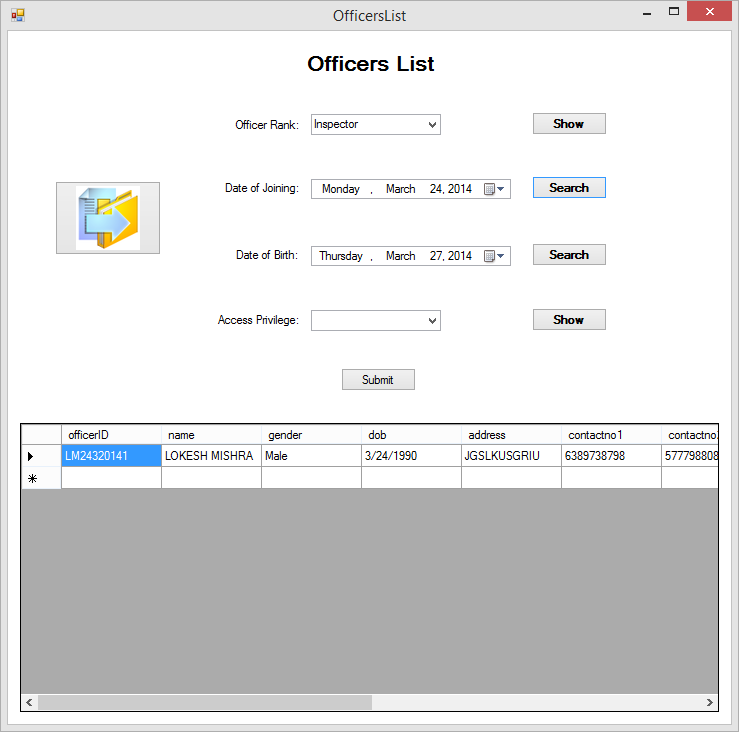
**Fig. 8.46 Case ID Generated**

****

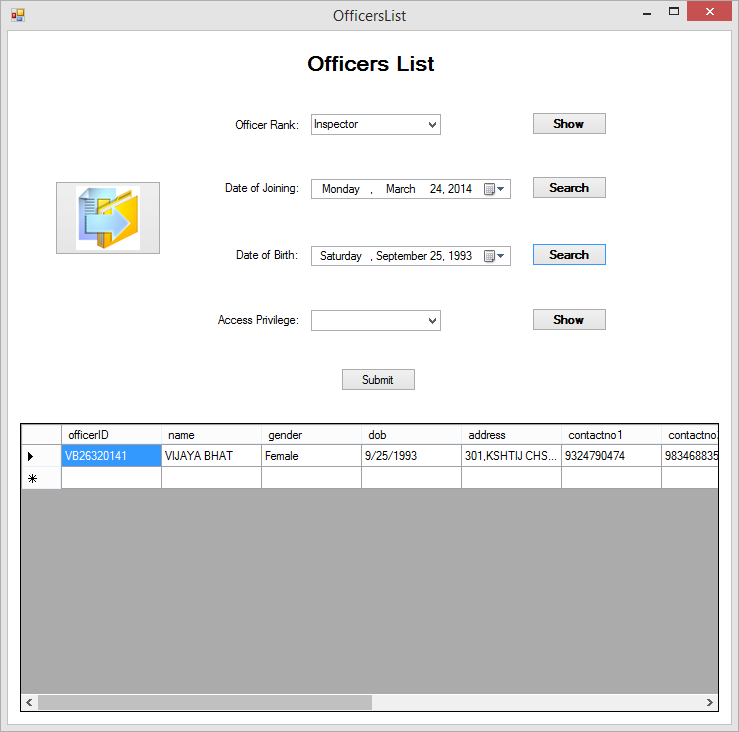
**Fig. 8.47 Officers list**

****

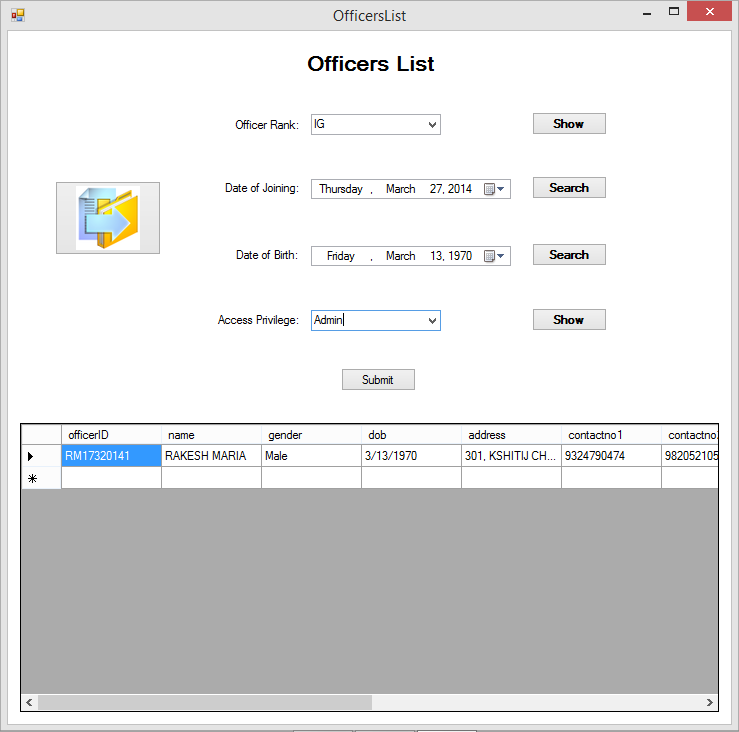
**Fig. 8.48 Officers filtered by rank**

****

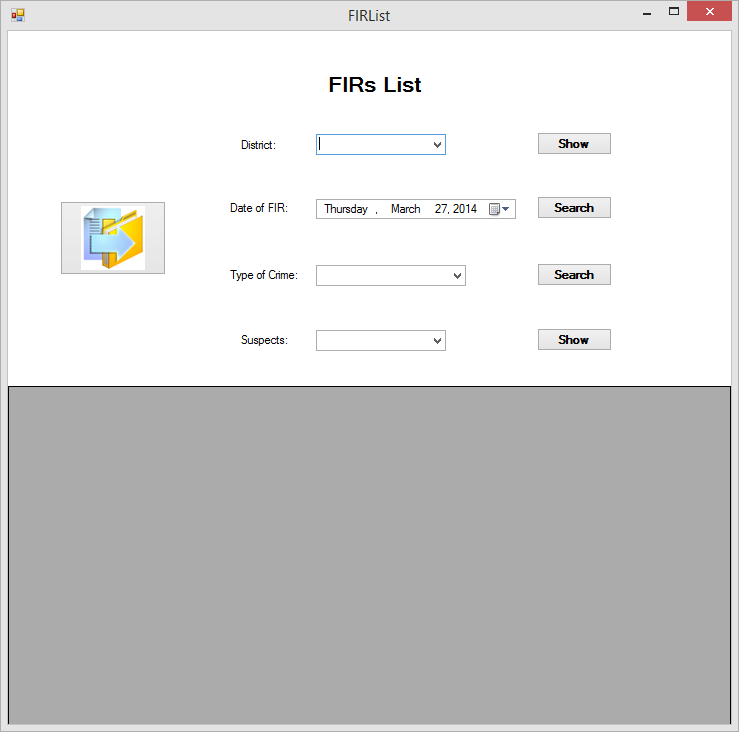
**Fig. 8.49 Officers filtered by date of joining**

****

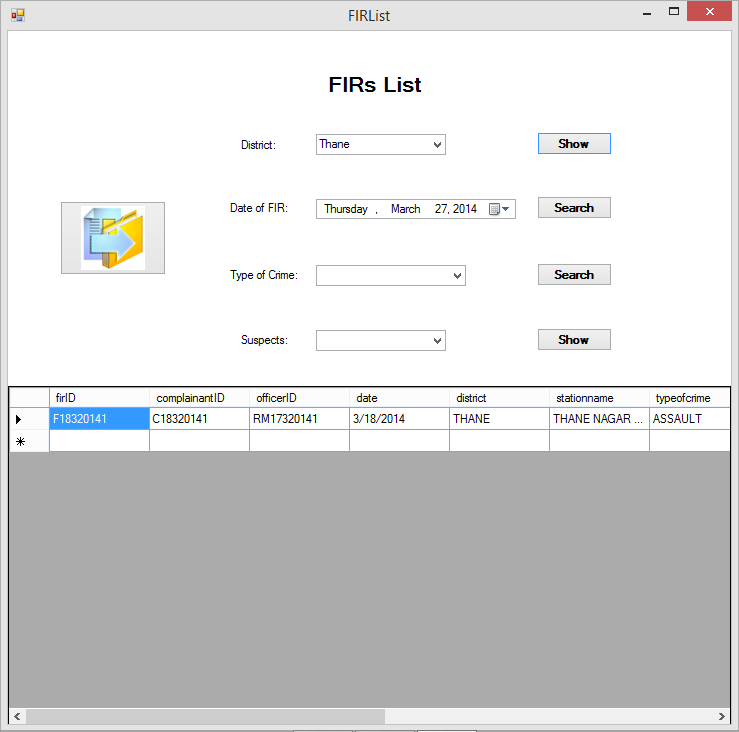
**Fig. 8.50 Officers filtered by date of birth**

****

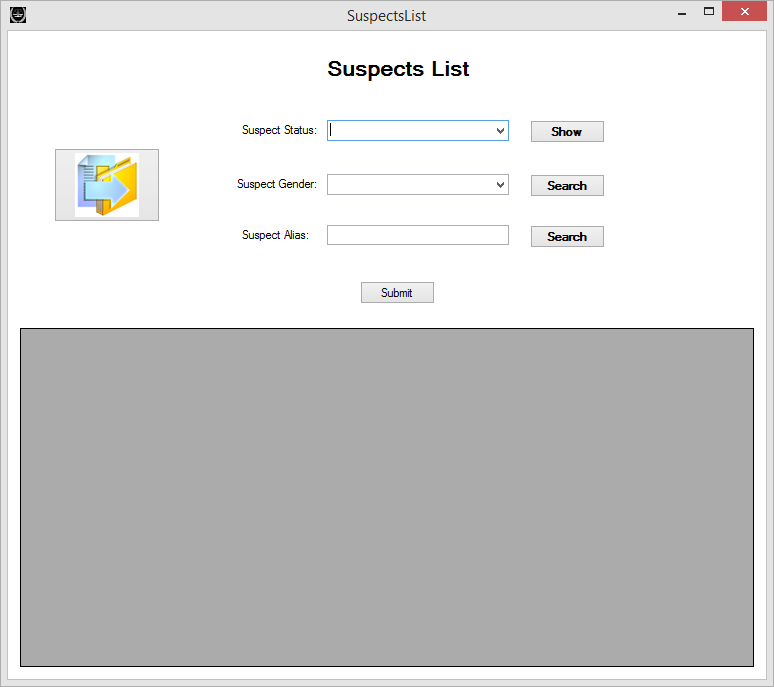
**Fig. 8.51 Officers filtered by access privilege**

****

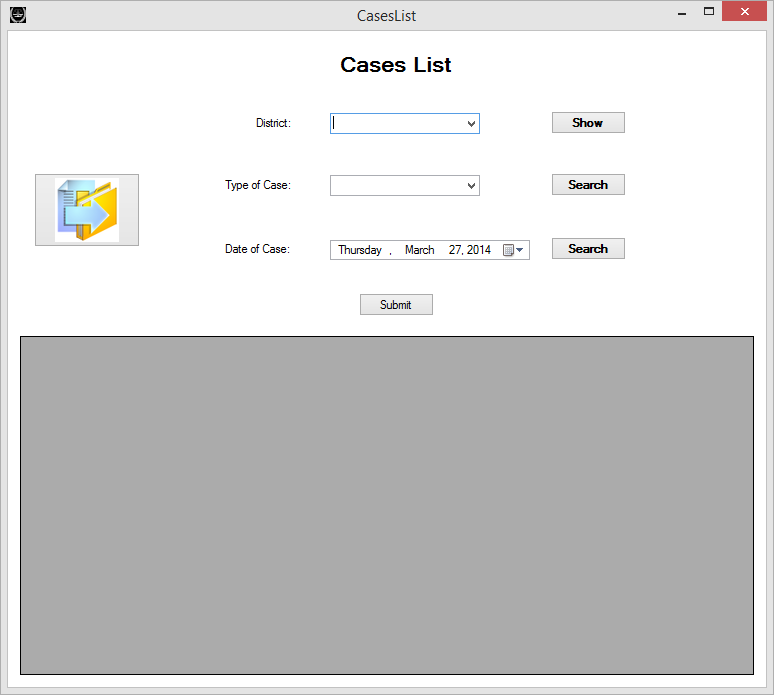
**Fig. 8.52 FIRs list**

****

**Fig. 8.53 FIRs list filtered by District**

****

**Fig. 8.54 Suspects list**

****

**Fig. 8.55 Cases list**

**CHATPER 9 – TESTING**

**9.1 Testing:**

Software testing is a process which is used to measure the quality of the software developed. It is also a process of uncovering errors in a program and making it a feasible task. It is useful for executing the program with the intention of finding bugs. In order to prove that a piece of software works, the software must be tested so as to determine if the requirements of the application are met. There are several different types of tests used throughout the development process.

* 1. **Types of testing:**

**9.2.1 White box testing:**

An internal perspective of the system as well as programming skills is used to design test cases. The tester chooses inputs to exercise paths through the code and to determine the appropriate outputs.

**9.2.2 Black box testing:**

Black box testing has little or no regard to the internal logical structure of the system. It only examines the fundamental aspect of the system. It makes sure that the input is properly accepted and output is correctly produced.

**9.2.3 Functional testing:**

Functional testing involves exercising the code with nominal input values which gives the expected results, by which the boundary values can be determined.

**9.2.4 Performance testing:**

Performance tests are designed to verify response time. If wrong data is entered, the system does not allow it and calculations cannot be performed.

**9.2.5 Unit testing:**

The first test in the development process is the unit test. The source code is normally divided into modules, which in turn are divided into smaller parts called units. These units have specific behaviors. The test done on these units of code is called unit test. Unit test depends upon the language on which the project is developed. Unit tests ensure that each unique path of the project performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**9.2.6 System testing:**

Several modules constitute a project. If the project is a long-term project, several developers write the modules. Once all the modules are integrated, several errors may arise. The testing done at this stage is called system testing. System testing ensures that the entire integrated software system meets the requirements. It tests a configuration to ensure known and predictable results. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**9.3Test cases:**

The two fundamental goals of a practical testing activity are — maximize the number of errors detected and minimize the number of test cases (i.e., minimize the cost). Hence, an ideal test case set is one that succeeds (meaning that its execution reveals no errors) only if there are no errors in the program.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr. No.** | **Name** | **Input** | **Expected output** | **Actual output** | **Remark** |
| 1 | Login | Enter access privilege, id and password | Admin panel for Admin & Officer panel for Officer | Admin panel for Admin & Officer panel for Officer | Successful |
| 2 | Add new FIR | All FIR details | Confirmation message | Confirmation message | Successful |
| 3 | Add new Complainant information | All Complainant details | Confirmation message | Confirmation message | Successful |
| 4 | Add new Suspect information | All Suspect details | Confirmation message | Confirmation message | Successful |
| 5 | Add new Case information | All Case details | Confirmation message | Confirmation message | Successful |
| 6 | Add new Case Disposal information | All Case Disposal details | Confirmation message | Confirmation message | Successful |
| 7 | Add new Non-Cognizable information | All Non-Cognizable details | Confirmation message | Confirmation message | Successful |
| 8 | Add new Officer information | All Officer details (n/a for Officer login) | Confirmation message | Confirmation message | Successful |
| 9 | Search for FIR/ Complainant/ Suspect/ Case/ Case Disposal/ Non-Cognizable/ Officer either by id or by name | Id or name | Details acquired | Details acquired | Successful |
| 10 | Edit Suspect information | Select Edit Suspect information | Confirmation message of updating | Confirmation message of updating | Successful |
| 11 | Edit Case information | Select Edit Case information | Confirmation message of updating | Confirmation message of updating | Successful |
| 12 | Edit Officer information | Select Edit Officer information (n/a for Officer login) | Confirmation message of updating | Confirmation message of updating | Successful |
| 13 | Generate FIR Report | Select report, enter FIR id | FIR report is generated | FIR report is generated | Successful |
| 14 | Generate Case Disposal Report | Select report, enter Case Disposal id | If Case has been disposed generate excel, else error | If Case has been disposed generate excel, else error | Successful |

**Table 9.3 Test cases**

**CHAPTER 10 – SYSTEM MAINTAINENCE AND EVALUATION**

**10.1 System maintenance:**

Maintenance is not a part of software development, but is an extremely important activity in the life of a software product. Maintenance includes all the activities after the installation of the software that are performed to keep the system operational.

Maintenance activities related to fixing of errors fall under corrective mechanisms. The maintenance related to modifications that may occur due to changes in the requirements is called ‘Adaptive Maintenance’.

A software may fail after operating correctly for some time. It is necessary to check that the vendor’s computer system is having the requirements specified, that too in full version and not trial period version. If the requirements are met, check for the bugs and errors and do possible modifications. These bugs should be removed. It is necessary to visit the organization frequently to check whether the system is maintained properly and is error free so as we do not encounter such problems in the future. An approach to problem solving should always be adopted.

The maintenance of the software developed for the Crime Record management is not a problem since it has been developed after a detailed study of the requirements of the user of the system. The needs of the maintainers are also kept in mind during the development process.

In most software organizations, the budget for software maintenance is much larger than for software development. However, there is much less management attention focus on software maintenance than on software development.

**CHAPTER 11 – TOOLS FOR DEVELOPING THE SYSTEM**

**11.1 Visual Studio 2010:**



It includes new modeling tools, such as the Architecture Explorer, which graphically displays projects and classes and the relationships between them. It supports UML activity diagram, component diagram, (logical) class diagram, sequence diagram, and use case diagram. Visual Studio Ultimate 2010 also includes Test Impact Analysis which provides hints on which test cases are impacted by modifications to the source code, without actually running the test cases. This speeds up testing by avoiding running unnecessary test cases.

It also provides easy drag and drop facility to design GUI. C# language has been used for development.

**11.2 Micr0osoft SQL Server:**



**Microsoft SQL Server** is a relational database management system (RDBMS) produced by Microsoft. Its primary query language is Transact-SQL, an implementation of the ANSI/ISO standard Structured Query Language (SQL).

SQL ("Structured Query Language") is an ANSI Standard computer language commonly used to access data stored in databases. SQL is a tool for managing, organizing, and retrieving data stored in a computer database. SQL works with relational databases which all organize data into tables, rows, and columns which correspond to files, records and fields. When you need to retrieve data from a database, you use the SQL language to make the request. The DBMS processes the SQL request, retrieves the requested data and returns it to you. This process of requesting data from a database and retrieving back the results is called a database query-hence the name Structured QueryLanguage.

**CHAPTER 12 – COST AND BENEFIT ANALYSIS**

**12.1 Cost analysis:**

For a given set of requirements it is desirable to know how much it will cost to develop the software, and how much time will be required for the development. These estimations are needed before development is initiated. The primary reason of cost and schedule estimation is to enable the developer to perform a cost benefit analysis and for project monitoring and control. The cost of a project is a function of many parameters. Foremost among them is the size of the project. Other factors that affect the cost are programmer ability, experience of the developers in the area, complexity of the project, and reliability requirements. It is also due to the requirements of software, hardware and human resources.

**12.1.1 Project cost using COCOMO:**

The Constructive Cost Model (COCOMO) is an algorithmic software cost estimation model developed by Barry W. Boehm. The model uses a basic regression formula with parameters that are derived from historical project data and current project characteristics.

COCOMO consists of a hierarchy of three increasingly detailed and accurate forms. The first level, Basic COCOMO is good for quick, early, rough order of magnitude estimates of software costs, but its accuracy is limited due to its lack of factors to account for difference in project attributes (Cost Drivers). Intermediate COCOMO takes these Cost Drivers into account and

Detailed COCOMO additionally accounts for the influence of individual project phases.

**12.2 Intermediate COCOMO:**

This computes the software development effort as a function of program size and a set of "cost drivers" that include subjective assessment of product, hardware, personnel and project attributes. This extension considers a set of four "cost drivers", as shown in table 12.1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Cost Drivers | Ratings | | | | |
| Very Low | Low | Nominal | High | Very high |
| **Product Attributes** |  |  |  |  |  |
| Required s/w reliability | 0.75 | 0.88 | 1.00 | 1.15 | 1.40 |
| Size of application database |  | 0.94 | 1.00 | 1.08 | 1.16 |
| Complexity of the product | 0.70 | 0.85 | 1.00 | 1.15 | 1.30 |
| **Hardware attributes** |  |  |  |  |  |
| Run-time performance constraints |  |  | 1.00 | 1.11 | 1.30 |
| Memory constraints |  |  | 1.00 | 1.06 | 1.21 |
| Volatility of the virtual machine environment |  | 0.87 | 1.00 | 1.15 | 1.30 |
| Required turnabout time |  | 0.87 | 1.00 | 1.07 | 1.15 |
| **Personnel attributes** |  |  |  |  |  |
| Analyst capability | 1.46 | 1.19 | 1.00 | 0.86 | 0.71 |
| Applications experience | 1.29 | 1.13 | 1.00 | 0.91 | 0.82 |
| Software engineer capability | 1.42 | 1.17 | 1.00 | 0.86 | 0.70 |
| Virtual machine experience | 1.21 | 1.10 | 1.00 | 0.90 |  |
| Programming language experience | 1.14 | 1.07 | 1.00 | 0.95 |  |
| **Project attributes** |  |  |  |  |  |
| Application of software engineering  Methods | 1.24 | 1.10 | 1.00 | 0.91 | 0.82 |
| Use of Software Tools | 1.24 | 1.10 | 1.00 | 0.91 | 0.83 |
| Required development schedule | 1.23 | 1.08 | 1.00 | 1.04 | 1.10 |

**Table 12.1**

Each of the 15 attributes receive a rating on a six-point scale that ranges from "very low" to "extra high" (in importance or value). An effort multiplier from the table below applies to the rating. The product of all effort multipliers results in an effort adjustment factor (EAF). Typical values for EAF range from 0.9 to 1.4.

**12.2.2 ADVANTAGES OF INTERMEDIATE COCOMO:**

The Intermediate COCOMO model can be applied to almost entire software project for easy and rough implementation during early stages. It can also be applied at the software component level for obtaining more accurate cost estimation.

**12.2.3 DISADVANTAGES OF INTERMEDIATE COCOMO:**

A Product with many components is difficult to estimate with Intermediate COCOMO and secondly the effort multipliers are not dependent on phase.

**12.2.4 CALCULATION:**

EAF=An Effort Adjustment factor that characterizes the Domain, Personnel, Environment and Tools used to produce artifacts of the process

KLOC= is the estimated number of thousands of delivered lines of code for the project.

Size=Size of end product measured by delivered source instructions (DSI).

Effort=Number of staff months.

Calculating EAF from the table gives:

EAF= 1\*1\*1\*1.11\*1.06\*0. 87\*0.87\*1\*1\*1\*1.10\*1\*1\*0.91\*1.04

EAF = 0.92

‘ai = 3.0’, ‘bi =1.12’, ‘c = 2.5’ and ‘d = 0.35’.

KLOC = 7.8 (Approx)

Using these values Effort Applied can be calculated as,

Effort (E) =3.0\*EAF (Size) 1.12

E = 3.0 \* 0.92 (7.8) 1.12

E = 27

Development time (D)= c (Effort applied)d

D = (2.5)(27)0.35

D = 7

People required (P) = Effort applied/ Development time

P= 27/7

P = 3.8 ~3

Independent costs = 15,000 (approx)

Infrastructure costs = 4,000 (approx)

**Total cost= Infrastructure costs\* People required + Miscellaneous costs.**

**Total cost = 4,000 \* 3 + 15,000**

**Total cost= Rs. 27,000**

**12.3 Benefit analysis:**

In the old system the work was done manually, so the cost of handling the system was increasing. The cost mainly includes the charges for registry maintenance, receipt books, files, accessories, and etc. to reduce the costs the new system was proposed. Positive aspects of the designed system which contributed to the benefit analysis are fast and easy storage of all information. It was also easy to retrieve any required details as fast as possible. There is no need for maintaining receipt books. The new system is very beneficial than the old one because the system is fully automated.

The metrics of benefit analysis are:

**Benefit/cost ratio (B/C):** Benefit/cost ratio pertains to the ratio between the net benefit in implementing a security solution and the costs of implementing. It demonstrates the capability for the organization to profit from its security investments.

**Estimation of loses:** For better accuracy it is highly recommend that losses be estimated for each misuse case.

**Estimation of costs:** Costs are then calculated on the number of man-hours multiplied by average hourly wage rates. The company provides average cost figures for employment in different roles. We found this process of cost estimation as the use of man-hours to estimate costs of implementation to be very effective.

**CHAPTER 13 – LIMITATIONS OF THE PROJECT**

**13.1 Limitations:**

1. The system won’t work effectively on older technologies.
2. Because it is a software, frequent maintenance would be required.
3. Basic knowledge of computers and their handling is required to effectively operate the software.

**CHAPTER 14 – CONCLUSION**

**14.1 Conclusion:**

At the end, we are able to cover all the basic requirements of the administrator and the officer as the system is feasible. They are allowed to make changes as per it is needed in the existing system. No paperwork is involved in this project. The project also allows the users to make observations about the existing systems and learn how superior systems can be developed from these existing systems .The system will be efficient and fast in response .It can be customized according to the needs of the users.

There is a lot of ease in using the software, as a very accessible user interface is used. Manual work is greatly reduced. Also, recording the information through an attractive interface and a fast pace gives a unique image to this software.

For making this project successful, a lot of research work was involved including interviews with various police officers. It has been a unique experience developing this project, due to which I have gained great knowledge and a new outlook about software development as well as project management.

**CHAPTER 15 - ANNEXURE**

**15.1 Diagrams list**

|  |  |  |
| --- | --- | --- |
| **Figure number** | **Name** | **Page no.** |
| Fig. 5.6.1 | ER Diagram | 21 |
| Fig. 5.6.2 | Database Schema | 26 |
| Fig. 5.6.3.1 | Use case add info for Admin | 28 |
| Fig. 5.6.3.2 | Use case search info for Admin | 28 |
| Fig. 5.6.3.3 | Use case edit info for Admin | 29 |
| Fig. 5.6.3.4 | Use case add info for Officer | 29 |
| Fig. 5.6.3.5 | Use case search info for Officer | 30 |
| Fig. 5.6.3.6 | Use case edit info for Officer | 30 |
| Fig. 5.6.4.1 | Sequence diagram add info for Officer | 41 |
| Fig. 5.6.4.2 | Sequence diagram search info for Officer | 42 |
| Fig. 5.6.4.3 | Sequence diagram edit info for Officer | 43 |
| Fig. 5.6.4.4 | Sequence diagram add info for Admin | 44 |
| Fig. 5.6.4.5 | Sequence diagram search info for Admin | 44 |
| Fig. 5.6.4.6 | Sequence diagram edit info for Admin | 45 |
| Fig. 5.6.5.1 | Context level DFD | 47 |
| Fig. 5.6.5.2 | Level 1 DFD for Admin | 48 |
| Fig. 5.6.5.3 | Level 1 DFD for Officer | 49 |
| Fig. 5.6.5.4 | Level 2 DFD for Login(Admin) | 50 |
| Fig. 5.6.5.5 | Level 2 DFD for Login(Officer) | 50 |
| Fig. 5.6.5.6 | Level 2 DFD for FIR | 51 |
| Fig. 5.6.5.7 | Level 2 DFD for Complainant | 51 |
| Fig. 5.6.5.8 | Level 2 DFD for Suspect | 52 |
| Fig. 5.6.5.9 | Level 2 DFD for Case | 52 |
| Fig. 5.6.5.10 | Level 2 DFD for Case Disposal | 53 |
| Fig. 5.6.5.11 | Level 2 DFD for Non-Cognizable | 53 |
| Fig. 5.6.5.12 | Level 2 DFD for Officer | 54 |
| Fig. 5.6.7.1 | Activity diagram for Add information (Admin) | 57 |
| Fig. 5.6.7.2 | Activity diagram for Search information (Admin) | 58 |
| Fig. 5.6.7.3 | Activity diagram for Edit information (Admin) | 59 |
| Fig. 5.6.7.4 | Activity diagram for Add information (Officer) | 60 |
| Fig. 5.6.7.5 | Activity diagram for Search information (Officer) | 61 |
| Fig. 5.6.7.6 | Activity diagram for Edit information (Officer) | 62 |
| Fig. 5.6.8 | Class Diagram | 64 |

**15.2 Table list:**

|  |  |  |
| --- | --- | --- |
| **Table name** | **Name** | **Page no.** |
| Table 5.6.1 | ER Notations | 21 |
| Table 5.6.5 | DFD Notations | 47 |
| Table 5.6.6 | Activity diagram notations | 56 |
| Table 5.6.8.1 | Officer table | 65 |
| Table 5.6.8.2 | Complainant table | 66 |
| Table 5.6.8.3 | Suspect table | 67 |
| Table 5.6.8.4 | FIR table | 68 |
| Table 5.6.8.5 | Case table | 69 |
| Table 5.6.8.6 | Case Disposal table | 70 |
| Table 5.6.8.7 | Non-Cognizable table | 71 |
| Table 8.4 | Test cases | 115 |
| Table 12.1 | Intermediate COCOMO | 125 |

**15.3 Screenshots:**

|  |  |  |
| --- | --- | --- |
| **Screenshot no.** | **Name** | **Page no.** |
| 8.1 | Login screen | 82 |
| 8.2 | Login screen with details | 82 |
| 8.3 | Message box showing correct login for administrator | 83 |
| 8.4 | Message box indicating incomplete details | 83 |
| 8.5 | Message box showing incorrect privilege selection for Admin Details | 84 |
| 88.6 | Message box indicating incomplete details | 84 |
| 8.7 | Show admin form after login | 85 |
| 8.8 | A verify close message | 85 |
| 8.9 | Message box showing correct login for Officer | 86 |
| 8.10 | Show main form after login | 86 |
| 8.11 | New account for officer | 87 |
| 8.12 | Adding data for officer | 87 |
| 8.13 | Providing incorrect information | 88 |
| 8.14 | Display of error message for privilege | 88 |
| 8.15 | Display of error message for incorrect password | 89 |
| 8.16 | Display of error message for providing 2 same contact numbers | 89 |
| 8.17 | Confirmation message | 90 |
| 8.18 | Generation and display of Officer’s ID | 90 |
| 8.19 | A verify close message | 91 |
| 8.20 | New Complainant | 91 |
| 8.21 | Providing Correct Information | 92 |
| 8.22 | Providing incomplete information | 92 |
| 8.23 | Incorrect Email format | 93 |
| 8.24 | Non-Selection of Officer | 93 |
| 8.25 | Non-Selection of gender | 94 |
| 8.26 | Confirmation Message | 94 |
| 8.27 | Generation and display of Complainant’s id | 95 |
| 8.28 | Select district | 95 |
| 8.29 | Select station | 96 |
| 8.30 | Correct details filled | 96 |
| 8.31 | Display of FID id | 97 |
| 8.32 | New Non-Cognizable complaint | 97 |
| 8.33 | Select station | 98 |
| 8.34 | Empty field error | 98 |
| 8.35 | Correct details filled | 99 |
| 8.36 | Data successfully added | 99 |
| 8.37 | Display complainant id | 100 |
| 8.38 | No information added | 100 |
| 8.39 | Suspect name not entered | 101 |
| 8.40 | Suspect details added successfully | 101 |
| 8.41 | Suspect ID generated | 102 |
| 8.42 | New Case | 102 |
| 8.43 | District not selected | 103 |
| 8.44 | FIR ID not selected | 103 |
| 8.45 | Case data successfully added | 104 |
| 8.46 | Case ID generated | 104 |
| 8.47 | Officers list | 105 |
| 8.48 | Officers list filtered by rank | 105 |
| 8.49 | Officers list filtered by date of joining | 106 |
| 8.50 | Officers list filtered by date of birth | 106 |
| 8.51 | Officers list filtered by access privilege | 107 |
| 8.52 | FIRs list | 107 |
| 8.53 | FIRs list filtered by district | 108 |
| 8.54 | Suspects list | 108 |
| 8.55 | Cases list | 109 |

**Bibliography:**

**Text-Book References:-**

* Software Engineering, Fifth Edition by Roger S. Pressman, Tata McGraw-Hill Publication.
* SQL Complete Reference Second Edition by James R. Groff and Paul Weinberg, Tata McGraw-Hill Publication.
* ASP.NET Wrox Publication

**13.3.2 Web References: -**

* www.wikipedia.com (Referred between 8th August’13 to 20th January’13)
* www.w3schools.com (Referred between 15th September’13 to 20th December’13 )
* www.youtube.com (Referred between 3rd October’13 to 31st January’13)
* www.codeproject.com (Referred between 3rd January to 10th March’14)
* www.sourcecode.com (Referred between 3rd January to 15th March’14)
* www.scribd.com (Referred between 5th January’14 to 3rd February’14)