**INTRODUCTION TO PROJECT**

# 1. Introduction

Humans are very good at recognizing faces and complex patterns. Even a passage of time doesn't effect this capability and therefore it would help if computers become as robust as humans in face recognition. Face recognition system can help in many ways :

1) Checking for criminal records .

2) Enhancement of security by using surveillance cameras in conjunction with face recognition system.

3) Finding lost children's by using the images received  from the cameras fitted at some public places .

4) Knowing in advance if some VIP is entering the hotel.

5) Detection of a criminal at public place.

6) Can be used in different areas of science for comparing a entity with a set of entities.

7) Pattern Recognition.

This project is a step towards developing a face recognition system which can recognize static images. It can be modified to work with dynamic images. In that case the dynamic images received from the camera can first be converted in to the static one's and then the same procedure can be applied on them. But then there are lots of other things that should be considered . Like distance between the camera and the person , magnification factor, view [top ,side, front] etc.

# 2. Overall Description

## 2.1 product Desription

The Face Detection will locate faces in a given image and separate them from the scene. The extracted faces will be further transformed using processing techniques like grayscale transformation (all faces used in recognition will be grayscale images because are less sensitive in background changes and more computationally efficient), histogram equalization

## 2.2 Product Features

The Face Recognition System will recognize faces provided by the detection module. It will use the Eigenfaces algorithm for recognition and Principal Component Analysis for dimensionality reduction. The training phase needs a subjects database – each person should have a different folder (unique folder name – ID) with more face pictures in different rotation and lighting conditions. The recognition module will return a list of IDs for the recognized persons, IDs used to extract information from the MySQL database.

1. Security level users

e) Bank Manager- this is the sophisticated user which is having some knowledge about system and handle the privilege for the different types of users.

f) Administrator- this apply the different types of security policies over the system to increase the robustness of the system.

## Operating Environment

On the following environment it will work more smoothly:-

1. Java JDK version >1.6.0

2. IDE: Netbeans 6.8

3. Emulator platform for mobile phone: Java ME Wireless Toolkit 2.5.2

4. MySql version >14.12

5. C++ Compiler: g++ version 14.12

6. Mobile phone with Java MIDP 2.0

## 2.5 Design and Implementation Constraints

It will needed windows environment. It will not work on linux platform. The 128MB minimum RAM is required for the system to use. It will only work on single pc not on the network.

## 2.6 User Documentation

The on-line help is required for the users so that they can use the system properly. The tool tip text is also being provided for the knowledge of the every function.

## 2.7 Assumptions and Dependencies

We are uses the following assumptions :-

1. In this we assume that there is always a guarantor corresponding to each customer.
2. In this software we assume the interest for the Educational loan is 10%, for car loan it is 14.5%, for house loan it is 16%.
3. In this we assume only few users like employee, customer, administrator,etc.

# 3. System Features

This section illustrates the functional features of the system. It gives the total description about the system functionalities that are provided by the system. The features are described below:-

## 3.1 User name and password feature

3.1.1 Description and Priority

This is the important feature of the system. Every user must have user name and password for using the system. This feature has the highest priority.

3.1.2 Stimulus/Response Sequences

After correctly entering the user id and password a new screen will be opened for the user according to their privileges. After entering the wrong user id and password this will be give message as a dialog box.

3.1.3 Functional Requirements

REQ 1:The main requirement is that user with a user id and password can access the system.

REQ 2: Different users have different privileges like the customer having only privilege of accessing the account information but can’t change any information..

REQ 3: There must be only three tries possible for the user id and password. The system will be blocked for that particular user id.

REQ 4: when they will enter wrong password the message will be displayed.

### 3.2.1 Description and Priority

This is also important feature of the system to create new account of customer for handling loan information. This has higher priority because no feature is used without any account.

### 3.2.2 Stimulus/Response Sequences

In this feature the employee creates the customer account by filling up the form of creating account if any information is wrongly written then it will show the message. If the information is correctly entered then the software show information of account created on the next screen that the account is created.

## 3.6 Due Record

### 3.6.1 Description and Priority

This feature include the managing the customer which don’t pay their installment at proper time. when employee want to see the due installment customer they see the list from this. This has high priority.

### 3.6.2 Stimulus /Response Sequences

It shows the report of all due customers which did not pay their installment at proper time.

3.6.3 Functional Requirements.

REQ 1: It includes the list of all due installment customers. And also include how many Installments.

REQ 2: it also includes that when a new entry is added to the software it immediately Show the message about new entry.

REQ 3: It also includes the facility of deleting the entry from the due record for the employee.

## 3.7 Search, Update, Delete and Report generate features

### 3.7.1 Description and Priority

These features are for saving the employee’s time. This has the low priority. These include earching a record, update a record, delete a record and make report of today’s transaction.

### 3.7.2 Stimulus /Response Sequences

After clicking the option it show you the report which you wanted.

3.7.3 Functional Requirements

REQ 1: It must include the searching of record by different types.

REQ 2: it also includes the option of updating the record information.

REQ 3: it must include the deleting of particular record.

REQ 4: it also include the report generation option. In this there is option of

report of particular type of loan customers. Report of today’stransaction,etc.

# 4 External Interface Requirements

## 4.1 User Interfaces

The system should be highly user friendly so the user interface mut be Graphical user interface. Every option should in the menus of the GUI.

## 4.2 Hardware Interfaces

It is not applicable yet.

## 4.3 Software Interfaces

1. Software component – Java jdk1.5
2. Databases- Oracle 9i
3. Windows Xp Sp2
4. Any text Editor

Inputs to the System

1. The customer personal information
2. The guarantor information
3. The amount taken, the loan type, no of installments,etc.
4. The installment amount
5. Print command

Outputs from the system

1. The reports generated

2. The printsd slip of the installment paid

3. Display the searched information and different types of forms according to the command

# 5. Other Nonfunctional Requirements

## 5.1 Performance Requirements

it will not decide yet.

## 5.2 Security Requirements

In the security requirements we require that each user has the authentication for the usage of the system. Each user has its user name and password for access the system. There should be different privileges for each type of user. The customer can only see the information of their account. They have no privilege to change any information in his account. The employee can’t do any technical change but have the permissions of change the account information. The administrator have permissions to change the technical things in the system.

## 5.3 Software Quality Attributes

1. Highly user friendly- GUI
2. High security system- login feature and different privileges level
3. reliable, robustness is the characterstics of the system.
4. Easily learnable for the new user.
5. Customer can’t do any change to the database.

# 6 Other Requirements

1. There is facility to print the reports generated.
2. This is in the system to print the slip for the payment by the bank or the installment

payment.

1. There is Help otion in the menu for taking help in using system.

**OBJECTIVES OF PROJECT**

There are some Objectives, which are Envisaged under the Project:-

**Practicality:** The system should be sable and can be operated with average intelligence.

**Efficiency:** There should be balance amongst various factors like accuracy, comprehensiveness on one hand and response timeliness of the system on the other hand.

**Cost:** It is desirable to aim for the system with a minimum cost subject to the condition that it must satisfy the entire requirement.

**Flexibility:** The system should be modifiable depending on the changing needs of the employee or department. Such modifications should entail extensive reconstructing. It should also be portable to different computer systems.

**Security:** This is very important aspect requiring rigorous designing of database including hardware reliability, fallback procedures and physical security of data.

The development of the proposed system is done keeping in view the problems in the existing system. The proposed system will not only overcome the limitations of the present system but will also provide the following characteristics.

The objective of the system would be:

To reduce the paper work involved in managing the information.

To reduce the time constraint involved.

To centralize all the data needed.

To maintain the information about the users/employees of the organization.

To generate various reports as required

To maintain the user rights.

**System Outline view**

After firming the requirements of the system, the summary chart or data flow diagram (DFD) of the proposed system is prepared. This gives the brief of the system with respect to the inputs being considered, the outputs reports, the data being transformed and the main processed involved in the system.

**STUDY OF EXITING SYSTEM**

It was difficult to get the comprehensive information about any user. Moreover it was a time consuming affair.

It was difficult to solve the problems those were arising during a particular interface to the administrator to maintain the different user rights for various users.

Moreover there are updates issues concerned with the maintenance of the information about the users of the organization. It was required to check for the updates on a regular basis.

This system will also provide information concerned with all users, which will be very helpful to the users for getting knowledge about any particular user.

**Significance Of Project**

1. Most current systems have a physical foundation that is the root cause to quite a number of problems. By maintaining multiple store fronts, itself being an expensive proposition, store prices are forced to rise. Thus, by using our product, our clients’ competitors are at a disadvantage because their costs are significantly higher than our costs, allowing our clients to sell the same goods at a lower price. As people become more accustomed to using the internet, they view ordering products and services online as a time-saving and cost-saving experience, which is the very essence of our online shopping system.

1. This project envisages bridging the gap between the seller, the retailer and the customer. A very high flexibility is being maintained in the design process so that this project can take the following path : -
2. A multiple merchant venue with each merchant having his/her own window which the customer can visit to browse and subsequently buy the products from
3. Maintaining the deliverable goods as well as services through single or multiple windows is also on the agenda.

**Technologies Used**

1. **Model:**
   1. Core Java
   2. Swing
   3. Events
   4. NetBeans IDE
   5. Utilities
2. **Design**
   1. Swing
   2. Awt
3. **Database**
   1. mysql

**SYTEM REQUIREMENTS**

**Product Definition:**

**Problem Recognition:**

A problem is well defined very rarely. It corps out with a vague feeling of some statements that lead to vague conclusions. So the first task is to get more crucial information by interviewing and meeting concerned people. It clarifies how the problem is felt, how often it occurs, how it affects the business and which departments are suffering with this. This phase consists of the following tasks.

**Problem Definition And Initial Investigation**

This was a preliminary investigation done with a view to have a “feel” of the working of the proposed system. This phase has been identified the end-user directly involved in the system who were the managers, assistant officer and database administrator, and the development department. By understanding the working of database, its flow and also after conducting meetings and interviews with the concerned persons of the department, a clear idea about the working was obtained. A flexible approach is adapted towards people who are interviewed. Short hand written notes are prepared based on the response of the employees. The interviews are preferably conducted at the work place of the person being interviewed. Detailed investigation is done in order to define the scope of the problem .The interview is concluded with a quick resume of the ground covered during the interview .The Questionnaire technique is combined with interviews to get the best result. Proper care has been taken in the design of such questionnaires so that the persons answering these questions dose not feel hesitant. An explanatory note that serves to gain cooperation and avoid misunderstanding by setting out the purpose of the exercise clearly accomplishes each questionnaire.

**Observation technique** is also used for fact finding. The work described at the time of interview is observed personally ads it reduces the chances of misunderstanding and omissions. Some important things observed are like the flow of information through the system and important data transactions, the data being maintained and the frequency of their updating.

By the end of this phase, idea as to how the information enters the system, how it is stored, how it is processed, how information changes affects the working of the system and finally the output format required by the end-user was collected. All the information generated from this phase acted as an input to the next phase.

**Function to be Provided**:

This phase provides the overall requirement for the system what is to be done. Input for this phase is the information collected through several data collecting schemes such as survey, cross-questioning-answering etc and the raw data obtained which is not properly ordered and not in the precise manner. So here this raw data is converted into useful information written in precise manner and thus output is a formal document. After collecting all the information and requirements, they were verified from the concerned persons by presenting a diagrammatic version of the proposed system. The points missing were added to the system specifications for the desired system. So this final document provides the system requirement specifications for the desired system. It helps in reducing the total development cost and also establishes the various points for validation and verification.

**User Requirements**

Since end users are the ones who are finally going to use the system, their requirements need to be identified. This involves questioning the end users what their expectations were. The main requirement of the end user is that the system should be easy to use and take less time. In addition to these another important factor was to eliminate the need for database conversion and migration that had to be carried out presently. After conducting interviews with the users a document called the software requirement specification was created. This is the most important document that forms the basis for system development. It should be consistent, complete, unambiguous, traceable and inter-related.

This document has the following components.

**Functional Requirements:** The functional requirements specify relationship between the inputs and outputs. All the operations to be performed on the input data to obtain output are to be specified. This includes specifying the validity checks on the input and output data, parameters affected by the operations and the other operations, which must be used to transform the inputs into outputs. Functional requirements specify the behavior of the system for valid input and outputs.

**Performance Requirements**

This section includes performance of the product that are set by user interaction and studying the existing system of the organization. These are stated in complete measurable terms, so that they can be verified during system evaluation phase. Some of the performance requirements are stated below.

User Friendly: the system produced is user friendly, understandable and easy to use so that the users of the system can easily learn to use the system. For this the system is made menu-driven with well-documented programs.

Time Element (response and processing time): the response time of the system is very less and takes less time to execute queries and triggers.

Maximum Throughput: the system gives maximum throughput with relevant output

Robustness: the system will be able to handle undesirable situations and errors encountered at various levels e.g. if the user supplies invalid input for processing, the system gracefully halts, displaying a message to the user indicating the cause of the error and prompting him it enter the correct input.

Flexibility: the system is flexible in nature so that likely changes and alterations can easily be made.

Information Security: records in the system must be safe, confidential and must be prevented from unauthorized access.

Moral and User Satisfaction: system will be able to satisfy the user requirements; this is the main and conspicuous measure of the system performance. Also the system must raise the moral of the user. The higher the moral, greater the expected work performance level.

**Methodology Adopted:**

Prototyping Model has been used for Software Development according o which a throwaway prototype of the proposed system, based on the currently known requirements is given to the user so that he has a fair idea about how the proposed system is going o be. This will help in deciding the interface, I/O requirements. It can be easily adjudged that I/O are big in number, can increase exponentially and may create a big chaos if not restricted properly. As the user spends some time on the prototype, he will become more precise about his own requirements. This prototype will provide him environment analogous to the proposed system’s environment. Because of object oriented support in J2EE, various concepts like (reusability, polymorphism, isolation etc) are already there but for the efficient management of system components, component based software engineering will also be exercised, which will help in resultant library of components, benefiting fast development. Because of lack of hierarchical structure in object oriented approach, there is no meaning of Bottom-up or Top-down testing. Testing will begin from the most rudimentary levels of the system and will move towards higher level components which will be based on design face rather than coding face. In little words it can be aid that “Cluster Testing” will be exercised to scrutinize all the parts and their associative functionality.

**Feasibility Analysis:**

A feasibility study is a test of a system proposal according to its workability impact on organization, ability to meet user needs and effective use of resources. The objective of a feasibility study is not to solve a problem but to acquire a sense of its scope. During the study, the problem definition is crystallized and the aspects of the problem to be included in the system are determined. After the initial investigation of the system that helped to have in-depth study of the existing system, understanding its strength and weaknesses and the requirements for the new proposed system.

1) Economic Feasibility: Economic Analysis is the most frequently used method for evaluating the effectiveness of a system. Most commonly known as cost/benefit analysis, the procedure is to determine the benefit and savings that are expected from a system and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the system.

2) Technical Feasibility: Technical Feasibility centers on the existing available operating system and to what extent it can be incorporated in any of the government applications.

In proposed system budget is not a constraint and all hardware required is technical feasible and mostly already exit. The software in the proposed system is technical feasible. So this project is technical feasible.

3) Behavioral Feasibility: People are inherently resistant to change, and the computer have been known to facilitate change. An adverbs reaction and resistance is always from the user staff. On the basis of feasibility analysis it is recommended that the project may be approved from design, development implementation.

**Introduction to J2EE Framework:**

Java is an object-oriented programming language with a built-in application programming interface (API) that can handle graphics and user interfaces and that can be used to create applications or applets. Because of its rich set of API's, similar to Macintosh and Windows, and its platform independence, Java can also be thought of as a platform in itself. Java also has standard libraries for doing mathematics.

Much of the syntax of Java is the same as C and C++. One major difference is that Java does not have pointers. However, the biggest difference is that you must write object oriented code in Java. Procedural pieces of code can only be embedded in objects. In the following we assume that the reader has some familiarity with a programming language. In particular, some familiarity with the syntax of C/C++ is useful.

In Java we distinguish between applications, which are programs that perform the same functions as those written in other programming languages, and applets, which are programs that can be embedded in a Web page and accessed over the Internet. Our initial focus will be on writing applications. When a program is compiled, a byte code is produced that can be read and executed by any platform that can run Java.

Java is platform independent. A Java program can run equally well on any architecture that has a Java enabled browser. With the release of Netscape Navigator 2.0 that includes Windows 95, Windows NT , Sun Solaris, Sun OS 4.1.3, SGI IRIX, OSF/1, HP-UX with more to come.

Java isn't just for web sites. Java is a programming language that lets you do almost anything you can do with a traditional programming language like Fortran or C++. However Java has learned from the mistakes of its predecessors. It is considerably cleaner and easier to use than those languages.

**Advantages of Java:**

**1) Simple**

Java has the bare bones functionality needed to implement its rich feature set. It does not add lots of syntactic sugar or unnecessary features.

**2) Object-Oriented:** Almost everything in Java is either a class, a method or an object. Only the most basic primitive operations and data types (int, for, while, etc.) are at a sub-object level.

**3) Platform Independent** Java programs are compiled to a byte code format that can be read and run by interpreters on many platforms including Windows 95, Windows NT, and Solaris 2.3 and later.

**4) Safe:** Java code can be executed in an environment that prohibits it from introducing viruses, deleting or modifying files, or otherwise performing data destroying and computer crashing operations.

**5) High Performance:** Java can be compiled on the fly with a Just-In-Time compiler (JIT) to code that rivals C++ in speed.

**6) Multi-Threaded:** Java is inherently multi-threaded. A single Java program can have many different things processing independently and continuously.

**OOP Principles:**

The object oriented programming languages provide mechanisms that help you implement the object oriented model.

**1) Data Hiding and Encapsulation****:** One of the important object-oriented techniques is hiding the data within the class and making it available only through the methods. This technique is known as encapsulation because it seals the data (and internal methods) safely inside the "capsule" of the class, where it can be accessed only by trusted users (i.e., by the methods of the class).

The most important reason is to hide the internal implementation details of your class. If you prevent programmers from relying on those details, you can safely modify the implementation without worrying that you will break existing code that uses the class.

Another reason for encapsulation is to protect your class against accidental or willful stupidity. A class often contains a number of interdependent fields that must be in a consistent state. If you allow a programmer (including yourself) to manipulate those fields directly, he may change one field without changing important related fields, thus leaving the class in an inconsistent state. If, instead, he has to call a method to change the field, that method can be sure to do everything necessary to keep the state consistent. Similarly, if a class defines certain methods for internal use only, hiding these methods prevents users of the class from calling them.

When all the data for a class is hidden, the methods define the only possible operations that can be performed on objects of that class. Once you have carefully tested and debugged your methods, you can be confident that the class will work as expected. On the other hand, if all the fields of the class can be directly manipulated, the number of possibilities you have to test becomes unmanageable.

There are other reasons to hide fields and methods of a class, as well:

Internal fields and methods that are visible outside the class just clutter up the API. Keeping visible fields to a minimum keeps your class tidy and therefore easier to use and understand.

If a field or method is visible to the users of your class, you have to document it. Save yourself time and effort by hiding it instead.

**2) Inheritance:** Inheritance is the process by which one object acquires the properties of another object. This is important because it supports the concept of hierarchical classification. Most knowledge is made by hierarchical classification.

Inheritance is a compile-time mechanism in Java that allows you to extend a class (called the base class or super class) with another class (called the derived class or subclass).

In Java, inheritance is used for two purposes:

Class inheritance - create a new class as an extension of another class, primarily for the purpose of code reuse. That is, the derived class inherits the methods of the base class.

Interface inheritance - create a new class to implement the methods defined as part of an interface for the purpose of sub typing. That is a class that implements an interface “conforms” to the interface.

In Java, unlike C++, these two types of inheritance are made distinct by using different language syntax. For class inheritance, Java uses the keyword extends and for interface inheritance Java uses the keyword implements.

For example:

class Base {

private int x;

public int f() { ... }

protected int g() { ... } // NOTE: protected access means visible to subclass only

}

class Derived extends Base {

private int y;

public void h() { y = g(); ... }

}

In Java, unlike C++, only single class inheritance is supported. I.e., for a given class, there is only one super class.

3) Polymorphism

Polymorphism is the capability of an action or method to do different things based on the object that it is acting upon. This is the third basic principle of object oriented programming.

Two types of polymorphism:

Overloading and Overriding.

Dynamic method binding.

Introduction to Mysql

**Concepts used**

Concepts used in the proposed system are RDBMS (Relational Database Management System) as Oracle (which is used as back end for the system) is RDBMS based and OOP (Object oriented Programming) as Visual Basic (which is the front end of the system) is based on the concepts of OOP.

**RDBMS ( Relational Database Management System) :-**

RDBMS stands for Relational Database Management System. An RDBMS is a computer program, which provides the user the facility to store and retrieve data in a manner consistent with a defined model, called the Relational model.

The Relational model has three components:

1) Structural component to build the model

* Relation (Table)
* Attributes (columns)
* Constraints

1. Integrity component, which defines the operational rules of the data, model.
2. Manipulation component, which is the component to manipulate the structure.

**DATA STRUCTURES:-**

The fundamentals rule of the Relational model is that the data is seen as tables. A table is formally referred to as a relation. Every relation has the same structure i.e. the same number of columns and rows. Each column which represents as attribute of the relation must be unique.

Oracle being an RDBMS, stores the data in tables. Many tables can be created and related to each other. All the tables form the Database.

**DATA INTEGRITY** :-

The Relational model also provides data integrity, which means also provides data integrity, which means that the data is accepted based on certain rules and therefore, data is valid. Data integrity is maintained using a set f rules referred to as integrity constraints.

**DATA MANIPULATION :-**

In order to manipulate the data within tables, the database access language is provided called SQL. SQL stands for Structured Query language.

Structure Query language is an English like language, which is used to store and retrieve data from a database. So, it is a database language to help users to extract information from a database easily.

**OOP :-**

The fundamental idea behind object Oriented languages is to combine into a single unit both data and functions that operate on the data. Such a unit is called and ‘object’

**Object Oriented Programming (OOP)** fall into two broad categories increased programming productivity and decreased maintenance. Costs Well –chosen classes and objects may be re-used in other applications. Objects help partition complex problems into simpler ones that are easier to implement.

Applications testing are generally shorter because objects are small and independent code. When a bug does occur, a particular object is usually implicated. This makes isolating and correcting the bug easier.

**Introduction to SQL:**

The SQL (R) software delivers a very fast, multi-threaded, multi-user, and robust SQL (Structured Query Language) database server. Mysql is intended for mission-critical, heavy-load production systems as well as for embedding into mass-deployed software. SQL is a trademark of SQL AB.

**Main Features of SQL**

The following list describes some of the important characteristics of the Mysql Database Software.

Internals and Portability

Written in C and C++.

Tested with a broad range of different compilers.

Works on many different platforms. Uses GNU Automake, Autoconf, and Libtool for portability.

APIs for C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, and Tcl are available.

Fully multi-threaded using kernel threads. This means it can easily use multiple CPUs if they are available.

Provides transactional and non-transactional storage engines.

A very fast thread-based memory allocation system.

Very fast joins using an optimized one-sweep multi-join.

In-memory hash tables which are used as temporary tables.

SQL functions are implemented using a highly optimized class library and should be as fast as possible. Usually there is no memory allocation at all after query initialization.

The SQL code is tested with Purify (a commercial memory leakage detector) as well as with Valgrind, a GPL tool.

The server is available as a separate program for use in a client/server networked environment. It is also available as a library that can be embedded (linked) into standalone applications. Such applications can be used in isolation or in environments where no network is available.

**2. Column Types**

Many column types: signed/unsigned integers 1, 2, 3, 4, and 8 bytes long, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET, ENUM, and OpenGIS geometry types.

Fixed-length and variable-length records.

**3. Commands and Functions**

Full operator and function support in the SELECT and WHERE clauses of queries. For example:

SQL> SELECT CONCAT(first\_name, ' ', last\_name)

-> FROM tbl\_nam

-> WHERE income/dependents > 10000 AND age > 30;

Full support for SQL GROUP BY and ORDER BY clauses. Support for group functions (COUNT(), COUNT(DISTINCT ...), AVG(), STD(), SUM(), MAX(), MIN(), and GROUP\_CONCAT()).

Support for LEFT OUTER JOIN and RIGHT OUTER JOIN with both standard SQL and ODBC syntax.

Support for aliases on tables and columns as required by SQL-92.

DELETE, INSERT, REPLACE, and UPDATE return the number of rows that were changed (affected). It is possible to return the number of rows matched instead by setting a flag when connecting to the server.

The SQL-specific SHOW command can be used to retrieve information about databases, tables, and indexes. The EXPLAIN command can be used to determine how the optimizer resolves a query.

Function names do not clash with table or column names. For example, ABS is a valid column name. The only restriction is that for a function call, no spaces are allowed between the function name and the `(' that follows it. You can mix tables from different databases in the same query (as of Version 3.22).

**4. Security:** A privilege and password system that is very flexible and secure, and allows host-based verification. Passwords are secure because all password traffic is encrypted when you connect to a server.

**5. Scalability and Limits:** Handles large databases. We use Oracle 9i/10g with databases that contain 50 million records. We also know of users that use Oracle 9i/10g with 60,000 tables and about 5,000,000,000 rows.

Up to 32 indexes per table are allowed. Each index may consist of 1 to 16 columns or parts of columns. The maximum index width is 500 bytes (this may be changed when compiling Oracle 9i/10g ). An index may use a prefix of a CHAR or VARCHAR column.

**6. Connectivity:** Clients may connect to the Oracle 9i/10g using TCP/IP sockets on any platform. On Windows systems in the NT family (NT, 2000, or XP), clients may connect using named pipes. On Unix systems, clients may connect using Unix domain socket files.

The Connector/ODBC interface provides SQL support for client programs that use ODBC (Open-DataBase-Connectivity) connections. For example, you can use MS Access to connect to your Oracle 9i/10g . Clients may be run on Windows or Unix. Connector/ODBC source is available.

The Connector/JDBC interface provides SQL support for Java client programs that use JDBC connections. Clients may be run on Windows or Unix. Connector/JDBC source is available.

The server can provide error messages to clients in many languages.

All data is saved in the chosen character set. All comparisons for normal string columns are case-insensitive.

Sorting is done according to the chosen character set (the Swedish way by default). It is possible to change this when the Oracle 9i/10g is started. To see an example of very advanced sorting, look at the Czech sorting code. Oracle 9i/10g supports many different character sets that can be specified at compile and runtime.

**7. Clients and Tools:** The Oracle 9i/10g has built-in support for SQL statements to check, optimize, and repair tables. These statements are available from the command line through the sqlcheck client. SQL also includes isamchk, a very fast command-line utility for performing these operations on ISAM tables.

All SQL programs can be invoked with the --help or -? options to obtain online assistance.

Database Models

A data model comprises

\* a data structure

\* a set of integrity constraints

\* operations associated with the data structure

Examples of data models include:

\* hierarchic

\* network

\* relational

Models other than the relational database module used to be quite popular. Each model type is appropriate to particular types of problem. The Relational model type is the most popular in use today, and the other types are not discussed further.

Relational Databases

The relational data model comprises:

\* relational data structure

\* relational integrity constraints

\* relational algebra or equivalent (SQL)

# SQL is an ISO language based on relational algebra

# relational algebra is a mathematical formulation

Relational Data Structure

A relational data structure is a collection of tables or relations.

\* A relation is a collection of rows or tuples

\* A tuple is a collection of columns or attributes

\* A domain is a pool of values from which the actual attribute values are taken.

Tuples and Domains

Figure : Tuples and Domains

Domain and Integrity Constraints

\* Domain Constraints

o limit the range of domain values of an attribute

o specify uniqueness and `nullness' of an attribute

o specify a default value for an attribute when no value is provided.

\* Entity Integrity

o every tuple is uniquely identified by a unique non-null attribute, the primary key.

\* Referential Integrity

o rows in different tables are correctly related by valid key values (`foreign' keys refer to primary keys).

Structure of a Table

In the design process tables are defined, and the relationships between tables identified. Remember a relationship is just a link between two concepts. Consider a table holding "drivers" and a table holding "car" information... Each car is owned by a driver, and therefore there is a link between "car" and "driver" to indicate which driver owns which car.

In the subsequent pages we will refer back to this driver and car arrangement. To make the examples easier, lets create some example data.

CAR

The CAR table has the following structure:

\* REGNO : The registration number of the car

\* MAKE : The manufacturer of the car

\* COLOUR: The colour of the car

\* PRICE : The price of the car when it was bought new

DRIVER

The DRIVER table has the following structure:

\* NAME : The full name of the driver

\* DOB : The data of birth of the driver

Relationship between CAR and DRIVER

The DRIVER and the CAR has a relationship between them of N:1. This indicates that a CAR can have only 1 DRIVER, but that a DRIVER can own more than 1 CAR simultaneously.

ER Diagram of DRIVER and CAR

Figure : ER Diagram of DRIVER and CAR

In the design section we can see that this requires a FOREIGN KEY in the CAR end of the relationship. This foreign key allows us to implement the relationship in the database. We will call this field OWNER.

Example Data

DRIVER

NAME DOB

Jim Smith 11 Jan 1980

Bob Smith 23 Mar 1981

Bob Jones 3 Dec 1986

CAR

REGNO MAKE COLOUR PRICE OWNER

F611 AAA FORD RED 12000 Jim Smith

J111 BBB SKODA BLUE 11000 Jim Smith

A155 BDE MERCEDES BLUE 22000 Bob Smith

K555 GHT FIAT GREEN 6000 Bob Jones

SC04 BFE SMART BLUE 13000

Columns or Attributes

Each column is given a name which is unique within a table

Each column holds data of one specified type. E.g.

integer decimal

character text data

-- the range of values can be further constrained

If a column of a row contains no data, we say it is NULL. For example, a car just off the production line might not have an owner in the database until someone buys the car. A NULL value may also indicate that the value is unavailable or inappropriate . This might be the case for a car which is being destroyed or a car where two people are arguing in court that they are both the owner.

Some important rules:

\* All rows of a table must be different in some way from all other rows.

\* Sometimes a row is referred to as a Tuple.

\* Cardinality is the number of ROWS in a table.

\* Arity is the number of COLUMNS in a table.

Primary Keys

A table requires a key which uniquely identifies each row in the table. This is entity integrity.

The key could have one column, or it could use all the columns. It should not use more columns than necessary. A key with more than one column is called a composite key.

A table may have several possible keys, the candidate keys, from which one is chosen as the primary key.

No part of a primary key may be NULL.

If the rows of the data are not unique, it is necessary to generate an artificial primary key.

In our example, DRIVER has a primary key of NAME, and CAR has a primary key of REGNO. This database will break if there are two drivers with the same name, but it gives you an idea what the primary key means...

Note that if for some reason JIM SMITH decided to change his name to "BRIAN SMITH", then not only would this have to be changed in DRIVER, but it would also have to be changed in CAR. If you changed it only in DRIVER, there would be some foreign keys pointing to DRIVER looking for a driver who does not exist. This would be an error called a REFERENTIAL INTEGRITY error, and the DBMS stops you making changes to the database which would result in such an error.

SQL Basics

Basic SQL Statements include:

\* CREATE - a data structure

\* SELECT - read one or more rows from a table

\* INSERT - one or more rows into a table

\* DELETE - one or more rows from a table

\* UPDATE - change the column values in a row

\* DROP - a data structure

In the remainder of this section only simple SELECT statements are considered.

Simple SELECT

The syntax of a SELECT statement is :

SELECT column FROM tablename

This would produce all the rows from the specified table, but only for the particular column mentioned. If you want more than one column shown, you can put in multiple columns separating them with commas, like:

SELECT column1,column2,column3 FROM tablename

If you want to see all the columns of a particular table, you can type:

SELECT \* FROM tablename

Lets see it in action on CAR...

SELECT \* FROM car;

REGNO MAKE COLOUR PRICE OWNER

F611 AAA FORD RED 12000 Jim Smith

J111 BBB SKODA BLUE 11000 Jim Smith

A155 BDE MERCEDES BLUE 22000 Bob Smith

K555 GHT FIAT GREEN 6000 Bob Jones

SC04 BFE SMART BLUE 13000

SELECT regno FROM car;

REGNO

F611 AAA

J111 BBB

A155 BDE

K555 GHT

SC04 BFE

SELECT colour,owner FROM car;

COLOUR OWNER

RED Jim Smith

BLUE Jim Smith

BLUE Bob Smith

GREEN Bob Jones

BLUE

In SQL, you can put extra space characters and return characters just about anywhere without changing the meaning of the SQL. SQL is also case-insensitive (except for things in quotes). In addition, SQL in theory should always end with a ';' character. You need to include the ';' if you have two different SQL queries so that the system can tell when one SQL statement stops and another one starts. If you forget the ';' the online interface will put one in for you. For these reasons all of the following statements are identical and valid.

SELECT REGNO FROM CAR;

SELECT REGNO FROM CAR

Select REGNO from CAR

select regno FROM car

SELECT

regno

FROM car;

Comments

Sometimes you might want to write a comment in somewhere as part of an SQL statement. A comment in this case is a simple piece of text which is meaningful to yourself, but should be ignored by the database. The characters '--', when they appear in a query, indicate the start of a comment. Everything after that point is ignored until the end of that line. The following queries are all equivalent.

SELECT regno

FROM car;

SELECT regno -- The registration number

FROM car -- The car storage table

;

Warning: You cannot put a comment immediately after a ';'. Comments are only supported within the text of an SQL statement. The following will cause SQL errors:

SELECT regno

FROM car; -- Error here as comment is after the query

-- Error here as comment is before the start of the query

SELECT regno

FROM car;

SELECT filters

Displaying all the rows of a table can be handy, but if we have tables with millions of rows then this type of query could take hours. Instead, we can add "filters" onto a SELECT statement to only show specific rows of a table. These filters are written into an optional part of the SELECT statement, known as a WHERE clause.

SELECT columns

FROM table

WHERE rule

The "rule" section of the WHERE clause is checked for every row that a select statement would normally show. If the whole rule is TRUE, then that row is shown, whereas if the rule is FALSE, then that row is not shown.

The rule itself can be quite complex. The simplest rule is a single equality test, such as "COLOUR = 'RED'".

Without the WHERE rule would show:

SELECT regno from CAR;

REGNO

F611 AAA

J111 BBB

A155 BDE

K555 GHT

SC04 BFE

From the database we know that only F611 AAA is RED, and the rest of the cars are either BLUE or GREEN. Thus a rule COLOUR = 'RED' is only true on the row with F611 AAA, and false elsewhere. With everything in a query:

SELECT regno from CAR

WHERE colour = 'RED';

REGNO

F611 AAA

An important point to note is that queries are case sensitive between the quotes. Thus 'RED' will work, but 'red' will produce nothing. The case used in the quotes must match perfectly the case stored in the table. SQL is not forgiving and if you forget you can be scratching you head for hours trying to fix it.

Note also that "colour" does not have to appear on the SELECT line as a column name. It can if you want to see the colour, but there is no requirement for it to be there. Therefore this will work too:

SELECT regno,colour from CAR

WHERE colour = 'RED';

REGNO COLOUR

F611 AAA RED

Comparisons

SQL supports a variety of comparison rules for use in a WHERE clause. These include =,!=,<>, <, <=, >, and >=.

Examples of a single rule using these comparisons are:

WHERE colour = 'RED' The colour attribute must be RED

WHERE colour != 'RED' The colour must be a colour OTHER THAN RED

WHERE colour <> 'RED' The same as !=

WHERE PRICE > 10000 The price of the car is MORE THAN 10000

WHERE PRICE >= 10000 The price of the car is EQUAL TO OR MORE THAN 10000

WHERE PRICE < 10000 The price of the car is LESS THAN 10000

WHERE PRICE <= 10000 The price of the car is EQUAL TO OR LESS THAN 10000

Note that when dealing with strings, like RED, you must say 'RED'. When dealing with numbers, like 10000, you can say '10000' or 10000. The choice is yours.

Dates

Date rules are some of the hardest rules to get right in writing SQL, yet there is nothing particularly complex about them. The hard part is working out what it means to be GREATER THAN a particular date.

In date calculations, you can use all the normal comparators.

SELECT name,dob from driver

NAME DOB

Jim Smith 11 Jan 1980

Bob Smith 23 Mar 1981

Bob Jones 3 Dec 1986

SELECT name,dob from driver

WHERE DOB = '3 Dec 1986'

NAME DOB

Bob Jones 3 Dec 1986

In other comparators, it is important to realise that a date gets bigger as you move into the future, and smaller as you move into the past. Thus to say 'DATE1 < DATE2' you are stating that DATE1 occurs before DATE2 on a calender. For example, to find all drivers who were born on or after the 1st Jan 1981 you would do:

SELECT name,dob from driver

WHERE DOB >= '1 Jan 1981'

NAME DOB

Bob Smith 23 Mar 1981

Bob Jones 3 Dec 1986

The syntax for dates does change slightly on difference database systems, but the syntax '1 Jan 2000' works in general on all systems. Oracle also allows dates like '1-Jan-2000' and '1-Jan-00'. If you specify a year using only the last two digits, Oracle uses the current date to compute the missing parts of the year, converting '00' to '2000'. Do not get confused by saying '87' for '1987' and ending up with '2087'!

BETWEEN

Sometimes when you are dealing with dates you want to specify a range of dates to check. The best way of doing this is using BETWEEN. For instance, to find all the drivers born between 1995 and 1999 you could use:

SELECT name,dob from driver

WHERE DOB between '1 Jan 1985' and '31 Dec 1999'

NAME DOB

Bob Jones 3 Dec 1986

Note that the dates have day of the month and month in them, and not just the year. In SQL, all dates must have a month and a year. If you try to use just a year the query will fail.

BETWEEN works for other things, not just dates. For instance, to find cars worth between 5000 and 10000, you could execute:

SELECT regno

FROM car

where price between 5000 and 10000;

REGNO PRICE

K555 GHT

NULL

The NULL value indicates that something has no real value. For this reason the normal value comparisons will always fail if you are dealing with a NULL. If you are looking for NULL, for instance looking for cars without owners using OWNER of CAR, all of the following are wrong!

SELECT regno from CAR WHERE OWNER = NULL WRONG!

SELECT regno from CAR WHERE OWNER = 'NULL' WRONG!

Instead SQL has a special comparison operator called IS which allows us to find NULL values. There is also an opposite to IS, called IS NOT, which finds all the values which are not NULL. So finding all the regnos of cars with current owners would be (note that if they have an owner, then the owner has a value and thus is NOT NULL):

SELECT REGNO from CAR

WHERE OWNER is not NULL

REGNO

F611 AAA

J111 BBB

A155 BDE

K555 GHT

And finding cars without owners would be:

SELECT REGNO from CAR

WHERE OWNER is NULL

REGNO

SC04 BFE

LIKE

When dealing with strings, sometimes you do not want to match on exact strings like ='RED', but instead on partial strings, substrings, or particular patterns. This could allow you, for instance, to find all cars with a colour starting with 'B'. The LIKE operator provides this functionality.

The LIKE operator is used in place of an '=' sign. In its basic form it is identical to '='. For instance, both of the following statements are identical:

SELECT regno FROM car WHERE colour = 'RED';

SELECT regno FROM car WHERE colour LIKE 'RED';

The power of LIKE is that it supports two special characters, '%' and '-'. These are equivalent to the '\*' and '?' wildcard characters of DOS. Whenever there is an '-' character in the string, any character will match. Whenever there is an '%' character in the string, 0 or more characters will match. Consider these rules:

name LIKE 'Jim Smith' Matches 'Jim Smith'

name LIKE '\_im Smith' Matches things like 'Jim Smith' or 'Tim Smith'

name LIKE '\_\_\_ Smith' Matches 'Jim Smith' and 'Bob Smith'

name LIKE '% Smith' Matches 'Jim Smith' and 'Bob Smith'

name LIKE '% S%' Matches 'Jim Smith' and 'Bob Smith'

name LIKE 'Bob %' Matches 'Bob Jones' and 'Bob Smith'

name LIKE '%' Matches anything not null

Note however that LIKE is more powerful than a simple '=' operator, and thus takes longer to run. If you are not using any wildcard characters in a LIKE operator then you should always replace LIKE with '='.

Logical Operators and Aggregation

Contents

\* Logical Operators

o AND

o OR

o NOT

o Precedence

o Parenthesis

\* DISTINCT

\* ORDER BY

\* IN

\* Aggregate Functions

o AVERAGE

o SUM

o MAX

o MIN

o COUNT

o COUNT DISTINCT

\* GROUP BY aggregation

\* HAVING

Logical Operators

In the previous section we saw how a single rule could be added to a query using a WHERE clause. While this is useful, usually more than a single rule is required to produce the correct result. To support multiple rules we need to make use of NOT, AND, OR and parentheses.

AND

The basic way of supporting multiple rules in a single query is by making use of AND. AND provides a way of connecting two rules together such that ALL the rules must be true before the row is shown. Lets make use again of the CAR table:

REGNO MAKE COLOUR PRICE OWNER

F611 AAA FORD RED 12000 Jim Smith

J111 BBB SKODA BLUE 11000 Jim Smith

A155 BDE MERCEDES BLUE 22000 Bob Smith

K555 GHT FIAT GREEN 6000 Bob Jones

SC04 BFE SMART BLUE 13000

Consider the case where a police eye witness spots a car driving away from a crime. The witness reports that the car was BLUE and had the character '5' somewhere in the REGNO field. Taking these rules seperately...

SELECT regno from CAR

WHERE colour = 'BLUE';

REGNO

J111 BBB

A155 BDE

SC04 BFE

SELECT regno from CAR

WHERE regno LIKE '%5%'

REGNO

A155 BDE

K555 GHT

We are looking for a REGNO in common to both these results, which means the car we are looking for is 'A155 BDE'. Rather than doing this ourselves we want the computer to identify the right car in a single query. The two rules in question are linked together with an AND.

SELECT regno from CAR

WHERE colour = 'BLUE' AND regno LIKE '%5%'

;

REGNO

A155 BDE

Remember that the layout of the SQL is independent of spaces and newlines, so this query is identical to:

SELECT regno

FROM CAR

WHERE colour = 'BLUE'

AND regno LIKE '%5%'

;

You can link as many rules together as you like. So for instance if the witness said that the car was BLUE, had a 5 in the registration number, and that someone said the car was owned by Bob, we could write a query:

SELECT regno

FROM CAR

WHERE colour = 'BLUE'

AND regno LIKE '%5%'

AND owner LIKE 'Bob %'

;

OR

AND allows us to link rules together such that all rules must be true to see that row. Think of AND as 'As well as'. Sometimes we want to say 'or that' or 'either' rather than 'As well as'. To do this we use OR. For instance, lets say that the police witness said

that the car colour was either RED or BLUE, and they were not sure which. If you said:

WHERE colour = 'RED' AND colour = 'BLUE'

then no rows would be produced, as you are saying you want rows where the colour is both RED and BLUE at the same time (RED as well as BLUE). What we need is either RED OR BLUE.

SELECT REGNO,COLOUR from CAR

WHERE colour = 'RED'

OR colour = 'BLUE';

REGNO COLOUR

F611 AAA RED

J111 BBB BLUE

A155 BDE BLUE

SC04 BFE BLUE

NOT

The NOT operator does the opposite of whatever comparison is being done. NOT is not frequently needed, as there is usually an opposite operator already. For instance, if you wanted the opposite of:

WHERE colour = 'RED'

You could say

WHERE colour != 'RED'

Using NOT you could also say

WHERE NOT colour = 'RED'

While not particularly useful in these simple examples, NOT comes into its own once you start to use parentheses.

Precedence

AND, OR, and NOT become more complex to understand when you mix them together in a single query. The problem is that the rules are combined together, not in the order you write them, but in the order of their precedence. This states that NOT is done first, then AND, and finally OR. This can make a BIG difference to your queries!

Consider the case of the police witness. Lets say that the car being looked for had a 5 in the registration number, and was either RED or BLUE.

SELECT REGNO,COLOUR from CAR

WHERE colour = 'RED' -- 1

OR colour = 'BLUE' -- 2

AND regno LIKE '%5%' -- 3

;

REGNO COLOUR

F611 AAA RED

A155 BDE BLUE

In this query, rule 3 and rule 2 and ANDed together first, as they have a higher precedent. Only then is rule 1 ORed in. Thus this query says "The car is BLUE with a 5 in the regno" OR "the car is RED". What was wanted was to have rules 1 and 2 done first, and then 3, so that the query says "The car is either RED or BLUE" AND "the car had a 5 in the regno". To do this we need to use parenthesis.

Parenthesis

Parenthesis, or brackets, are used to instruct the database which rules should be done first. The database uses a simple ruleset to understand brackets. If you have any brackets, then the rule in the brackets is done first. If you have brackets within brackets, then the inner brackets are done first. In the example above, the right query can be generated as:

SELECT REGNO,COLOUR from CAR

WHERE (colour = 'RED'

OR colour = 'BLUE')

AND regno LIKE '%5%'

;

REGNO COLOUR

A155 BDE BLUE

The following queries are all identical in function to the above query...

SELECT REGNO,COLOUR from CAR

WHERE (colour = 'RED' OR colour = 'BLUE')

AND regno LIKE '%5%';

SELECT REGNO,COLOUR from CAR

WHERE ( (colour = 'RED' OR colour = 'BLUE')

AND regno LIKE '%5%');

Do not use brackets where they are not needed, as it makes the query harder for users to understand whats going on.

DISTINCT

Lets say you want a list of all the colours of cars in the database. The COLOUR field of CAR gives you this, and thus:

SELECT colour FROM car;

COLOUR

RED

BLUE

BLUE

GREEN

BLUE

This result was not the ideal one wanted. BLUE for some reason appears 3 times. It does this because BLUE appears 3 times in the original data. Sometimes this duplication is what is wanted, other times we want only to see the colours appearing once. To tell the database to show the rows only once, you can use the keyword DISTINCT. This appears immediately after the word SELECT. DISTINCT effectively means that all rows which appear must be unique, and any duplicate rows will be deleted.

SELECT DISTINCT colour FROM car;

COLOUR

BLUE

GREEN

RED

ORDER BY

When a query is executed the results are displayed in an almost random order. The order is dependent on how the database management system was written. This is fine usually, but sometimes giving the data out in a particular order would make the data must more useful. There is a special clause, ORDER BY, which can be added to the end of a query to give the data a particular order.

SELECT make FROM car;

MAKE

FORD

SKODA

MERCEDES

FIAT

SMART

To order alphabetically (which in SQL is known as ascending or ASC) you can use ORDER BY or ORDER BY ASC.

SELECT make FROM car

ORDER BY make;

MAKE

FIAT

FORD

MERCEDES

SKODA

SMART

This is identical to

SELECT make FROM car

ORDER BY make ASC;

To order things in the reverse ordering, you can use ORDER BY DESC.

SELECT make FROM car

ORDER BY make DESC;

MAKE

SMART

SKODA

MERCEDES

FORD

FIAT

For complex orderings involving more than one column, you can specify multiple columns in the ORDER BY statement, simply by separating each column name with a comma. Thus a query to sort cars by colour and then make would look like:

SELECT make,colour FROM car

ORDER BY colour,make;

MAKE COLOUR

SKODA BLUE

SMART BLUE

MERCEDES BLUE

FIAT GREEN

FORD RED

IN

\* IN (list of values) determines whether a specified value is in a set of one or more listed values.

List the registration numbers of cars which are either SKODA or SMART

SELECT regno,make

FROM car

WHERE make = 'SKODA' or make='SMART'

;

REGNO MAKE

J111 BBB SKODA

SC04 BFE SMART

This can be rewritten using IN.

SELECT regno,make

FROM car

WHERE make IN ('SKODA','SMART')

;

A good way to think of IN is to consider it as "is one of the following".

Aggregate Functions

Operators exist in SQL to give results based on the statistics of a group of values stored in the database. Such operators include "what is the maximum number" and "what is the average". These functions are called SET or AGGREGATE functions.

AVERAGE

To calculate the average of a column you use the AVG function.

SELECT price FROM car;

PRICE

12000

11000

22000

6000

13000

SELECT avg(price) FROM car;

avg(price)

12800

SUM

To calculate the SUM of all values in a column you use the SUM function.

SELECT sum(price) FROM car;

sum(price)

64000

MAX

To calculate the maximum or biggest value present in a particular column you can use the MAX function.

SELECT max(price) FROM car;

sum(price)

22000

MIN

To calculate the minimum or smallest value present in a particular column you can use the MIN function.

SELECT min(price) FROM car;

sum(price)

6000

COUNT

To work out how many rows are in a particular query result you can use the COUNT function.

Using "count(column)" counts how many rows exist in the answer where that column is NOT NULL. Using "count(\*)" counts how many rows exist independent of NULL values.

SELECT count(price) FROM car;

sum(price)

5

In this case, the following SQL produces the same answer.

SELECT count(\*) FROM car;

COUNT DISTINCT

Sometimes you do not want to count how many rows are in a particular column, but how many different values are stored in a column. There is a special variation of count which allows you to do that, known as COUNT DISTINCT. Its syntax is a little unusual...

SELECT count(colour) from car;

sum(price)

5

SELECT count(DISTINCT colour) from car;

sum(price)

3

GROUP BY aggregation

The aggregate functions are excellent when all you want is a single number answer. Frequently what is needed is statistical analysis in groups. For instance, what is the maximum cost of a car given its colour. Here we are wanting two columns, one the car colour, and the second column the highest cost. Intuitively one might think:

SELECT colour,max(price)

FROM car

;

If you were to run this query it would produce a "group by" error.

Instead, what you have to do is consider all aggregate functions in your query, and over which columns they are going to be grouped. In this case we are grouping on colour, and want the maximum price within each "colour" group. To tell the computer this we use GROUP BY.

SELECT colour,price

FROM car

;

COLOUR PRICE

RED 12000

BLUE 11000

BLUE 22000

GREEN 6000

BLUE 13000

SELECT colour,max(price)

FROM car

GROUP BY colour

;

COLOUR max(PRICE)

RED 12000

BLUE 22000

GREEN 6000

If you are ever confused by what to put in the GROUP BY, then here is a simple rule which is 99% accurate... If you have a SELECT line with aggregate functions, then you need a GROUP BY listing all the column names from the SELECT line which are not used by the functions. In this example "price" and "colour" are columns from SELECT, but as "price" is used in MAX, only "colour" needs to go into the GROUP BY statement.

HAVING

One annoying feature of SQL is that aggregate functions are executed at almost the last stage of the query process. This makes writing queries like "Which owners own more than 1 car" quite complex. Ideally we would like to write:

SELECT owner from car where count(owner) > 1;

The problem is that this does not work! Aggregate functions cannot appear in a WHERE clause, so this query is illegal... To get around this you can have the HAVING clause. HAVING works in an identical way to WHERE, except that it runs very late in the process and allows aggregate functions. It is also VERY expensive for the database to use, so do not use it until it is absolutely essential.

Our query can now be rewritten thus:

SELECT owner,count(regno)

FROM car

GROUP BY owner

HAVING count(regno)>1;

This query also shows how many cars the owner owns. You do not have to have the function in the HAVING on the SELECT line. The following also works:

SELECT owner

FROM car

GROUP BY owner

HAVING count(regno)>1;

If you remember count(\*) counts how many rows there are in the answer. With a GROUP BY, it counts how many rows are in each group. The difference between a count with \* or with a column name is that using a column name makes the count ignore NULL entries in that column, whereas with \* NULL entries are counted too. In our example, REGNO is never NULL, so the query is also identical to:

SELECT owner

FROM car

GROUP BY owner

HAVING count(\*)>1;

JOINs and VIEWs

Contents

\* Multiple source tables

o JOIN condition

o Traditional JOIN

o Modern JOIN

o OUTER JOIN

o FULL OUTER JOIN

\* Naming

\* Aliases

\* Self Joins

\* VIEWs

o DROP View

Multiple source tables

Sometimes you will need to write a query which uses more than a single table. This is perfectly acceptable in SQL, but needs a little care... It is very easy to produce multi-table queries which produce mostly rubbush.

The basic concept for producing multi-table queries is that all the tables you need must be listed in the FROM clause of the query. For example, lets try to write a query which lists the owner name, date of birth, and the registration number, for each car in the database. REGNO is in CAR, but DOB is in DRIVER. Therefore both tables are needed. The basic query looks like:

SELECT name,dob,regno

FROM car,driver

;

The order in which the tables appear in the FROM line are irrelevent. However, this query does not produce the right answer. The reason for this is that the DBMS does not understand how to relate one table to the other.

JOIN condition

In order to usefully join multiple tables together we need to explain to the database how they are joined. The FROM clause takes all rows in all the tables listed, and forms a new table which contains all combinations of the original rows. Most of the time this results in rubbish. Look at this example.

SELECT \*

FROM car

;

REGNO MAKE COLOUR PRICE OWNER

F611 AAA FORD RED 12000 Jim Smith

J111 BBB SKODA BLUE 11000 Jim Smith

A155 BDE MERCEDES BLUE 22000 Bob Smith

K555 GHT FIAT GREEN 6000 Bob Jones

SC04 BFE SMART BLUE 13000

SELECT \*

FROM driver

;

NAME DOB

Jim Smith 11 Jan 1980

Bob Smith 23 Mar 1981

Bob Jones 3 Dec 1986

SELECT \*

FROM car,driver

;

REGNO MAKE COLOUR PRICE OWNER NAME DOB

F611 AAA FORD RED 12000 Jim Smith Jim Smith 11 Jan 1980

J111 BBB SKODA BLUE 11000 Jim Smith Jim Smith 11 Jan 1980

A155 BDE MERCEDES BLUE 22000 Bob Smith Jim Smith 11 Jan 1980

K555 GHT FIAT GREEN 6000 Bob Jones Jim Smith 11 Jan 1980

SC04 BFE SMART BLUE 13000 Jim Smith 11 Jan 1980

F611 AAA FORD RED 12000 Jim Smith Bob Smith 23 Mar 1981

J111 BBB SKODA BLUE 11000 Jim Smith Bob Smith 23 Mar 1981

A155 BDE MERCEDES BLUE 22000 Bob Smith Bob Smith 23 Mar 1981

K555 GHT FIAT GREEN 6000 Bob Jones Bob Smith 23 Mar 1981

SC04 BFE SMART BLUE 13000 Bob Smith 23 Mar 1981

F611 AAA FORD RED 12000 Jim Smith Bob Jones 3 Dec 1986

J111 BBB SKODA BLUE 11000 Jim Smith Bob Jones 3 Dec 1986

A155 BDE MERCEDES BLUE 22000 Bob Smith Bob Jones 3 Dec 1986

K555 GHT FIAT GREEN 6000 Bob Jones Bob Jones 3 Dec 1986

SC04 BFE SMART BLUE 13000 Bob Jones 3 Dec 1986

In our query, we are only interested in table combinations which obey the rules of the FOREIGN KEY relationship which joins these two tables. If you remember, the PRIMARY KEY of DRIVER (NAME) was copied into CAR as a FOREIGN KEY (named OWNER). Thus this FROM generated table needs to be filtered so that only the rows where NAME = OWNER appear. Note that this FROM

generated table containing all the combinations of the listed tables is known as the cartesian cross product. We will return to the subject of the cross product in the relational algebra chapter.

Now, in order to get our query working properly, we put in the JOIN condition NAME = OWNER. There are two basic ways to do this, which we will call traditional and modern. Both ways are usually referred to as an INNER JOIN.

Traditional JOIN

To put the join condition NAME = OWNER into a query using the traditional approach is simply to list it in the WHERE clause as a rule. So...

SELECT \*

FROM car,driver

WHERE owner = name

;

REGNO MAKE COLOUR PRICE OWNER NAME DOB

F611 AAA FORD RED 12000 Jim Smith Jim Smith 11 Jan 1980

J111 BBB SKODA BLUE 11000 Jim Smith Jim Smith 11 Jan 1980

A155 BDE MERCEDES BLUE 22000 Bob Smith Bob Smith 23 Mar 1981

K555 GHT FIAT GREEN 6000 Bob Jones Bob Jones 3 Dec 1986

Modern JOIN

To put the join condition NAME = OWNER into a query using the modern approach, you rewrite the FROM line to say:

FROM table1 JOIN table2 ON (rules)

So in our example:

SELECT \*

FROM car JOIN driver ON (owner = name)

;

REGNO MAKE COLOUR PRICE OWNER NAME DOB

F611 AAA FORD RED 12000 Jim Smith Jim Smith 11 Jan 1980

J111 BBB SKODA BLUE 11000 Jim Smith Jim Smith 11 Jan 1980

A155 BDE MERCEDES BLUE 22000 Bob Smith Bob Smith 23 Mar 1981

K555 GHT FIAT GREEN 6000 Bob Jones Bob Jones 3 Dec 1986

OUTER JOIN

You might have noticed a result in the previous query (when there were no join conditions) where there was a NULL in the OWNER field. This is for a car with no current owner. Once the join condition was inserted into the query the rows with NULL owners were filtered out. This is usually exactly what is desired, but

sometimes we want the join condition to be obeyed if the fields are not NULL, and the rules to be broken when there is a NULL. Such JOINs are called OUTER JOINS. In the modern JOIN syntax you simply insert either the word LEFT or the word RIGHT in front of the word JOIN.

To decide if the right word is LEFT of RIGHT, you have to consider where the NULL values will be. In our example query, the NULL value is in the OWNER field, which belongs to the CAR table. The current JOIN is:

FROM car JOIN driver on (owner = name)

^ ^ ^

| | |

| | +-------- To the right of JOIN

| +--------------- The JOIN statement

+------------------------- To the left of JOIN

As the CAR table has the NULL values, and CAR appears to the left of the word JOIN in the query, the right keyword to use is LEFT JOIN. The query becomes:

SELECT \*

FROM car LEFT JOIN driver ON (owner = name)

;

REGNO MAKE COLOUR PRICE OWNER NAME DOB

F611 AAA FORD RED 12000 Jim Smith Jim Smith 11 Jan 1980

J111 BBB SKODA BLUE 11000 Jim Smith Jim Smith 11 Jan 1980

A155 BDE MERCEDES BLUE 22000 Bob Smith Bob Smith 23 Mar 1981

K555 GHT FIAT GREEN 6000 Bob Jones Bob Jones 3 Dec 1986

SC04 BFE SMART BLUE 13000

The OUTER JOIN fills in the missing data (for the things which do not satisfy the rules) with NULLs. Note that if you swap CAR and DRIVER around in the JOIN statement you can write it as a RIGHT JOIN just as easily...

SELECT \*

FROM driver RIGHT JOIN car ON (owner = name)

;

The order of the rules in ON have no significance in deciding what is right and what is left.

**SOFTWARE REQUIREMENT SPECIFICATION (SRS)**

**System Analysis:** System Analysis is the process of studying the business processors and procedures, generally referred to as business system, to see how they can operate and whether improvement is needed. This may involve examining data moments and storage, machines and technology used in the system., programs that control the machines, people providing inputs, doing the processing and receiving the outputs.

**1) Investigation Phase:** The Investigation Phase is also known as the Fact-Finding stage or the analysis of the current system. This is detailed study conducted with the purpose of wanting to fully understand the existing system and to identify the basic information requirement.

**2) Detailed Study of the Existing System:** This phase provides the overall requirement for the system what is to be done. Input for this phase is the information collected through several data collecting schemes such as survey, cross-questioning-answering etc and the raw data obtained which is not properly ordered and not in the precise manner. So here this raw data is converted into useful information written in precise manner and thus output is a formal document. After collecting all the information and requirements, they were verified from the concerned persons by presenting a diagrammatic version of the proposed system. The points missing were added to the system specifications for the desired system. So this final document provides the system requirement specifications for the desired system. It helps in reducing the total development cost and also establishes the various points for validation and verification.

**Software Requirement Specification:** Software Requirement Specification is produced at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by establishing a complete information description, a detailed functional description, a representation of system behavior, and indication of performance requirement and design constraints appropriate validation criteria, and other information pertinent to requirement.

The introduction to SRS specification states the goals and objectives of the software, describing it in the context of the computer based system. The Information Description provides a detailed description of the problem that the software must solve. Information content, flow and structure are documented. A description of each function required to solve the problem is presented in the functional description. Validation Criteria is the mos important and ironically the most often neglected section of the software requirement specification. SRS can be used for the different purposes. Here are the major uses:

**1) Statement of User Need:** Since end users are the ones who are finally going to use the system, their requirements need to be identified. This involves questioning the end users what their expectations were. The main requirement of the end user is that the system should be easy to use and take less time. In addition to these another important factor was to eliminate the need for database conversion and migration that had to be carried out presently. After conducting interviews with the users a document called the software requirement specification was created. This is the most important document that forms the basis for system development. It should be consistent, complete, unambiguous, traceable and inter-related.

This document has the following components.

**Functional Requirements:** The functional requirements specify relationship between the inputs and outputs. All the operations to be performed on the input data to obtain output are to be specified. This includes specifying the validity checks on the input and output data, parameters affected by the operations and the other operations, which must be used to transform the inputs into outputs. Functional requirements specify the behavior of the system for valid input and outputs.

**2) Statements of requirements for implementation:** This section includes performance of the product that are set by user interaction and studying the existing system of the organization. These are stated in complete measurable terms, so that they can be verified during system evaluation phase. Some of the performance requirements are stated below.

**User Friendly:** the system produced is user friendly, understandable and easy to use so that the users of the system can easily learn to use the system. For this the system is made menu-driven with well-documented programs.

**Time Element** (response and processing time): the response time of the system is very less and takes less time to execute queries and triggers.

**Maximum Throughput:** the system gives maximum throughput with relevant output

**Robustness:** the system will be able to handle undesirable situations and errors encountered at various levels e.g. if the user supplies invalid input for processing, the system gracefully halts, displaying a message to the user indicating the cause of the error and prompting him it enter the correct input.

**Flexibility:** the system is flexible in nature so that likely changes and alterations can easily be made.

**Information Security:** records in the system must be safe, confidential and must be prevented from unauthorized access.

**Moral and User Satisfaction:** system will be able to satisfy the user requirements; this is the main and conspicuous measure of the system performance. Also the system must raise the moral of the user. The higher the moral, greater the expected work performance level.

**SYSTEM DESIGN**

After the analysis phase we have with us the details of the existing system and the requirements of the user for the new system. This phase diverts focus from the problem domain to the solution domain. It acts as a bridge between the requirement phase and its solution. The design phase focuses on the detailed implementation of the system recommended in the feasibility study. Emphasis is on translating performance specifications into design specifications.

**The External Design**

External design consists of conceiving, planning out and specifying the externally observable characteristics of the software product. These characteristics include user displays or user interface forms and the report formats, external data sources and the functional characteristics, performance requirements etc. External design begins during the analysis phase and continues into the design phase.

**Design objectives:-**

The primary objective of the design is to deliver the requirements as specified in the feasibility report. These are the some of the objectives, which I kept in mind.

**1) Practicality:** The system is quite stable and can be operated by the people with average intelligence.

**2) Efficiency:**  I tried to involve accuracy, timeliness and comprehensiveness of the system output.

**3) Cost:** It is desirable to aim for the system with a minimum cost subject to the condition that it must satisfy the entire requirement.

**4) Flexibility:** I have tried that the system should be modifiable depending on the changing needs of the user. Such modifications should entail extensive reconstructing or recreation of software. It should also be portable to different computer systems.

**5)Security:** This is very important aspect which I followed in this designing phase and tried to covers the areas of hardware reliability, fallback procedures, and physical security of data.

**Characteristics of a Well Defined System**

In design an efficient and effective system is of great importance to consider the human factor and equipment that these will require to use. System analyst must evaluate the capabilities and limitations of the personal and corresponding factors of the equipment itself.

**The characteristics associated with effective system operations are:**

1) Accessibility

1) Decision Making Ability

3) Economy

4) Flexibility

5) Reliability

6) Simplicity

Success is a new system pivots on its acceptance or non-acceptance by the organization.

**Personnel:**

If the operating system is convinced that the new system will not benefit them, it appears one, and the system is in serious trouble. To overcome this resistance participation by operating personal during all phases of the changeover is necessary because they constitute the organization, which must use alive in with newly design system. An effective system produces not only information at the lowest cost pertinent and timely for making decision.

**DATABASE DESIGN:**

The overall objective in the development of the database technology has been to treat data as an organizational resource and as an integrated whole. Database management system allows data to be protected and organize separately from other resources. Database is an integrated collection of data. The most significant of data as seen by the programs and data as stored on the direct storage access storage devices. This is the difference between logical and physical data. The organization of data in the database aims to achieve free major objectives:

1) Data Integration

2) Data Integrity

3) Data Independence

**DATA FLOW DIAGRAM**

In our DFD, we give names to data flows, processes, and data stores. Although the names are descriptive of the data, they do not give details. So the following the DFD, our interest is to build some structured place to keep details of the contents of data flow, processes, and data store. A data dictionary is a structured repository of data about data. It is a set of rigorous definition of all DFD data element and data structure

**4.2.1) DFD Symbols**

In the DFD, there are five symbols,

1. **A Square** defines a source (originator) or destination of system data.
2. **An Arrow** identifies data flow- data in motion .It is pipeline through which information flows.
3. **A circle** or a **bubble** (or a oval bubble) represents a process that transforms incoming data flow(s) into outgoing data flow(s)
4. **An Open rectangle** is a data store-data at rest, or temporary repository of data.

5 **AN HORIZONTAL LINE** represents data stored or data at rest or a temporary rest

repository of data.

The DFD was first developed by “Larry Constatine” as a way of expressing system requirements in a graphical form. A DFD, also referred to as a bubble chart has a purpose of clarifying system requirements and identifying major transformations that will become the program in this system design.

* A square defines a source of destination or system data.
* An arrow line identifies the data flow or data in motion. It is a pipeline through which information flows.

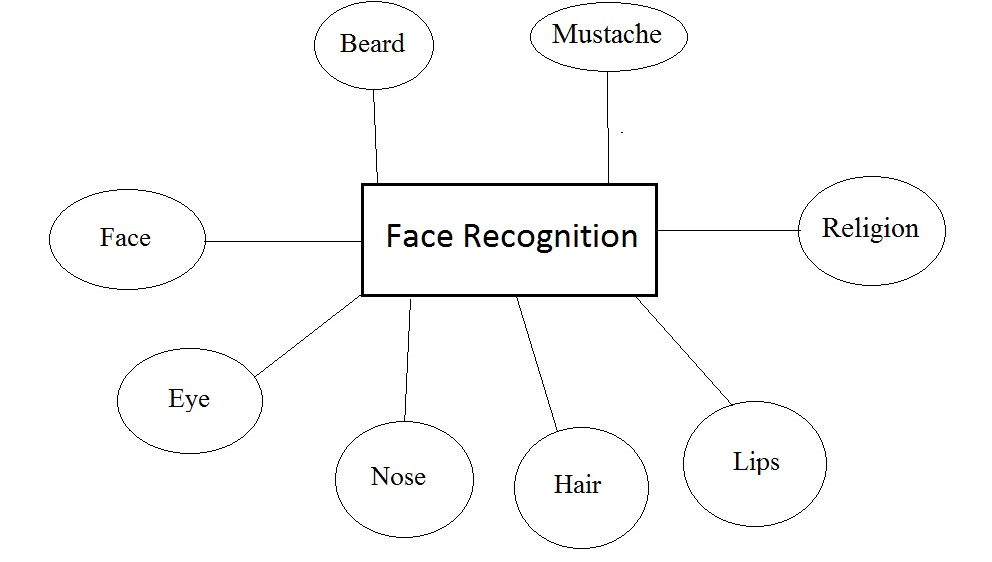
* A circle or bubble represents a process transform incoming data flow in to outgoing data flow.
* A horizontal line represents data stored or data at rest or a temporary rest repository of data.
* An open rectangle refers to the database storage

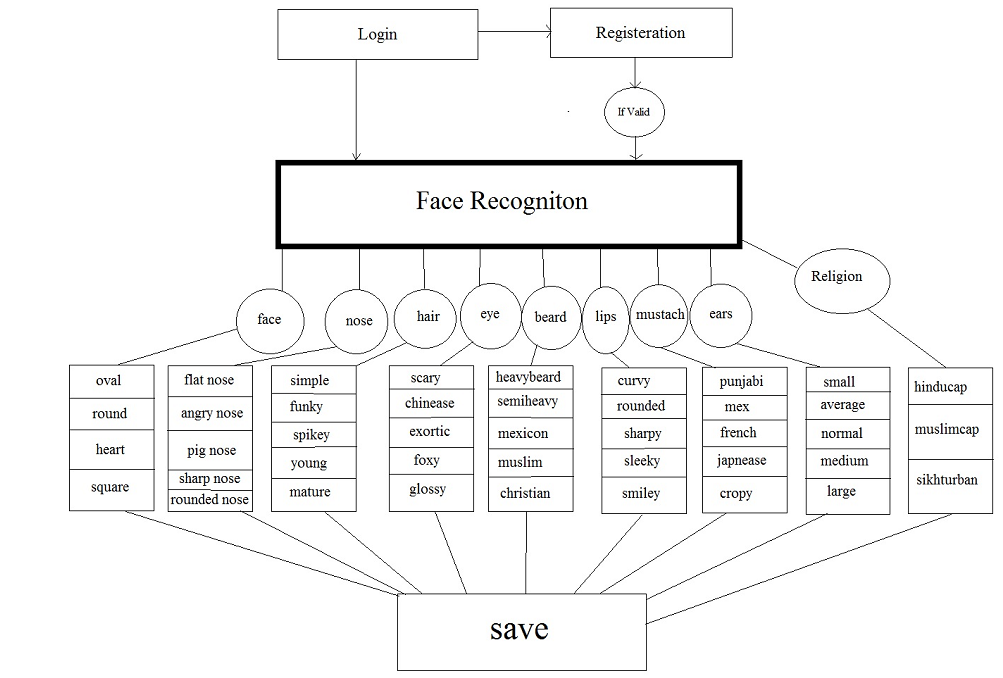
1**) Data Flow Diagram: Level 0**

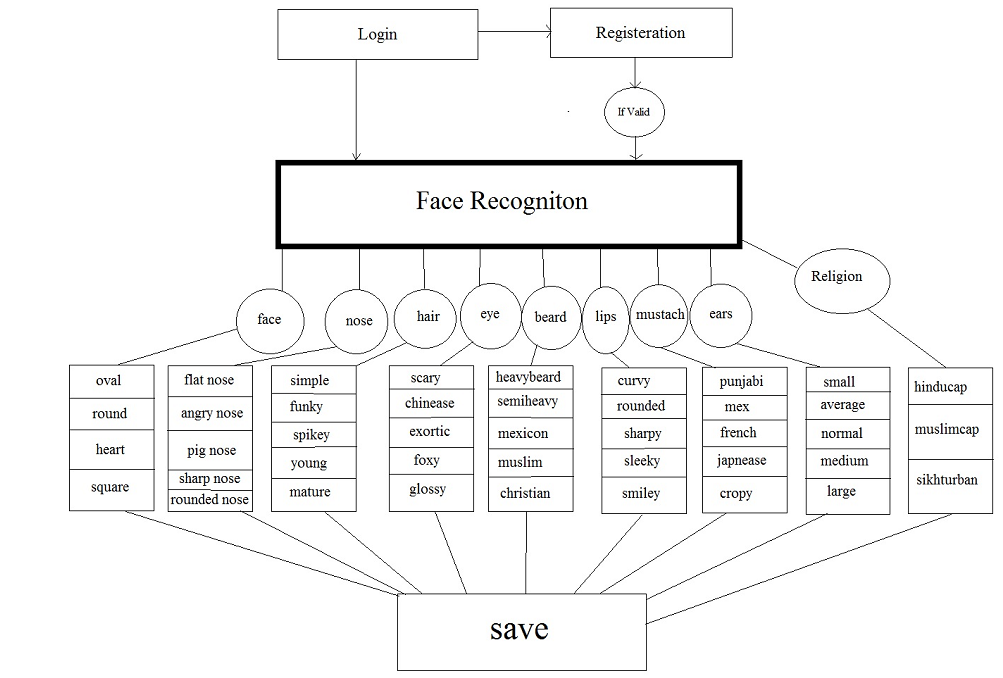
**FACE**

**DETECTION**

**2) Data Flow Diagram: Level 1**

**DATA FLOW DIAGRAM OF BANKING SYSTEM**

****

**SYSTEM TESTING**

The purpose of system testing is to identify and correct errors in the candidate system. Testing is and important element of software quality assurance ad represents the ultimate review of specification, design and coding. The increasing visibility of the software as a system element and the cost associated with a software failure are motivated forces for well planned, through testing.System testing was conducted in order to detect errors and for comparing then the final system with the requirement specification reports, i.e. whether the system meets requirements. During testing the software was executed with the set of test cases and the output of programs for the test cases was evaluated to determine if the program is performing as it was expected to. Testing presents an interesting challenge for the software engineers attempt to build software from an abstract concept to an acceptable implementation. In testing engineer create a series of test cases that occurs when errors are uncovered. Testing is the process of executing a program for finding errors. A good test is one that has the high probability of finding an uncovered error. A successful error is one that uncovers undiscovered errors. The term error is used to refer the difference between actual output of the software and the current output. Fault is a condition that causes the software to fail to perform its required function. Software reliability is defined as a required function. Software reliability is defined as the probability that the software will not undergoes failures for a specified times under specified condition. Failure is the inability of a system or a component to perform a required function according to its specification. Different levels of testing were employed for software to make it error free, fault free and reliable.

**Unit Testing**:

Unit testing was conducted first. Different modules of the software were tested against the specifications produced during design of the modules. Verification of the code produced during the coding phase was done. Each module was tested separately.

Unit testing focuses verification effort on the smallest unit of software design module. This uncovers errors within the boundary of a module. Unit testing is actually White box testing both the external things as well as the internal codes are tested. In testing, the interfaces are tested in order to ensure the proper flow of data in and out of the module. The boundary testing is done to ensure that the module keeps the limit of it. All independent paths are tested to ensure that all statements are tested at least once. At last the error path is also tested.

Unit testing comprises the set of tests performed by an individual programmer prior to integration of the unit into a larger system. There are four categories of test that can be performed on a program unit

1) Functional Unit

2) Performance Unit

3) Stress Unit

4) Structure Unit

**System Testing:**

Then system testing was conducted. Here the entire software system was tested.

The reference document used for this process was requirement document and the goal was to see if the software meets its requirements.

System testing includes the thorough testing of the product. System testing is actually a series of different tests whose primary purpose is to fully exercise the computer based system. The tests are recovery testing: this checks the recovery of the system when failure occurs. This is to ensure that there are recovery procedures for error occurrences.

System testing involves unit testing, integration testing, acceptance testing. Careful planning and scheduling are required to ensure that modules will be available for integration into the evolving software product when needed. A test plan has the following steps:

1) Prepare test plan

2) Specify conditions for user acceptance testing

3) Prepare test data for program testing

4) Prepare test data for transaction path testing

5) Plan user testing

6) Compile/Assemble program

7) Prepare job performance aids

8) Prepare operational documents

**Objectives of Testing.**

First of all objectives should be clear.

1) Testing as a process of executing a program with the intent of finding errors.

2) To perform testing, test cases are designed. A test case is a particular made up of artificial situation upon which a program is exposed so as to find errors. So a good test case is one that finds undiscovered errors.

3) If testing is done properly, it uncovers errors and after fixing those errors we have software that is being developed according to specifications.

The above objective implies a dramatic change in viewpoint .The move counter to the commonly held view than a successful test is one in which no errors are found. In fact, our objective is to design tests that a systematically uncover different classes of errors and do so with a minimum amount of time and effort.

**Testing principles**

Before applying methods to design effective test cases, software engineer must understand the basic principles that guide the software testing process. Some of the most commonly followed principles are:

All test should be traceable to customer requirements as the objective of testing is to uncover errors, it follows that the most severe defects (from the customers point of view) are those that causes the program to fail to meet its requirements.

Tests should be planned long before the testing begins. Test planning can begin as soon as the requirement model is complete. Detailed definition of test cases can begin as soon as the design model has been solidated. Therefore, all tests can be planned and designed before any code can be generated.

The Pareto principle applies to software testing stated simply the Pareto principle implies that 80 percent of all errors uncovered during testing will likely be traceable to 20 percent of all program modules. The problem of course, is to isolate these suspects’ modules and to thoroughly test them.

Testing should begin “in the small “and progress towards testing “in large”. The first tests planned and executed generally focus on individual modules. As testing progresses, testing shifts focus in an attempt to find errors in integrated clusters of modules and ultimately in the entire system.

Exhaustive testing is not possible. The number of paths permutations for impossible to execute every combination of paths during testing. It is possible however to adequately

cover program logic and to ensure that all conditions in the procedural design have been exercised.

To be most effective, an independent third party should conduct testing. By “most effective”, we mean testing that has the highest probability of finding errors (the primary objective of testing).

Software project management is an umbrella activity within software engineering. It begins before any technical activity is intimated and continues throughout the definition, development, and maintenance of computer software.

Three p’s 1 have a substantial influence on software project management –people, problem, and process. People must be organized into effective teams, motivated to do high quality software work, and coordinated to achieve effective communication. The problem communicated from customer to developer, partitioned (decomposed) into its

constitute parts, and positioned for work by the software team. The process must be adapted to the people and the problem. A common process framework is selected, an appropriate software engineering paradigm is applied, and a set of work is chosen to get the job done.

The pivotal element in all software projects is people. Software engineers can be organized in a number of different team structures that range from traditional control hierarchies to “Open paradigm” team. A variety of coordination and communication techniques can be applied to support the work of the team. In general, formal reviews and informal person-to-person communication have the most value for the practitioners.

The project management activity encompasses measurement and metrics, estimation, risk analysis, schedules, tracking and control, and control. Each of these steps was followed during project also.

**Test information flow**

Testing is a complete process. For testing we need two types of inputs:

Software configuration –It includes software requirement specification, design specification and source code of program. Software configuration is required so that testers know what is to be expected and tested.

Test configuration – it is basically test plan and procedure. Test configuration is testing plan that is, the way how the testing will be conducted on the system. It specifies the test cases and their expected value. It also specifies if any tools for testing are to be used.

Test cases are required to know what specific situations need to be tested. When tests are evaluated, test results are compared with actual results and if there is some error, then debugging is done to correct the error. Testing is a way to know about quality and reliability. Error rate that is the occurrence of errors is evaluated. This data can be used to predict the occurrence of errors in future.

Test case design

We know, test cases are integral part of testing. So we need to know more about test cases and how these test cases are designed. The most desired or obvious expectation from the test cases is that it should be able to find most errors with the least amount of time and effort.

A software product can be tested in two ways. In first approach, only overall functioning of the product is tested. Inputs are given and outputs ate checked. This approach is called black box testing. It does not care about the internal functioning of the product.

The other approach is called white box testing. Here the internal functioning of the product is tested. Each procedure is tested for its accuracy. It is more intensive than black

box testing. But for the overall product both these techniques are crucial. There should be sufficient number of tests in both categories to test the overall product.

**Basic Testing Methods**

**White box testing**

White box testing is performed to reveal problems with the internal structure of a program. This requires the tester to have detailed knowledge of the internal structure. A common goal of white box testing is to ensure a test case exercises every path through a program. A fundamental strength that all white box strategies share is that the entire software implementation is taken into account during testing, which facilitates error detection even when software specification is vague or incomplete. The effectiveness or thoroughness of white box testing is commonly expressed in terms of test or code coverage metrics, which measure the fraction of code exercised by test cases.

**Basic Path Testing**

It is a white box technique. It was proposed by Tom McCabe. These tests guarantee to execute every statement in the program at least one time during testing. Basic set is the set of all execution paths of a procedure.

**Black Box Testing**

Black box tests are performed to access how well a program meets its requirements, looking for incorrect or missing functionality. Functional tests typically exercise code with valid or nearly valid input for which the expected output is known. This includes concepts such as ‘boundary values’.

Performance tests evaluate response time, memory usage, throughput, device utilization and execution time. Stress tests push the system to or beyond its specified limits to evaluate its robustness and error handling capabilities. Reliability tests monitor system response to representative user input, counting failures over time to measure or certify reliability.

**Black box testing uncovers the following types of errors**

1) Incorrect or missing functions

2) Interface errors

3) External database access

4) Performance errors

5) Initialization and termination errors

The following techniques are employed during black box testing

**Integration Testing**

One of the most difficult aspects of software development is the integration and testing of large untested subsystems. The integrated system frequently fails in significant and mysterious ways and it’s difficult to fix it.

Integration testing exercises several units that have been combined to form a module, subsystem or system. Integration testing focuses on the interfaces between units, to make sure the units work together. The nature of this phase is certainly ‘white box’, as we must have knowledge of the units to recognize if we have been successful in focusing them together in the module.

**SYSTEM IMPLEMENTATION**

**INTRODUCTION**

Software testing is used in association with [verification and validation](http://en.wikipedia.org/wiki/Verification_and_Validation) (V&V). Verification is the checking of or testing of items, including software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques such as reviews, inspections, and walkthroughs. Validation is the process of checking what has been specified is what the user actually wanted. Verification: Have we built the software right? (i.e. does it match the specification). Validation: Have we built the right software? (i.e. Is this what the customer wants?)

Implementation is the stage in the project where the theoretical design is turned into the working system and is giving confidence to the new system for the users i.e. will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of method to achieve the change over, an evaluation, of change over methods. A part from planning major task of preparing the implementation is education of users. The more complex system is implemented, the more involved will be the system analysis and design effort required just for implementation. An implementation coordinating committee based on policies of individual organization has been appointed. The implementation process begins with preparing a plan for the implementation for the system. According to this plan, the activities are to be carried out, discussions may regarding the equipment has to be acquired to implement the new system.

Implementation is the final and important phase. The most critical stage is in achieving a successful new system and in giving the users confidence that the new system will work and be effective. The system can be implemented only after thorough testing is done and if it found to working according to the specification. This method also offers the greatest security since the old system can take over if the errors are found or inability to handle certain types of transaction while using the new system.

At the beginning of the development phase a preliminary implementation plan is created to schedule and manage the many different activities that must be integrated into plan. The implementation plan is updated throughout the development phase, culminating in a change over plan for the operation phase. The major elements of implementation plan are test plan, training plan, equipment installation plan, and a conversion plan.

**Types Of Implementation**

There are three types of implementation:

Implementation of a computer system to replace a manual system.

Implementation of a new computer system to replace an existing system.

Implementation of a modified application to replace an existing one, using the same computer.

Successful implementation may not guarantee improvement in the organization using the new system, but improper installation will prevent it. It has been observed that even the best system cannot show good result if the analysts managing the implementation do not attend to every important detail. This is an area where the systems analysts need to work with utmost care.

**IMPLEMENTATION TOOLS:**

**Training personnel**

**Conversion Procedures**

**Post-implementation review**

**Training of Personnel involved with system**

Even well designed system can succeed or fail because of the way they are operated and used. Therefore, the quality of training received by the personal involved with the system in various capacities helps or hinders and may even prevent the successful implementation of management information system.

Those who are directly or indirectly related with the system development work must know in detail what must know in detail what their roles will be, how they can make efficient use of the system and what the system will or will not do for them. Both system operators and users need training.

**System Operators Training**

Running of the system successfully depend on the personnel working in the Computer Centre. They are Responsible for providing the necessary support. Their training must ensure that they are able to handle all possible operations, both routine and extra-ordinary in nature.

If the system calls for the installation of new equipment, such as new computer system, special terminals or different data entry machines, the operators training should include such fundamentals as how to turn the equipment on and use it, how to power off and a knowledge of what constitutes normal operations. The operators should also be trained on different type of malfunctioning, how to recognize them and what steps should also be taken whenever they arise.

**User Training**

User may be trained on use equipment, particularly in the case where, e.g. a micro computer is in use and individual involved is both operator and user. In such cases, user must be given training on how to operate and user. In such cases, user must be given training on how to operator the system also. Questions that may be trivial to the analyst, such as how to turn on a terminal, how to insert a diskette into a micro-computer or when it is safe to turn off equipment with out danger of data loss are significant problems to new users who are not familiar.

Inmost of the cases user training deals with the operation of the system itself, with proper attention given to data handling techniques. It is imperative that users be properly trained in methods of entering transaction, editing data, formulating inquiries, deleting and inserting of records. No training is complete without familiarizing users with simple systems maintenance activities. Weakness in any aspect of training may lead of awkward situation that creates user frustration and error.

**Conversion Methods**

A conversion is the process of changing from the old system to the new one. It must be properly planned and executed. Four methods are common in use. They are Parallel Systems, Direct Conversion, Pilot System and Phase In method. Each method should be considered in the light of the opportunities that it offers and problems that it may create. In general, system conversion should be accomplished in shortest possible time. Long conversion periods create problems for all persons involved including both analysts and users.

**Parallel systems:**

The most secure method of converting from an old to new system is to run both systems in parallel. This method is safest one because it ensures that in case of any problem in using new system, the organization can still fall back to the old system without the loss of time and money.

The disadvantages of parallel systems approach are:

It doubles operating costs.

The new system may not get fair trial.

**Direct conversion:**

This method converts from the old system to new system abruptly, sometimes over a weekend or even overnight. The old system is used until a planned conversion day, when it is replaced by the new system.

**Pilot system:**

Pilot approach is often preferred in the case of the new system which involves new techniques or some drastic changes in organization performance. In this method, a working version of the system is implemented in one part of the organization, such as a single work area or department**.**

**Phase –IN- method:**

This method is used when it is not possible to install a new system throughout an organization all at once. The conversion of files, training of personnel or arrival of equipment may force the staging of the implementation over a period of time, ranging from weeks to months.

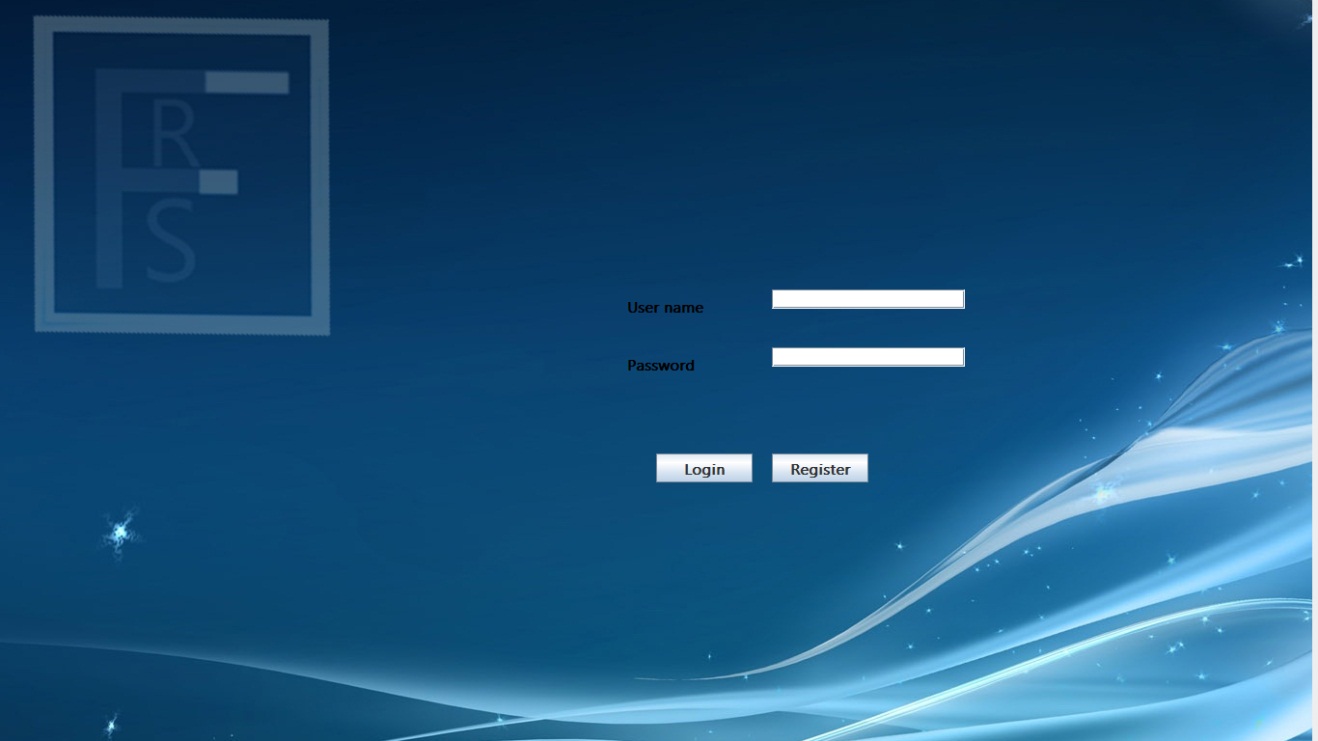
**POST IMPLEMENTATION REVIEW**

After the system is implemented and conversion is complete, a review should be conducted to determine whether the system is meeting expectations and where improvements are needed. A post implementation review measures the systems performance against predefined requirement. It determines how well the system continues to meet the performance specifications.

**SNAP SHOTS**

Login Page

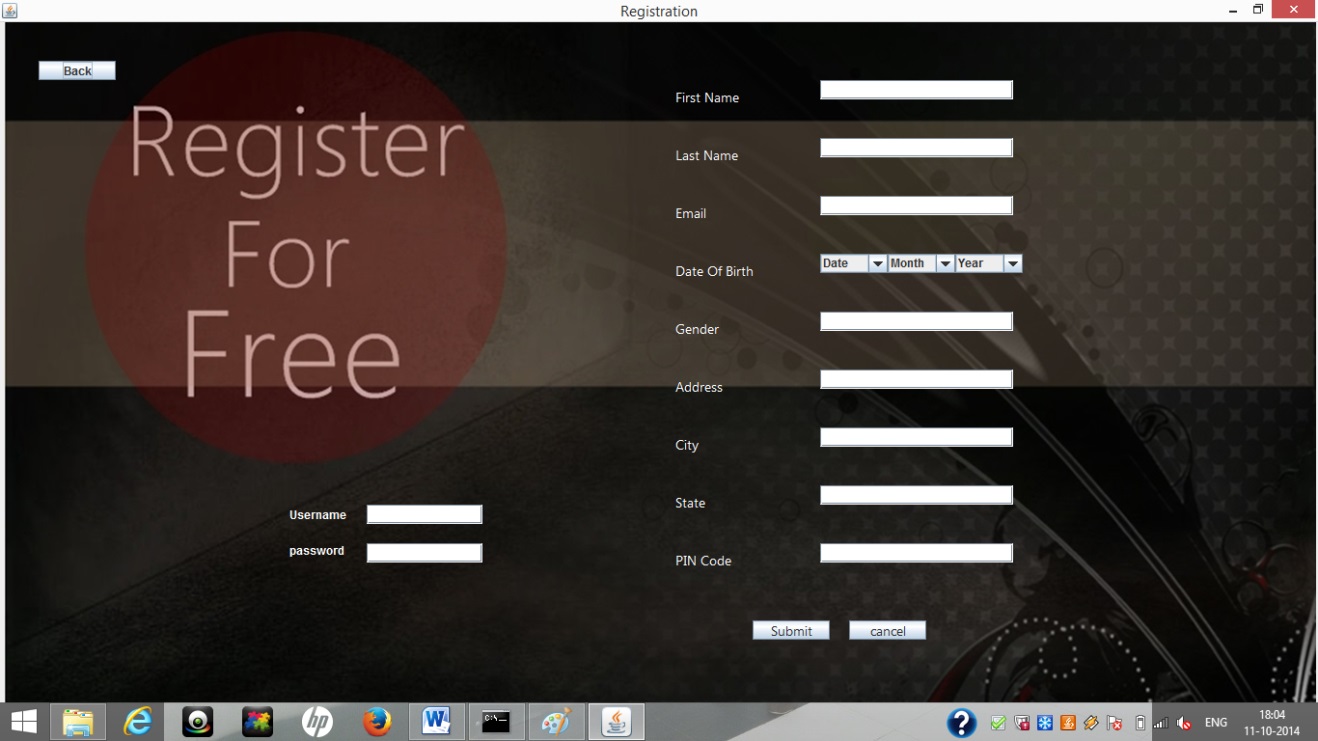
login



Home Page



New Account

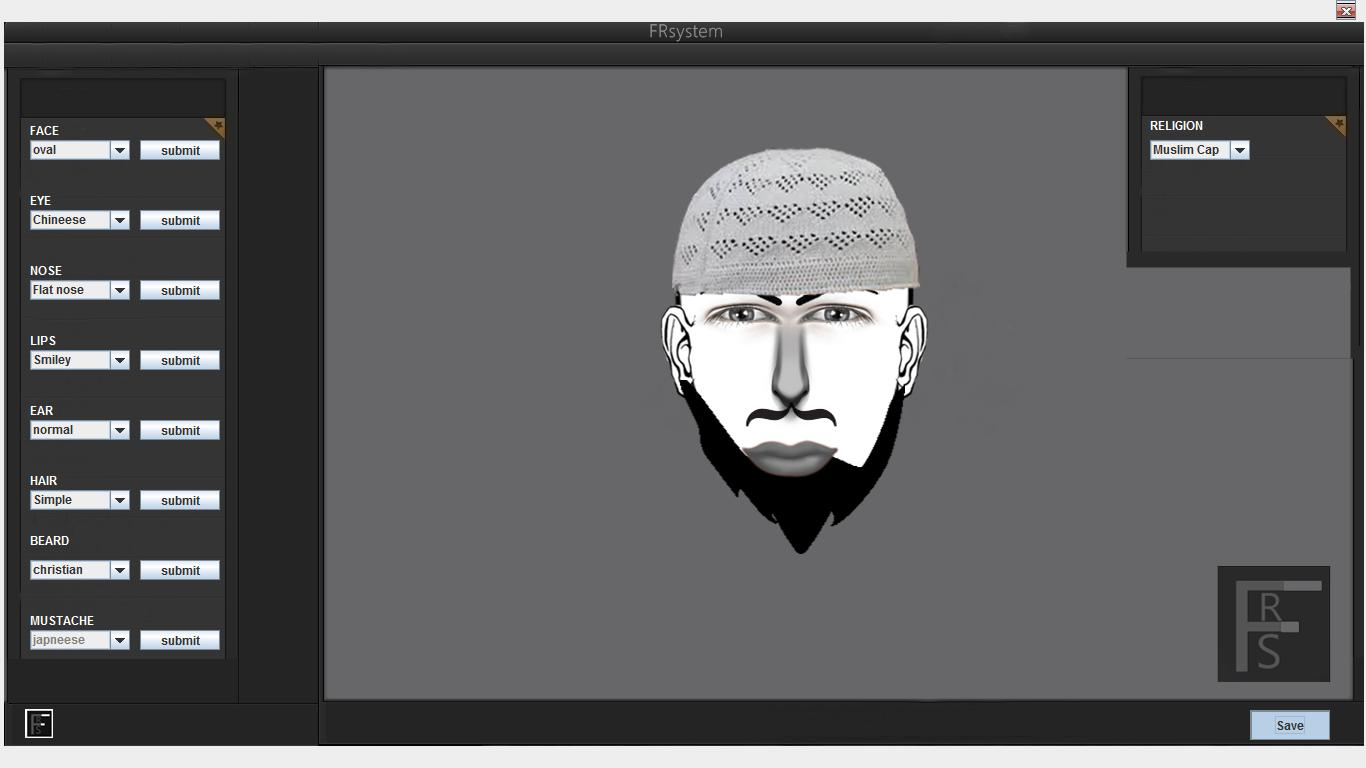


Different faces



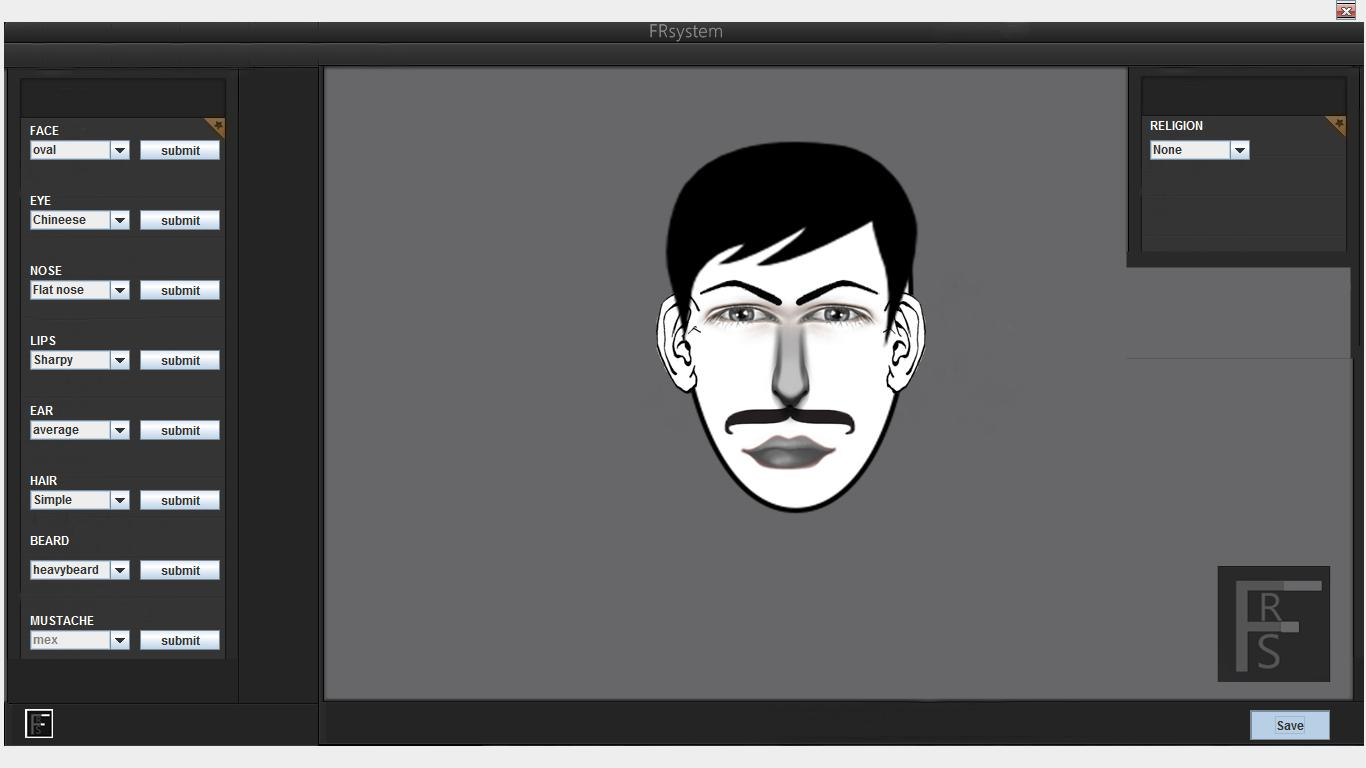












**MAINTANANCE AND SECURITY ASPECTS**

System maintenance is an extremely important activity in the life of a software product that starts after the implementation phase. It is one form of change or rework. For proper maintenance of this system, I am spending a good time on documentation so that I or some other person can easily understand this system later when there is a need for maintenance.

Because of the full development being Object Oriented and MS support with this product, this system can be easily connected to any other MS product with a very small effort. Except Web support, more modules can also be easily added because of the 3 tier approach in design; any development or modification is as simple as creating a new application. We don’t have to follow the old patterns; we can just start with a new style and add the new modules in to the older one without any problem.

**TYPES OF MAINTAINENCE**

Software system maintenance covers a wide range of activities, including correcting coding and design errors, updating documentation and test data and upgrading user support. It is the process of changing the system to maintain its ability to survive. The system design can adopt any of the under given system maintenance strategies.

**Corrective Maintenance**

The design software can be maintained by corrective maintenance that is concerned with fixing reported errors in the software. Coding errors are relatively cheap to fix as compared to design errors and requirement errors. The requirement errors are more expensive, as redesigning of the entire system is to fix them.

**Adaptive Maintenance**

The design software can be maintained by adaptive maintainence if case there is a need to change the system environment such as a different hardware platform or for using it with different operating system.

**Predictive Maintenance**

The design system can be maintained by predictive maintainence if there is need to implement new functional requirements. They are generated due to the changing requirements of the software customers as their organization or business changes.

**Maintenance of the system follows the following steps**

**System Documentation**

System documentation is a process that emphasizes on documentation the requirement, design, plan, code implementation and testing of the system as whole.

In a product life cycle, there will be a number of documents like requirements specification design, documents, user manuals, release documents etc. that will be generated. The types of documentation that are covered under system documentation also include training material, user manuals, operations manual, and release document. The documentation will be reviewed and approved by concerned users of the documentation. The purpose of the system documentation process is to develop and maintain the documentation that will be used to install and operate the product. This process also ensures that the documentation developed is synchronized with latest product release.

Our product document typically consists of the following:

**Deliverable (file name size etc.):** It includes the code after making changes in the original code.

**Test Summary Report:** It includes the result generated by executing the test cases designed by us and the test cases given to us by the customer and transporter. List of all the corresponding changes made in the existing code and reason why the changes are necessary.

These documents not only ensures that the system is well designed and is fully complaint with the needs of the user but also helps in further maintenance programmers of the software system. A well documented system helps the maintenance of the software by providing the maintenance programmers and insight.

Security Aspects

System security is a vital aspect when it comes to developing a system. The system should ensure the facility of preventing unauthorized personnel from accessing the information and the data within the system. The system should provide total protection for each user’s information so that the integrity of the data is sustained.

The proposed system ensures the security and the integrity of data. This is done by providing a password login system for each authorized user. At the starting of the application, the user is asked for username and password and only the correct information allows further navigation in the project. However, the System Administrator has access to all kinds of information.

**Access Rights**

Access rights is who can read / modify the data in various tables / files. In this system, the following people have the following access rights:

**Operator of the site:** He has access to all kinds of information, including those of the clients / users and the providers, except the information related to the employees of different providers.

**Client / User:** He can read and modify his own data but he has no access to the information related to the administrator, providers and their employees and other clients.

**CONCLUSION**

The system has been developed for the given condition and is found working effectively. The developed system is flexible and changes whenever can be made easy. Using the facilities and functionalities of Java, the software has been developed in a neat and simple manner, thereby reducing the operators work.

The speed and accuracy are maintained in proper way. The user friendly nature of this software developed in Java framework is very easy to work with both for the higher management as well as other employees with little knowledge of computer. The results obtained were fully satisfactory from the user point of view.

The system was verified with valid as well as invalid data in each manner. The system is run with an insight into the necessary modifications that may require in the future. Hence the system can be maintained successfully without much network.

**FUTURE PLANS:**

In future we will do some major advances in the project and it may be extended for the major one. This may include many features like replying can be user specific and we can see the replies according to selecting the user. We may make to look it attractive by adding some flash animation to the pages.

Going further to access this project on Wifi we can buy the desired web space and after that it can be made to run on the Wifi and can be accessed from any region. Thus it will be available on net every moment.

Link in the documents are more important than simple text. Number and types of links and other web structures can be given higher weights to improve the overall classification of the document.

Web structures can also be given importance for classification. The higher font size labels can be given more importance while classifying a document than other text. These can be done by giving the different weights to keywords depending upon the font size.

In this project the features like remote login can also be made possible. The features like online chatting on the blogs and giving the comments can also be made possible.

**BIBLIOGRAPHY**

1) [www.google.com](http://www.google.com)

2) [www.wikipedia.org](http://www.wikipedia.org)

3) Jewei Han and Micheline Kamber(2005) “Elementary implementation of Tomcat”.

4) The complete reference for Java by Helbert Schildt

5) Google News [www.googlenews.com](http://www.googlenews.com)

6) Microsoft Network [www.msn.com](http://www.msn.com)