Criminal Face Identification System

INDEX

ABSTRACT:

Title: Criminal Face Identification System

Objective:

Criminal record generally contains personal information about particular person along with photograph. To identify any Criminal we need some identification regarding person, which are given by eyewitness. In most cases the quality and resolution of the recorded image segments is poor and hard to identify a face. To overcome this sort of problem we are developing software. Identification can be done in many ways like finger print, eyes, DNA etc. One of the applications is face identification. The face is our primary focus of attention in social inters course playing a major role in conveying identify and emotion. Although the ability to infer intelligence or character from facial appearance is suspect, the human ability to recognize face is remarkable.

INTRODUCTION

PURPOSE OF THE PROJECT:

This project is aimed to identify the criminals in any investigation department. Here the technique is we already store some images of the criminals in our database along with his details and that images are segmented into many slices say eyes, hairs, lips, nose, etc. These images are again stored in another database record so to identify any criminals; eyewitnesses will see the images or slices that appear on the screen by using it we develop the face, which may or may not be matched with our images. If any image is matched up to 99% then we predict that he is only the criminal. Thus using this project it provides a very friendly environment for both operator and eyewitness to easily design any face can identify criminals very easy.

PROJECT OBJECTIVE:

This project is intended to identify a person using the images previously taken. The identification will be done according the previous images of different persons.

PROJECT SCOPE:

The scope of the project is confined to store the image and store in the database. When a person has to be identified the images stored in the database are compared with the existing details.

OVERVIEW OF THE PROJECT:

This project is aimed to identify the criminals in any investigation department. Here the technique is we already store some images of the criminals in our database along with his details and those images are segmented into many slices say eyes, hairs, lips, nose, etc. These images are again stored in another database record so to identify any criminals; eyewitnesses will see the images or slices that appear on the screen by using it we develop the face, which may or may not be matched with our images. If any image is matched up to 99% then we predict that he is only the criminal. Thus using this project it provides a very friendly environment for both operator and eyewitness to easily design any face can identify criminals very easy.

PROBLEM AREA DESCRIPTION

The project is aimed at identifying the criminals with the help of eye witness. There are mainly four modules in our project. They are Adding, Deleting, Updating and identifying the criminals. There are mainly three roles in our project. They are:

- Administrator
- Operator
- Eyewitness

The administrator is responsible for providing User id's and passwords. He provides authentication to the users. He creates deletes and Updates the User ids and Passwords.

The operator, who belongs to the investigating department, is responsible for entering the criminal details and maintains them. He adds, deletes and updates the criminal details. He also constructs the criminal face with the help of eye witness.

The eyewitness identifies the criminals with the help of cropped parts stored in a different database by the operator. The eyewitness selects a cropped part from the database and that cropped part will be freeze by the operator in this way, complete face of the criminal is constructed and the details of that criminal is retrieved from the database. We can also construct a new image from those cropped parts which we consider as an imaginary face of the criminal.

SYSTEM ANALYSIS

The first step in developing anything is to state the requirements. This applies just as much to leading edge research as to simple programs and to personal programs, as well as to large team efforts. Being vague about your objective only postpones decisions to a later stage where changes are much more costly.

The problem statement should state what is to be done and not how it is to be done. It should be a statement of needs, not a proposal for a solution. A user manual for the desired system is a good problem statement. The requestor should indicate which features are mandatory and which are optional, to avoid overly constraining design decisions. The requestor should avoid describing system internals, as this restricts implementation flexibility. Performance specifications and protocols for interaction with external systems are legitimate requirements. Software engineering standards, such as modular construction, design for testability, and provision for future extensions, are also proper.

Many problems statements, from individuals, companies, and government agencies, mixture requirements with design decisions. There may sometimes be a compelling reason to require a particular computer or language; there is rarely justification to specify the use of a particular algorithm. The analyst must separate the true requirements from design and implementation decisions disguised as requirements. The analyst should challenge such pseudo requirements, as they restrict flexibility. There may be politics or organizational reasons for the pseurequirements, but at least the analyst should recognize that these externally imposed design decisions are not essential features of the problem domain.

A problem statement may have more or less detail. A requirement for a conventional product, such as a payroll program or a billing system, may have considerable detail. A requirement for a research effort in a new area may lack many details, but presumably the research has some objective, which should be clearly stated.

Most problem statements are ambiguous, incomplete, or even inconsistent. Some requirements are just plain wrong. Some requirements, although precisely stated, have unpleasant consequences on the system behavior or impose unreasonable implementation costs. Some requirements seem reasonable at first but do not work out as well as the request or thought. The problem statement is just a starting point for understanding the problem, not an immutable document. The purpose of the subsequent analysis is to fully understand the problem and its implications. There is no reasons to expect that a problem statement prepared without a fully analysis will be correct.

The analyst must work with the requestor to refine the requirements so they represent the requestor's true intent. This involves challenging the requirements and probing for missing information. The psychological, organizational, and political considerations of doing this are beyond the scope of this book, except for the following piece of advice: If you do exactly what the customer asked for, but the result does not meet the customer’s real needs, you will probably be blamed anyway.

Existing System:

This system is manual system only. Here, have a facility to store the criminal images. If you want to compare the criminal images with the existing images it is manual process. This process is very slow to give the result. It is very critical to find the criminal images.

Proposed System:

To overcome the drawbacks that were in the existing system we develop a system that will be very useful for any investigation department. Here the program keeps track of the record number of each slice during the construction of identifiable human face and calculate maximum number of slices of the similar record number. Based on this record number the program retrieves the personal record of the suspect (whose slice constituted the major parts of the constructed human face) on exercising the “locate” option.

Advantages:

- Very fast and accurate.
- No need of any extra manual effort.
- No fever of data loss.
- Just need a little knowledge to operate the system.
- Doesn’t require any extra hardware device.
- At last very easy to find the criminals

Overview:

Addition, Clipping, Construction and updating of the criminal record and face. Comparing the image with the faces that are there in our database. If any new images are found then it should be entered into our database by add image module and then it should be segmented into different slices.

FEASIBILITY REPORT

Feasibility Study

A feasibility study is a high-level capsule version of the entire System analysis and Design Process. The study begins by classifying the problem definition. Feasibility is to determine if it's worth doing. Once an acceptance problem definition has been generated, the analyst develops a logical model of the system. A search for alternatives is analyzed carefully. There are 3 parts in feasibility study.

Operational Feasibility:

Question that going to be asked are

- Will the system be used if it developed and implemented.
- If there was sufficient support for the project from the management and from the users.
- Have the users been involved in planning and development of the Project.
- Will the system produce poorer result in any respect or area?

This system can be implemented in the organization because there is adequate support from management and users. Being developed in Java so that the necessary operations are carried out automatically.

Technical feasibility:

- Does the necessary technology exist to do what is been suggested

- Does the proposed equipment have the technical capacity for using the new system?
- Are there technical guarantees of accuracy, reliability and data security?
- The project is developed on Pentium IV with 256 MB RAM.
- The environment required in the development of system is any windows platform
- The observer pattern along with factory pattern will update the results eventually
- The language used in the development is JAVA 1.5 & Windows Environment

Financial and Economical Feasibility:

The system developed and installed will be good benefit to the organization. The system will be developed and operated in the existing hardware and software infrastructure. So there is no need of additional hardware and software for the system.

SYSTEM REQUIREMENT SPECIFICATION

Functional Requirements:

By conducting the requirements analysis we listed out the requirements that are useful to restate the problem definition.

- Insert the image into database
- Split the image into no of parts.
- Merge the parts.
- Identify the image.
- Draw image manually.
- Maintain separate login for admin and operator.
- Maintain information about each criminal

Module Description

Well structured designs improve the maintainability of a system. A structured system is one that is developed from the top down and modular, that is, broken down into manageable components. In this project we modularized the system so that they have minimal effect on each other.

This application is designed into five independent modules which take care of different tasks efficiently.

1. **User Interface Module.**
2. **Admin Module.**
3. **Client Module.**
4. **Database Operations Module.**
5. **Splitting and Merging Module.**
6. **Identify Module.**

User Interface Module:

Actually every application has one user interface for accessing the entire application. In this application also we are providing one user interface for accessing this application. The user interface designed completely based on the end users. It is provide friendly accessing to the users. This user interface has attractive look and feel. Technically I am using the swings in core java for preparing this user interface.

Admin Module:

User requirements	**Elaboration**	**Further Elaboration**
Create	Assign new user id & password for an employee.	
Delete	Administrator can delete the user id & password of unwanted employee.	
Update	First the details of employees are to be obtained by using user id & password.	After obtaining the original details the updated details are submitted.

Client Module:

User requirements	Elaboration	Further Elaboration
Login	Employee log in to home page by entering id & password.	
Adding details	Personal details of criminal store in to data base	Images are cropped and saved in database.
Update process	Enter criminal id and obtain his details	Update the details and images of existing criminal
Delete process	Enter criminal id	Delete the details and image of unwanted criminal
Logout	Logout in to the home page	

Splitting and Merging Module:

Requirements	Elaboration	Further Elaboration
View clippings	View all clips and select the clip shown by eyewitness	Compare the clippings with images of criminals
Construction	Construct the face of criminal by clubbing all freezed clippings	

Database Operations Module:

ADD MODULE: The add module is helpful in adding the details of the criminals along with the details of the criminal photo. While adding the details of the criminal, we crop the image of the criminal and store those cropped parts in a separate database.

DELETE MODULE : This module deletes the criminal details along with the photo. The operator first submits the criminal id and searches for the availability of the id in the database. If that id is available in the database, then the operator may delete the record of that particular r criminal.

UPDATE MODULE : The operator first enters the criminal id and searches for the availability of that id .If that id is available in the database , then the details of that criminal are retrieved and the operator can update the details of that criminal and that updated details of the criminal are stored in the database again for future retrieval.

Identify Module:

The cropped parts of the criminals, along with the criminal Id are viewed by the eyewitness .The eyewitness selects particular cropped part of the criminal and it is freeze by the operator., then complete face of the criminal is constructed and the details of the criminal is retrieved.

SDLC METHDOLOGIES

SDLC METHDOLOGIES:

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

SPIRAL MODEL was defined by Barry Boehm in his 1988 article, "A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

The steps for Spiral Model can be generalized as follows:

- The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
- A preliminary design is created for the new system.
- A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and

represents an approximation of the characteristics of the final product.

- A second prototype is evolved by a fourfold procedure:

    1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
    2. Defining the requirements of the second prototype.
    3. Planning an designing the second prototype.
    4. Constructing and testing the second prototype.

- At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involve development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer's judgment, result in a less-than-satisfactory final product.

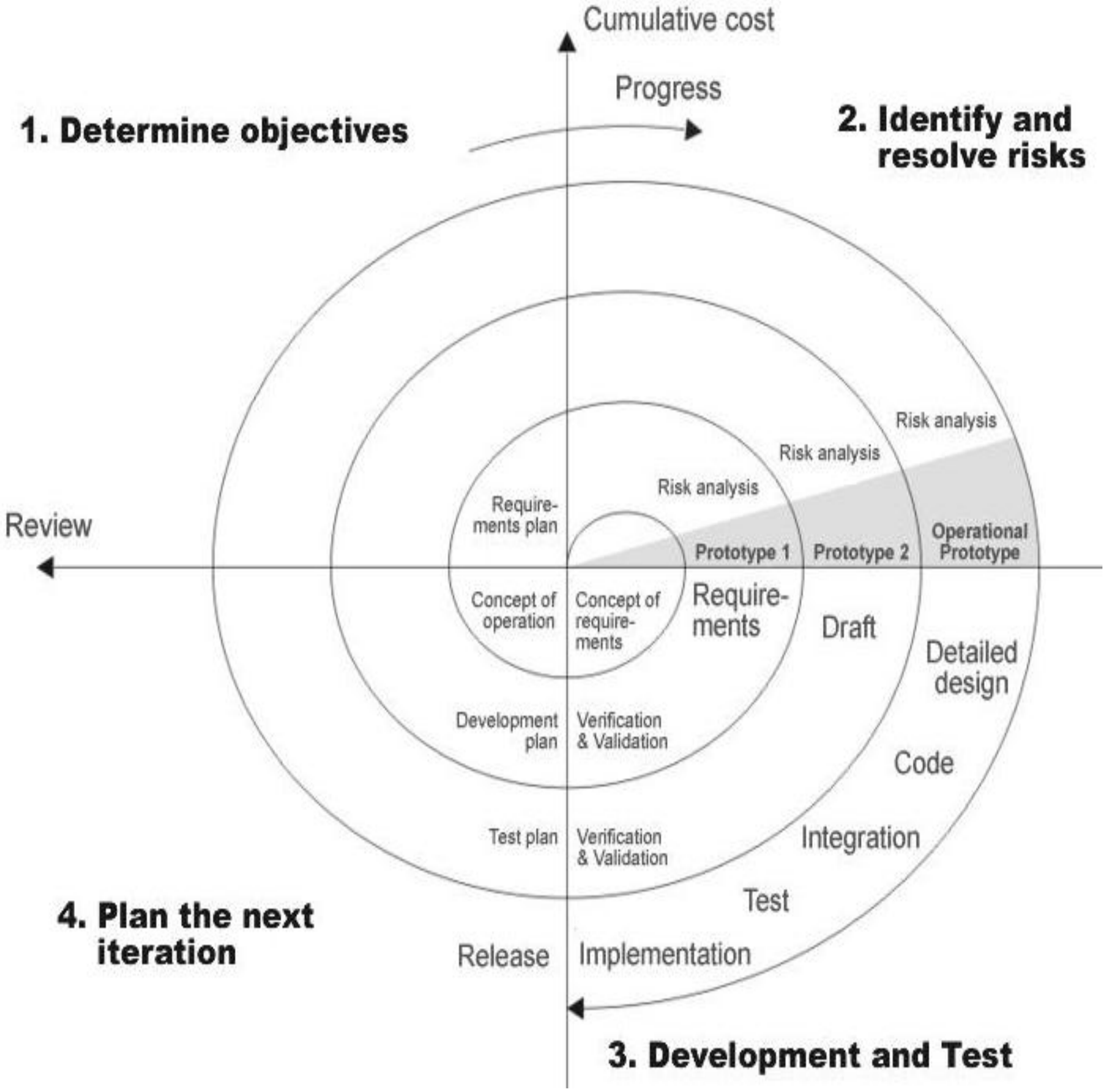
- The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.

- The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.

- The final system is constructed, based on the refined prototype.

- The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

The following diagram shows how a spiral model acts like:



<u>Fig 1.0-Spiral Model</u>

ADVANTAGES:

- Estimates(i.e. budget, schedule etc .) become more relistic as work progresses, because important issues discoved earlier .
- It is more able to cope with the changes that are software development generally entails.
- Software engineers can get their hands in and start woring on the core of a project earlier.

APPLICATION DEVELOPMENT

N-TIER APPLICATIONS

N-Tier Applications can easily implement the concepts of Distributed Application Design and Architecture. The N-Tier Applications provide strategic benefits to Enterprise Solutions. While 2-tier, client-server can help us create quick and easy solutions and may be used for Rapid Prototyping, they can easily become a maintenance and security night mare

The N-tier Applications provide specific advantages that are vital to the business continuity of the enterprise. Typical features of a real life n-tier may include the following:

- Security
- Availability and Scalability
- Manageability
- Easy Maintenance

- Data Abstraction

The above mentioned points are some of the key design goals of a successful n-tier application that intends to provide a good Business Solution.

DEFINITION

Simply stated, an n-tier application helps us distribute the overall functionality into various tiers or layers:

- Presentation Layer
- Business Rules Layer
- Data Access Layer
- Database/Data Store

Each layer can be developed independently of the other provided that it adheres to the standards and communicates with the other layers as per the specifications.

This is the one of the biggest advantages of the n-tier application. Each layer can potentially treat the other layer as a 'Block-Box'.

In other words, each layer does not care how other layer processes the data as long as it sends the right data in a correct format.

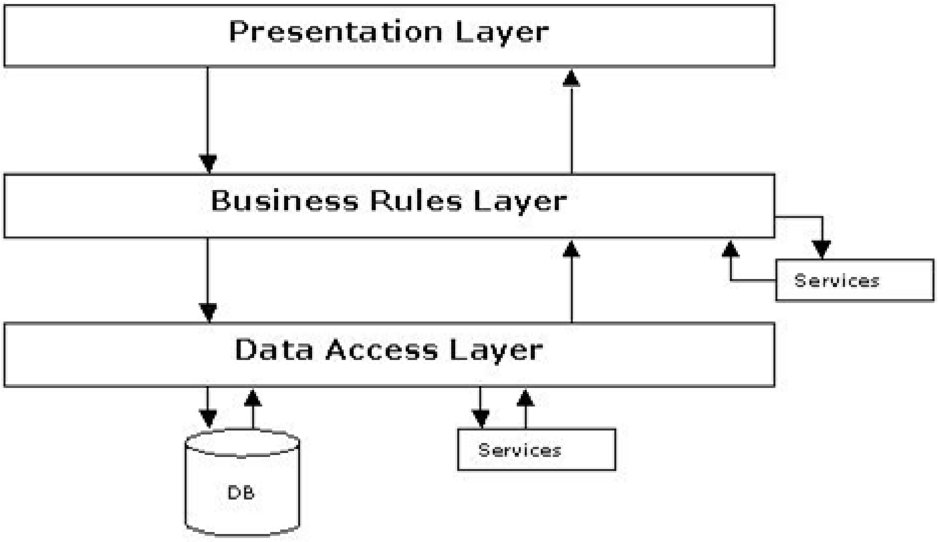


Fig 1.1-N-Tier Architecture

1. **THE PRESENTATION LAYER**

   Also called as the client layer comprises of components that are dedicated to presenting the data to the user. For example: Windows/Web Forms and buttons, edit boxes, Text boxes, labels, grids, etc.

2. **THE BUSINESS RULES LAYER**

   This layer encapsulates the Business rules or the business logic of the encapsulations. To have a separate layer for business logic is of a great advantage. This is because any changes in Business Rules can be easily handled in this layer. As long as the interface between the layers remains the same, any changes to the functionality/processing logic in this layer can be made without impacting the others. A lot of client-server apps failed to implement successfully as changing the business logic was a painful process.

3. THE DATA ACCESS LAYER

This layer comprises of components that help in accessing the Database. If used in the right way, this layer provides a level of abstraction for the database structures. Simply put changes made to the database, tables, etc do not affect the rest of the application because of the Data Access layer. The different application layers send the data requests to this layer and receive the response from this layer.

4. THE DATABASE LAYER

This layer comprises of the Database Components such as DB Files, Tables, Views, etc. The Actual database could be created using SQL Server, Oracle, Flat files, etc.

In an n-tier application, the entire application can be implemented in such a way that it is independent of the actual Database. For instance, you could change the Database Location with minimal changes to Data Access Layer. The rest of the Application should remain unaffected.

SOFTWARE REQUIREMENT

Software Requirements:

Operating System	:	Windows
Graphical User Interface	:	Java Swing, AWT.
Application Logic	:	Java 7.
Database	:	Oracle 10g
IDE/Workbench	:	My Eclipse 6.0.

HARDWARE REQUIREMENT

Hardware Requirements:

- **System Configuration**

Processor	:	Pentium III – 900 MHz
Hard Disk	:	20 GB
RAM	:	128 MB

SYSTEM DESIGN

During analysis, the focus is on what needs to be done, independent of how it is done. During design, decisions are made about how the problem will be solved, first at high level, then at increasingly detailed levels.

System design is the first design stage in which the basic approach to solving the problem is selected. During system design, the overall structure and style are decided. The system architecture is the overall organization of the system into components called subsystems. The architecture provides the context in which more detailed decisions are made in later design stages. By making high level decisions that apply to the entire system, the system designer partitions the problem into subsystems so that further work can be done by several designers working independently on different subsystems.

The system designer must make the following decisions:

- Organize the system into subsystems.
- Identify the concurrency inherent in the problem.
- Allocate subsystems to processors and tasks.
- Choose an approach for management of data stores.
- Handle access to global resources.
- Choose the implementation of control in software.
- Handle boundary conditions.
- Set trade-off priorities.

Breaking the System into Subsystems:

The first step in system design is to divide the system into small number of components. Each major component of a system is called a sub system. Each subsystem encompasses aspects of the system that share some common property – similar functionality, the same physical location, or execution on the same kind of hardware.

A subsystem not an object nor a function but a package of classes, associations, operations, events, and constraints that are interrelated and that have a reasonably well-defined and small interface with other subsystems. A subsystem usually identified by the services it provides. A service is a group of related functions that share some common purpose such as I/O processing. A subsystem defines a coherent way of looking at one aspect of the problem.

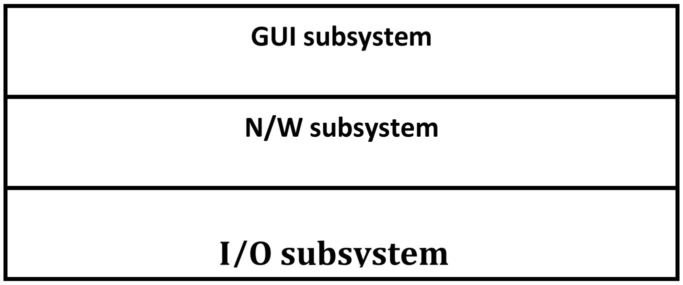
Each subsystem has a well-defined interface to the rest of the system. The interface specifies the form of all interactions and the information flow across subsystem boundaries but does not specify how the sub system is implemented internally. Each subsystem then can be designed independently without affecting the others.

The decomposition of systems into subsystems may be organized as a sequence of horizontal layers or vertical partitions.

A layered system is an ordered set of virtual worlds, each built in terms of the ones below it and providing the basis of implementation for the ones above it. The objects in each layer can be independent, although there is often some correspondence between objects in different layers. Knowledge is one-way only: a subsystem knows about the layers below it, but has no knowledge of the above layers. Each layer may have its own set of classes and operations. Each layer is implemented in terms of the classes and operations of lower layers.

Layered architecture comes in two forms: closed and open. In a closed architecture, each layer is built only in terms of the immediate lower layer. In a open architecture, a layer can use features of any lower layer to any depth.

We decomposed our system into three subsystems as layers. The three layers are closed architecture form. The three layers are GUI layer, Network layer and I/O layer. The purpose of GUI layer is to provide an efficient user interface to the user to interact with the system. It is built upon the Network layer which provides basic FTP services. The lowest layer is the I/O layer that provides services like reading or writing file to and from local and remote systems.



When top-level subsystems are identified, the designer should show the information flow among the sub systems. There are several architectural frameworks that are common in existing systems. They are batch transformation, continuous transformation, interactive interface, dynamic simulation, real-time system and transaction manager.

In the architectural frameworks specified above, our system will best suit in interactive interface architecture, since there are large number of interactions between system and user.

An interactive interface is a system that is dominated by interactions between the system and external agents, such as humans, devices or other programs. The external agents are independent of system, so their inputs can't be controlled, although the system may solicit responses from them.

Identifying Concurrency:

One important goal of system design is to identify which objects must be active concurrently and which objects have activity that is mutually exclusive. The latter objects can be folded together in a single thread of control or task. But there is no part that is concurrent in our system.

Allocating Subsystems to Processor:

In this step system designer estimates the hardware resources required and the implementation choice of either hardware or software. In our system all the subsystems will be implemented in software. The hardware requirements are general such as Pentium – III, 128 MB of RAM.

Management of Data Stores:

In this stage the system designer decides what format is used to store the data stores. There are DBMS systems or file systems and others. Here in our project there are no data stores except files. We then definitely prefer files to download and upload.

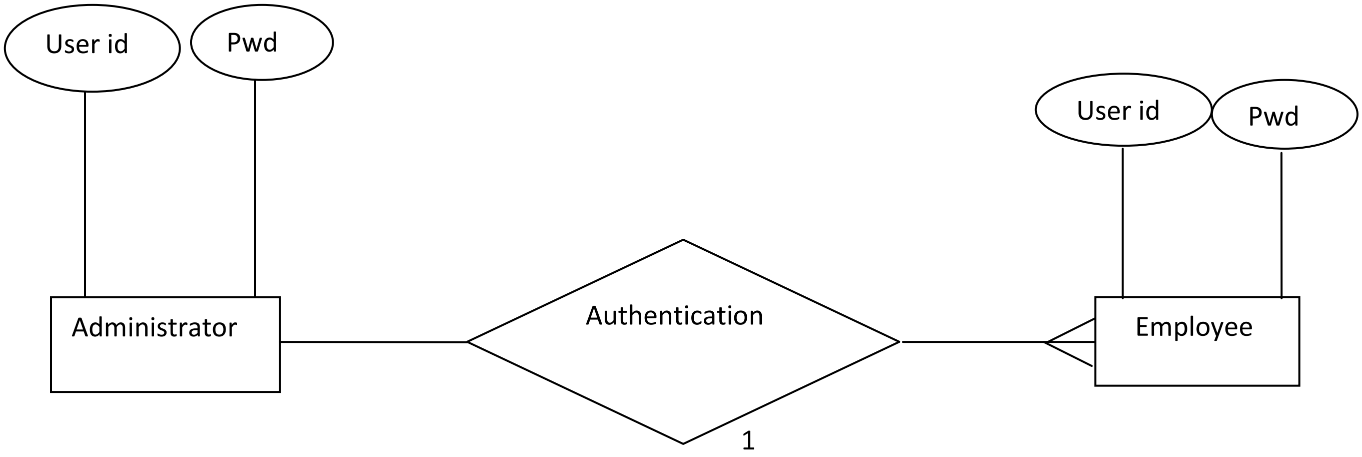
Choosing Software Control Implementation:

During the analysis, all interactions are shown as events between objects. But the system designer must choose among several ways to implement control in software. There are two kinds of control flows in a software system: internal and external. External control is the flow of externally visible events among the objects in the system. There are three kinds of control for external events: procedure driven, event driven sequential and concurrent. Internal control is the flow of control within a process.

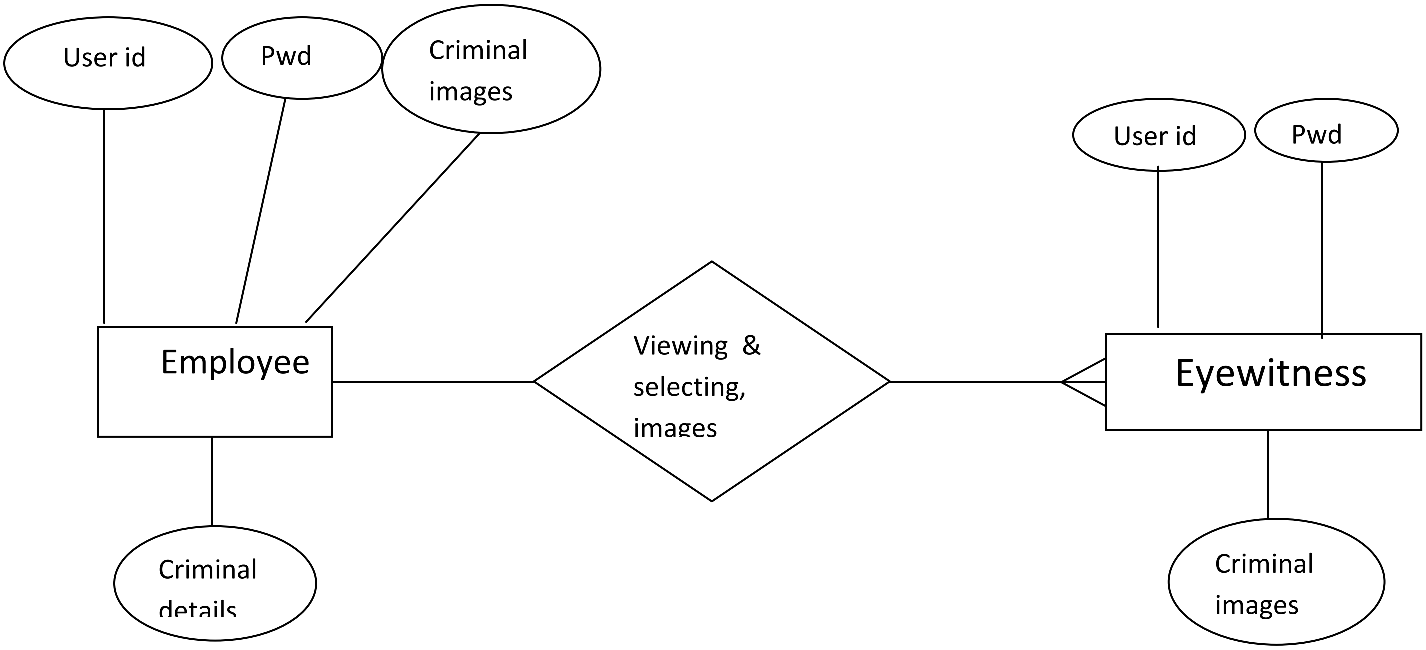
Our system falls in external control flow like event driven control. Events are fired by external agent such as user in different order but sequential not concurrent.

PROCESS FLOW

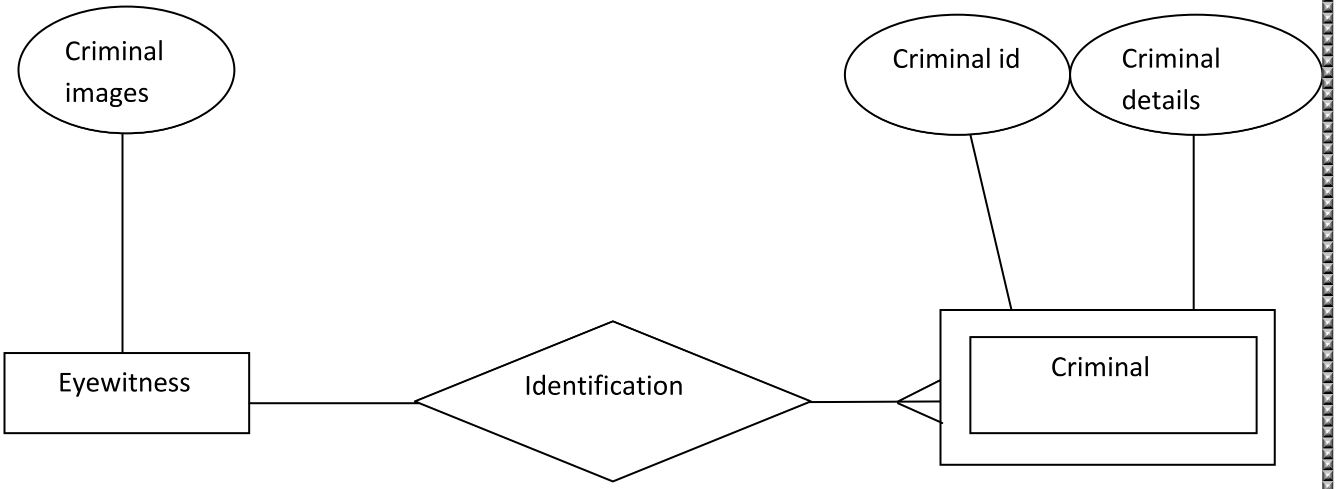
E-RDIAGRAMS BETWEEN ADMINISTRATOR AND EMPLOYEE:



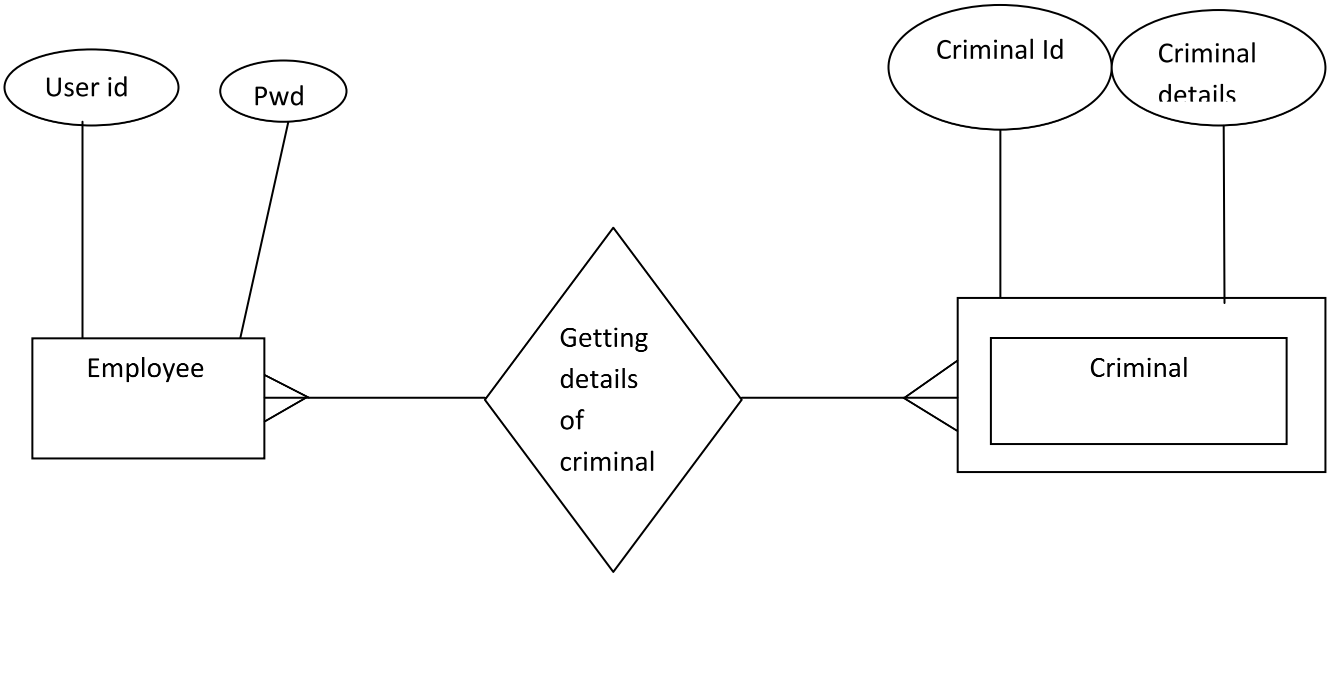
E-R DIAGRAMS BETWEEN EMPLOYEE AND EYEWITNESS:



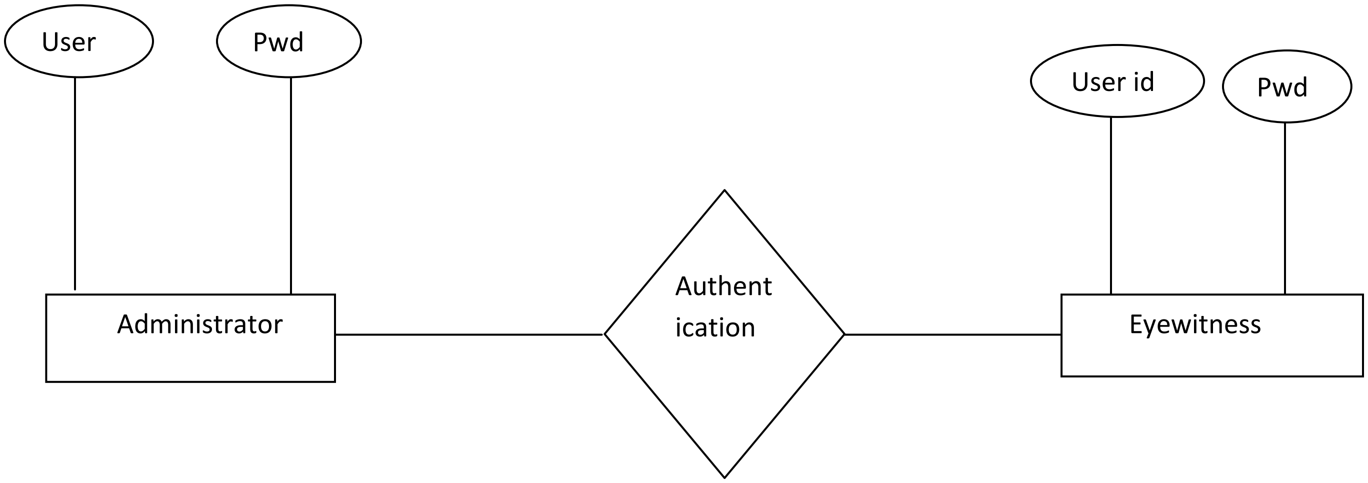
E-R DIAGRAM BETWEEN EYEWITNESS AND CRIMINAL:



E-R DIAGRAM BETWEEN EMPLOYEE AND CRIMINAL:



E-R DIAGRAM BETWEEN EYEWITNESS AND ADMINISTTRATOR



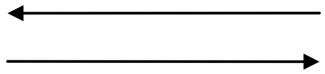
DATA FLOW DIAGRAMS

DATA FLOW DIAGRAMS:

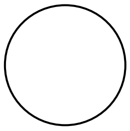
A graphical tool used to describe and analyze the moment of data through a system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed. The transformation of data from input to output, through processes, may be described logically and independently of the physical components associated with the system. The DFD is also know as a data flow graph or a bubble chart.

DFDs are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system's structure charts. The Basic Notation used to create a DFD's are as follows:

1. Dataflow: Data move in a specific direction from an origin to a destination.

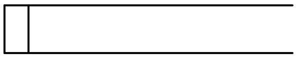


2. Process: People, procedures, or devices that use or produce (Transform) Data. The physical component is not identified.



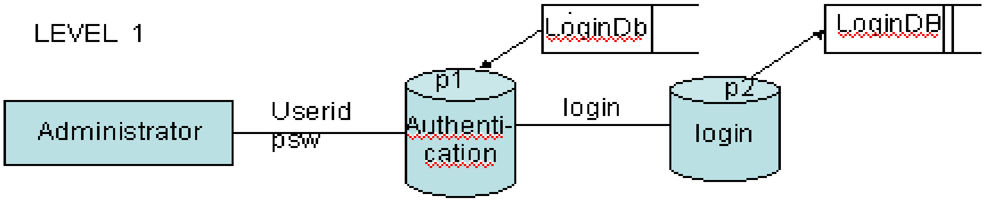
3. Source: External sources or destination of data, which may be People, programs, organizations or other entities.

4. Data Store: Here data are stored or referenced by a process in the System.

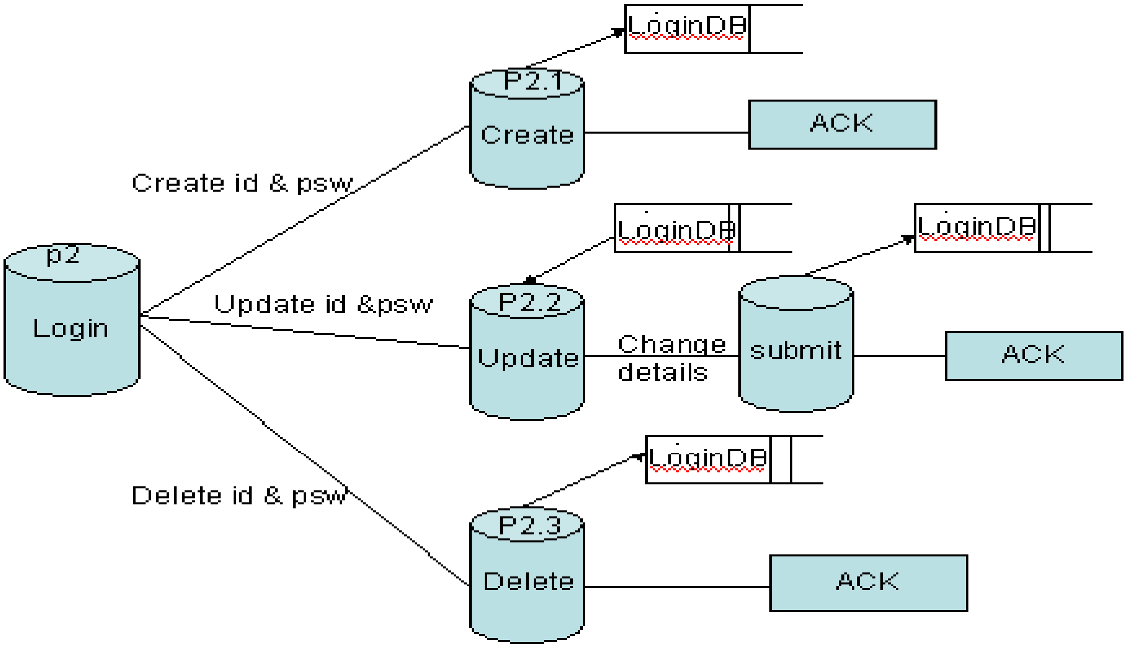


Data Flow Diagrams:

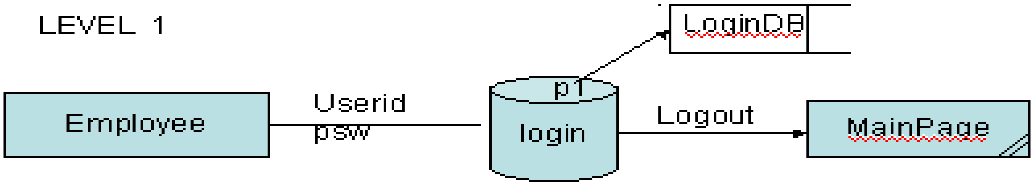
LEVEL 1



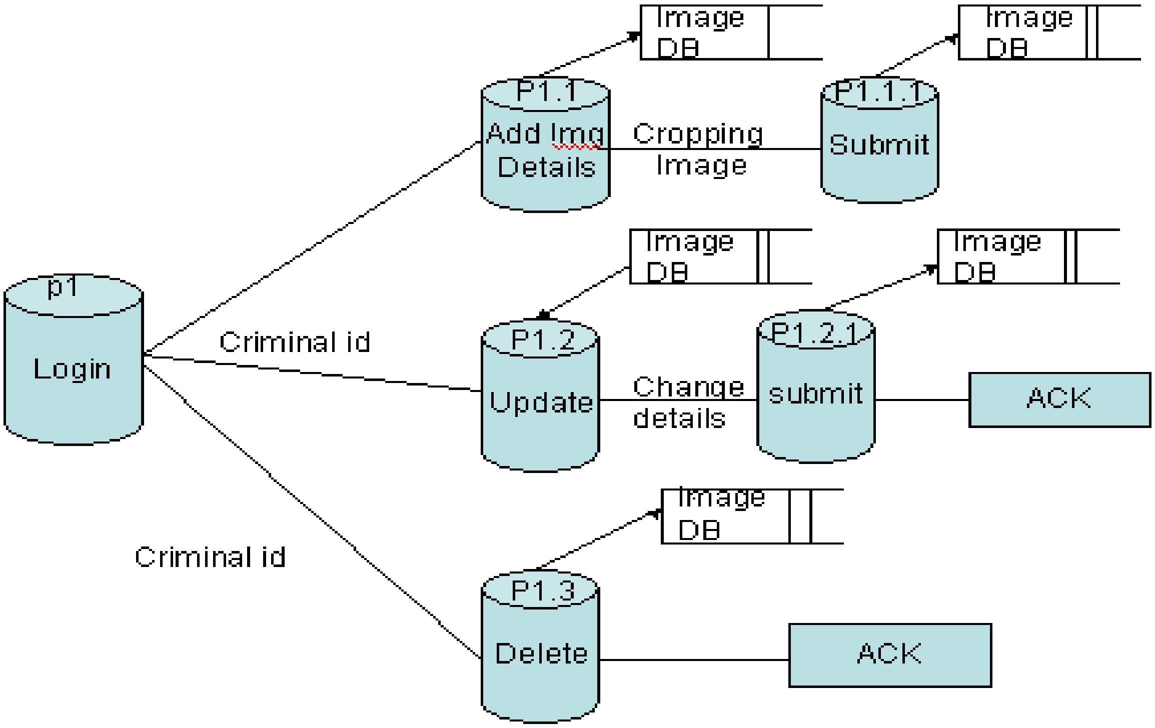
LEVEL 2



DFD FOR Client:

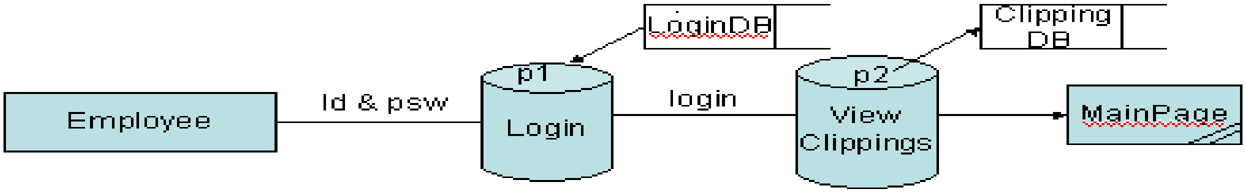


LEVEL-2:

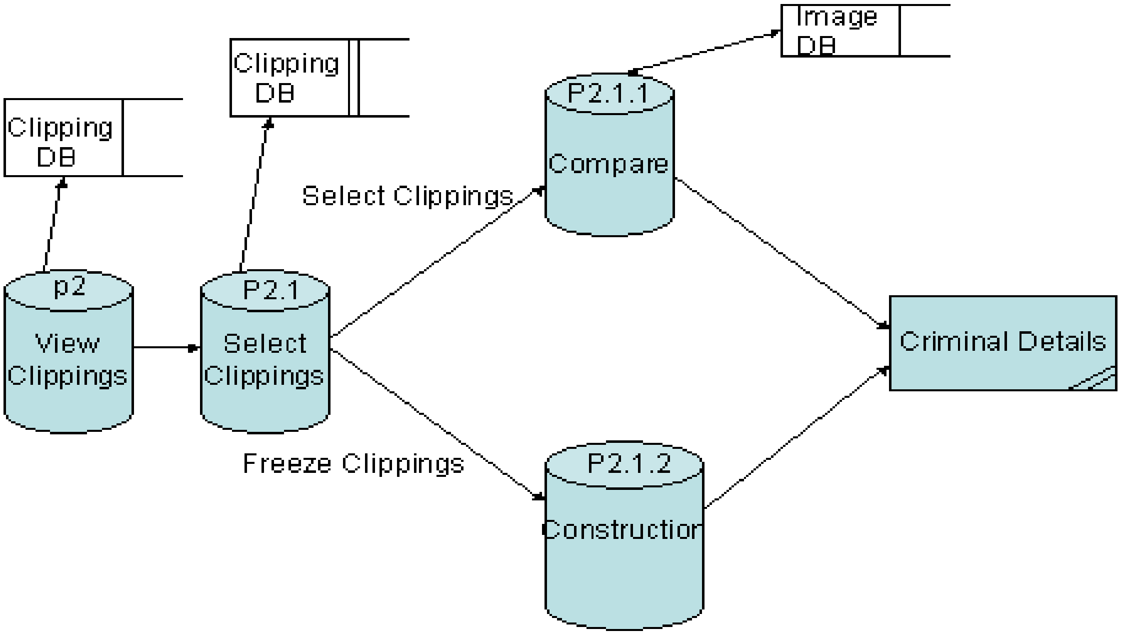


DFD FOR Split & Merge:

LEVEL 1:



LEVEL 2:



UML Diagrams

The Unified Modeling Language (UML) is an open method used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to write a system's blueprints, including conceptual components such as:

- actors,
- business processes and
- system components and activities

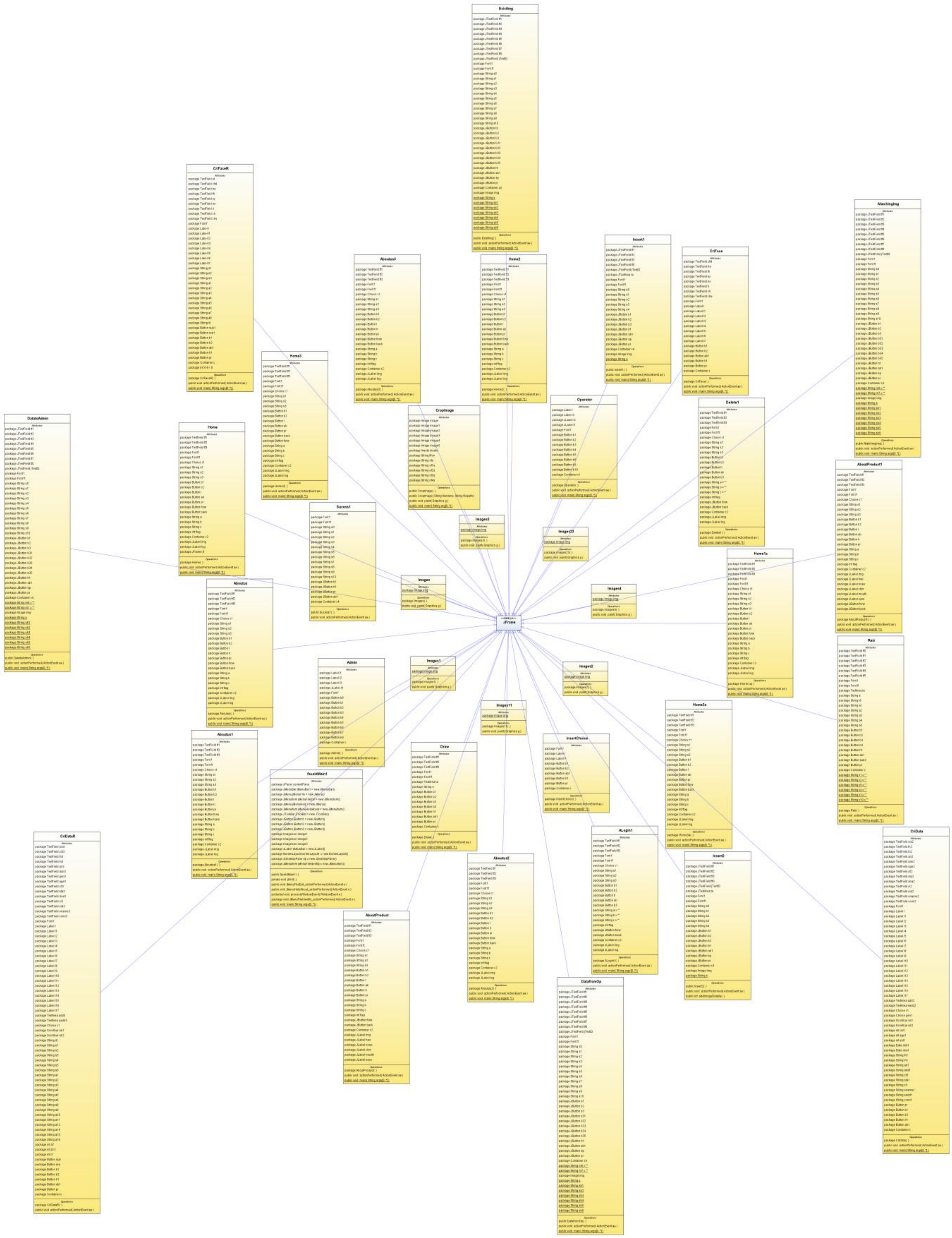
as well as concrete things such as:

- programming language statements,
- database schemas, and
- Reusable software components.

UML combines best practices from data modeling concepts such as entity relationship diagrams, business modeling (work flow), object modeling and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies.[3] UML has succeeded the concepts of the Booch method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling language. UML aims to be a standard modeling language which can model concurrent and distributed systems. UML is not an industry standard, but is taking shape under the auspices of the Object Management Group (OMG). OMG has initially called for information on object-oriented methodologies, that might create a rigorous software modeling language. Many industry leaders have responded in earnest to help create the standard.[1]

UML models may be automatically transformed to other representations (e.g. Java) by means of QVT-like transformation languages, supported by the OMG. UML is extensible, offering the following mechanisms for customization: profiles and stereotype.

Class Diagram:


Existing
CrFaceR
Aboutus3
Home2
Insert1
CrFace
MatchingImg
Home3
DeleteAdmin
Home
CropImage
Operator
Delete1
AboutProduct1
Sucess1
Images3
Images23
Images
Images4
Aboutus
JFrame
Home1s
Retr
Admin
Images1
Images2
Images11
Home2s
Aboutus1
SwebMain1
Draw
InsertChoice
CrDataR
ALogin1
Aboutus2
Insert2
CrData
AboutProduct
DataFromOp

Use case Description:

In software engineering, a use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

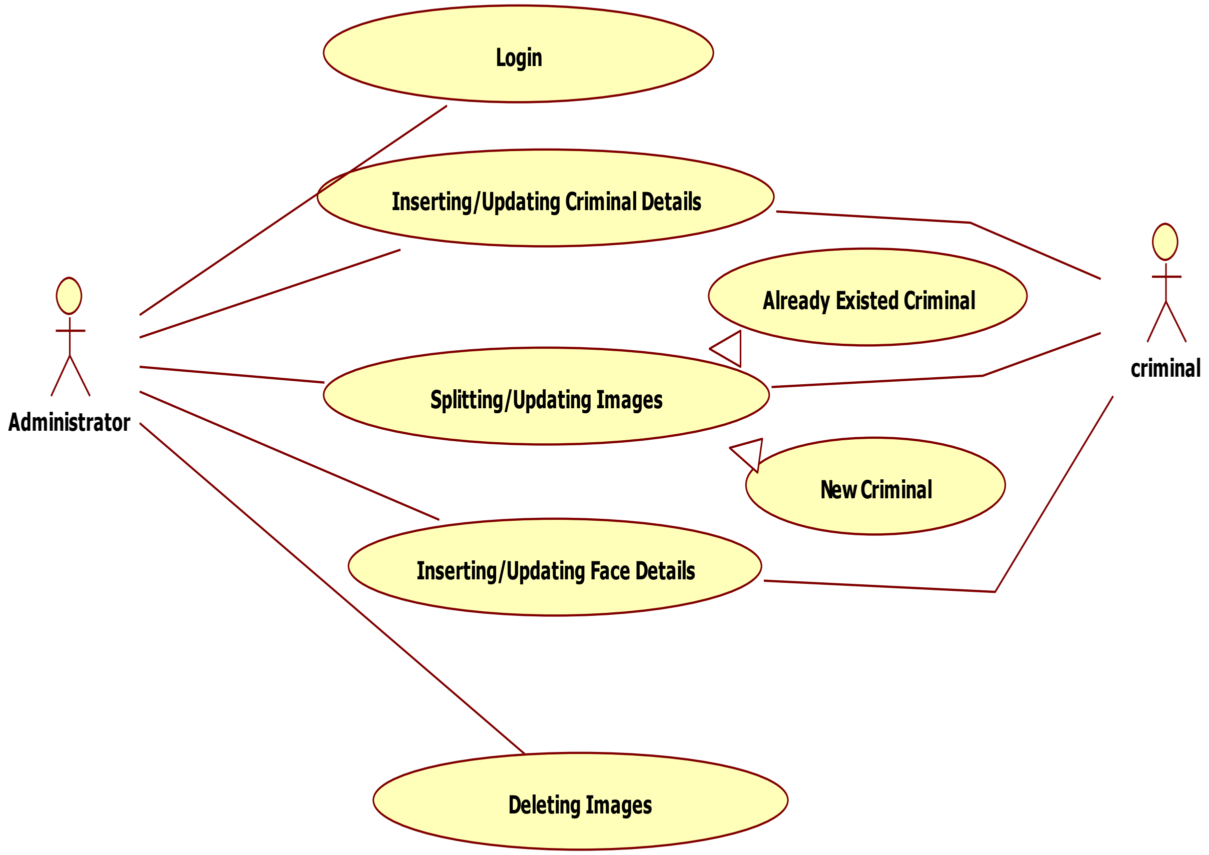
The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

Use Case diagrams are formally included in two modeling languages defined by the OMG. Both the UML and SysML standards define a graphical notation for modeling use cases with diagrams. One complaint about the standards has been that they do not define a format for describing these use cases. Generally, both graphical notation and descriptions are important as they document the use case, showing the purpose for which an actor uses a system.

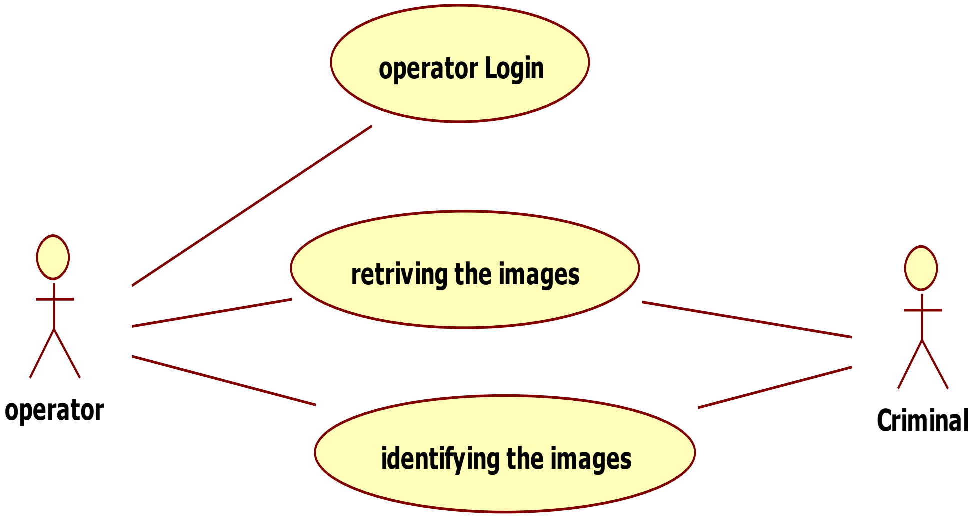
The use case diagram shows the position or context of the use case among other use cases. As an organizing mechanism, a set of consistent, coherent use cases promotes a useful picture of system behavior, a common understanding between the customer/owner/user and the development team.

Use case Diagram:

Administrator:


Login
Inserting/Updating Criminal Details
Already Existed Criminal
Splitting/Updating Images
New Criminal
Inserting/Updating Face Details
Deleting Images
Administrator
criminal

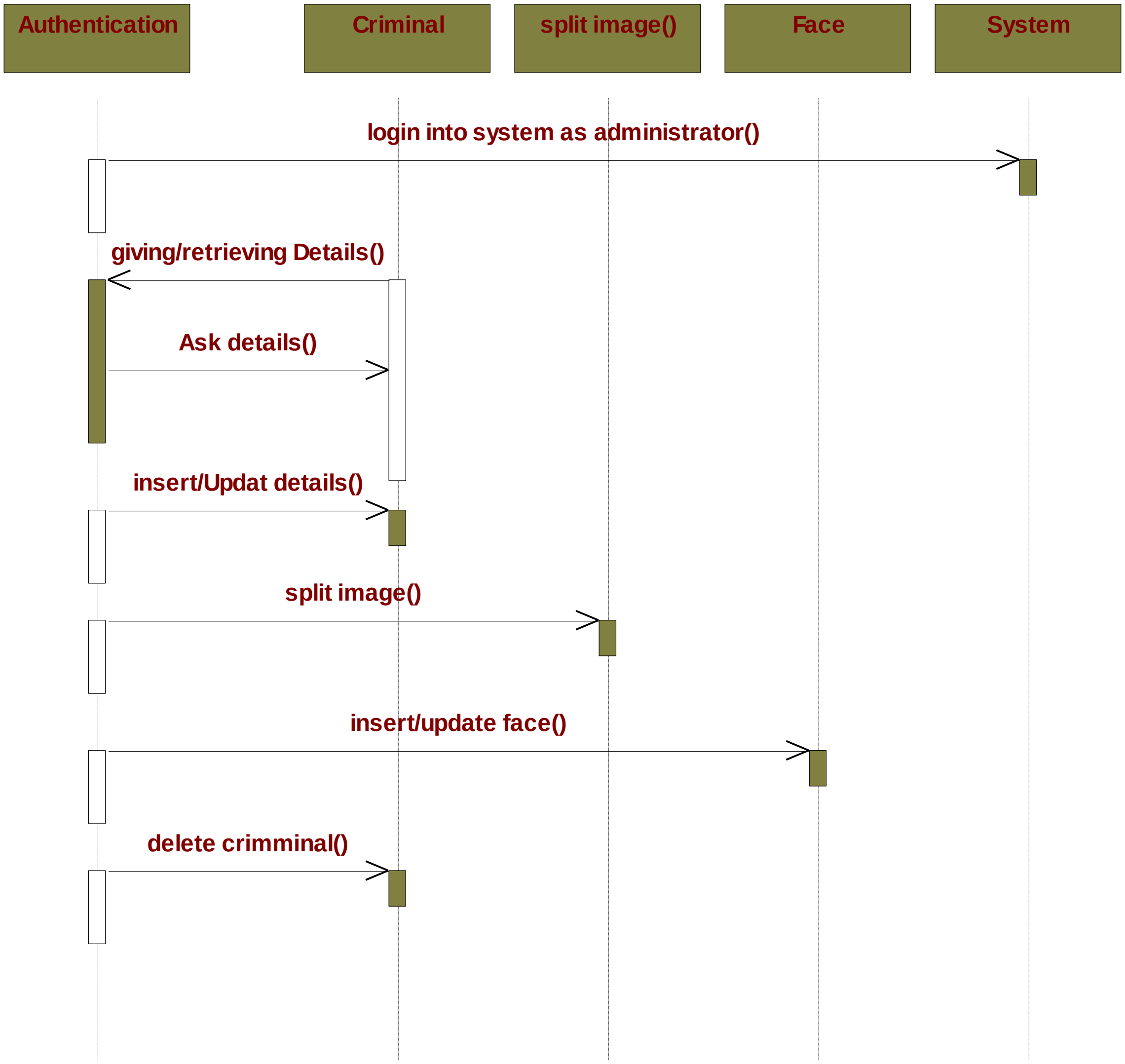
Client (Operator):



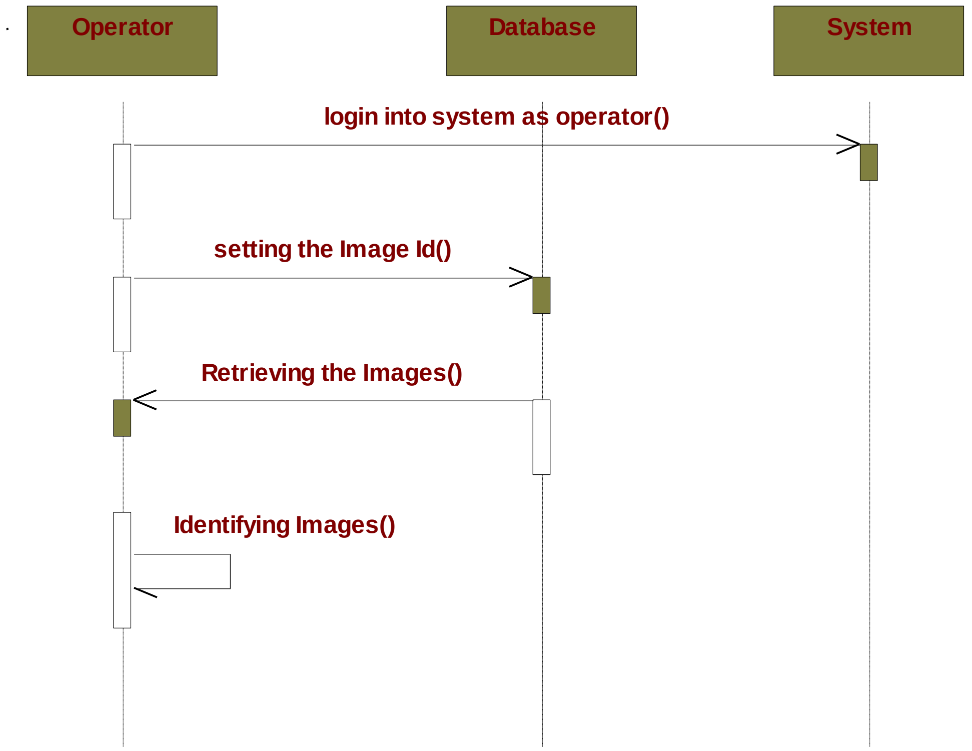
Sequence Diagram:

A sequence diagram in Unified Modelling 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. A sequence diagram shows, as parallel vertical lines ("lifelines"), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

Administrator:


Authentication
Criminal
split image()
Face
System
login into system as administrator()
giving/retrieving Details()
Ask details()
insert/Updat details()
split image()
insert/update face()
delete crimminal()

Operator:



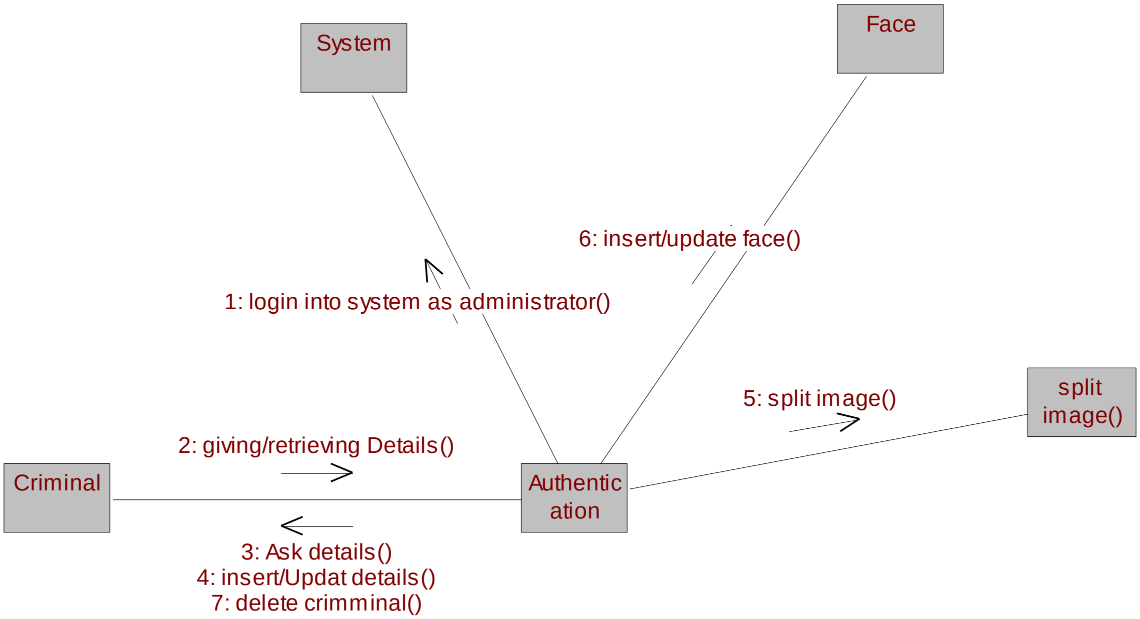
Collaboration Diagram:

A Communication diagram models the interactions between objects or parts in terms of sequenced messages. Communication diagrams represent a combination of information taken from Class, Sequence, and Use Case Diagrams describing both the static structure and dynamic behavior of a system.

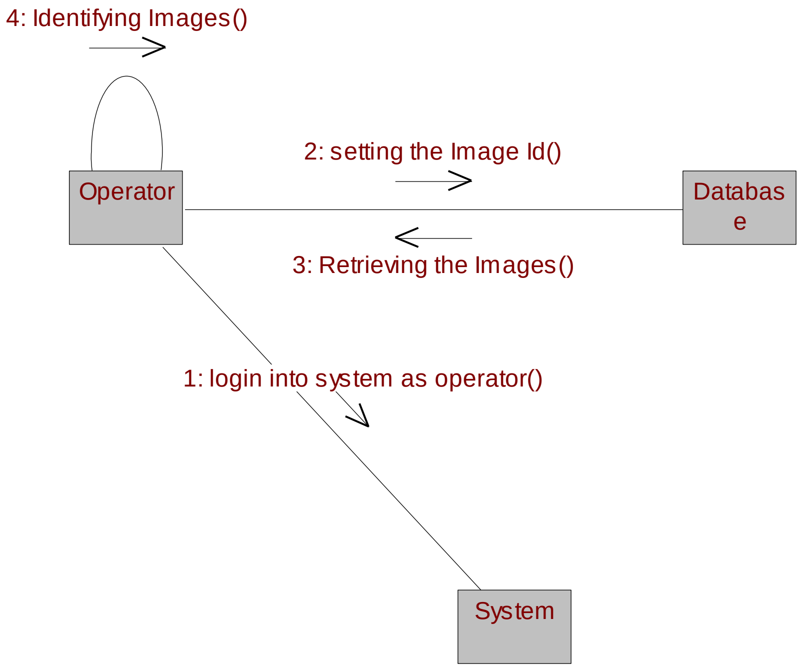
However, communication diagrams use the free-form arrangement of objects and links as used in Object diagrams. In order to maintain the ordering of messages in such a free-form diagram, messages are labeled with a chronological number and placed near the link the message is sent over. Reading a communication diagram involves starting at message 1.0, and following the messages from object to object.

Communication diagrams show a lot of the same information as sequence diagrams, but because of how the information is presented, some of it is easier to find in one diagram than the other. Communication diagrams show which elements each one interacts with better, but sequence diagrams show the order in which the interactions take place more clearly.

Administrator:



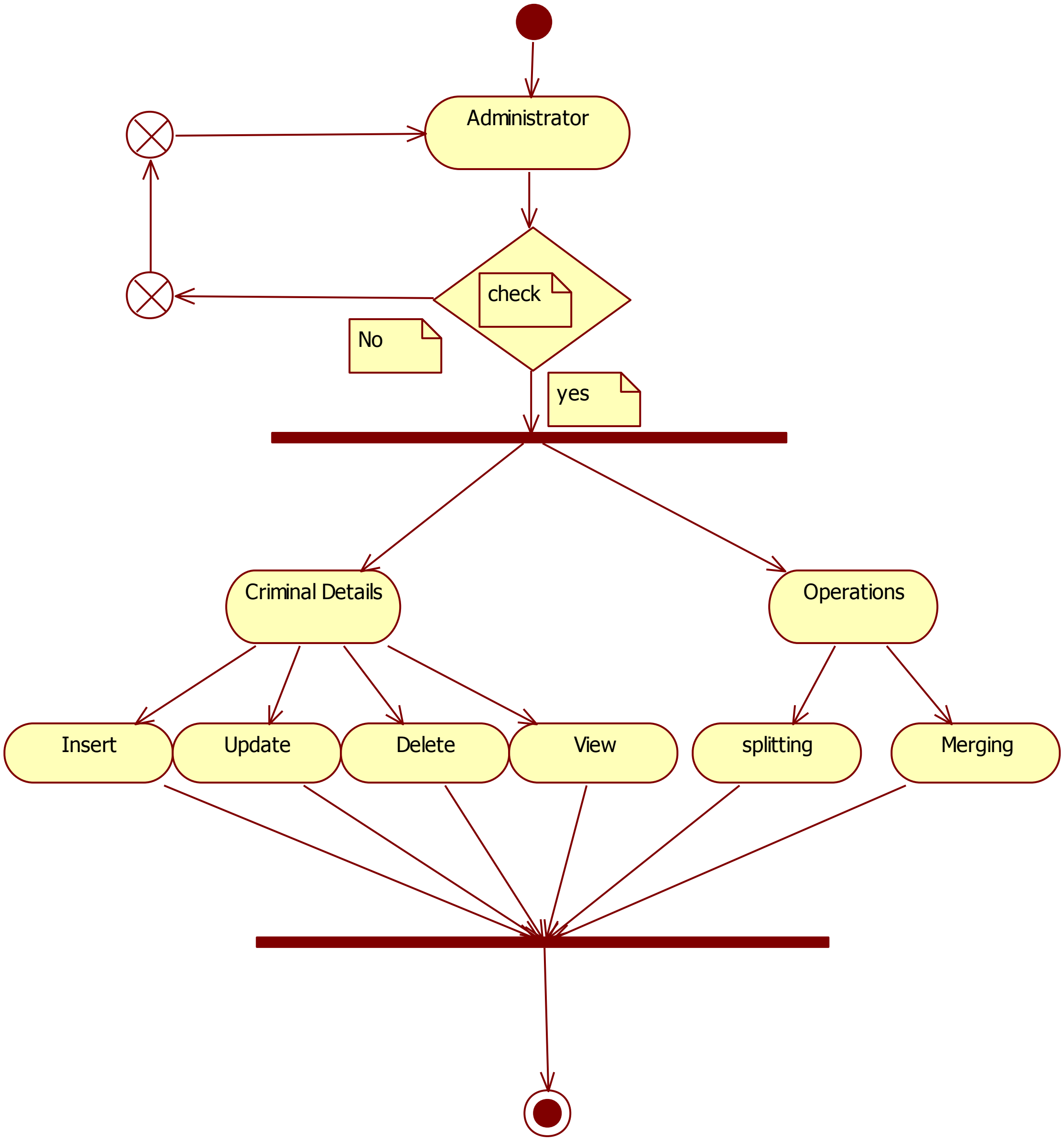
Operator:


4: Identifying Images()
2: setting the Image Id()
Operator
Databas
e
3: Retrieving the Images()
1: login into system as operator()
System

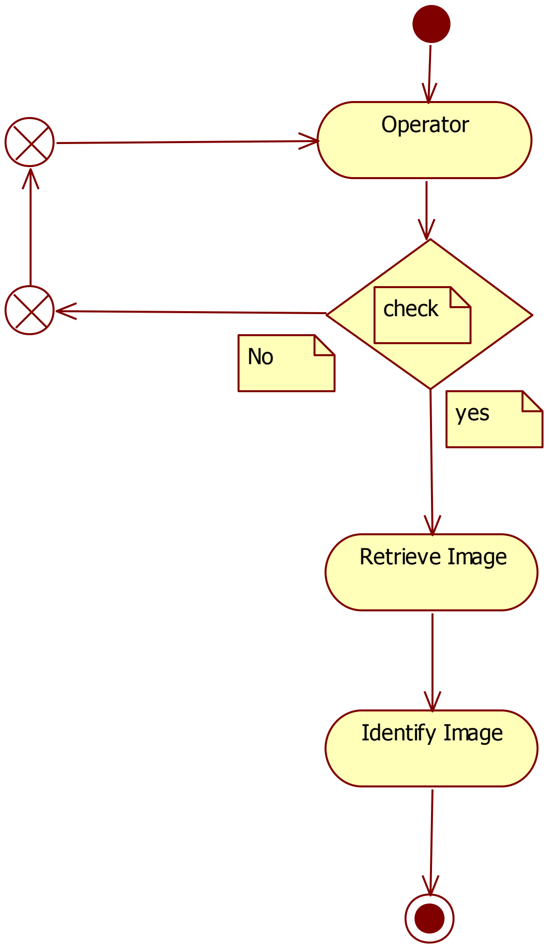
Activity Diagram:

Activity diagrams are a loosely defined diagram technique for showing workflows of stepwise activities and actions, with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. n SysML the activity diagram has been extended to indicate flows among steps that convey physical element (e.g., gasoline) or energy (e.g., torque, pressure).

Administrator:

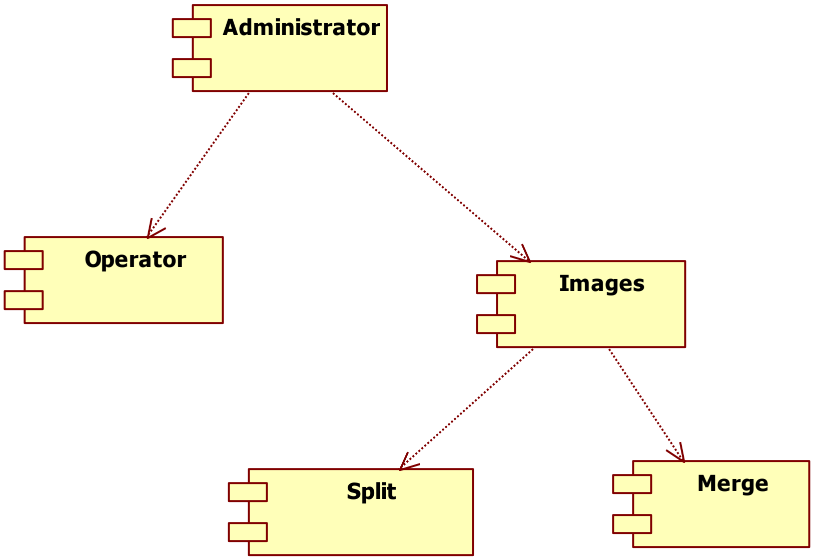


Operator:



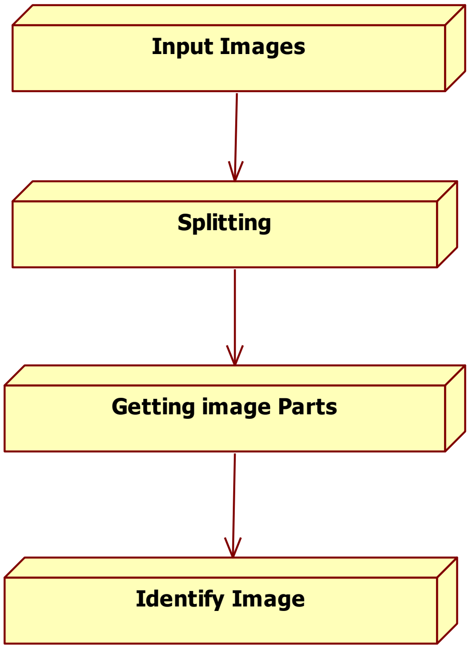
Component Diagram:

A component diagram in the Unified Modeling Language, depicts how components are wired together to form larger components and or software systems. Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components. An assembly connector is a "connector between two components that defines that one component provides the services that another component requires. An assembly connector is a connector that is defined from a required interface or port to a provided interface or port." When using a component diagram to show the internal structure of a component, the provided and required interfaces of the encompassing component can delegate to the corresponding interfaces of the contained components.



Deployment Diagram:

A deployment diagram in the Unified Modeling Language serves to model the physical deployment of artifacts on deployment targets. Deployment diagrams show "the allocation of Artifacts to Nodes according to the Deployments defined between them." Deployment of an artifact to a node is indicated by placing the artifact inside the node. Instances of nodes (and devices and execution environments) are used in deployment diagrams to indicate multiplicity of these nodes. For example, multiple instances of an application server execution environment may be deployed inside a single device node to represent application server clustering.



TECHNOLOGY DESCRIPTION

FEATURES OF THE LANGUAGE USED:

About Java:

Initially the language was called as “oak” but it was renamed as “Java” in 1995. The primary motivation of this language was the need for a platform-independent (i.e., architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

- Java is a programmer’s language.
- Java is cohesive and consistent.
- Except for those constraints imposed by the Internet environment, Java gives the programmer, full control.

Finally, Java is to Internet programming where C was to system programming.

Swings:

Swing, which is an extension library to the AWT, includes new and improved components that enhance the look and functionality of GUIs. Swing can be used to build Standalone swing GUI Apps as well as Servlets and Applets. It employs a model/view design architecture. Swing is more portable and more flexible than AWT.

Swing is built on top of AWT and is entirely written in Java, using AWT’s lightweight component support. In particular, unlike AWT, t he architecture of Swing components makes it easy to customize both their appearance and behavior. Components from AWT and Swing can be mixed, allowing you to add Swing support to existing AWT-based programs. For example, swing components such as JSlider, JButton and JCheckbox could be used in the same program with standard AWT labels, textfields and scrollbars. You could subclass the existing Swing UI, model, or change listener classes without having to reinvent the entire implementation. Swing also has the ability to replace these objects on-the-fly.

- 100% Java implementation of components
- Pluggable Look & Feel
- Lightweight components
- **Uses MVC Architecture**

  Model represents the data

  View as a visual representation of the data

  Controller takes input and translates it to changes in data
- **Three parts**

  Component set (subclasses of JComponent)

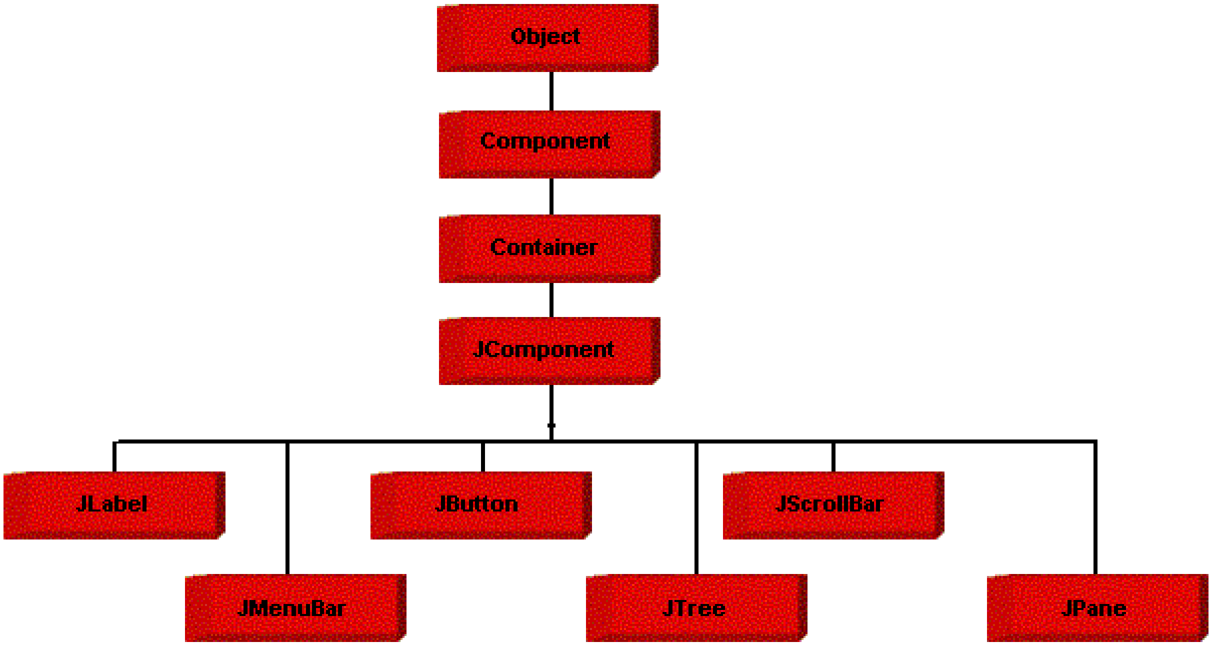
  Support classes

  Interfaces

In Swing, classes that represent GUI components have names beginning with the letter J. Some examples are JButton, JLabel, and JSlider. Altogether there are more than 250 new classes and 75 interfaces in Swing — twice as many as in AWT.

Java Swing class hierarchy

The class JComponent, descended directly from Container, is the root class for most of Swing's user interface components.



Swing contains components that you'll use to build a GUI. I am listing you some of the commonly used Swing components. To learn and understand these swing programs, AWT Programming knowledge **is not** required.

Applications and Applets

An application is a program that runs on our Computer under the operating system of that computer. It is more or less like one creating using C or C++. Java's ability to create Applets makes it important. An Applet is an application designed, to be transmitted over the Internet and executed by a Java –compatible web browser. An applet is actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can react to the user input and dynamically change.

FEATURES OF JAVA:

Security

Every time you that you download a “normal” program, you are risking a viral infection. Prior to Java, most users did not download executable programs frequently, and those who did scan them for viruses prior to execution. Most users still worried about the possibility of infecting their systems with a virus. In addition, another type of malicious program exists that must be guarded against. This type of program can gather private information, such as credit card numbers, bank account balances, and passwords. Java answers both of these concerns by providing a “firewall” between a networked application and your computer. When you use a Java-compatible Web browser, you can safely download Java applets without fear of virus infection or malicious intent.

Portability

For programs to be dynamically downloaded to all the various types of platforms connected to the Internet, some means of generating portable executable code is needed .As you will see, the same mechanism that helps ensure security also helps create portability. Indeed, Java’s solution to these two problems is both elegant and efficient.

The Byte code

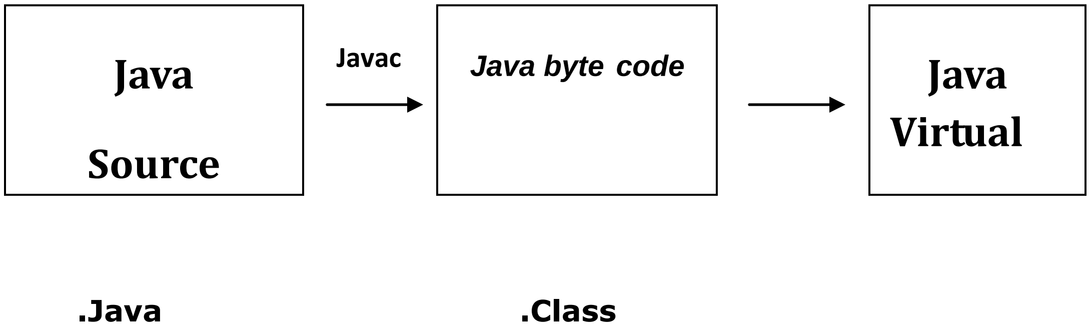
The key that allows the Java to solve the security and portability problem is that the output of Java compiler is Byte code. Byte code is a highly optimized set of instructions designed to execute by the Java run-time system, which is called the Java Virtual Machine (JVM). That is, in its standard form, the JVM is an interpreter for byte code.

Translating a Java program into byte code helps makes it much easier to run a program in a wide variety of environments. The reason is, once the run-time package exists for a given system, any Java program can run on it.

Although Java was designed for interpretation, there is technically nothing about Java that prevents on-the-fly compilation of byte code into native code. Sun has just completed its Just In Time (JIT) compiler for byte code. When the JIT compiler is a part of JVM, it compiles byte code into executable code in real time, on a piece-by-piece, demand basis. It is not possible to compile an entire Java program into executable code all at once, because Java performs various run-time checks that can be done only at run time. The JIT compiles code, as it is needed, during execution.

Java Virtual Machine (JVM)

Beyond the language, there is the Java virtual machine. The Java virtual machine is an important element of the Java technology. The virtual machine can be embedded within a web browser or an operating system. Once a piece of Java code is loaded onto a machine, it is verified. As part of the loading process, a class loader is invoked and does byte code verification makes sure that the code that’s has been generated by the compiler will not corrupt the machine that it’s loaded on. Byte code verification takes place at the end of the compilation process to make sure that is all accurate and correct. So byte code verification is integral to the compiling and executing of Java code.



The above picture shows the development process a typical Java programming uses to produce byte codes and executes them. The first box indicates that the Java source code is located in a. Java file that is processed with a Java compiler called **JAVA**. The Java compiler produces a file called a. class file, which contains the byte code. The class file is then loaded across the network or loaded locally on your machine into the execution environment is the Java virtual machine, which interprets and executes the byte code.

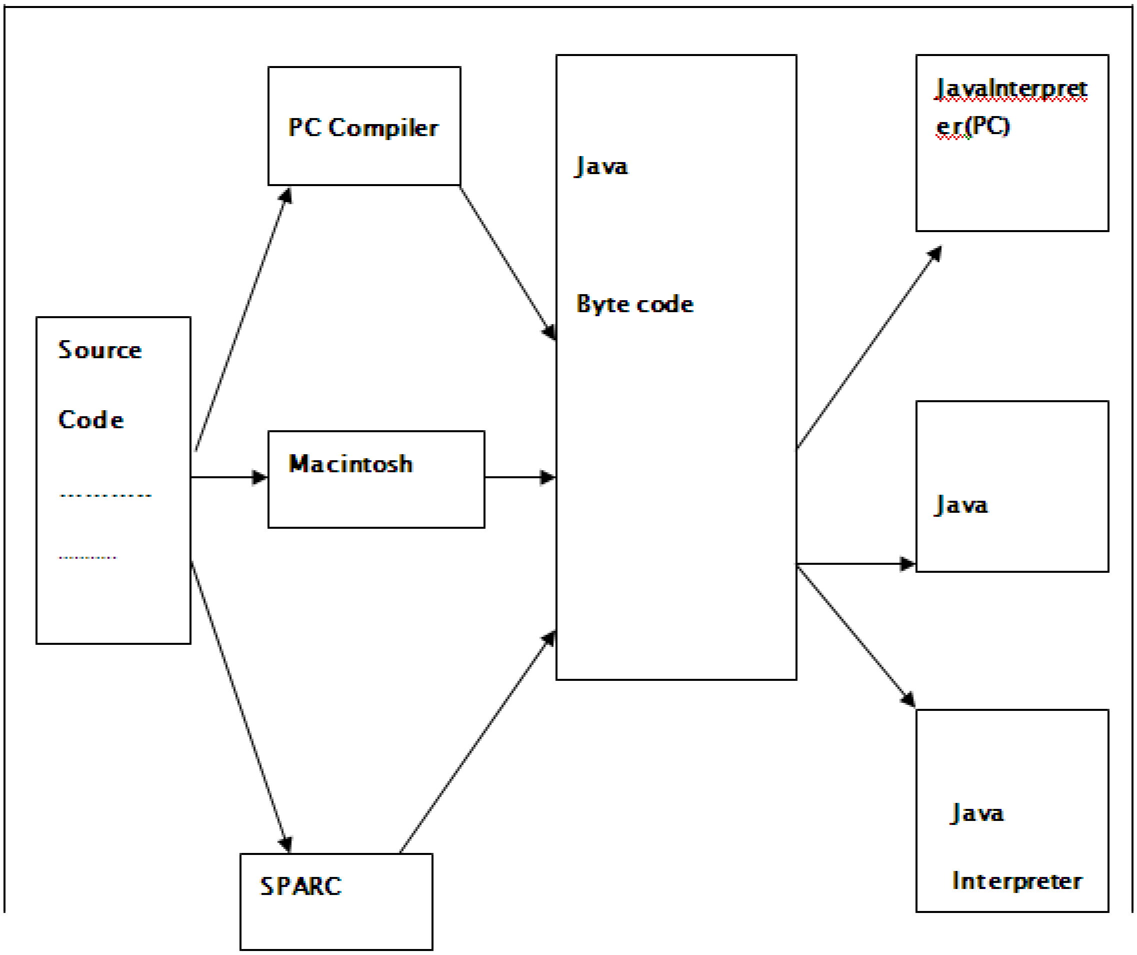
Java Architecture

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet.

Compilation of Code

When you compile the code, the Java compiler creates machine code (called byte code) for a hypothetical machine called Java Virtual Machine (JVM). The JVM is supposed to execute the byte code. The JVM is created for overcoming the issue of portability. The code is written and compiled for one machine and interpreted on all machines. This machine is called Java Virtual Machine.

Compiling and interpreting Java Source Code



During run-time the Java interpreter tricks the byte code file into thinking that it is running on a Java Virtual Machine. In reality this could be a Intel Pentium Windows 95 or Suns ARC station running Solaris or Apple Macintosh running system and all could receive code from any computer through Internet and run the Applets.

SIMPLE

Java was designed to be easy for the Professional programmer to learn and to use effectively. If you are an experienced C++ programmer, learning Java will be even easier. Because Java inherits the C/C++ syntax and many of the object oriented features of C++. Most of the confusing concepts from C++ are either left out of Java or implemented in a cleaner, more approachable manner. In Java there are a small number of clearly defined ways to accomplish a given task.

Object-Oriented

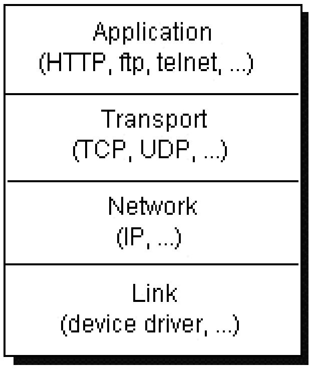
Java was not designed to be source-code compatible with any other language. This allowed the Java team the freedom to design with a blank slate. One outcome of this was a clean usable, pragmatic approach to objects. The object model in Java is simple and easy to extend, while simple types, such as integers, are kept as high-performance non-objects.

Robust

The multi-platform environment of the Web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. The ability to create robust programs was given a high priority in the design of Java. Java is strictly typed language; it checks your code at compile time and run time. Java virtually eliminates the problems of memory management and de-allocation, which is completely automatic. In a well-written Java program, all run time errors can –and should –be managed by your program.

What is networking?

Computers running on the Internet communicate to each other using either the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP), as this diagram illustrates:



When you write Java programs that communicate over the network, you are programming at the application layer. Typically, you don't need to concern yourself with the TCP and UDP layers. Instead, you can use the classes in the java.net package. These classes provide system-independent network communication. However, to decide which Java classes your programs should use, you do need to understand how TCP and UDP differ.

TCP

When two applications want to communicate to each other reliably, they establish a connection and send data back and forth over that connection.This is analogous to making a telephone call. If you want to speak to Aunt Beatrice in Kentucky, a connection is established when you dial her phone number and she answers. You send data back and forth over the connection by speaking to one another over the phone lines. Like the phone company, TCP guarantees that data sent from one end of the connection actually gets to the other end and in the same order it was sent. Otherwise, an error is reported.

TCP provides a point-to-point channel for applications that require reliable communications. The Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), and Telnet are all examples of applications that require a reliable communication channel. The order in which the data is sent and received over the network is critical to the success of these applications. When HTTP is used to read from a URL, the data must be received in the order in which it was sent. Otherwise, you end up with a jumbled HTML file, a corrupt zip file, or some other invalid information.

Definition: TCP (*Transmission Control Protocol*) is a connection-based protocol that provides a reliable flow of data between two computers.

UDP

The UDP protocol provides for communication that is not guaranteed between two applications on the network. UDP is not connection-based like TCP. Rather, it sends independent packets of data, called *datagrams*, from one application to another. Sending datagrams is much like sending a letter through the postal service: The order of delivery is not important and is not guaranteed, and each message is independent of any other.

Definition: UDP (*User Datagram Protocol*) is a protocol that sends independent packets of data, called datagram's, from one computer to another with no guarantees about arrival. UDP is not connection-based like TCP.

For many applications, the guarantee of reliability is critical to the success of the transfer of information from one end of the connection to the other. However, other forms of communication don't require such strict standards. In fact, they may be slowed down by the extra overhead or the reliable connection may invalidate the service altogether.

Consider, for example, a clock server that sends the current time to its client when requested to do so. If the client misses a packet, it doesn't really make sense to resend it because the time will be incorrect when the client receives it on the second try. If the client makes two requests and receives packets from the server out of order, it doesn't really matter because the client can figure out that the packets are out of order and make another request. The reliability of TCP is unnecessary in this instance because it causes performance degradation and may hinder the usefulness of the service.

Another example of a service that doesn't need the guarantee of a reliable channel is the ping command. The purpose of the ping command is to test the communication between two programs over the network. In fact, ping needs to know about dropped or out-of-order packets to determine how good or bad the connection is. A reliable channel would invalidate this service altogether.

The UDP protocol provides for communication that is not guaranteed between two applications on the network. UDP is not connection-based like TCP. Rather, it sends independent packets of data from one application to another. Sending datagram's is much like sending a letter through the mail service: The order of delivery is not important and is not guaranteed, and each message is independent of any others.

Note: Many firewalls and routers have been configured not to allow UDP packets. If you're having trouble connecting to a service outside your firewall, or if clients are having trouble connecting to your service, ask your system administrator if UDP is permitted.

FTP Protocol:

File Transfer Protocol (**FTP**) is a standard network protocol used to exchange and manipulate files over an Internet Protocol computer network, such as the Internet. FTP is built on a client-server architecture and utilizes separate control and data connections between the client and server applications. Client applications were originally interactive command-line tools with a standardized command syntax, but graphical user interfaces have been developed for all desktop operating systems in use today. FTP is also often used as an application component to automatically transfer files for program internal functions. FTP can be used with user-based password a While data is being transferred via the data stream, the control stream sits idle. This can cause problems with large data transfers through firewalls which time out sessions after lengthy periods of idleness. While the file may well be successfully transferred, the control session can be disconnected by the firewall, causing an error to be generated.

The FTP protocol supports resuming of interrupted downloads using the REST command. The client passes the number of bytes it has already received as argument to the REST command and restarts the transfer. In some commandline clients for example, there is an often-ignored but valuable command, "reget" (meaning "get again"), that will cause an interrupted "get" command to be continued, hopefully to completion, after a communications interruption.

Resuming uploads is not as easy. Although the FTP protocol supports the APPE command to append data to a file on the server, the client does not know the exact position at which a transfer got interrupted. It has to obtain the size of the file some other way, for example over a directory listing or using the SIZE command.

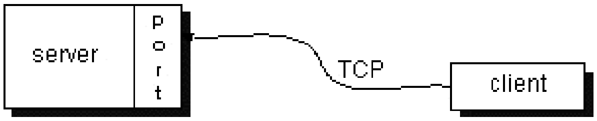
In ASCII mode (see below), resuming transfers can be troublesome if client and server use different end of line characters.

Understanding Ports

Generally speaking, a computer has a single physical connection to the network. All data destined for a particular computer arrives through that connection. However, the data may be intended for different applications running on the computer. So how does the computer know to which application to forward the data? Through the use of ports.

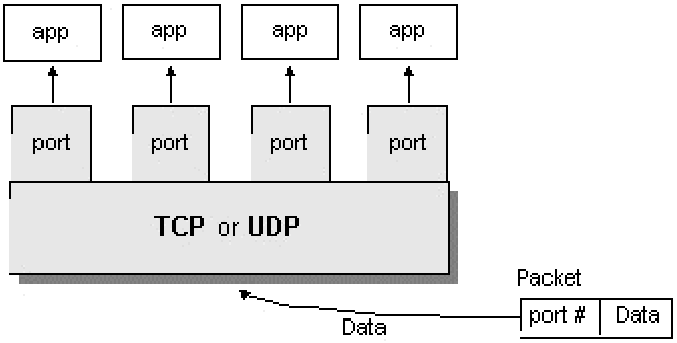
Data transmitted over the Internet is accompanied by addressing information that identifies the computer and the port for which it is destined. The computer is identified by its 32-bit IP address, which IP uses to deliver data to the right computer on the network. Ports are identified by a 16-bit number, which TCP and UDP use to deliver the data to the right application.

In connection-based communication such as TCP, a server application binds a socket to a specific port number. This has the effect of registering the server with the system to receive all data destined for that port. A client can then rendezvous with the server at the server's port, as illustrated here:



Definition: The TCP and UDP protocols use ports to map incoming data to a particular process running on a computer.

In datagram-based communication such as UDP, the datagram packet contains the port number of its destination and UDP routes the packet to the appropriate application, as illustrated in this figure:



Port numbers range from 0 to 65,535 because ports are represented by 16-bit numbers. The port numbers ranging from 0 - 1023 are restricted; they are reserved for use by well-known services such as HTTP and FTP and other system services.

These ports are called *well-known ports*. Your applications should not attempt to bind to them.

Networking Classes in the JDK

Through the classes in java.net, Java programs can use TCP or UDP to communicate over the Internet. The URL, URL Connection, Socket, and Server Socket classes all use TCP to communicate over the network. The Datagram Packet, Datagram Socket, and Multicast Socket classes are for use with UDP.

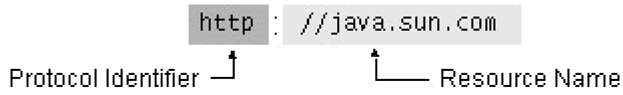
What Is a URL*?*

If you've been surfing the Web, you have undoubtedly heard the term URL and have used URLs to access HTML pages from the Web.

It's often easiest, although not entirely accurate, to think of a URL as the name of a file on the World Wide Web because most URLs refer to a file on some machine on the network. However, remember that URLs also can point to other resources on the network, such as database queries and command output.

Definition: URL is an acronym for *Uniform Resource Locator* and is a reference (an address) to a resource on the Internet.

The following is an example of a URL which addresses the Java Web site hosted by Sun Microsystems:



As in the previous diagram, a URL has two main components:

- Protocol identifier
- Resource name

Note that the protocol identifier and the resource name are separated by a colon and two forward slashes. The protocol identifier indicates the name of the protocol to be used to fetch the resource. The example uses the Hypertext Transfer Protocol (HTTP), which is typically used to serve up hypertext documents. HTTP is just one of many different protocols used to access different types of resources on the net. Other protocols include File Transfer Protocol (FTP), Gopher, File, and News.

The resource name is the complete address to the resource. The format of the resource name depends entirely on the protocol used, but for many protocols, including HTTP, the resource name contains one or more of the components listed in the following table:

Host Name	The name of the machine on which the resource lives.
Filename	The pathname to the file on the machine.

Port Number	The port number to which to connect (typically optional).
Reference	A reference to a named anchor within a resource that usually identifies a specific location within a file (typically optional).

For many protocols, the host name and the filename are required, while the port number and reference are optional. For example, the resource name for an HTTP URL must specify a server on the network (Host Name) and the path to the document on that machine (Filename); it also can specify a port number and a reference. In the URL for the Java Web site java.sun.com is the host name and the trailing slash is shorthand for the file named /index.html.

Sequence of socket calls for connection-oriented protocol:

System Calls

Socket - create a descriptor for use in network communication. On success, socket system call returns a small integer value similar to a file descriptor Name.

Bind - Bind a local IP address and protocol port to a socket

When a socket is created it does not have nay notion of endpoint address. An application calls bind to specify the local; endpoint address in a socket. For TCP/IP protocols, the endpoint address uses the socket address in structure. Servers use bind to specify the well-known port at which they will await connections.

Connect - connect to remote client

After creating a socket, a client calls connect to establish an actual connection to a remote server. An argument to connect allows the client to specify the remote endpoint, which include the remote machines IP address and protocols port number. Once a connection has been made, a client can transfer data across it.

Accept () - accept the next incoming connection

Accept creates a new socket for each new connection request and returns the descriptor of the new socket to its caller. The server uses the new socket only for the new connections it uses the original socket to accept additional connection requests once it has accepted connection, the server can transfer data on the new socket.

Return Value:

This system-call returns up to three values

- An integer return code that is either an error indication or a new socket description
- The address of the client process
- The size of this address

Listen - place the socket in passive mode and set the number of incoming TCP connections the system will en-queue. Backlog - specifies how many connections requests can be queued by the system while it wants for the server to execute the accept system call it us usually executed after both the socket and bind system calls, and immediately before the accept system call.

Send; send to, recv and recvfrom system calls

These system calls are similar to the standard read and write system calls, but additional arguments are requested.

Close - terminate communication and de-allocate a descriptor. The normal UNIX close system call is also used to close a socket.

CODING

```
import java.awt.*;
Import java.awt.event.*;
Import javax.swing.*;


public class Operator extends JFrame implements ActionListener
                          {
Label l,l3;
JLabel l2,l1;
Font f;
Button b1,b2,b3,b4,b5,b8,b9,b10;
Container c1;


Operator()
                          {
       c1=getContentPane();
      c1.setLayout(new FlowLayout());


       setBackground(Color.LIGHT_GRAY);
```

```
          f=new Font("Arial",Font.BOLD,20);

           setFont(f);

                              ImageIcon i=new ImageIcon("criminal1.jpg");



                               l=new Label("         Welcome  Chandu!!!!!!!!
      ");
                               l2=new JLabel("     Hi,
      ",i,JLabel.LEADING);

                               l3=new Label("          This is Operator
      ",Label.CENTER);



          b3=new Button("   Identifying of Images                  ");
```

```
b4=new Button("   Images from Data Base      ");

//b5=new Button("   Images from Eyewitnesses   ");

b8=new Button("   Drawing of Images          ");

                    // b9=new Button("   Matching of Images          ");

                   //b10=new Button("   Data to Administrator      ");



                     addWindowListener(new WindowAdapter()
                    {
                          public void windowClosing(WindowEvent  e)
                          {
                                System.exit(0);
                          }
                    });
```

```
  l.setForeground(new Color(100,100,100));
  l.setFont(new Font("Arial",Font.BOLD,30));
  l2.setForeground(new Color(100,100,100));
  l2.setFont(new Font("Arial",Font.BOLD,20));
  l3.setForeground(new Color(0,64,64));
  l3.setFont(new Font("Arial",Font.BOLD,20));

  b3.setForeground(new Color(255,255,255));
  b3.setBackground(new Color(0,64,64));
  b3.setFont(new Font("Arial",Font.BOLD,15));

                   b4.setForeground(new Color(255,255,255));
  b4.setFont(new Font("Arial",Font.BOLD,15));
  b4.setBackground(new Color(0,64,64));

                  // b5.setForeground(new Color(255,255,255));
 //  b5.setFont(new Font("Arial",Font.BOLD,15));
 // b5.setBackground(new Color(0,64,64));
```

```
                    b8.setForeground(new Color(255,255,255));
b8.setFont(new Font("Arial",Font.BOLD,15));
b8.setBackground(new Color(0,64,64));

/*b9.setForeground(new Color(255,255,255));
b9.setFont(new Font("Arial",Font.BOLD,15));
b9.setBackground(new Color(0,64,64));

b10.setForeground(new Color(255,255,255));
b10.setFont(new Font("Arial",Font.BOLD,15));
b10.setBackground(new Color(0,64,64));*/



b3.addActionListener(this);
b4.addActionListener(this);
//b5.addActionListener(this);
b8.addActionListener(this);
```

```
    //b10.addActionListener(this);

    //b9.addActionListener(this);




                          c1.add(l);

                          c1.add(l2);

                          c1.add(l3);


//    c1.add(b5);

    c1.add(b4);

    c1.add(b3);

                          c1.add(b8);

                        //  c1.add(b9);

                       //   c1.add(b10);




    setSize(500,600);

    setVisible(true);
```

```
                                }

    public void actionPerformed(ActionEvent ae)

             {

                   if(ae.getSource()==b5)

                    {

                                        Insert1 d=new Insert1();

                                        d.setSize(700,600);

                                        d.setVisible(true);

                                        }

        if(ae.getSource()==b3)

                    {

                                  System.out.println("Face Framing page opening");

                                  faceIdMain1 f1=new faceIdMain1();

                                f1.setSize(800,800);

                                f1.setVisible(true);


                                     }


                                       if(ae.getSource()==b8)
```

```
{

              Draw d1=new Draw();

             d1.setSize(600,400);

             d1.setVisible(true);



                  }



                        if(ae.getSource()==b9)

{

              MatchingImg mi=new MatchingImg();

             mi.setSize(700,600);

             mi.setVisible(true);



                  }



                        if(ae.getSource()==b10)

{

              DatafromOp da=new DatafromOp();

             da.setSize(600,400);
```

```
                da.setVisible(true);


                    }




                     if(ae.getSource()==b4)

     {

                     Retr r1=new Retr();

                      r1.setSize(600,600);

                      r1.setVisible(true);

                     }


 }

public static void main(String args[])

 {

 Operator op= new Operator();

}}
```

TESTING

Software testing is a critical element of software quality assurance and represents the ultimate reviews of specification, design and coding. Testing represents an interesting anomaly for the software. During earlier definition and development phases, it was attempted to build software from an abstract concept to a tangible implementation. No system is error free because it is so till the next error crops up during any phase of the development or usage of the product. A sincere effort however needs to be put to bring out a product that is satisfactory.

The testing phase involves the testing of development system using various data. Preparation of the test data plays a vital role in system testing. After preparing the test data, the system under study was tested using those data. While testing the system, by using the test data, errors were found and corrected by using the following testing steps and corrections were also noted for future use. Thus, a series of testing is performed on the proposed system before the system is ready for implementation.

The various types of testing done on the system are:

- Integration testing
- Validation testing
- Unit testing
- Output testing
- User Acceptance testing

Unit testing:

Unit testing focuses on verification effort on the smallest unit of software design module. Using the unit test plans prepared in the design phase of the system development as a guide, important control paths are tested to uncover errors with in the boundary of the modules. The interfaces of the modules are tested to ensure proper flow of information into and out of the modules under consideration boundary conditions were checked. All independent paths were exercised to ensure that all statements in the module have been executed at least once and all error-handling paths were tested.

Each unit is thoroughly tested to check if it might fail in any possible situation. This testing is carried during the programming state itself. At the end of this testing phase each module is found to be have an adverse effect working satisfactorily, as regard to the expected output from the module.

Integration Testing:

Data can be lost across an interface, one module can on another; sub-functions when combined may not produce the desired major function: global data structures can present problems. Integration testing is a systematic technique for the program structure while at the same time concluding tests to uncover errors associated with interface. All modules are combined in this testing step. Then the entire program is tested as a whole. Each of the module is integrated and tested separately and later all modules are tested together for sometime to ensure the system as a whole works well without any errors.

Validation Testing:

At the culmination of the integration testing, the software is completely assembled as a package, interfacing errors have been uncovered and corrected, and a final series of software validation testing began. Here we test if the system functions in a manner that can be reasonably expected by the customer. The system is tested against the system requirement specification.

Output Testing:

After performing validation testing, the next phase is output testing of the proposed system, since no system can be useful if it does not produce the desired output in the specified format. The output generated or displayed by the system under consideration is tested by asking the user about the format required by them, here, the output format is considered in two ways: One is on the screen and the other is on the printed form. Beta testing is carried output by the client, and minor errors that have been discovered by the client are rectified to improve the user friendliness of the system.

Object-Oriented Testing

The overall objectives of the object-oriented testing – to find the maximum number of errors with a minimum amount effort – is identical to the to the objective of conventional software testing. But the strategy and tactics for OO testing differ significantly. The view of testing broadens to include the review of both the analysis and design model. In addition, the focus of testing moves away from the procedural component and toward the class.

Because the OO analysis and design models and the resulting source code are semantically coupled, testing begins during these engineering activities. For this reason, a review of CRC, object relationships, and object behavior models can be viewed as first stage testing. As a result of this first stage testing, we encountered few problems in OOA done at analysis time. We have gone back and remodeled with new errorless classes and their relationships. The documented model is the revised model of earlier analysis model.

Once OOP has been accomplished, unit testing is applied for each class. Class testing uses a variety of methods: fault-based, random, and partition test methods. Each of those methods exercises the operations encapsulated by the class. Test sequences are designed to ensure that relevant operations are exercised. The state of the class, represented by the values of its attributes, is examined to determine if errors exist.

Integration test can be accomplished using a thread-based or use-based strategy. Thread-based strategy integrates the set of classes that collaborate to respond to one input or event. Use-based testing constructs the system in layers, beginning with those classes that do not make use of server classes. Integration test case design methods can also make use of random and partition tests. In addition, scenario based testing and the tests derived from behavioral models can be used to test a class and its collaborators. A test sequence tracks the flow of operations across class collaborations.

OO system validation testing is black box oriented and can be accomplished by applying the same black box methods known for conventional software. However scenario based testing dominates the validation of OO systems, making the use case a primarily driver for validation testing.

We mainly concentrated on scenario based testing strategy. Some of the test cases for scenario based testing are given below.

Test case #1

Use Case: download a file

Background: User wants to download a file from Criminal Face Identification System

Event Sequence:

1. Select file to down load in remote tree panel.
2. Right click and then click download.

Test case #2

Use Case : upload a file

Background : User wants to upload a file to the server from client.

Event Sequence:

1. Click file in local file view tree.
2. Right click and select upload.

Test case #3

Use Case : Remove file on remote server

Background : User wants to delete a file on the server..

Event Sequence:

1. Select file on remote file view tree panel.
2. Right click and select ‘Delete’.

Test case #4

Use Case : Disconnect to server

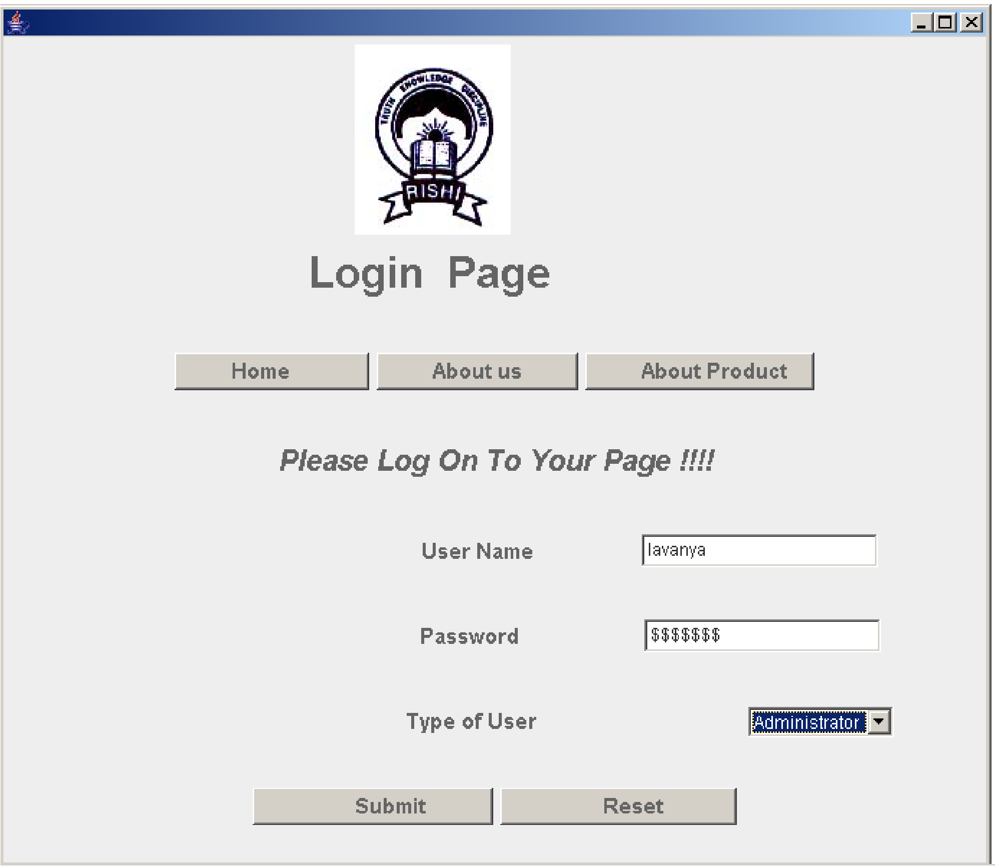
Background : User wants to close the session.

Event Sequence:

1. Click disconnect option in menu

In this way we tested using different test cases that found lot of errors which were corrected by recoding of that related procedures.

SCREENS



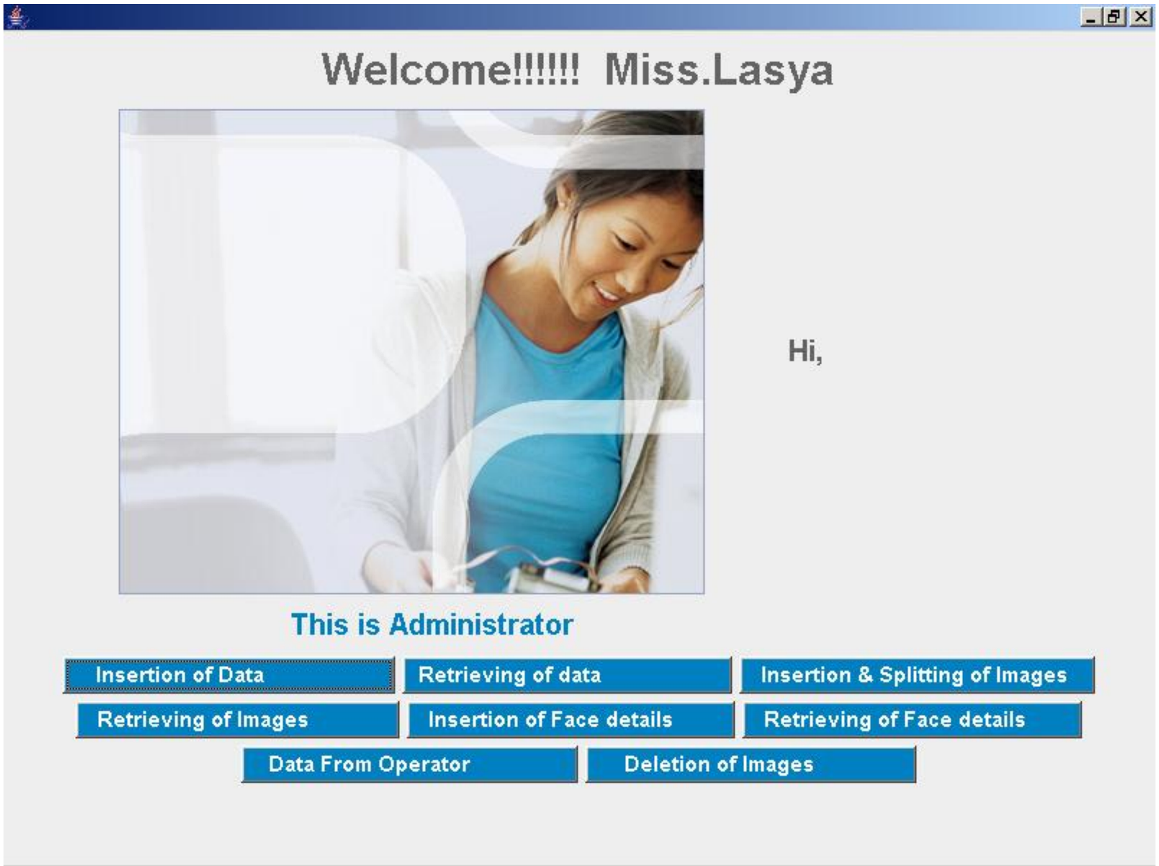
Login Page

Please Log On To Your Page !!!!

User Name lavanya

Password $$$$$$$

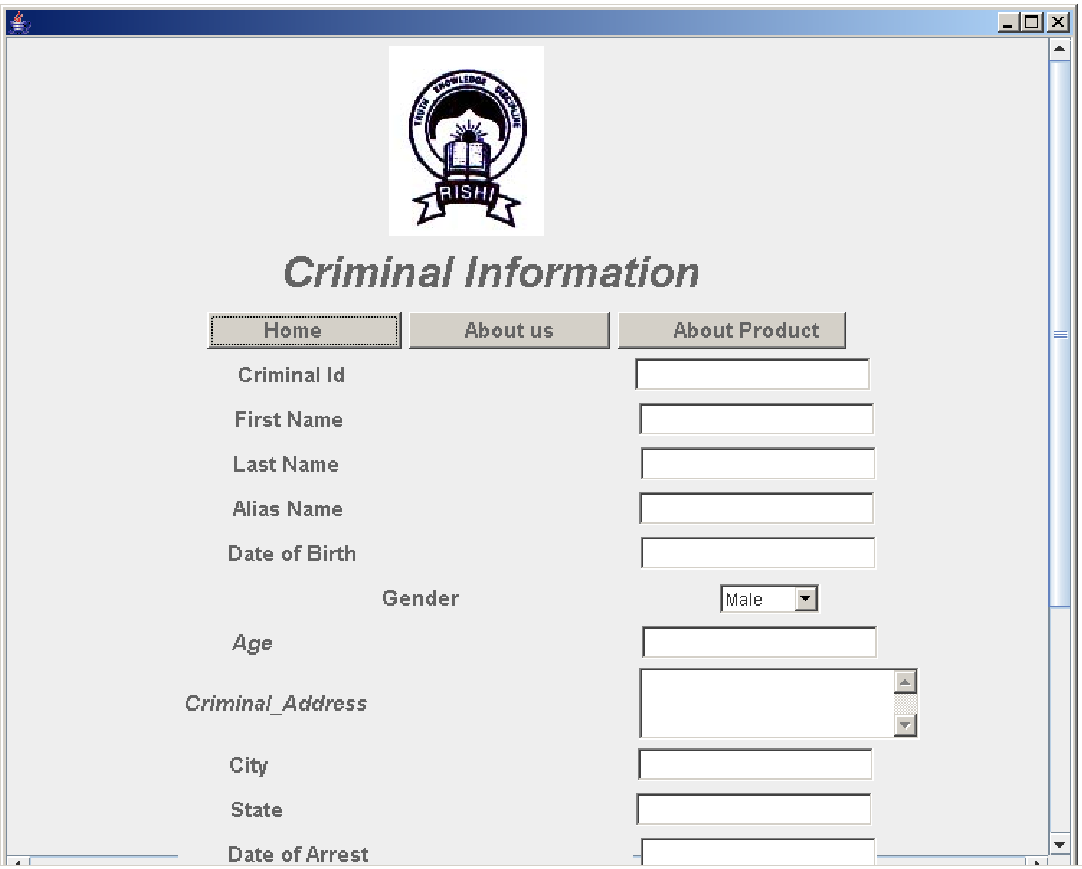
Type of User Administrator



Welcome!!!!!! Miss.Lasya

Hi,

This is Administrator



Criminal Information

Home | About us | About Product

Criminal Id

First Name

Last Name

Alias Name

Date of Birth

Gender: Male

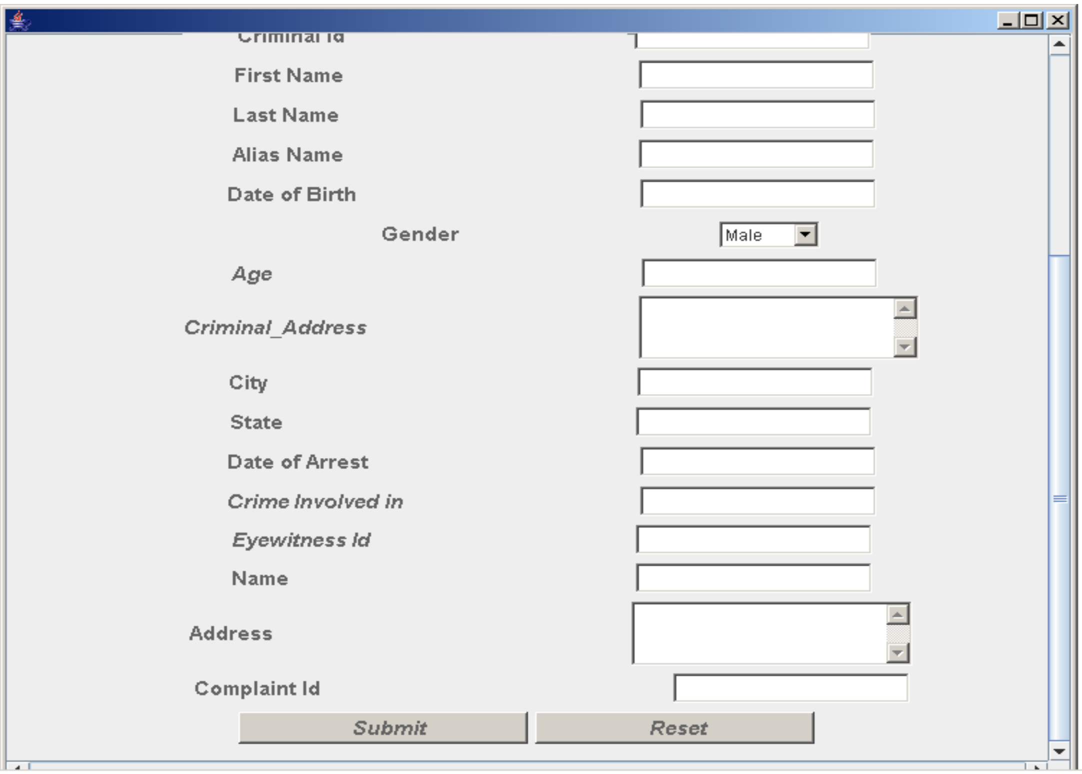
Age

Criminal_Address

City

State

Date of Arrest



Criminal Id

First Name

Last Name

Alias Name

Date of Birth

Gender: Male

Age

Criminal_Address

City

State

Date of Arrest

Crime Involved in

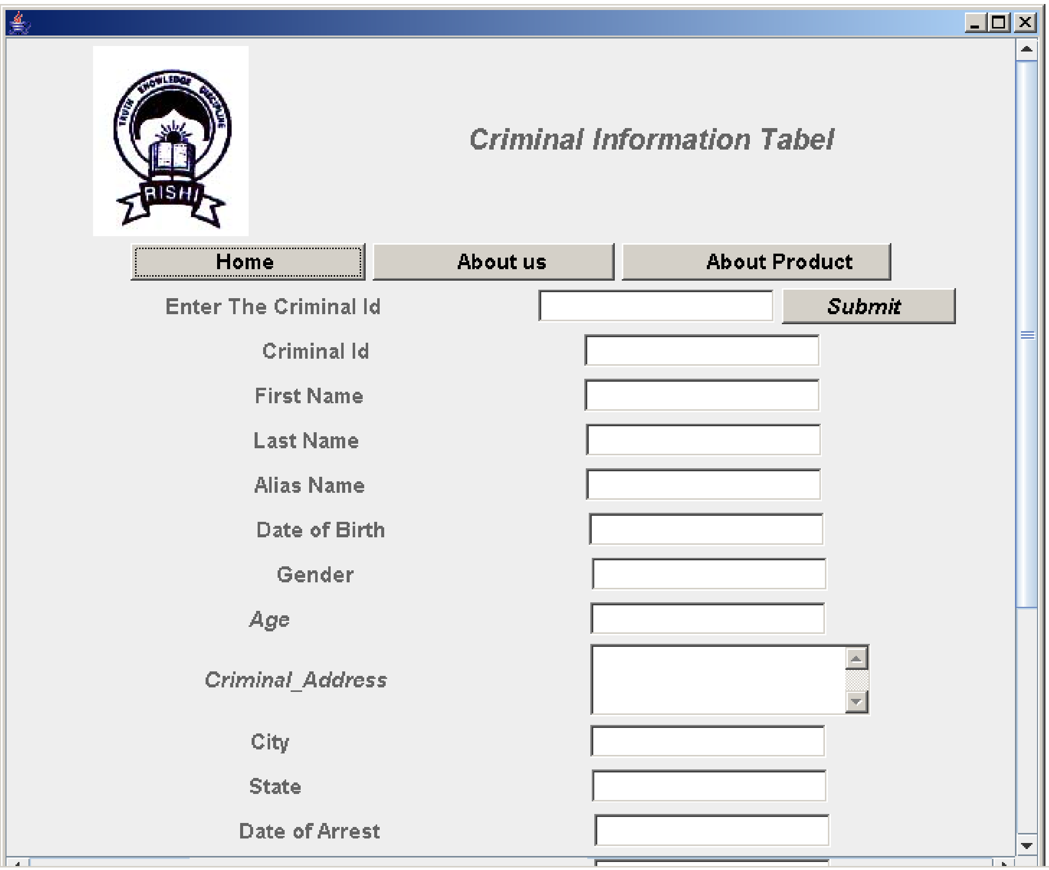
Eyewitness Id

Name

Address

Complaint Id

Submit | *Reset*



Criminal Information Tabel

Enter The Criminal Id

Criminal Id

First Name

Last Name

Alias Name

Date of Birth

Gender

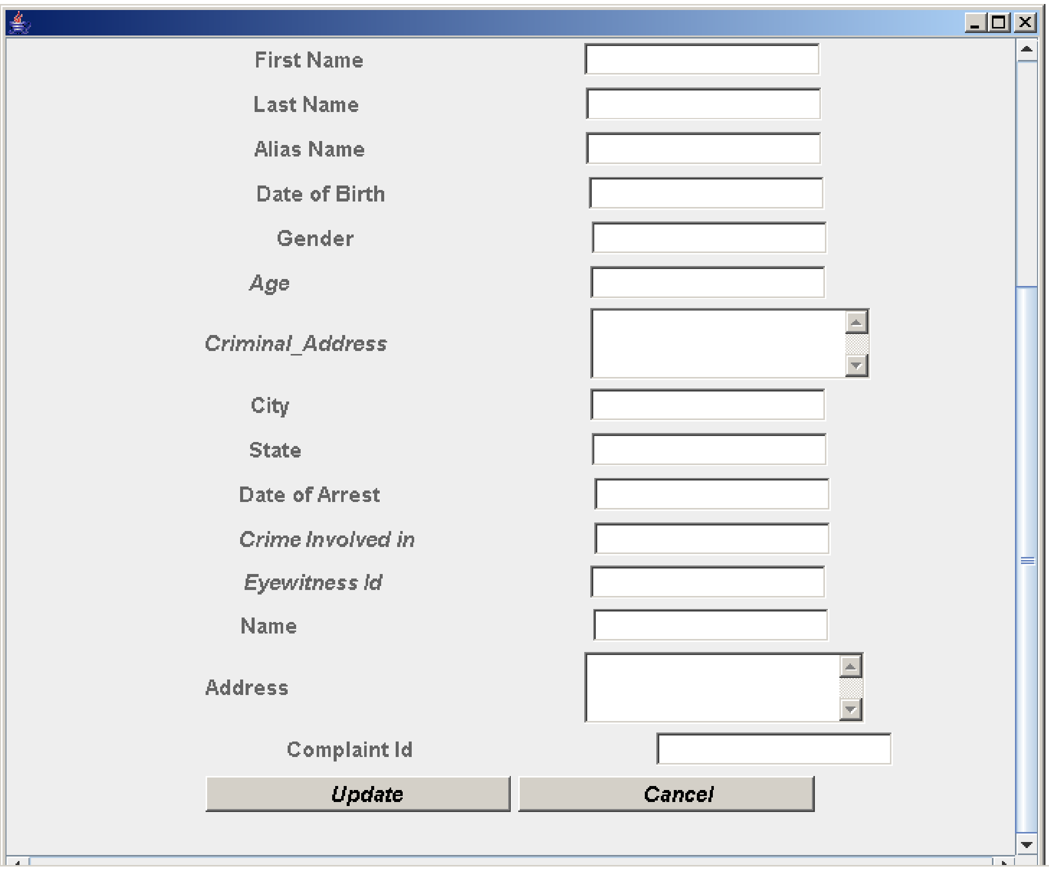
Age

Criminal_Address

City

State

Date of Arrest



First Name

Last Name

Alias Name

Date of Birth

Gender

Age

Criminal_Address

City

State

Date of Arrest

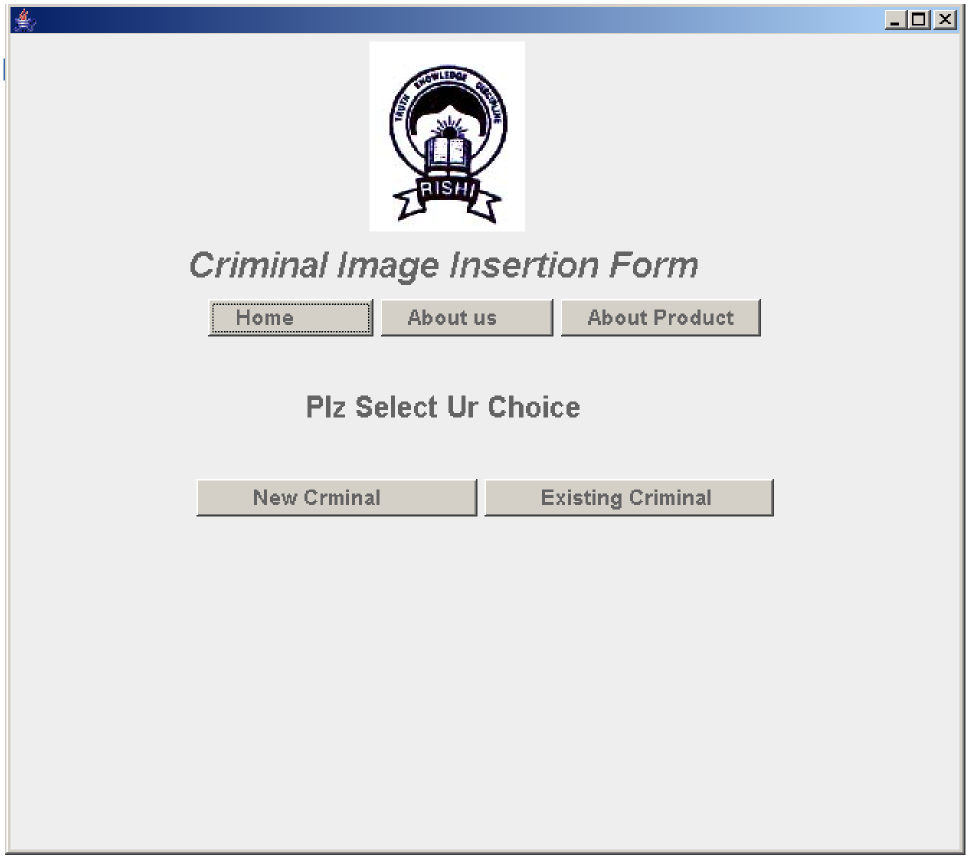
Crime Involved in

Eyewitness Id

Name

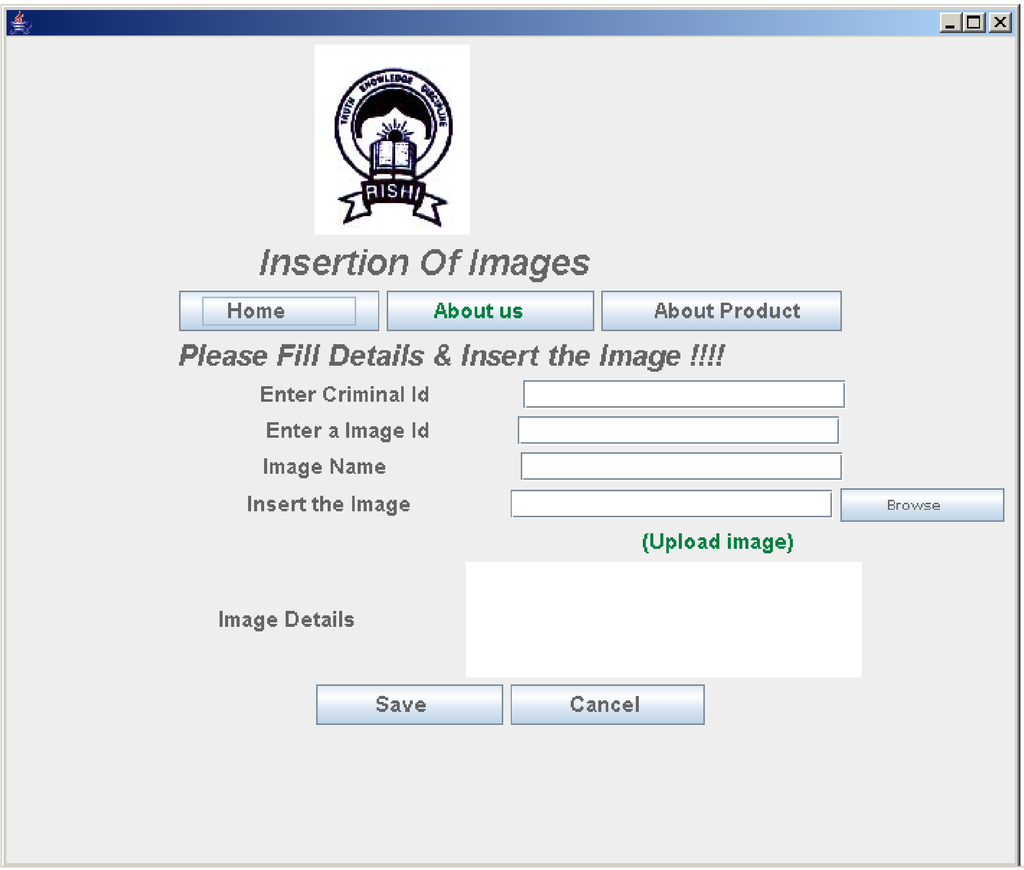
Address

Complaint Id



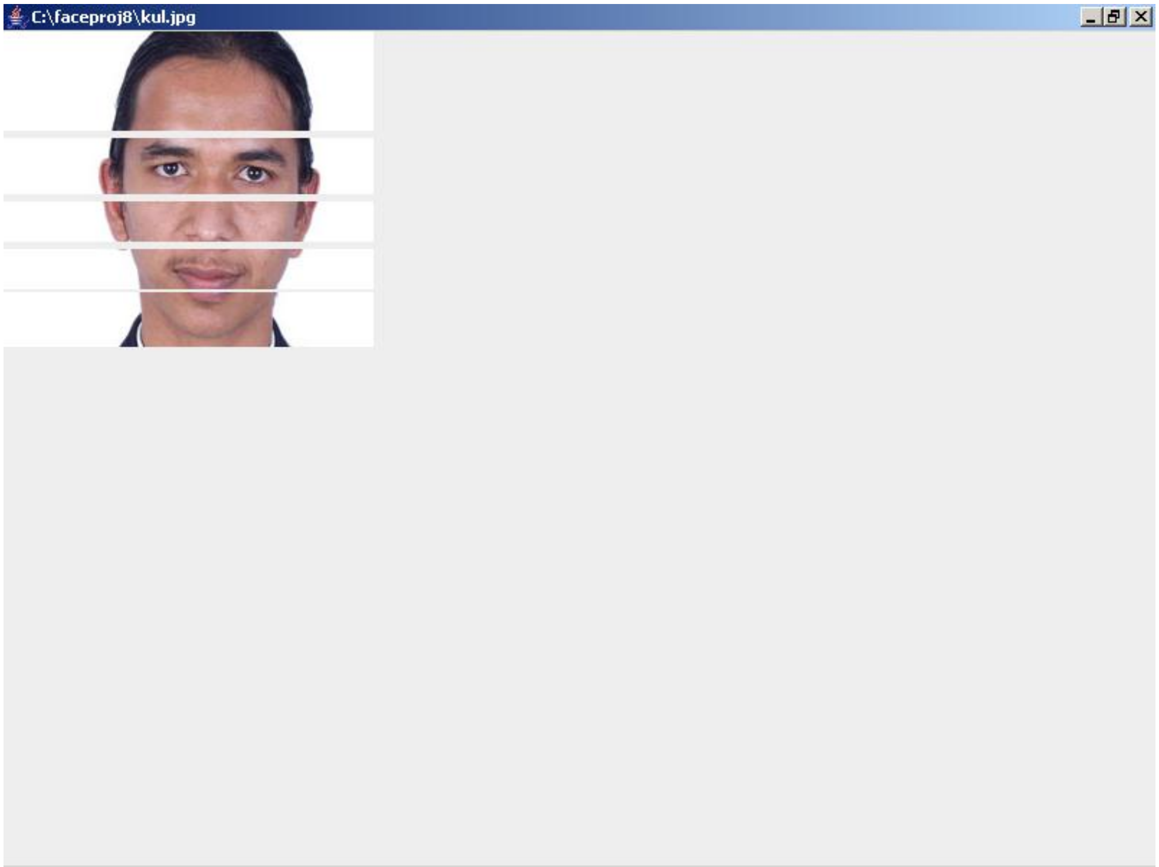
Criminal Image Insertion Form

Plz Select Ur Choice

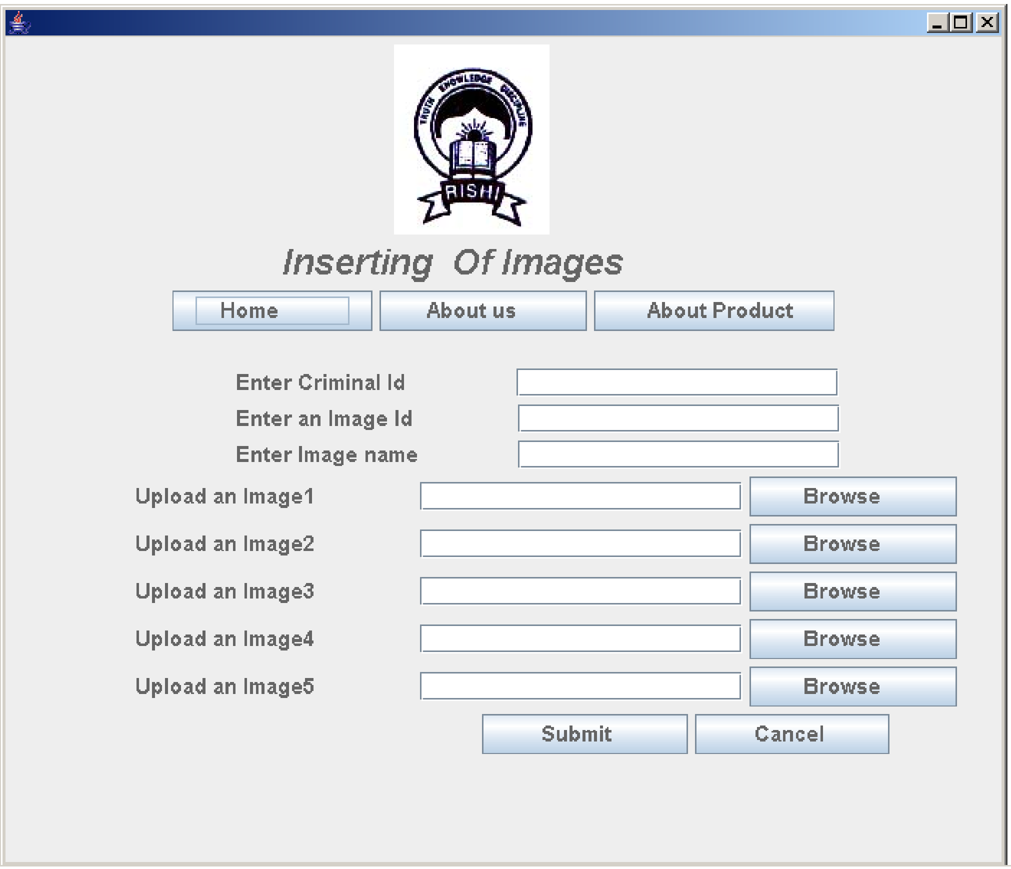


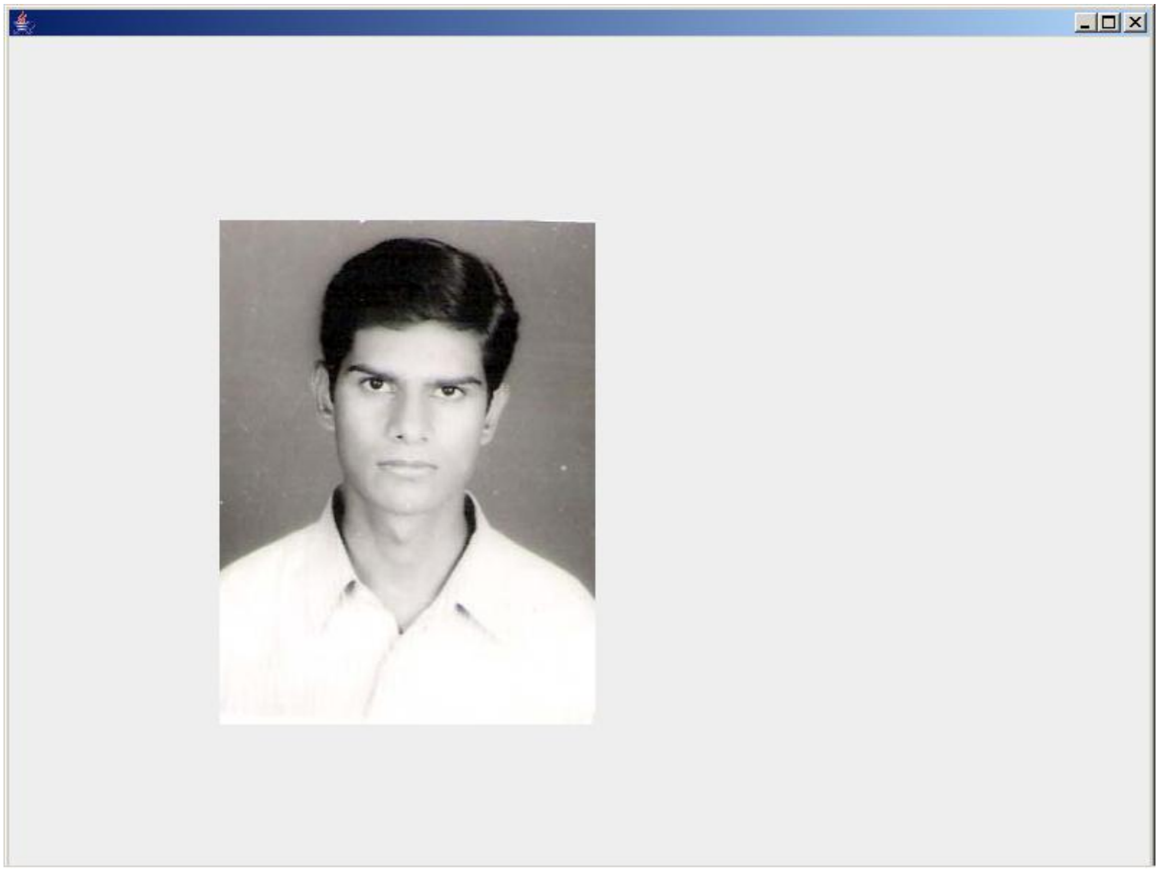
Insertion Of Images

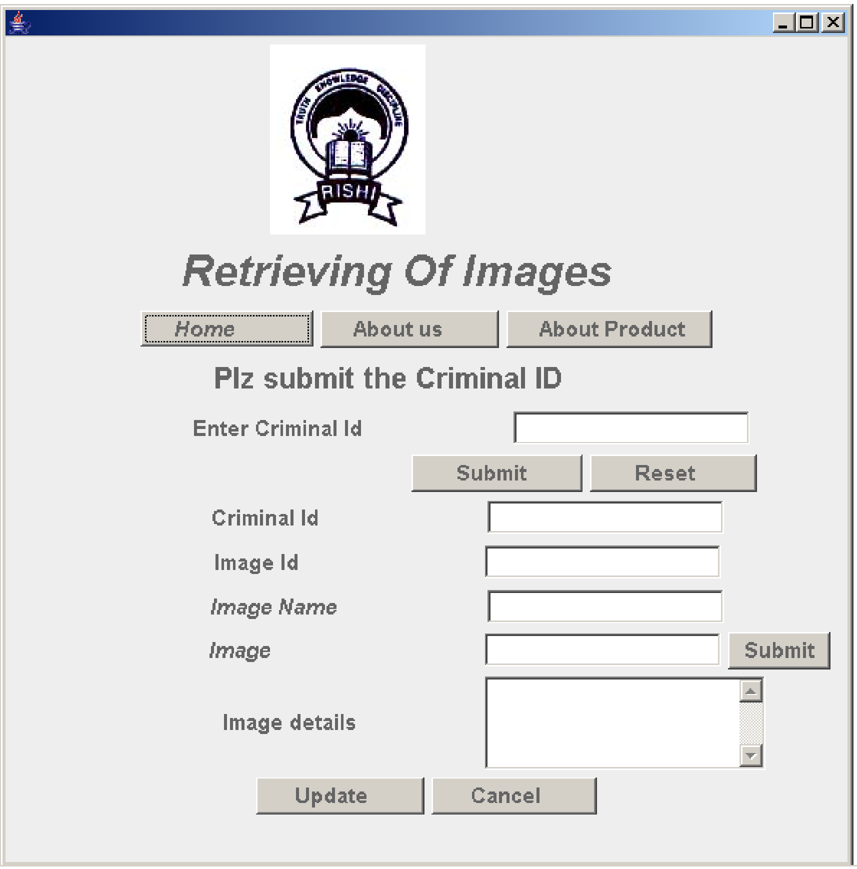
Please Fill Details & Insert the Image !!!!



C:\faceproj8\kul.jpg

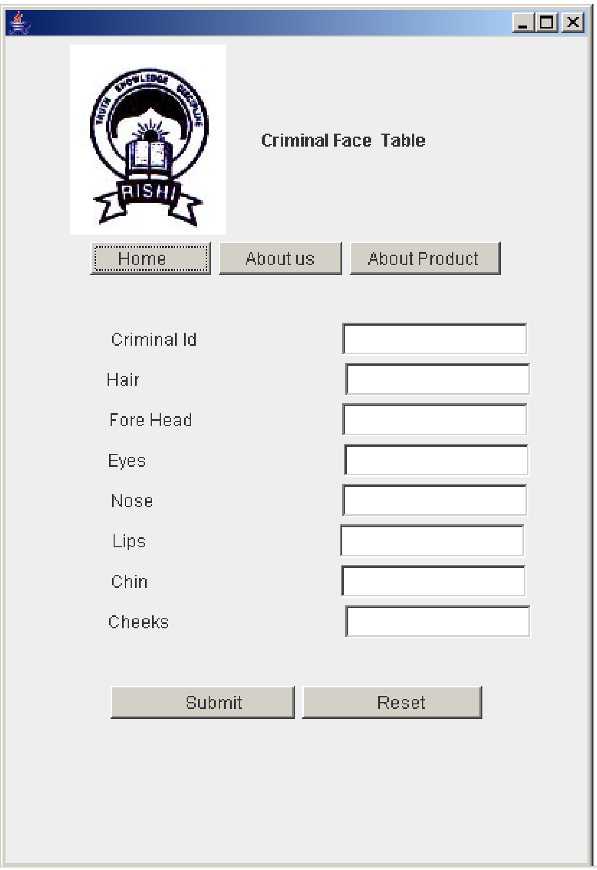

RISHI
TRUTH KNOWLEDGE DISCIPLINE
Inserting Of Images
Home
About us
About Product
Enter Criminal Id
Enter an Image Id
Enter Image name
Upload an Image1
Browse
Upload an Image2
Browse
Upload an Image3
Browse
Upload an Image4
Browse
Upload an Image5
Browse
Submit
Cancel

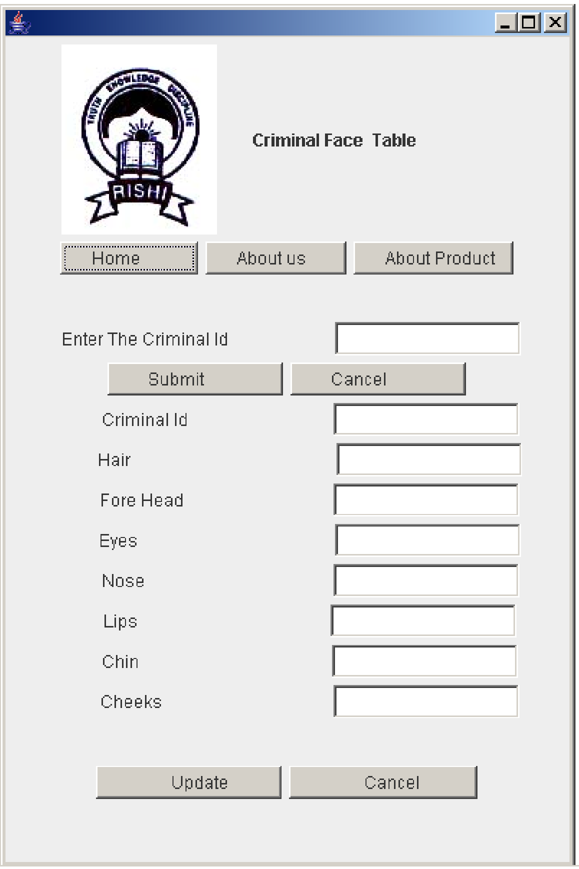




Retrieving Of Images

Plz submit the Criminal ID





Criminal Face Table

Enter The Criminal Id

Criminal Id

Hair

Fore Head

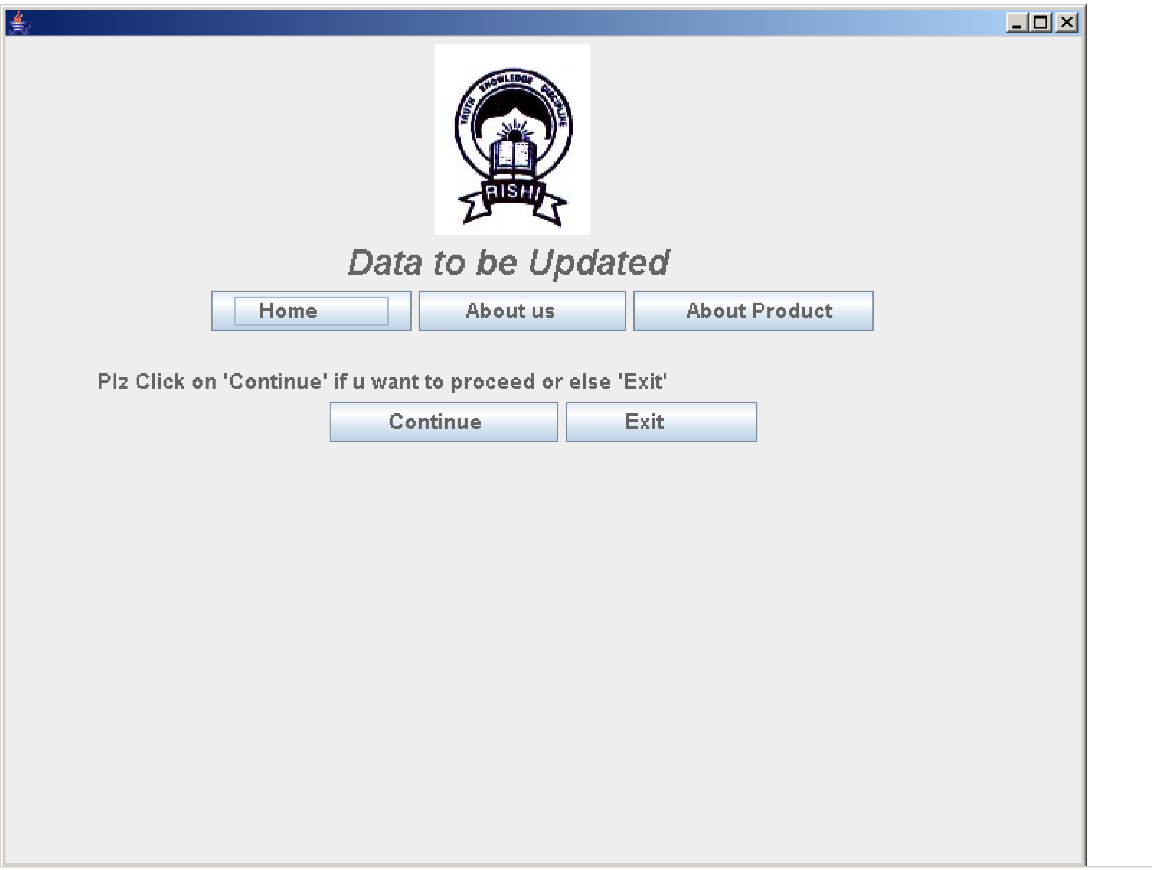
Eyes

Nose

Lips

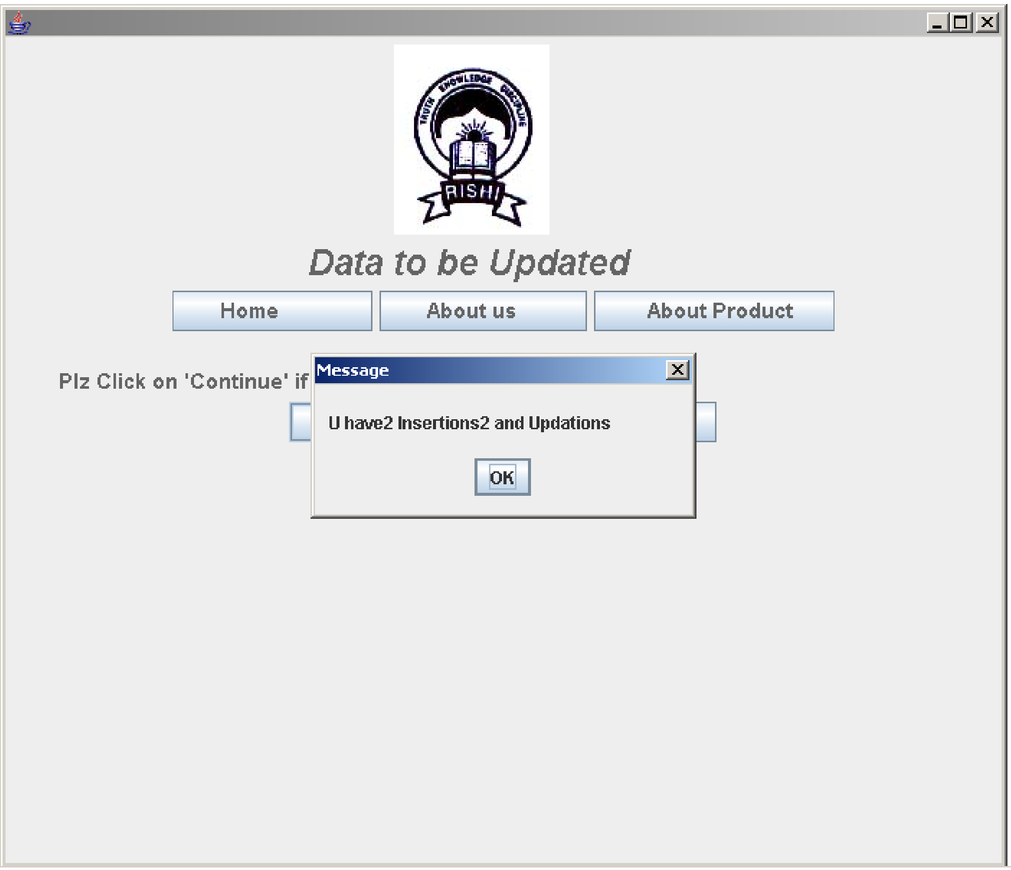
Chin

Cheeks



Data to be Updated

Plz Click on 'Continue' if u want to proceed or else 'Exit'

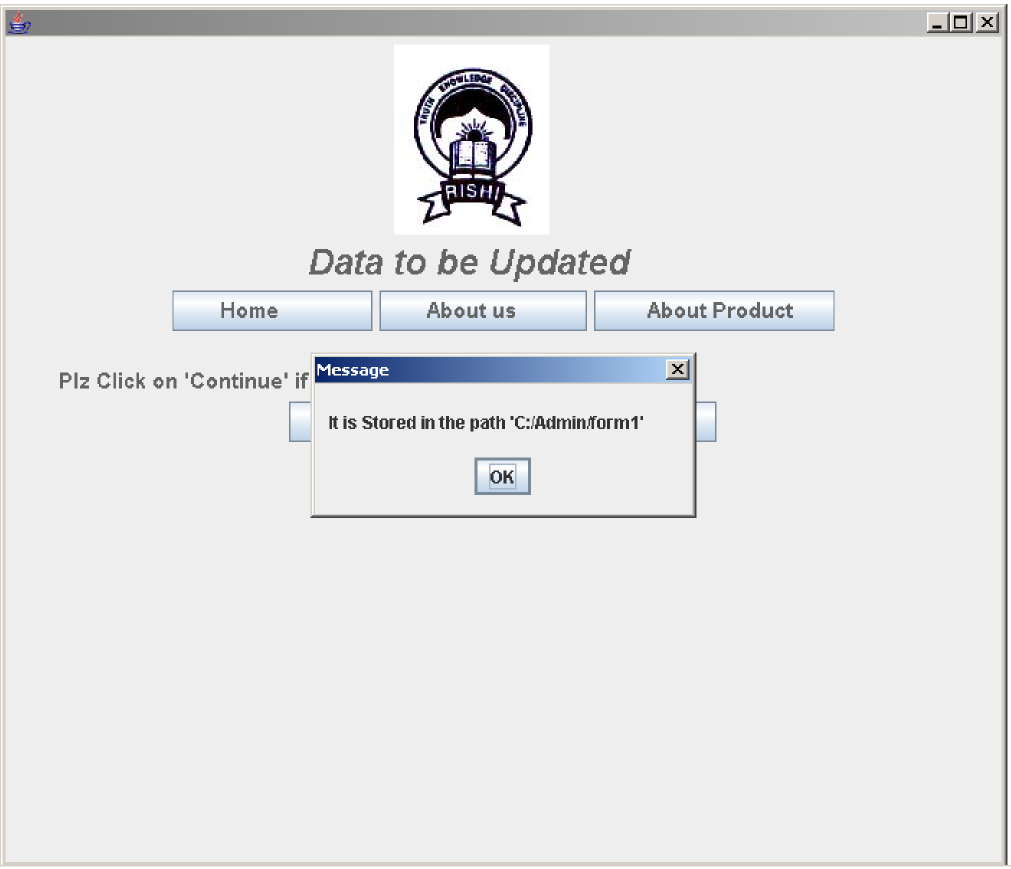


Data to be Updated

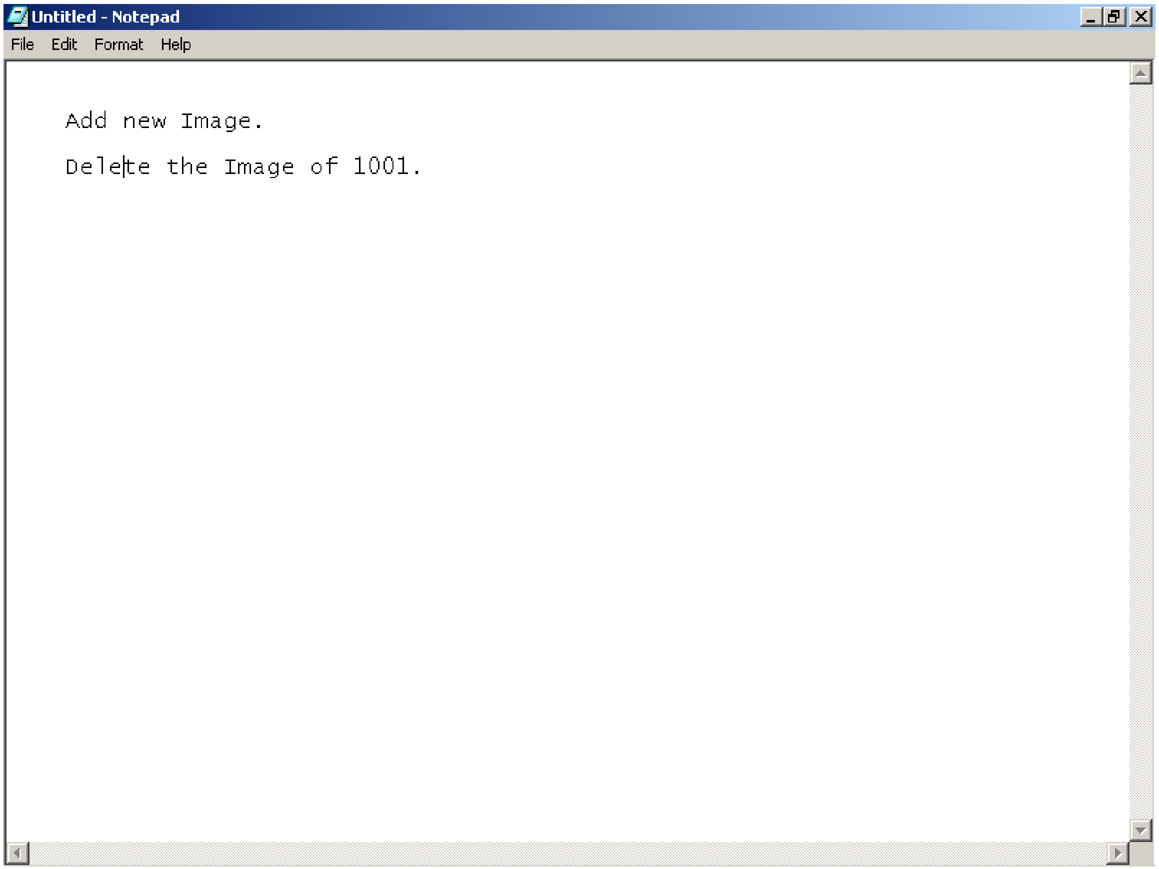
Plz Click on 'Continue' if

Message

U have2 Insertions2 and Updations

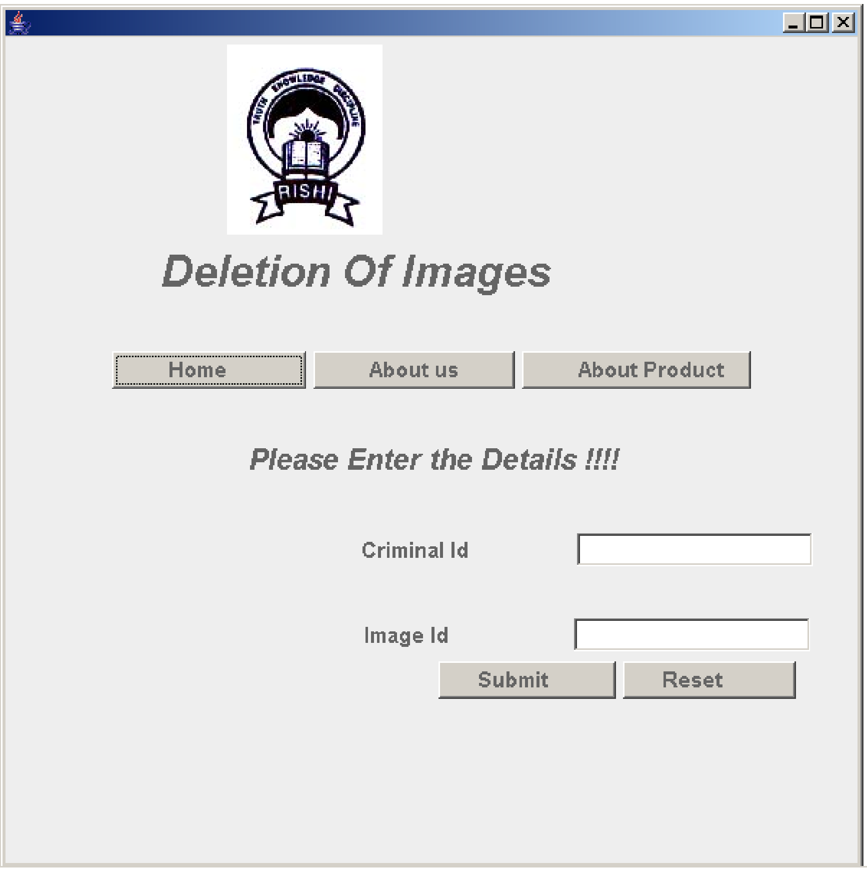


It is Stored in the path 'C:/Admin/form1'



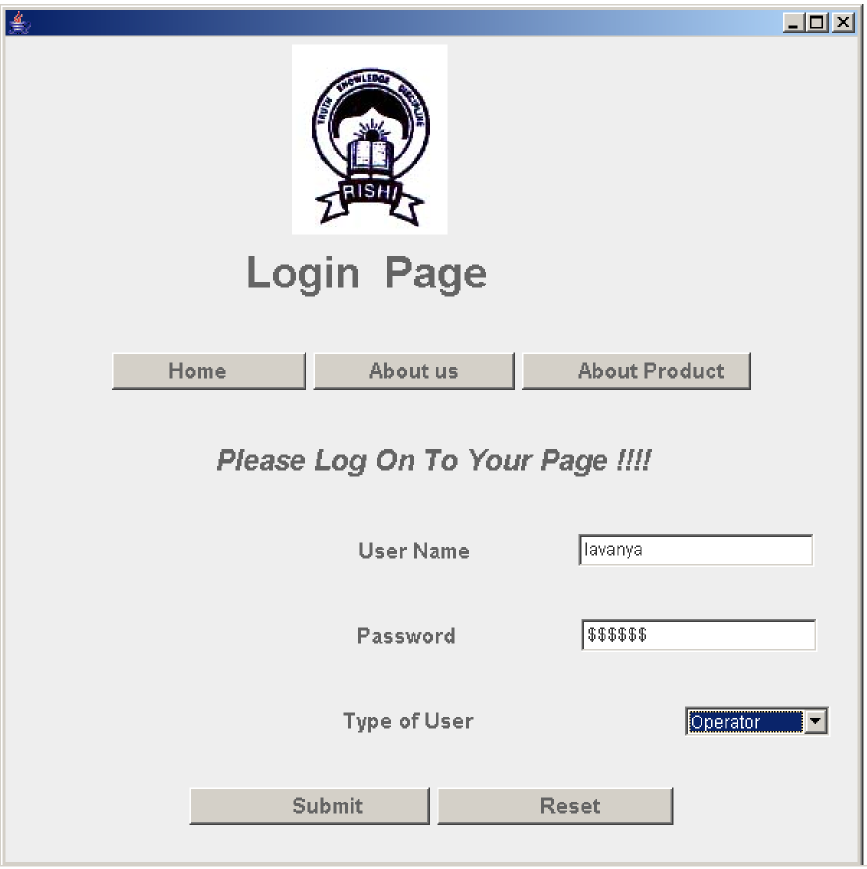
Add new Image.

Delete the Image of 1001.



Deletion Of Images

Please Enter the Details !!!!



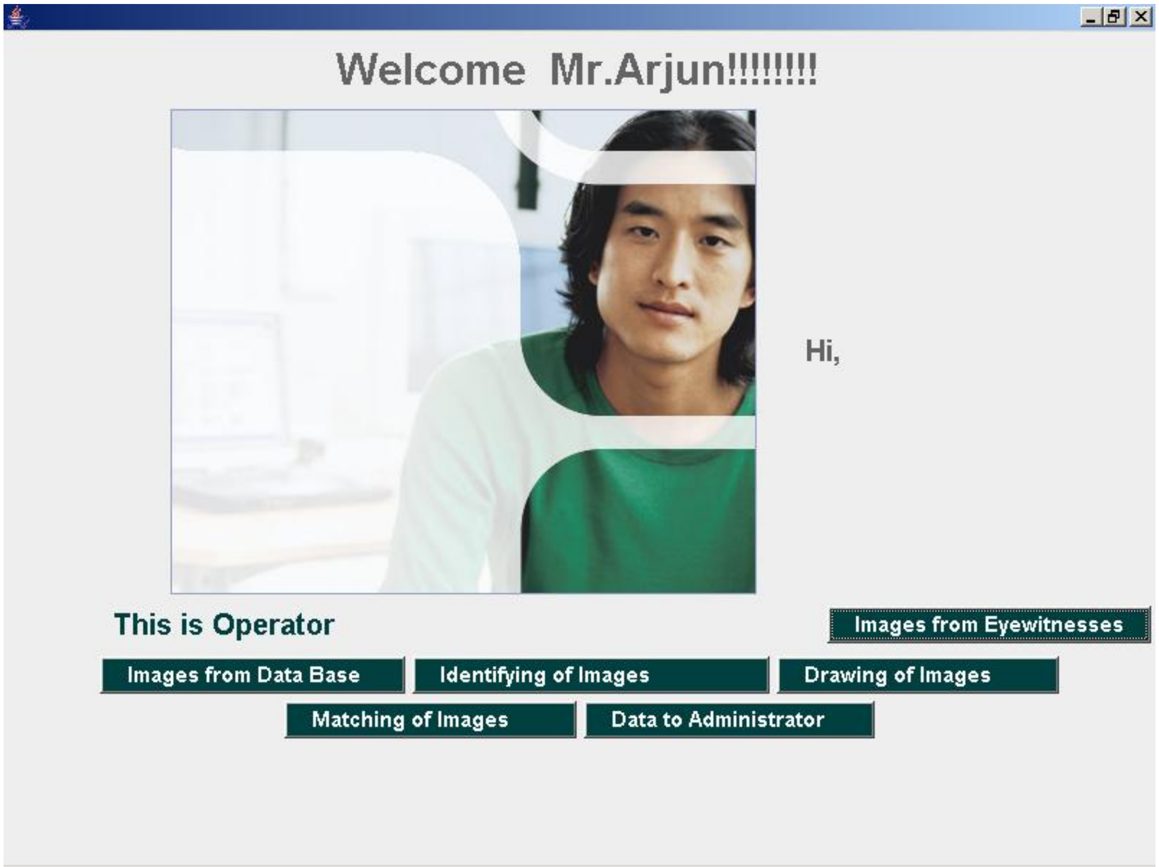
Login Page

Please Log On To Your Page !!!!

User Name: lavanya

Password: $$$$$$

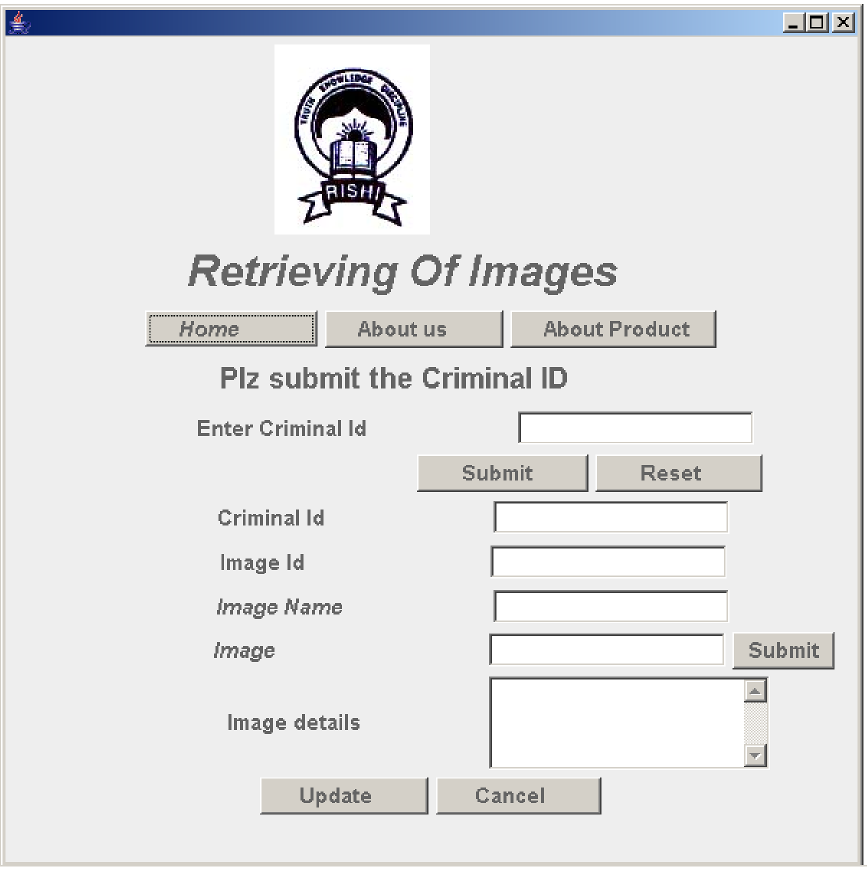
Type of User: Operator



Welcome Mr.Arjun!!!!!!!!

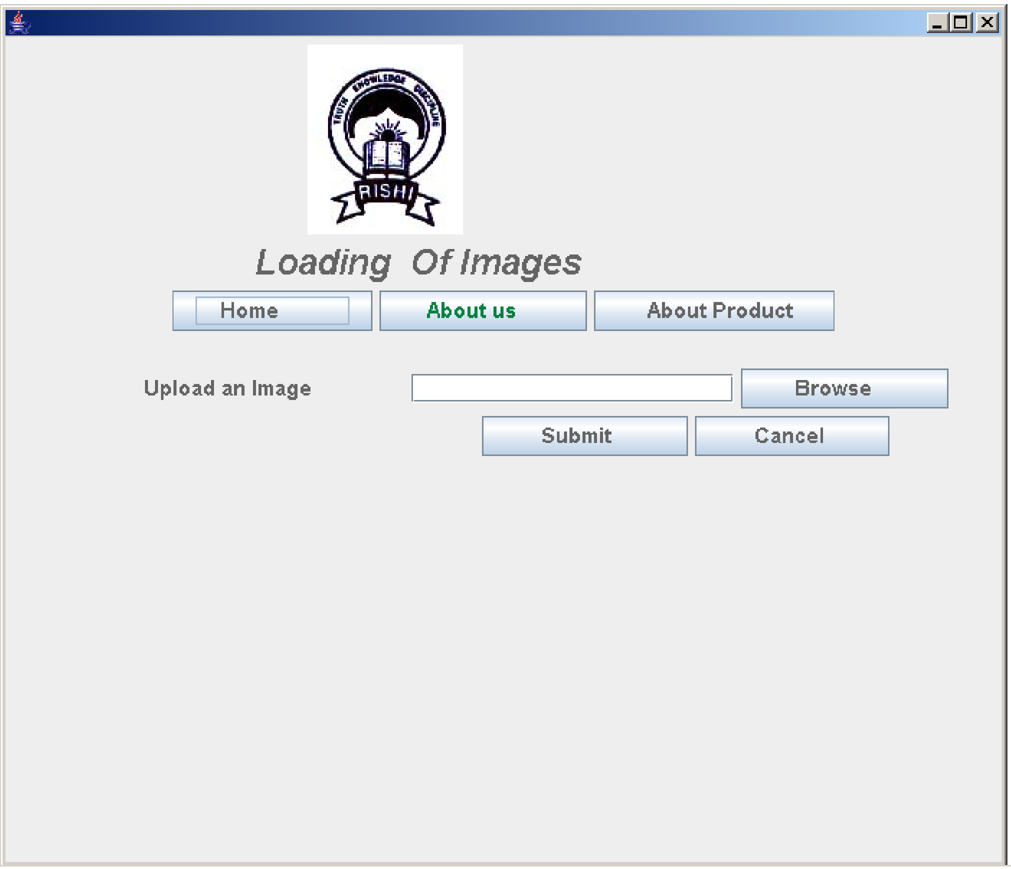
Hi,

This is Operator

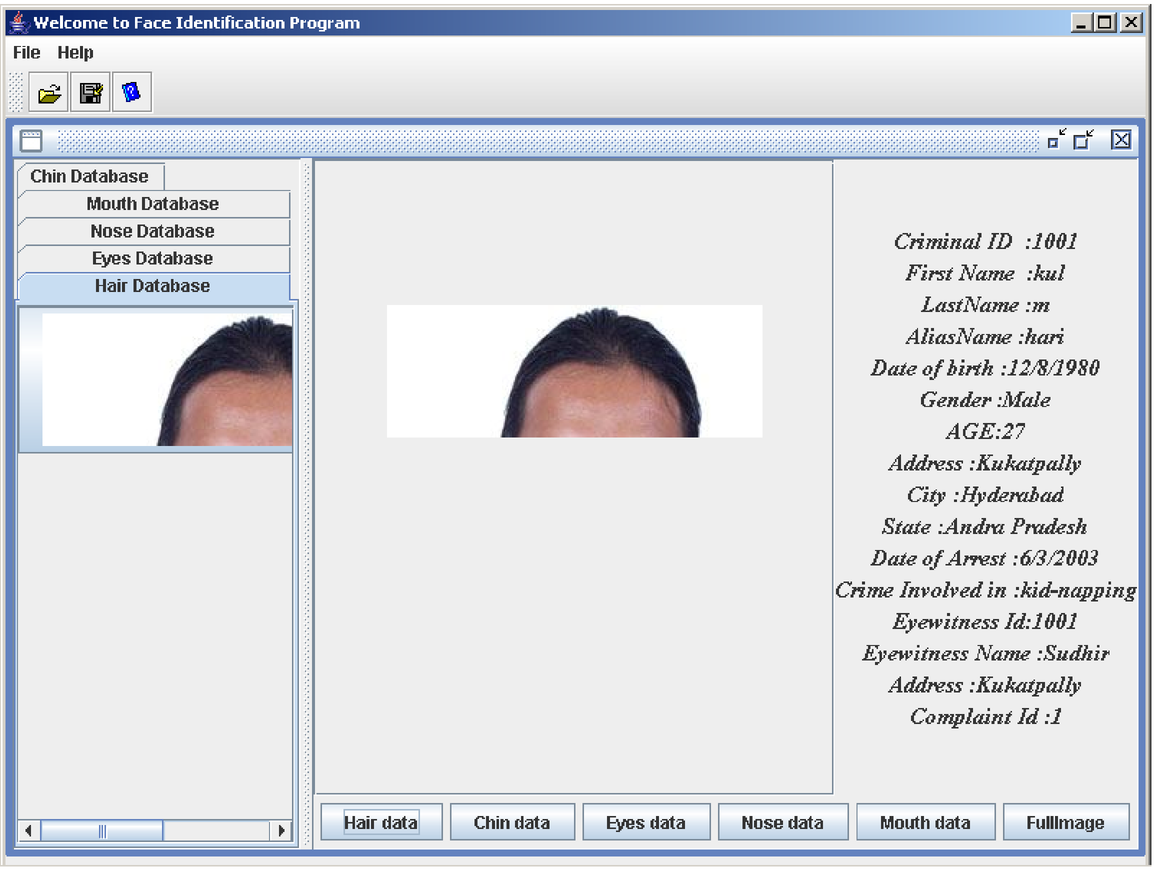


Retrieving Of Images

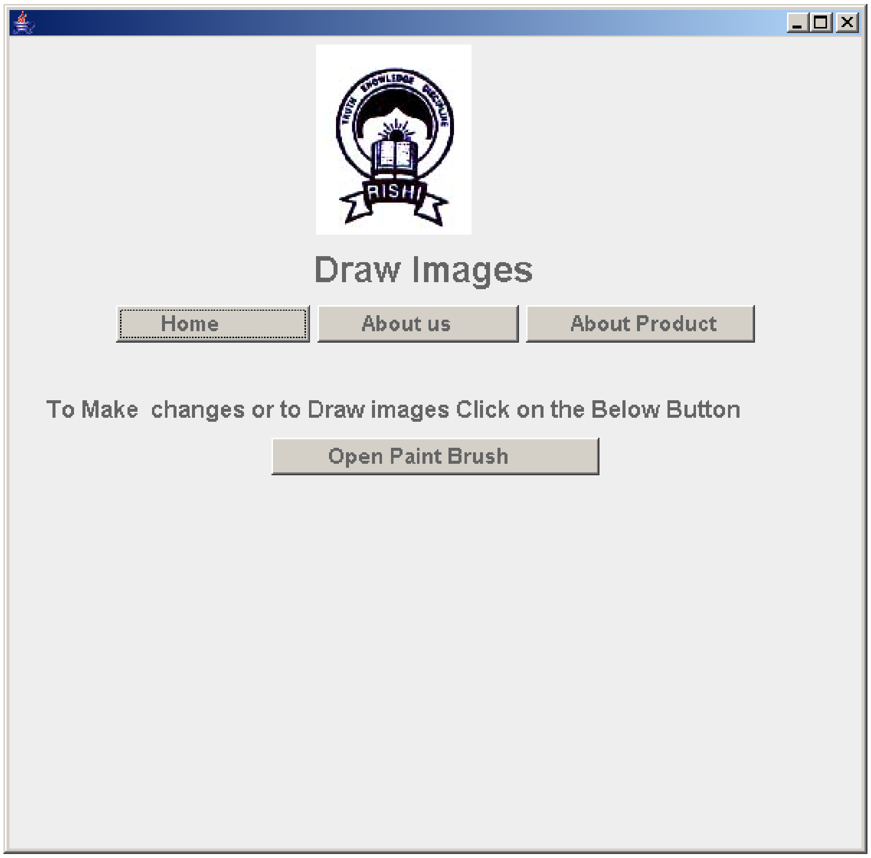
Plz submit the Criminal ID



TRUTH KNOWLEDGE DISCIPLINE
RISHI
Loading Of Images
Home
About us
About Product
Upload an Image
Browse
Submit
Cancel

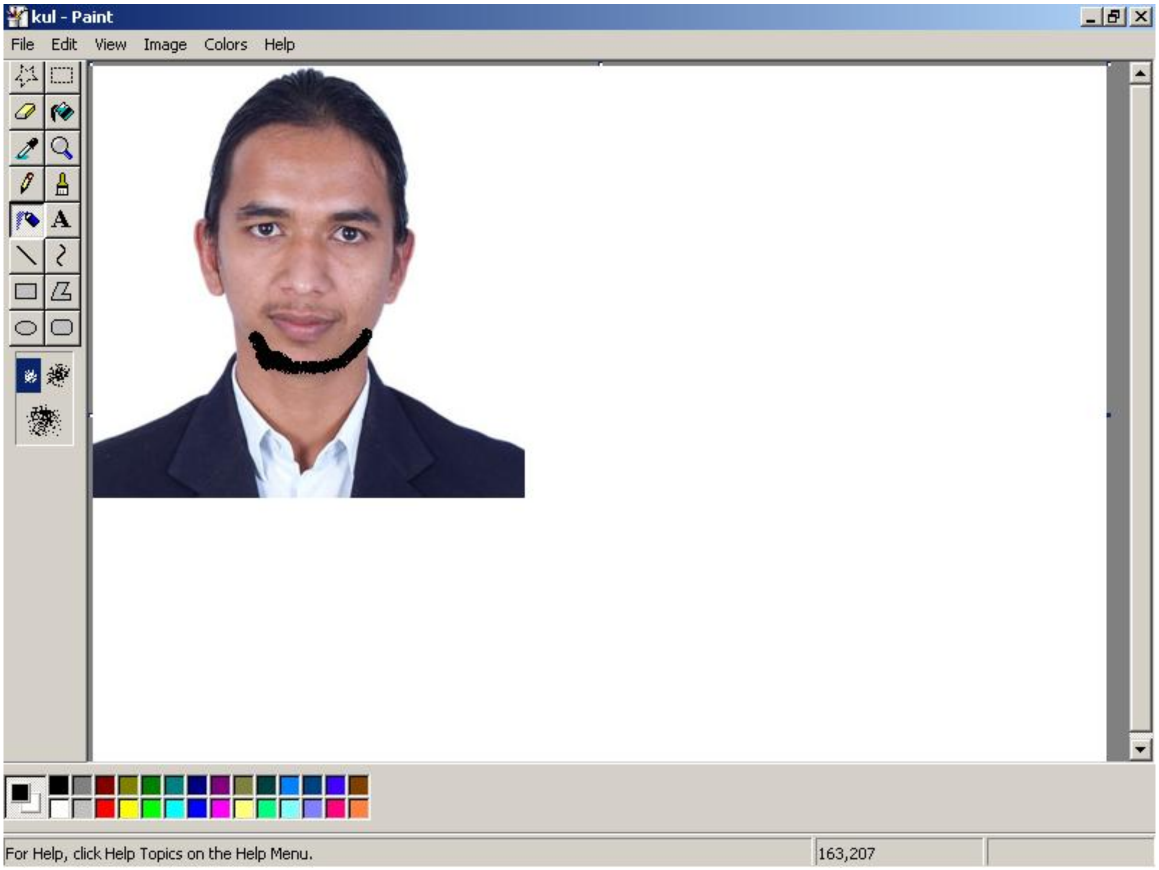


Criminal ID :1001
First Name :kul
LastName :m
AliasName :hari
Date of birth :12/8/1980
Gender :Male
AGE:27
Address :Kukatpally
City :Hyderabad
State :Andra Pradesh
Date of Arrest :6/3/2003
Crime Involved in :kid-napping
Eyewitness Id:1001
Eyewitness Name :Sudhir
Address :Kukatpally
Complaint Id :1

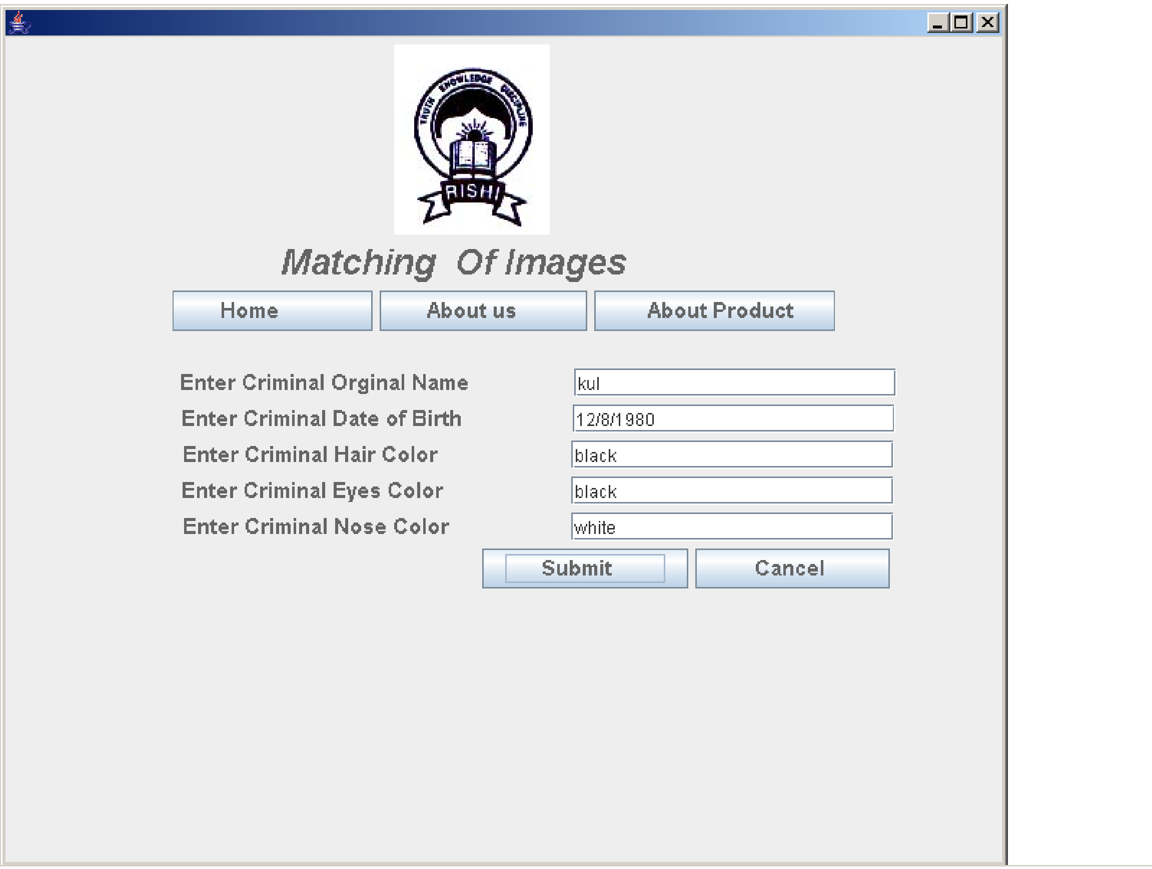


Draw Images

To Make changes or to Draw images Click on the Below Button



kul - Paint
File Edit View Image Colors Help
For Help, click Help Topics on the Help Menu.
163,207



Enter Criminal Orginal Name kul

Enter Criminal Date of Birth 12/8/1980

Enter Criminal Hair Color black

Enter Criminal Eyes Color black

Enter Criminal Nose Color white

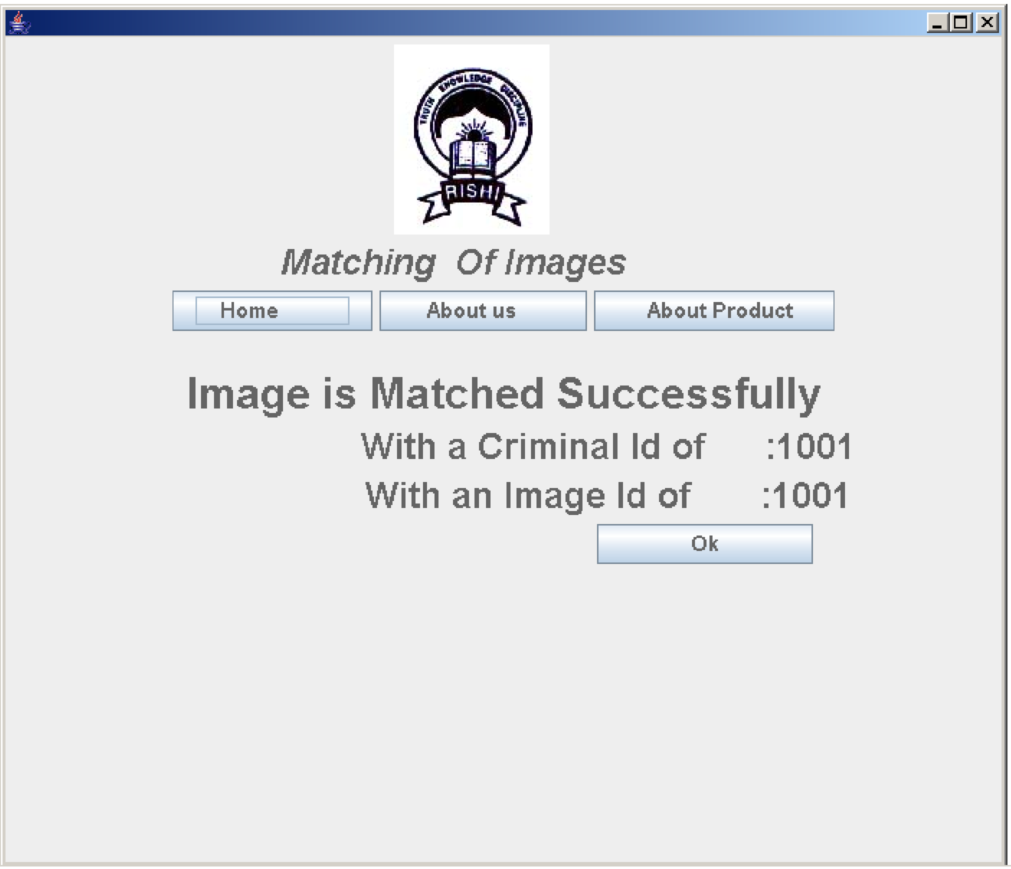
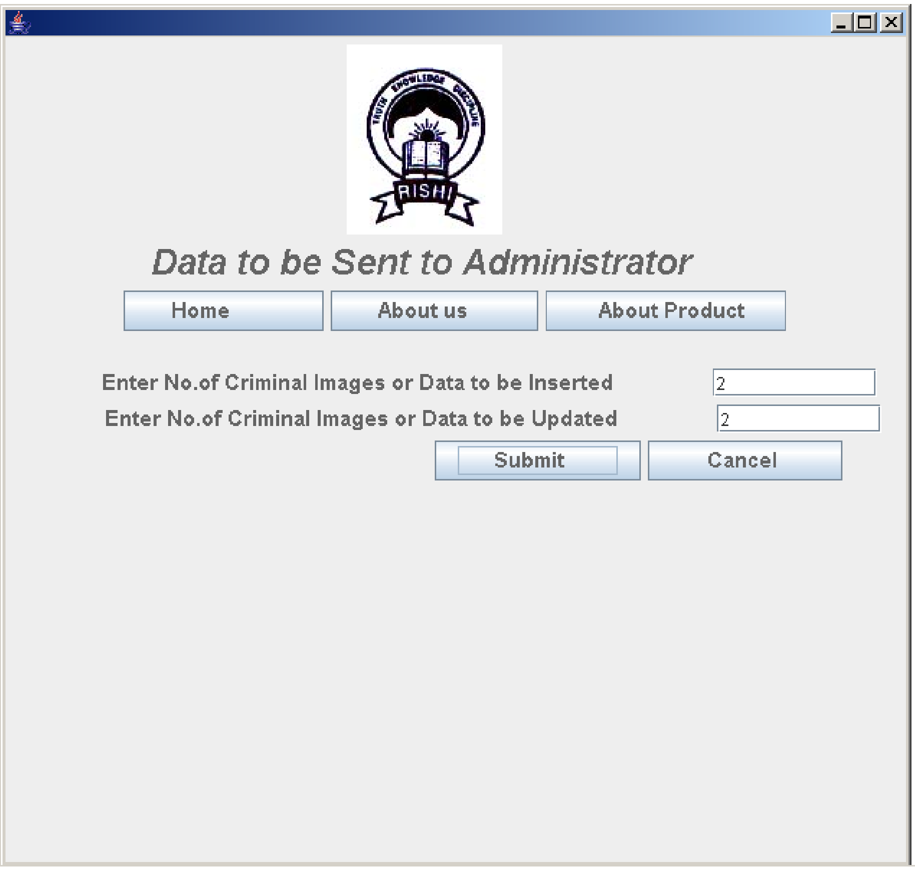


Image is Matched Successfully

With a Criminal Id of :1001

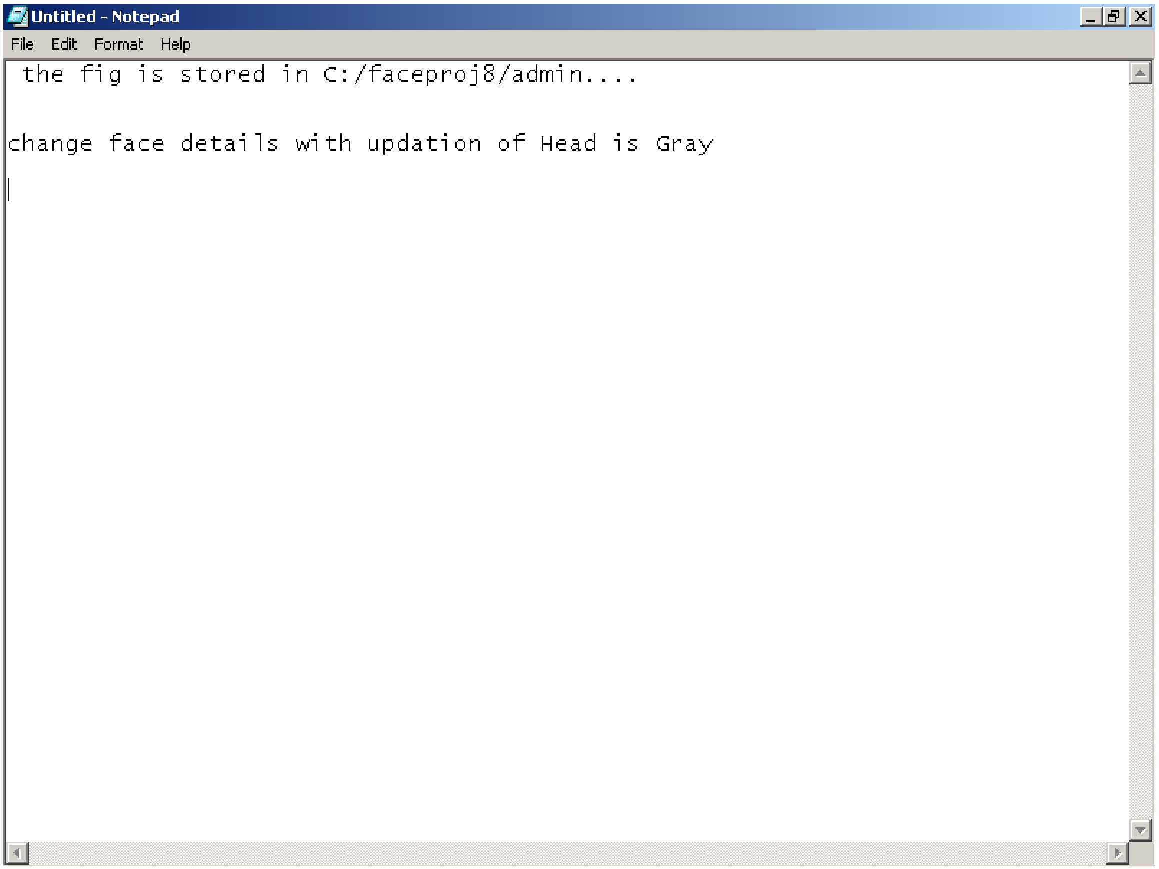
With an Image Id of :1001



Data to be Sent to Administrator

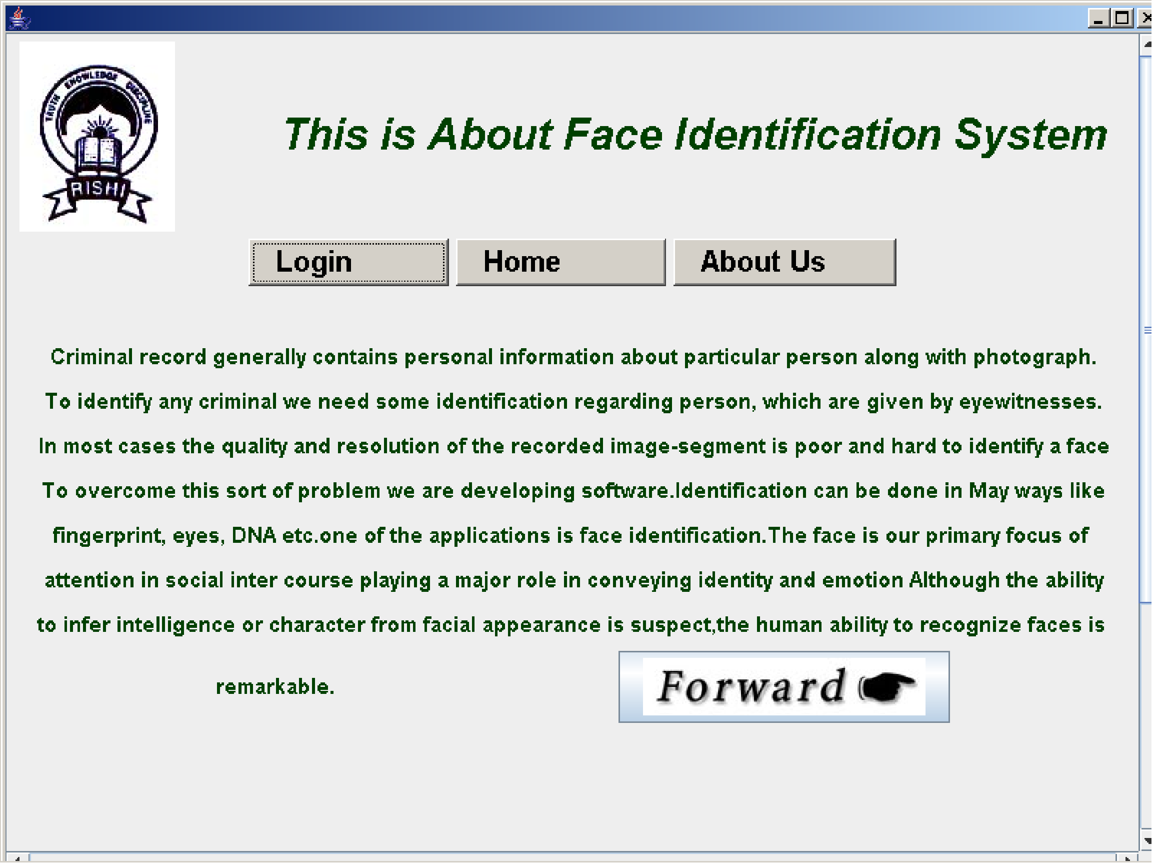
Enter No.of Criminal Images or Data to be Inserted 2

Enter No.of Criminal Images or Data to be Updated 2



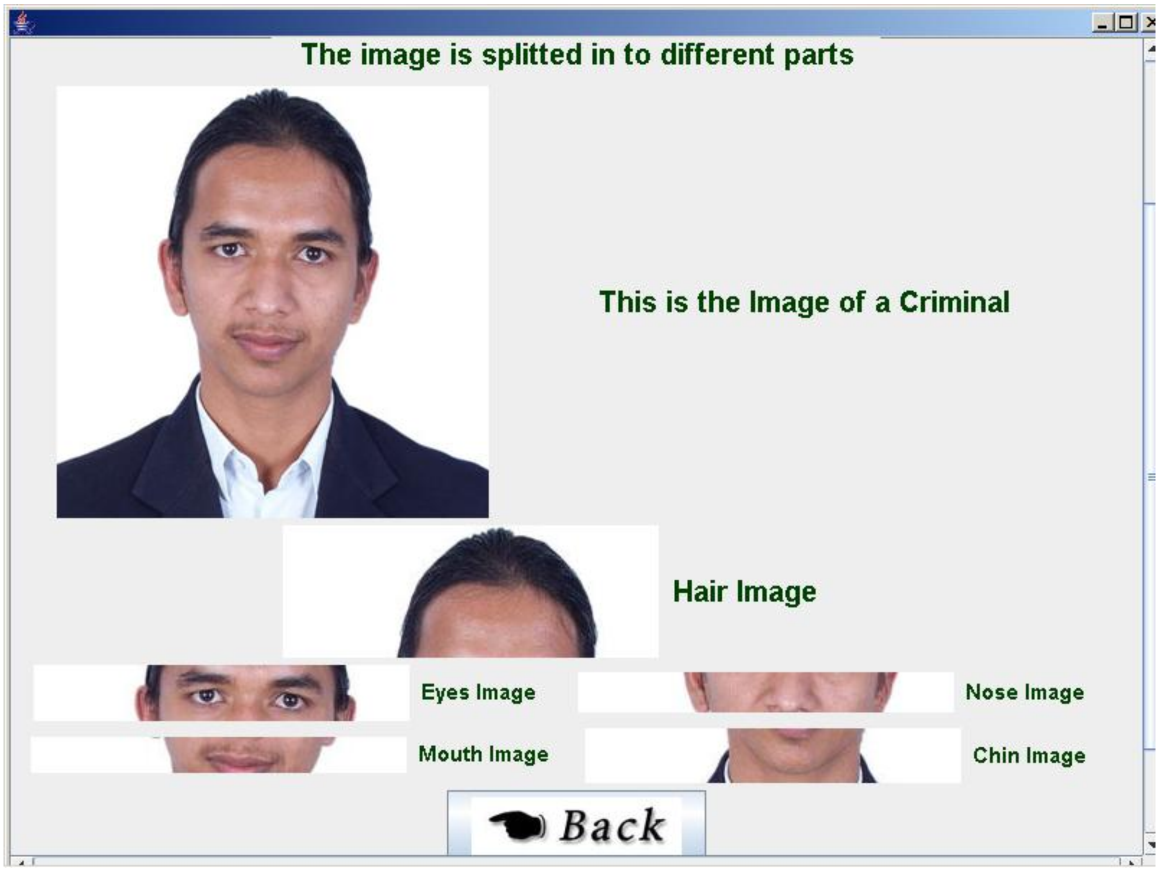
```
 the fig is stored in C:/faceproj8/admin....

change face details with updation of Head is Gray
```



This is About Face Identification System

Criminal record generally contains personal information about particular person along with photograph. To identify any criminal we need some identification regarding person, which are given by eyewitnesses. In most cases the quality and resolution of the recorded image-segment is poor and hard to identify a face To overcome this sort of problem we are developing software.Identification can be done in May ways like fingerprint, eyes, DNA etc.one of the applications is face identification.The face is our primary focus of attention in social inter course playing a major role in conveying identity and emotion Although the ability to infer intelligence or character from facial appearance is suspect,the human ability to recognize faces is remarkable.



The image is splitted in to different parts

This is the Image of a Criminal

Hair Image

Eyes Image

Nose Image

Mouth Image

Chin Image

FUTURE ENHANCEMENTS

The Future enhancements of this project include the following:

- The criminal photos may be of any size.
- By selecting any one cropped part of the criminal, we can get the full image of the criminals along with details.
- New face constructed by different cropped parts can be saved.

BIBLIOGRAPHY

Java (Swing & XML):

Java in a Nut Shell	O’Rielly
Using Java2 Platform	Joseph Weber
The Complete Reference Java 1.2	Herbert Schildt
Core Java	Kenneth Paul

System Development:

Fundamentals Of System Concepts	Jerry Fitz Gerald
System Analysis And Design	Elias M. Awad
Object Oriented Modeling And Design	James Rumabah