**INTRODUCTION**

The “**E-Transaction Interface on Multiple Banking System”** is the designed targeted at the future banking solution for the users who is having multiple bank accounts at the multiple banks. This interface integrates all existing banks and provides business solutions for both retail and corporate.

This system acts as a standard interface between the clients and all the banks that register with the system and clients who maintains accounts in various banks don’t have to visit individual bank’s website to make money transactions instead he can directly log on to “E-Transaction Interface on Multiple Banking System” and make any kind of request and get his work fulfilled and in the backend the system will take care of all the obligation required in order to carry on transaction smoothly.

# Existing System:

Let’s consider a condition when a bank customer is having bank accounts in more than one bank. The online banking system available at present is bank specific. Each bank is having its own interface to interact with the bank. A customer can login to the bank and make the transactions using the online banking provided by the bank. The way he interacts with different banks varies. The user must learn how to interact with each system.

**Objective**

The main Vision of this project is to eliminate all the diversities amongst banks, which generally client faces at the time of any transaction. By doing so Client will use to only one Systematic Standard way of banking and there by they will be at ease using this interface.

The kind of functionality it’s capable of providing also reveals the kind of banking facilities that a customer could get online. Of course, the bank that implements this solution decides the features available to customers.

**Modules**

1) Banker Module

This module deals with the Accounts Pending, Pending Transfers, and Reports regarding transactions.

2) Customer Module

In this module, a customer can register him self to the system, customer can login into the system. A customer can add a new account, view the account information, Transfer amount, A customer can also see the Transaction Reports.

3) Admin Module

The admin module will be used by the administrator of the site, The admin can accept or reject the request from the banker as well as the request from the user. The requests are in the form of bank registration, customer registration.

**DRAWBACKS OF EXISTING SYSTEM**

A user requires accessing the system on the fly. The user interfaces designed by the different banks will confuse the user. He requires learning how to use each and every user interface of the bank in which he is having accounts. This process may be time consuming and too irritating for the user also. When he transfers the accounts, He may probably prone to click the different action when shifting from one bank user interface to other.

**Proposed System and its features:**

The e-Transaction Interface provides the following system features.

1. This system provides a Common User Interface for the customers to log on to any bank.
2. Here the user interface is Graphical User Interface.
3. This application is a Web based Application.
4. Being a web based application it doesn’t require any client side installation.
5. Any number of users can interact with the system simultaneously.
6. Eradicates the time consumed to learn how to use all the user interfaces of every bank in which a customer is having account.
7. The transactions are secure.

**Theoretical Background**

**Yash Infosoft Pvt Ltd** offers a comprehensive range of software and IT solutions, including packaged applications for the School, College, Hospital, Logistic and Pharmacy, Manufacturing, and Retail & Distribution industries. In addition, it offers a broad range of software services such as Custom Software Development, Web designing and development, among others

The Company’s Delivery Model provides for the best resources to be drawn from its vast talent pool all over India offer optimal solutions.  
  
Yash Infosoft integrates its products and services to create customized solutions to allow you to undertake technology-based business transformation that allows reorganization in line with today's dynamic digital business environment.

* Our Services:
* Enterprise Resource Planning
* School Management System
* College Management System
* Hospital Management System
* Logistic Administration Software
* Library Administration Software
* Web Based Solution
* Search Engine Optimization

**PROJECT CATEGORY**

**Server Client**

#### Over view:

With the varied topic in existence in the fields of computers, Client Server is one, which has generated more heat than light, and also more hype than reality. This technology has acquired a certain critical mass attention with its dedication conferences and magazines. Major computer vendors such as IBM and DEC; have declared that Client Servers is their main future market. A survey of DBMS magazine reveled that 76% of its readers were actively looking at the client server solution. The growth in the client server development tools from $200 million in 1992 to more than $1.2 billion in 1996.

Client server implementations are complex but the underlying concept is simple and powerful. A client is an application running with local resources but able to request the database and relate the services from separate remote server. The software mediating this client server interaction is often referred to as MIDDLEWARE.

The typical client either a PC or a Work Station connected through a network to a more powerful PC, Workstation, Midrange or Main Frames server usually capable of handling request from more than one client. However, with some configuration server may also act as client. A server may need to access other server in order to process the original client request.

The key client server idea is that client as user is essentially insulated from the physical location and formats of the data needs for their application. With the proper middleware, a client input from or report can transparently access and manipulate both local database on the client machine and remote databases on one or more servers. An added bonus is the client server opens the door to multi-vendor database access indulging heterogeneous table joins.

#### What is a Client Server

Two prominent systems in existence are client server and file server systems. It is essential to distinguish between client servers and file server systems. Both provide shared network access to data but the comparison dens there! The file server simply provides a remote disk drive that can be accessed by LAN applications on a file-by-file basis. The client server offers full relational database services such as SQL-Access, Record modifying, Insert, Delete with full relational integrity backup/ restore performance for high volume of transactions, etc. the client server middleware provides a flexible interface between client and server, who does what, when and to whom.

#### Why Client Server?

Client server has evolved to solve a problem that has been around since the earliest days of computing: how best to distribute your computing, data generation and data storage resources in order to obtain efficient, cost effective departmental an enterprise wide data processing. During mainframe era choices were quite limited. A central machine housed both the CPU and DATA (cards, tapes, drums and later disks). Access to these resources was initially confined to batched runs that produced departmental reports at the appropriate intervals. A strong central information service department ruled the corporation. The role of the rest of the corporation limited to requesting new or more frequent reports and to provide hand written forms from which the central data banks were created and updated. The earliest client server solutions therefore could best be characterized as “SLAVE-MASTER”.

#### Time-sharing changed the picture. Remote terminal could view and even change the central data, subject to access permissions. And, as the central data banks evolved in to sophisticated relational database with non-programmer query languages, online users could formulate adhoc queries and produce local reports with out adding to the MIS applications software backlog. However remote access was through dumb terminals, and the client server remained subordinate to the Slave/Master.

#### Front end or User Interface Design

The entire user interface is planned to be developed in browser specific environment with a touch of Intranet-Based Architecture for achieving the Distributed Concept.

The browser specific components are designed by using the HTML standards, and the dynamism of the designed by concentrating on the constructs of the Java Server Pages.

#### Communication or Database Connectivity Tier

The Communication architecture is designed by concentrating on the Standards of Servlets and Enterprise Java Beans. The database connectivity is established by using the Java Data Base Connectivity.

The standards of three-tire architecture are given major concentration to keep the standards of higher cohesion and limited coupling for effectiveness of the operations.

#### Features of The Language Used

In my project, I have chosen Java language for developing the code.

#### About Java

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

* Java is a programmer’s language.
* Java is cohesive and consistent.
* Except for those constraints imposed by the Internet environment, Java gives the programmer, full control.
* Finally, Java is to Internet programming where C was to system programming.

#### Importance of Java to the Internet

Java has had a profound effect on the Internet. This is because; Java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the Server and the Personal computer. They are: Passive information and Dynamic active programs. The Dynamic, Self-executing programs cause serious problems in the areas of Security and probability. But, Java addresses those concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

#### Java can be used to create two types of programs

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

#### Features Of Java Security

Every time you see that when you download a “normal” programs, you are risking a viral infection. Prior to Java, most users did not download executable programs frequently, and those who did scanned them for viruses prior to execution. Most users still worried about the possibility of infecting their systems with a virus. In addition, another type of malicious program exists that must be guarded against. This type of program can gather private information, such as credit card numbers, bank account balances, and passwords. Java answers both these concerns by providing a “firewall” between a network application and your computer.

When you use a Java-compatible Web browser, you can safely download Java applets without fear of virus infection or malicious intent.

#### Portability

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

#### The Byte code

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

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

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

#### Java Virtual Machine (JVM)

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

**Overall Description**

# Java Source

## Java Compiler

# JavaVM

Java

.Class

Picture showing the development process of JAVA Program

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

#### Java Architecture

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

#### Compilation of code

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

**Compiling and interpreting Java Source Code**

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

**Source**

**Code**

**………..**

**………..**

**………..**

**…………**

# PC Compiler

**Macintosh**

**Compiler**

**SPARC**

###### Compiler

**Java**

**Byte code**

**(Platform**

**Indepen**

**dent)**

**Java**

**Interpreter**

**(PC)**

**Java**

**Interpreter**

**(Macintosh)**

**Java**

**Interpreter**

**(Spare)**

Simple

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

**Object-Oriented**

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

Robust

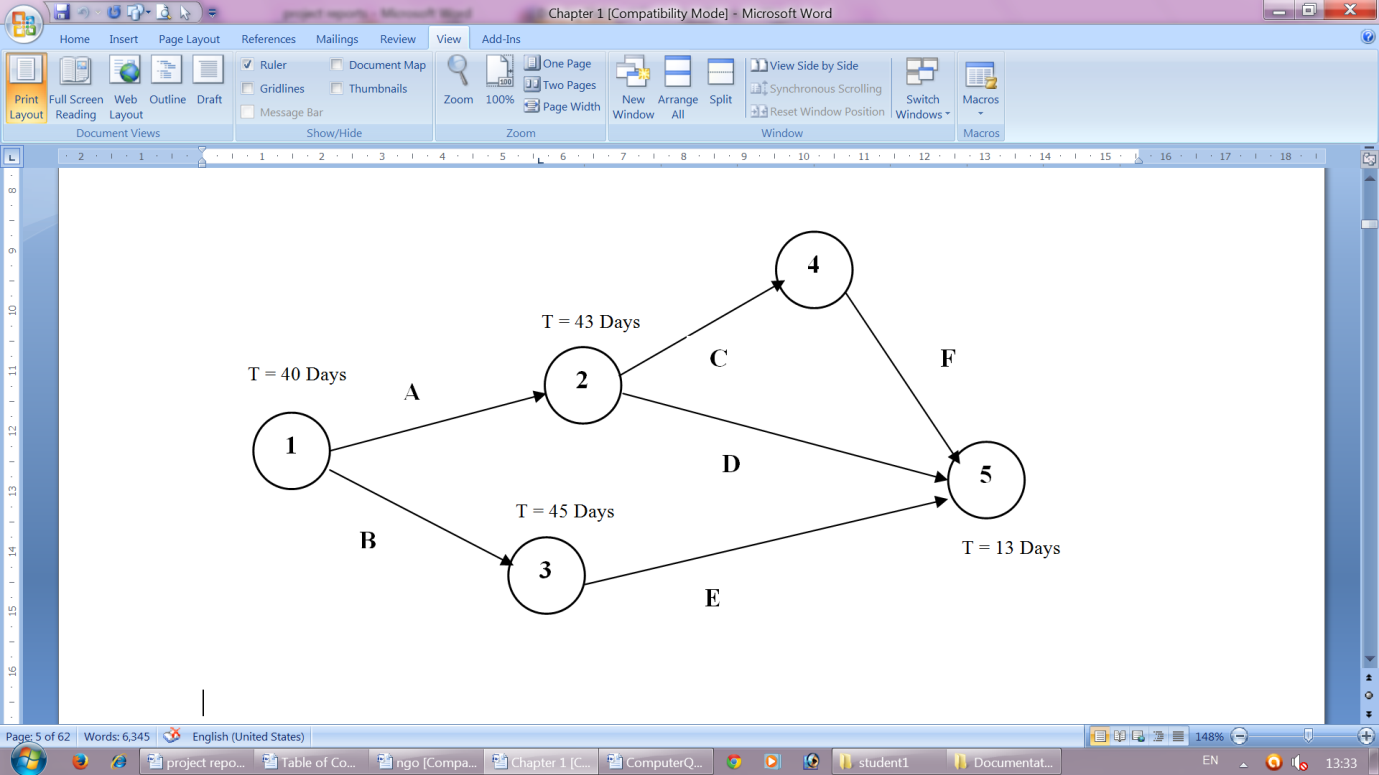
The multi-platform environment of the Web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. The ability to create robust programs was given a high priority in the design of Java. Java is strictly typed language; it checks your code at compile time and run time.

Java virtually eliminates the problems of memory management and deallocation, which is completely automatic. In a well-written Java program, all run time errors can –and should –be managed by your program.

**System Planning (PERT Chart, Gantt Chart)**

**PERT Chart:**

The Program Evaluation and Review Technique is a network model that allows for randomness in activity completion time.



PERT Chart Legend:

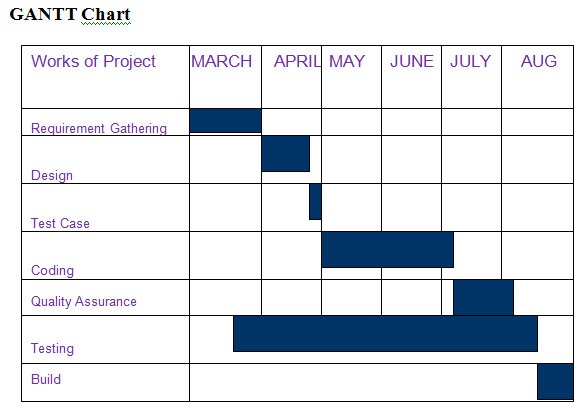
1) System Analysis Task

2) System Design Task

3) Coding Task

4) Testing Task

5) Implementation Task



**PROJECT PLANNING**

**Spiral Model**

This model can be considered as the model, which combines the strengths of various other models. Conventional software development processes do not take uncertainties into account. Important software projects have failed because of unforeseen risks.

The following are the primary activities in this model:

* Finalizing objectives: The objectives are set for the particular phase of the project.
* Risk Analysis: The risks are identified to the extent possible. They are analyzed and necessary steps are taken.
* Development: Based on the risks that are identified, an SDLC model is selected and is followed.
* Planning: At this point, the work done till this time is renewed.

**Features of Spiral Model:**

* + - Risk assessment is involved at every activity.
    - Most realistic approach to development for large systems.

In the spiral model, these phases are followed iteratively. Following figure depicts the Boehm’s Spiral Model:

Determines objectives alternatives, constraints

Evaluates alternatives, identify, resolve risks

Plan next phase

Develop next level p objectives alternatives, constraints

review

Risk Analysis

Risk Analysis

Risk Analysis

Requirements plan Life Cycle plan

Operation concept

Integration and test plan

Development plan

Prototype2

Final Prototype

**SYSTEM ANALYSIS**

**Performance Requirements:**

Performance is measured in terms of ease of use of user interface.

The document is prepared keeping is view of the academic constructs of my Masters Degree from university as partial fulfillment of my academic purpose the document specifies the general procedure that that has been followed by me, while the system was studied and developed. The general document was provided by the industry as a reference guide to understand my responsibilities in developing the system, with respect to the requirements that have been pin pointed to get the exact structure of the system as stated by the actual client.

The system as stated by my project leader the actual standards of the specification were desired by conducting a series of interviews and questionnaires. The collected information was organized to form the specification document and then was modeled to suite the standards of the system as intended.

**Document Conventions:**

The overall documents for this project use the recognized modeling standards at the software industries level.

Unified modeling language concepts to give a generalized blue print for the overall system.

The standards of flow charts at the required states that are the functionality of the operations need more concentration.

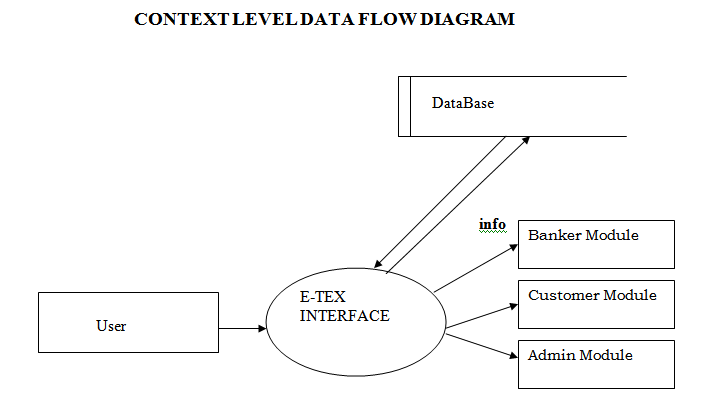
Scope of the Development Project:

Database Tier: The concentration is applied by adopting wide range of DBMS.

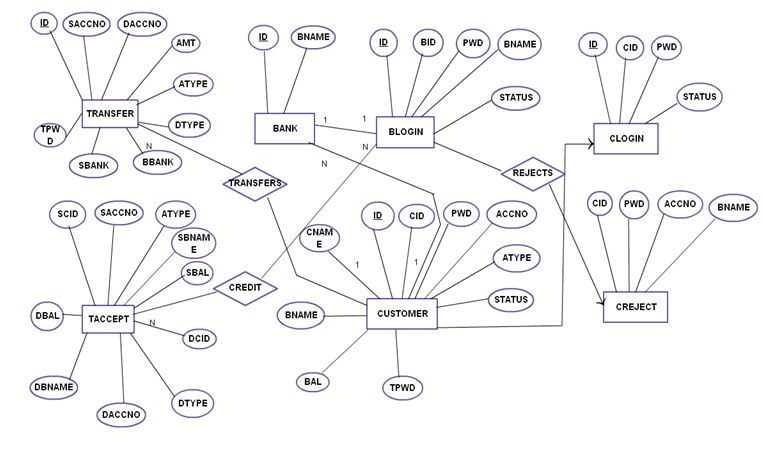
User Tier: The user interface is developed in a browser specific environment to have web based architecture. The components are designed using HTML standards and Java server pages power the dynamic of the page design. Java Script for Loading graphics and Validations.

Data Base Connectivity Tier: The communication architecture is designed by concentrated on the standards of Servlets and JSP’s. The database connectivity is established using the JDBC.

**DATA FLOW DIAGRAM :**



**E-R Diagram:**

****

**UNIFIED MODELING LANGUAGE DIAGRAMS**

* + The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.
  + A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.
  + User Model View
    1. This view represents the system from the users perspective.
    2. The analysis representation describes a usage scenario from the end-users perspective.

**Structural model view**

◆ In this model the data and functionality are arrived from inside the system.

◆ This model view models the static structures.

**Behavioral Model View**

◆ It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

**Implementation Model View**

* + In this the structural and behavioral as parts of the system are represented as they are to be built.

**Environmental Model View**

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

UML is specifically constructed through two different domains they are

* UML Analysis modeling, which focuses on the user model and structural model views of the system.
* UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

Use case Diagrams represent the functionality of the system from a user’s point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view.

Actors are external entities that interact with the system. Examples of actors include users like administrator, bank customer …etc., or another system like central database.

**Use case Model.**

SYSTEM NAME

Use case 1

Use case 2

Use case n

Actor

Actor

# Use Cases of E-Transaction Interface on Multiple Banking System

E-Transaction Interface

LOGIN

BANK REGISTRATION

CUSTOMER REGISTRATION

AMOUNT TRANSCATION

ACCOUNT CREATON

VIEW TRANSCATION STATUS

TRANSACTION VERIFICATION

ACCOUNT VERIFICATION

REPORTS GENEATION

Banker

Admin

Customer

|  |  |
| --- | --- |
| *Use case name* | Login |
| *Participating actors* | Customer, Admin, Banker |
| *Flow of events* | The Actor will give the user name and password to the system. The system will verify the authentication. |
| *Entry Condition* | The actor will enter the system by using username and password |
| *Exit condition* | If un authenticated should be exited |
| *Quality Requirements* | Password must satisfy the complexity requirements. |

|  |  |
| --- | --- |
| *Use case name* | Bank Registration |
| *Participating actors* | Admin |
| *Flow of events* | The administrator will all the details submitted by banker and creates and place in the application. |
| *Entry Condition* | Must satisfy all the norms given by the e-transaction interface site. |
| *Exit condition* | Successful or Un successful completion of creation of account. |
| *Quality Requirements* | All fields are mandatory. |

|  |  |
| --- | --- |
| *Use case name* | Customer Registration |
| *Participating actors* | Customer |
| *Flow of events* | The customer must enter all his personal details. |
| *Entry Condition* | View Home page |
| *Exit condition* | Registered customer should be successfully logged out. Error Message should be displayed on Un successful creation. |
| *Quality Requirements* | Best Error Handling techniques. Check on Mandatory fields. |

|  |  |
| --- | --- |
| *Use case name* | Amount Transaction |
| *Participating actors* | Customer |
| *Flow of events* | The customer selects the target account and must give the amount to be transferred across the accounts. |
| *Entry Condition* | User should have an account and the destination account should be valid one. |
| *Exit condition* | Transaction must be rolled back if exceptions are raised. |
| *Quality Requirements* | Target account must be verified, amount entered must be valid. Cross checking of account balance. |

|  |  |
| --- | --- |
| *Use case name* | Amount Transaction |
| *Participating actors* | Customer |
| *Flow of events* | The customer selects the target account and must give the amount to be transferred across the accounts. |
| *Entry Condition* | User should have an account and the destination account should be valid one. |
| *Exit condition* | Transaction must be rolled back if exceptions are raised. |
| *Quality Requirements* | Target account must be verified, amount entered must be valid. Cross checking of account balance. |

|  |  |
| --- | --- |
| *Use case name* | Account creation |
| *Participating actors* | Customer |
| *Flow of events* | The customer must create the bank account by giving the necessary data. |
| *Entry Condition* | Actor must know the proper bank account number and the bank. |
| *Exit condition* | Not Applicable |
| *Quality Requirements* | Not Applicable |

|  |  |
| --- | --- |
| *Use case name* | View Transaction Status |
| *Participating actors* | Customer |
| *Flow of events* | The customer can view the whether the transaction is approved or rejected.. |
| *Entry Condition* | Not Applicable. |
| *Exit condition* | Not Applicable |
| *Quality Requirements* | Good Report Design |

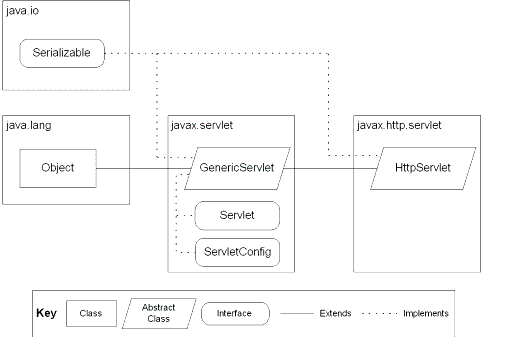
|  |  |
| --- | --- |
| *Use case name* | Transaction Verification |
| *Participating actors* | Banker |
| *Flow of events* | The Banker must verify the transaction before it was going to be committed. |
| *Entry Condition* | Banker must check the validity constraints |
| *Exit condition* | Banker must complete the task before exiting the system |
| *Quality Requirements* | Not Applicable |

|  |  |
| --- | --- |
| *Use case name* | Account Verification |
| *Participating actors* | Banker |
| *Flow of events* | The Banker can accept or reject the account that can be added to the customer. |
| *Entry Condition* | He must verify the account that is that account belongs to specified person or not. |
| *Exit condition* | Not Applicable |
| *Quality Requirements* | No Applicable |

|  |  |
| --- | --- |
| *Use case name* | Reports Generation |
| *Participating actors* | Banker |
| *Flow of events* | The Banker can view the reports regarding the accounts and transactions in specified time. |
| *Entry Condition* | Duration of the Reports |
| *Exit condition* | Not Applicable |
| *Quality Requirements* | Good Report Design |

**CLASS DIAGRAMS**

Class diagrams describe the structure of the system in terms of classes and objects. The servlet api class diagram will be as follows.



JSP: Implicit Objects

**SEQUENCE DIAGRAMS**

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically.

**Sequence Diagram 1**

Time

Admin

: Login

: Create Bank Account

: Log Out

Use url

Press login button

Press Create Account

**Sequence Diagram 2**

Press login button()

DB

: Accounts

: Login

Customer

Get login page ()

Validate norms()

Press create accounts button()

# Sequence Diagram 3

View last transaction ()

Customer

:TX

: TXVerify

: DB

Press Amount submit ()

Wait for acceptance()

**Sequence Diagram 4**

Reports()

Select Validate or Reject Account()

Select Validate or Reject

Transaction ()

:Banker Privel

: Login

: DB

: Tran.Validate

:Account

Validate

Press Accept/Reject Button

Press login button()

Banker

**Feasibility Report**

**1. System Analysis Concentration:**

Before planning a replacing for a new system it is essential to have through knowledge about the existing system along with estimation of how lost computes can be used to make its operations more effective.

System analysis is the process of collecting and interpreting facts, disposing problem and use the information about the existing system, which is also called as system study.

System analysis is about understanding situation but not solving the problem.

System analysis is performed to determine whether a not it is feasible to design and information system laved on the policies and plans of an organization. To determine the user requirements and to eliminate the weakness of the present system a few general requirements are concerned.

**GENERAL REQUIREMENTS:**

The new system should be cost effective

To improve productivity and service and services.

To enhance user interface.

To improve information presentation and durability.

To upgrade systems reliability, availability and flexibility.

To address human factors for better and uses acceptance.

Feasibility Report

**TECHINICAL FEASIBILITY:**

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, .at this point in time, not too many detailed design of the system, making it difficult to access issues like performance, costs on (on account of the kind of technology to be deployed) etc. A number of issues have to be considered while doing a technical Analysis.

1. Understand the different technologies involved in the proposed system:

Before commencing the project, we have to be very clear about what are the technologies that are to be required for the development of the new system.

1. Find out whether the organization currently possesses the required technologies:
   * Is the required technology available with the organization?
   * If so is the capacity sufficient?

For instance –

“Will the current printer be able to handle the new reports and forms required for the new system?”

**OPERATIONAL FEASIBILITY:**

Proposed projects are beneficial only if they can be turned into information systems that will meet the organizations operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to Implementation? Here are questions that will help test the operational feasibility of a project:

* Is there sufficient support for the project from management from users? If the current system is well liked and used to the extent that persons will not be able to see reasons for change, there may be resistance.
* Are the current business methods acceptable to the user? If they are not, Users may welcome a change that will bring about a more operational and useful systems.
* Have the user been involved in the planning and development of the project?
* Early involvement reduces the chances of resistance to the system and in
* General and increases the likelihood of successful project.

Since the proposed system was to help reduce the hardships encountered. In the existing manual system, the new system was considered to be operational feasible.

**ECONOMIC FEASIBILITY:**

Economic feasibility attempts 2 weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system.

A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be a useful point of reference to compare actual costs as the project progresses. There could be various types of intangible benefits on account of automation. These could include increased customer satisfaction, improvement in product quality better decision making timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale.

## SOFTWARE REQUIREMENT SPECIFICATION

**REQUIREMENT SPECIFICATION:**

The application “e-Transaction interface on Multiple Banking System” is designed to serve as the common user interface between the banks.

## INTRODUCTION

Purpose: The main purpose for preparing this document is to give a general insight into the analysis and requirements of the existing system or situation and for determining the operating characteristics of the system.

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

## Developers Responsibilities Overview:

The developer is responsible for:

1. Developing the system, which meets the SRS and solving all the requirements of the system?
2. Demonstrating the system and installing the system at client's location after the acceptance testing is successful.
3. Submitting the required user manual describing the system interfaces to work on it and also the documents of the system.
4. Conducting any user training that might be needed for using the system.
5. Maintaining the system for a period of one year after installation.

Functional Requirements is the specification of the function that the system must support. Where as non functional requirements are the constraint on the operation of the system that is not related directly to function of system.

## Functional Requirements:

Inputs: The major inputs for the “E-transaction Interface on Multiple Banking System” are the user name and passwords of the customer, Administrator and the Banker, The account number and the bank name associated with it, Transaction Passwords, Transaction Amount.

Output: The outputs are the Amount that is transferred between the accounts. The verification reports of the bank. Bank Statements, Account status.

**Software Engineering Paradigm applied**

Software engineering paradigm provides or the facilities to the software engineer to used the different type of model for the development of the software or package. When we develop the software according to the software engineering paradigm [Model] then we have not facing the problem in the development of the any type of the software or package software engineering paradigm is using different type of the mold.

Detail of the some important model is given below: -

* + Data Model

**Software Requirement Specification**

### Required Hardware

### Pentium IV Processor.

### 256 MB RAM.

### 40GB Hard Disk space.

### Ethernet card with an Internet and Internet zone.

**Required Software**

Server Side:

### Microsoft Windows 2000 Prof

### Internet explorer 5.0 or above or Netscape navigator.

### Oracle 9i or later versions.

### JDK 1.4 or Later versions.

### Apache Tomcat Web Server 4.0 or above.

### TCP/IP Protocol suite.

### Client Side:

### Any Web Browser on any operating system.

**System Design**

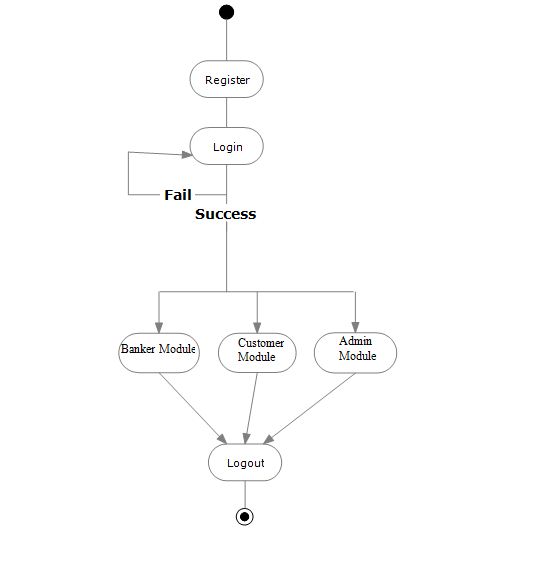
**Designing :**

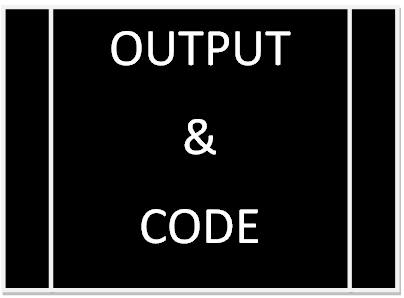
* Mapping system requirements to an abstract representation of a specific system implementation
* Draw to meet the cost and performance constraints
  + Program Flow Chart

DFD

* + E-R Diagram
  + Class Diagram
  + Use Case Diagram
  + Sequence Diagram
* In this design phase we design the system making use of study phase and the data flow diagrams.
* We make use the general access methods for designing.
* We consider the top down approach.
* In the design phase we determine the entitle and their attributes and the relationships between the entities.
* We do both logical and the physical design of the system.

**Program Flow Chart:**

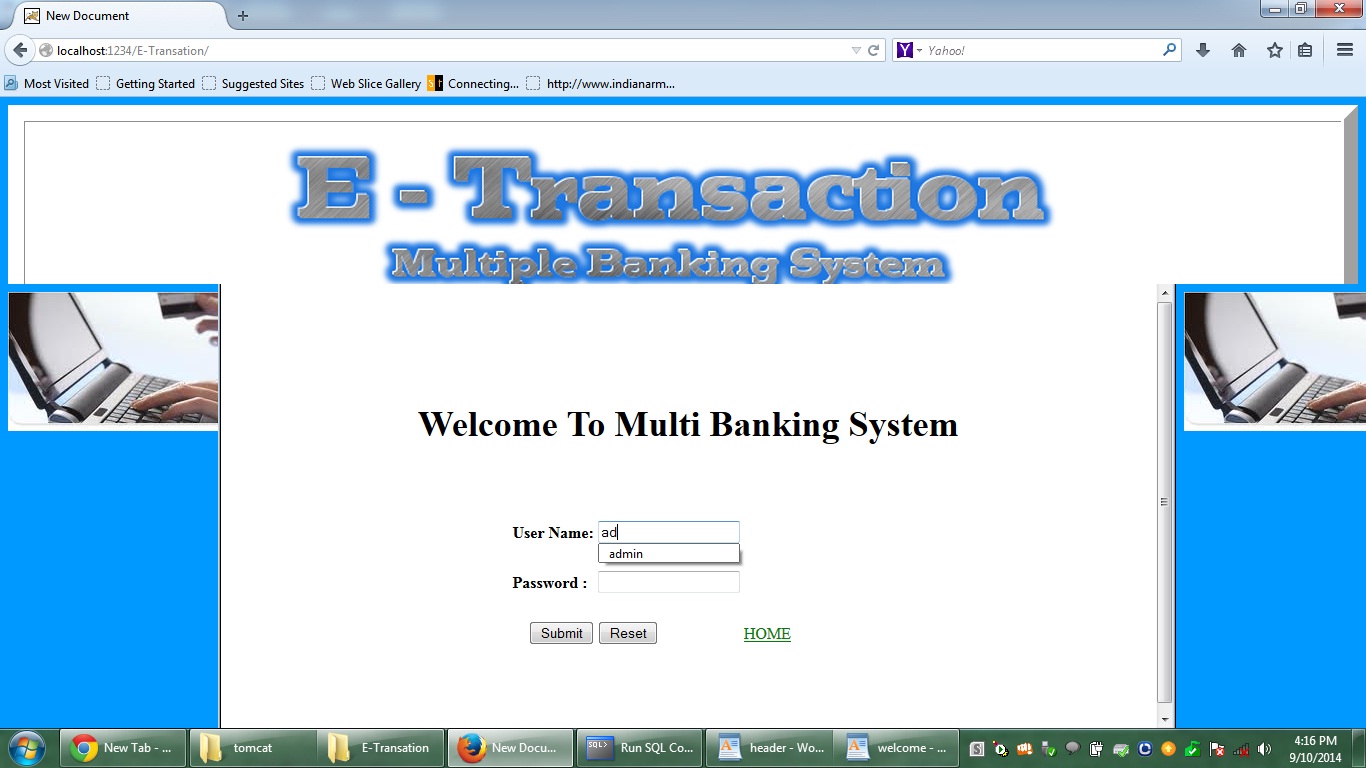
****



**Home Page**

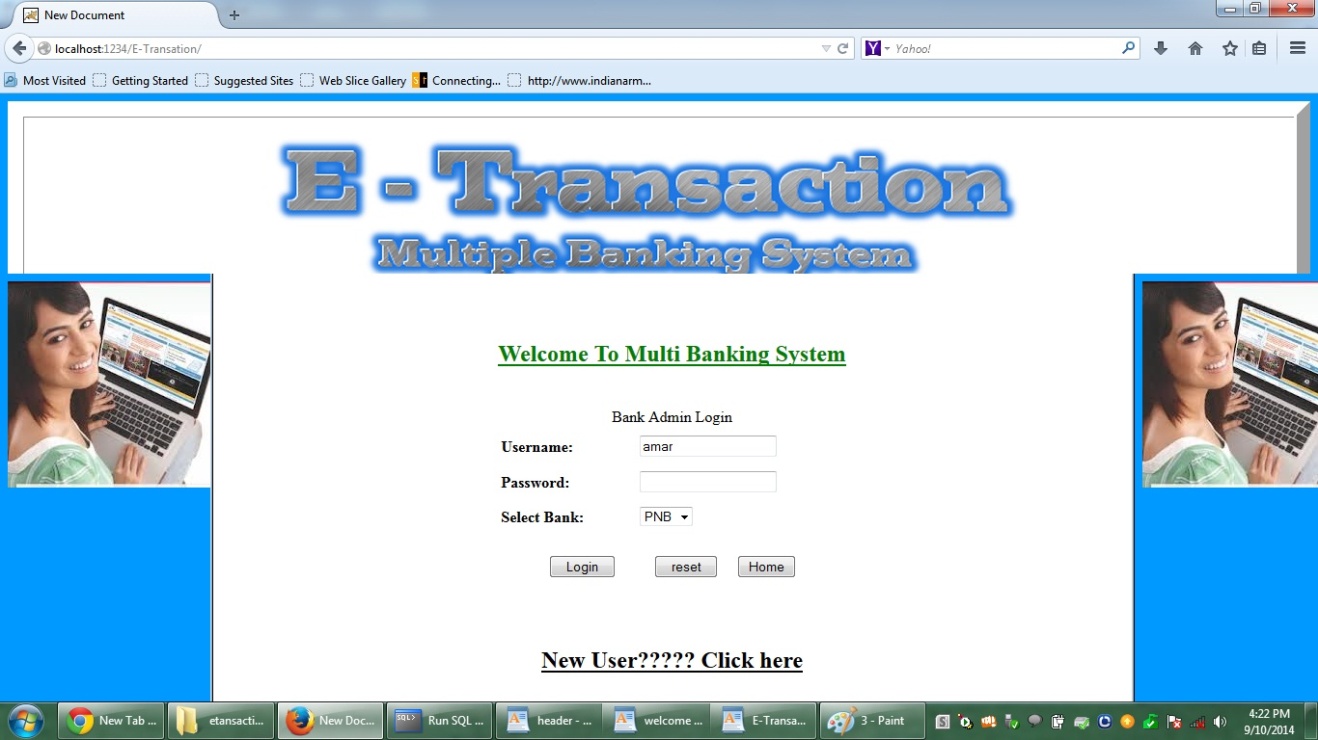
****

**Admin Login**

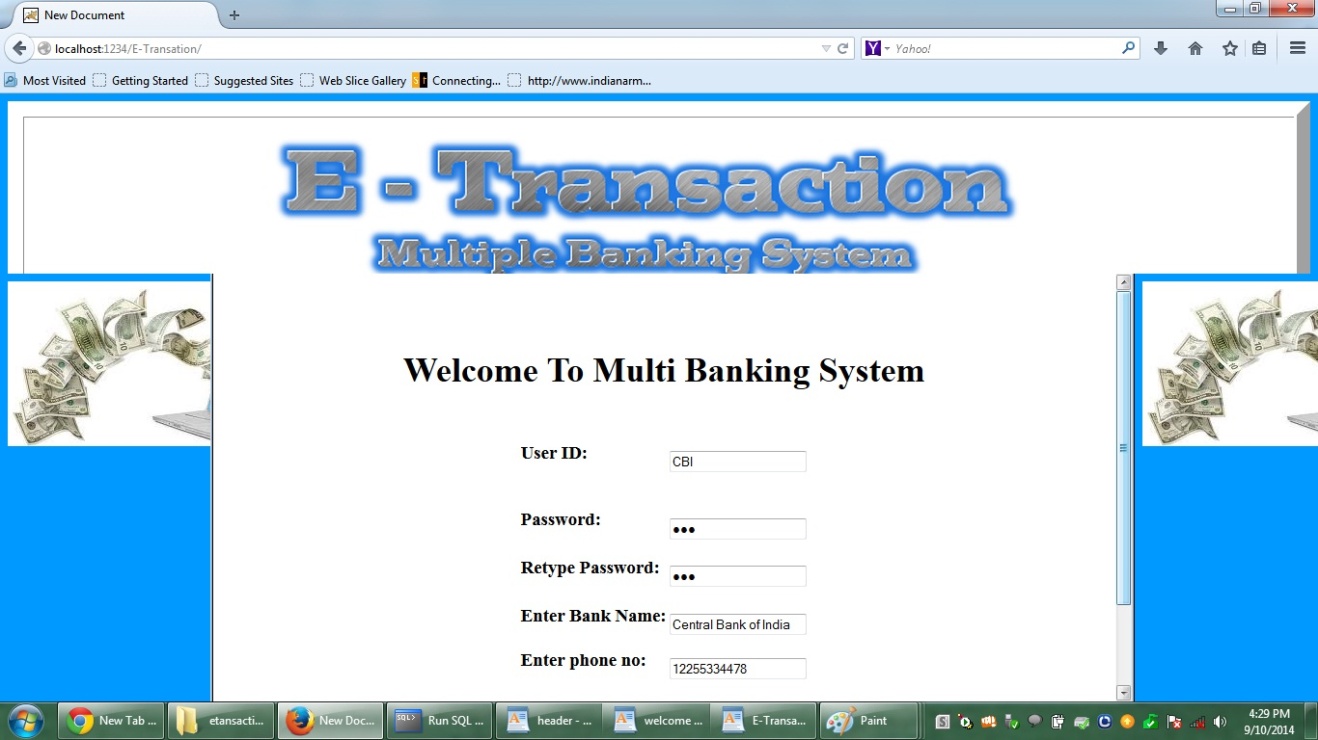
****

**After login**

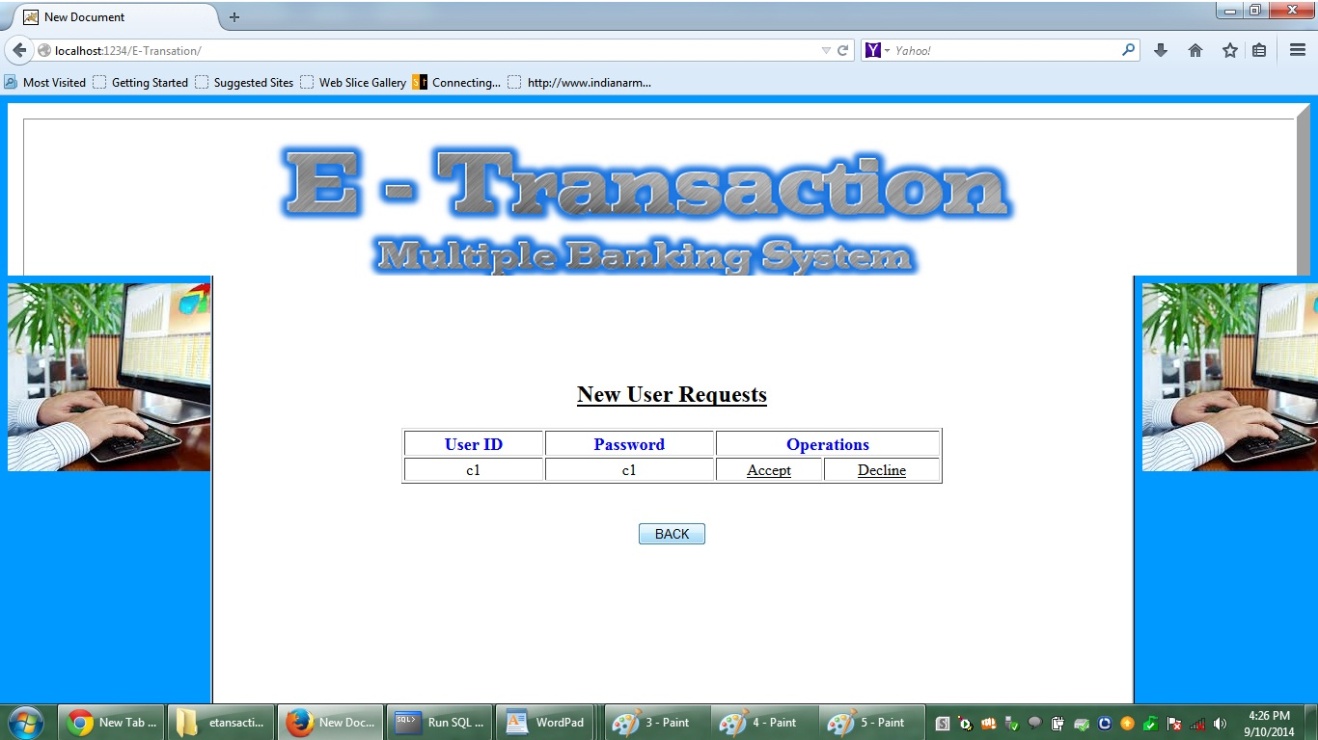
****

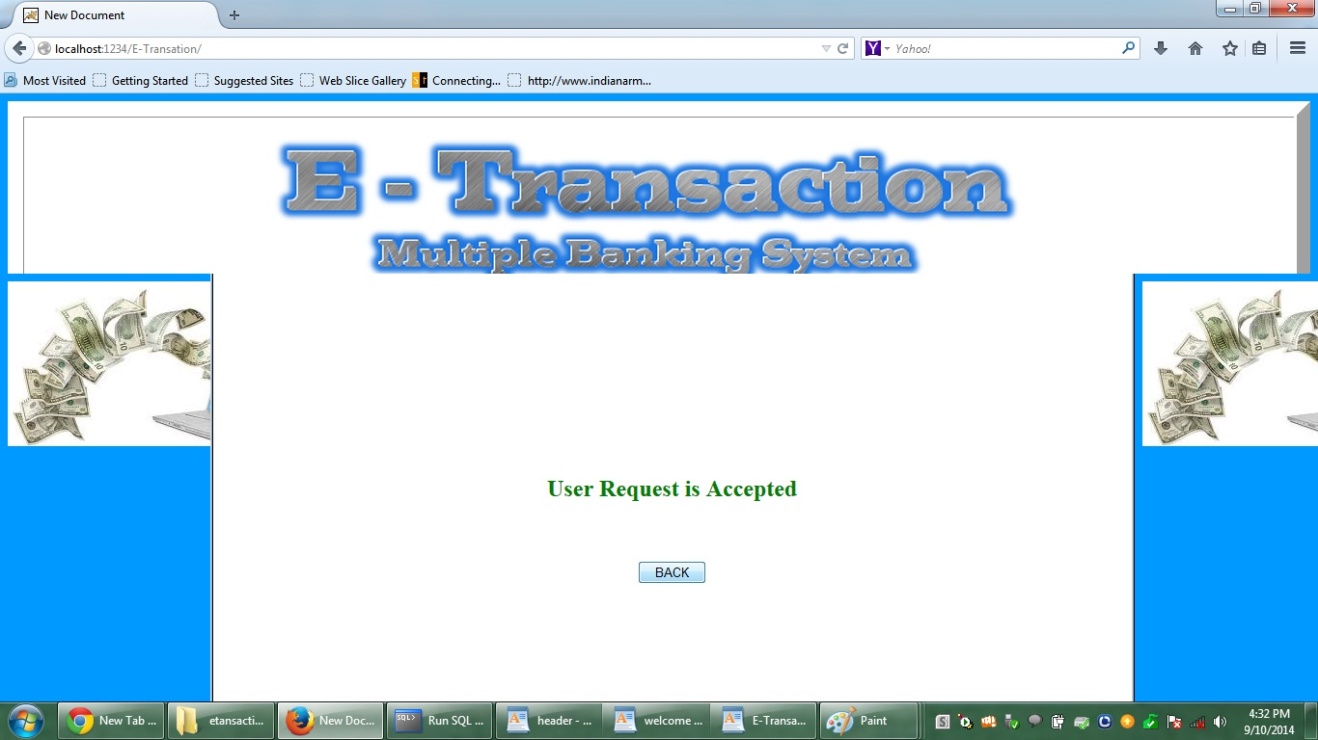
**Bank Login **

**Register Bank**

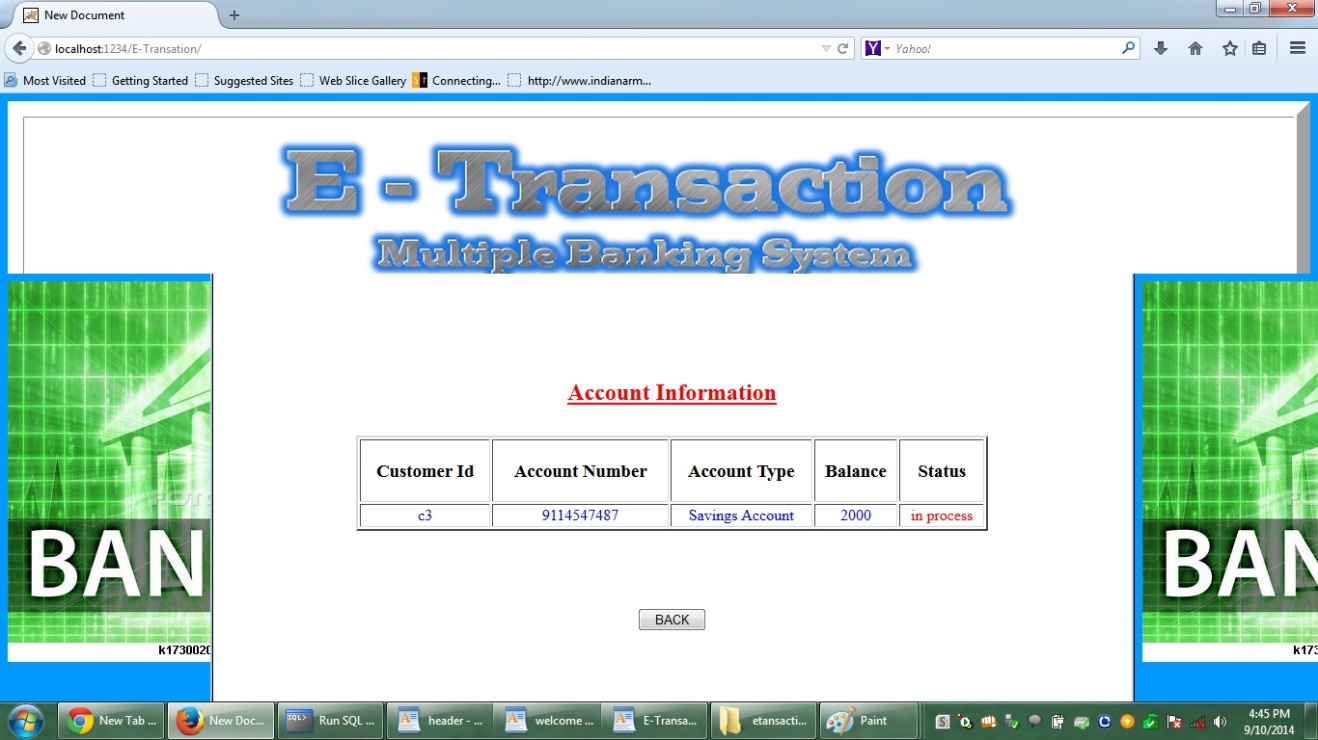


**User Request**

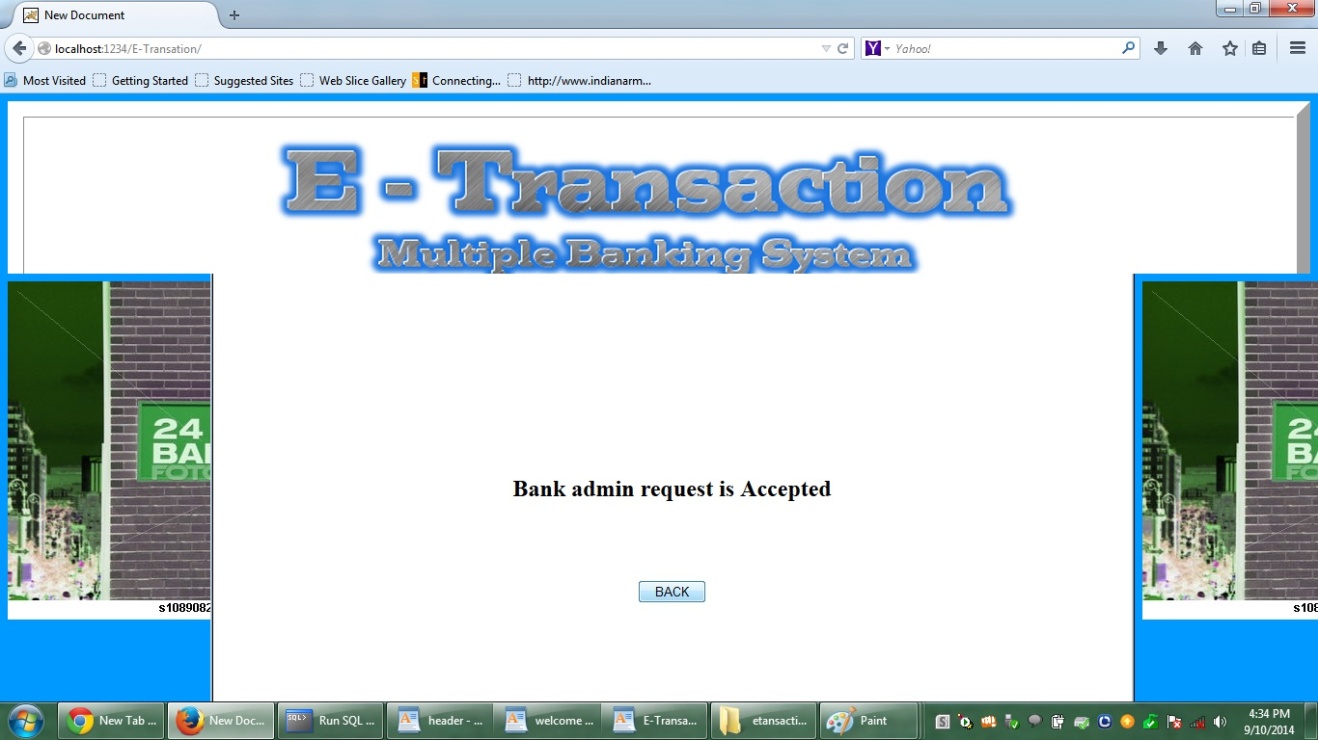
****

**Successfully accepted by admin**

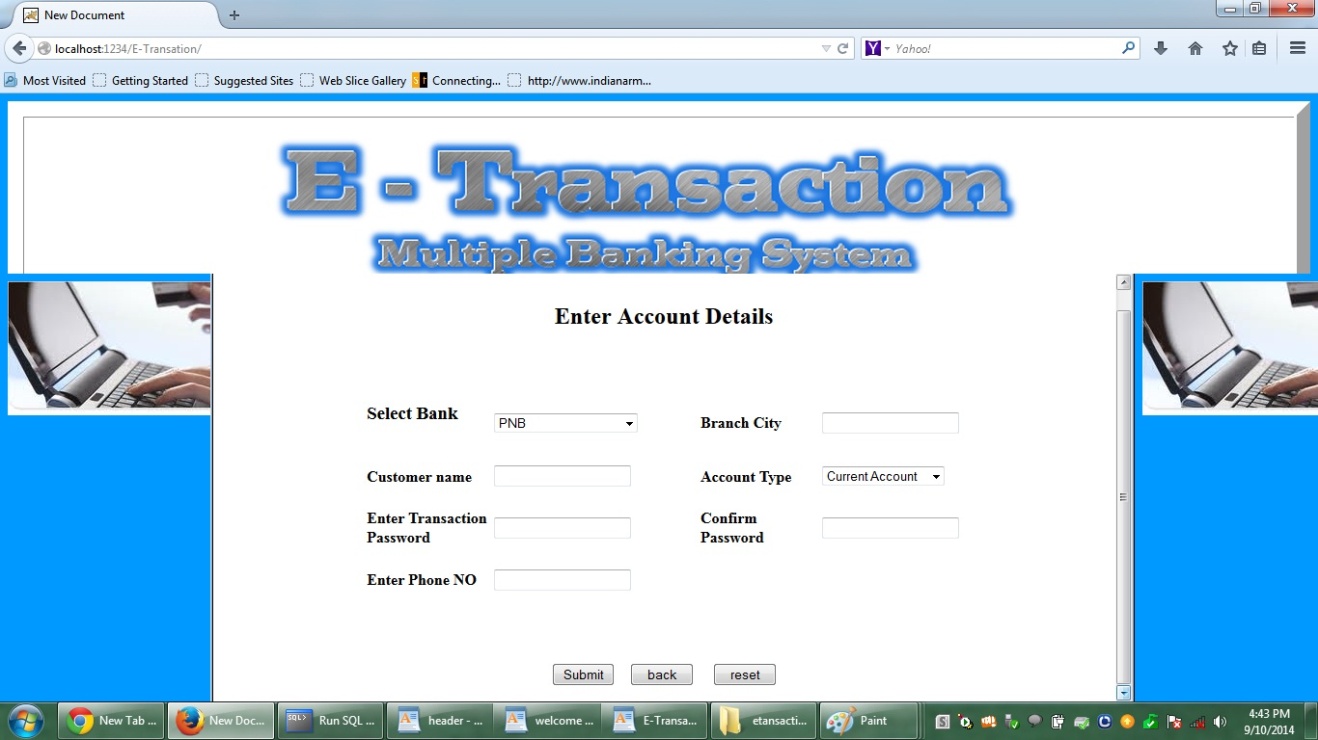
**Account Information**

****

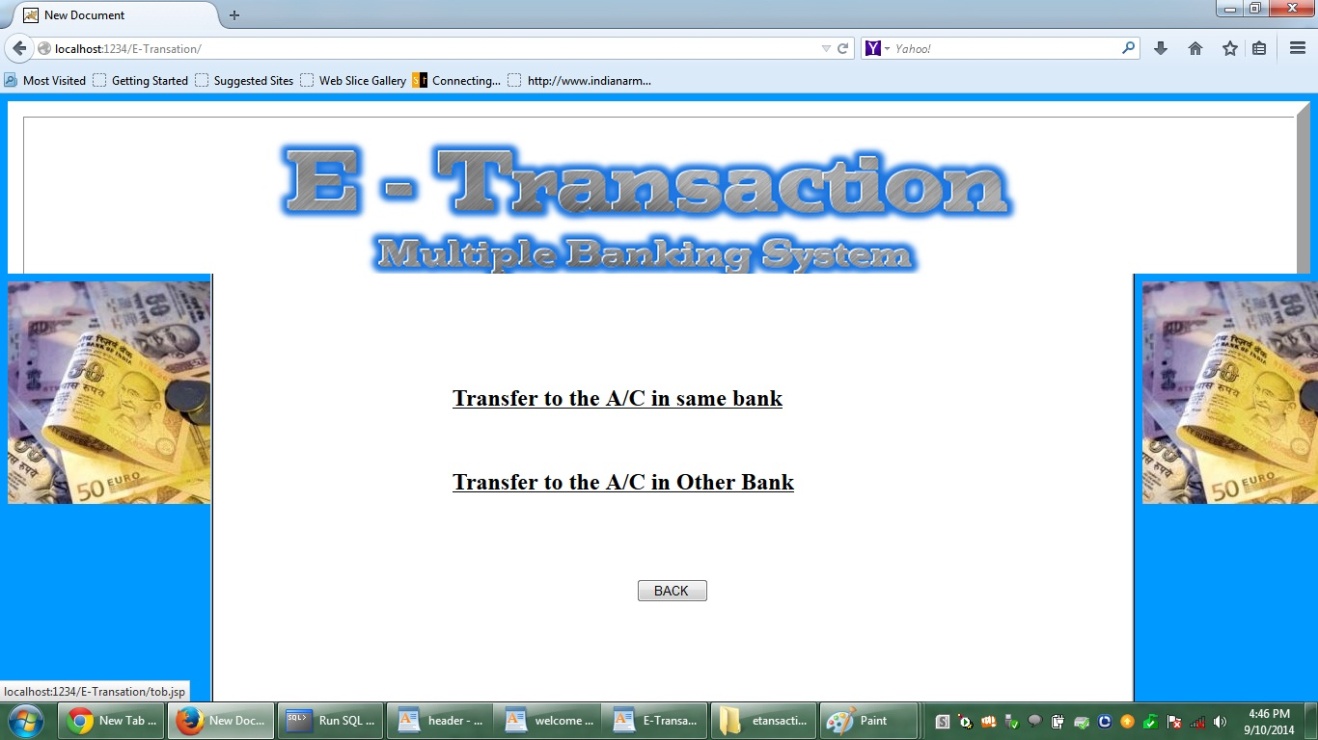
**Accepted by admin**

****

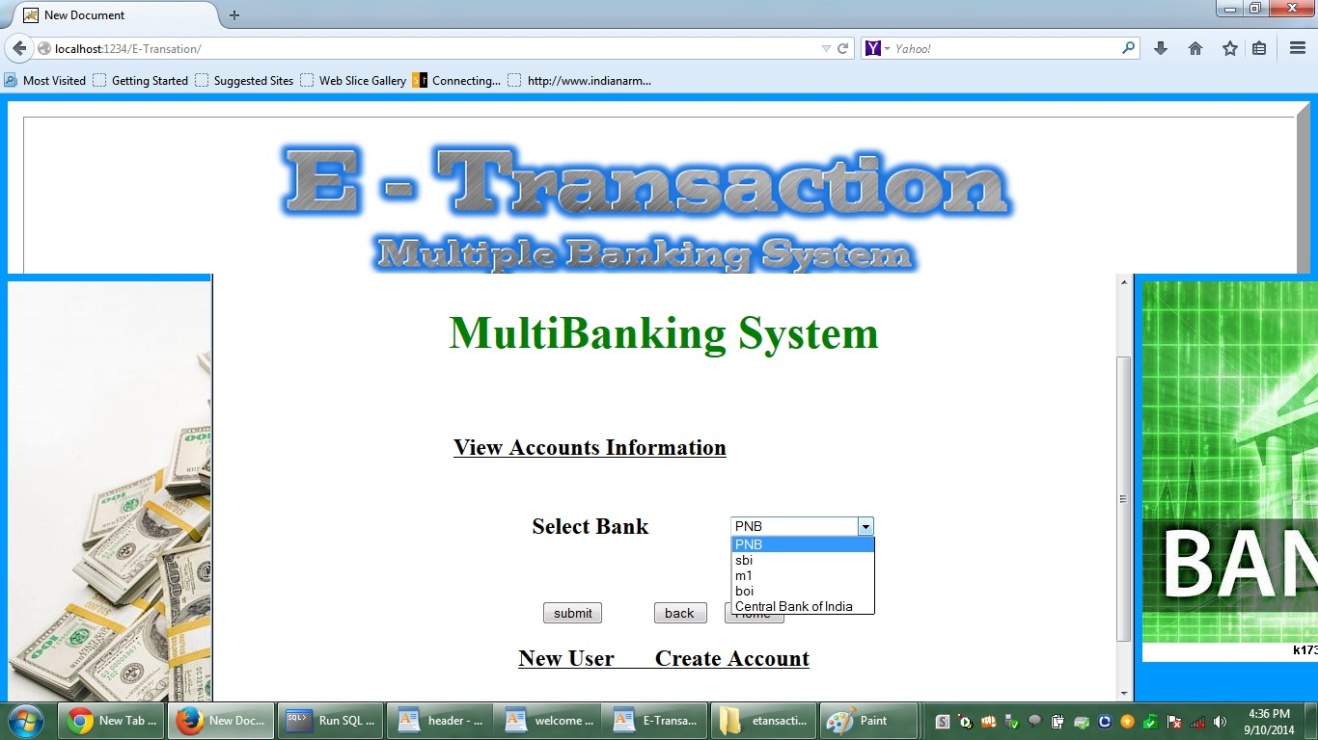
**Account Details**



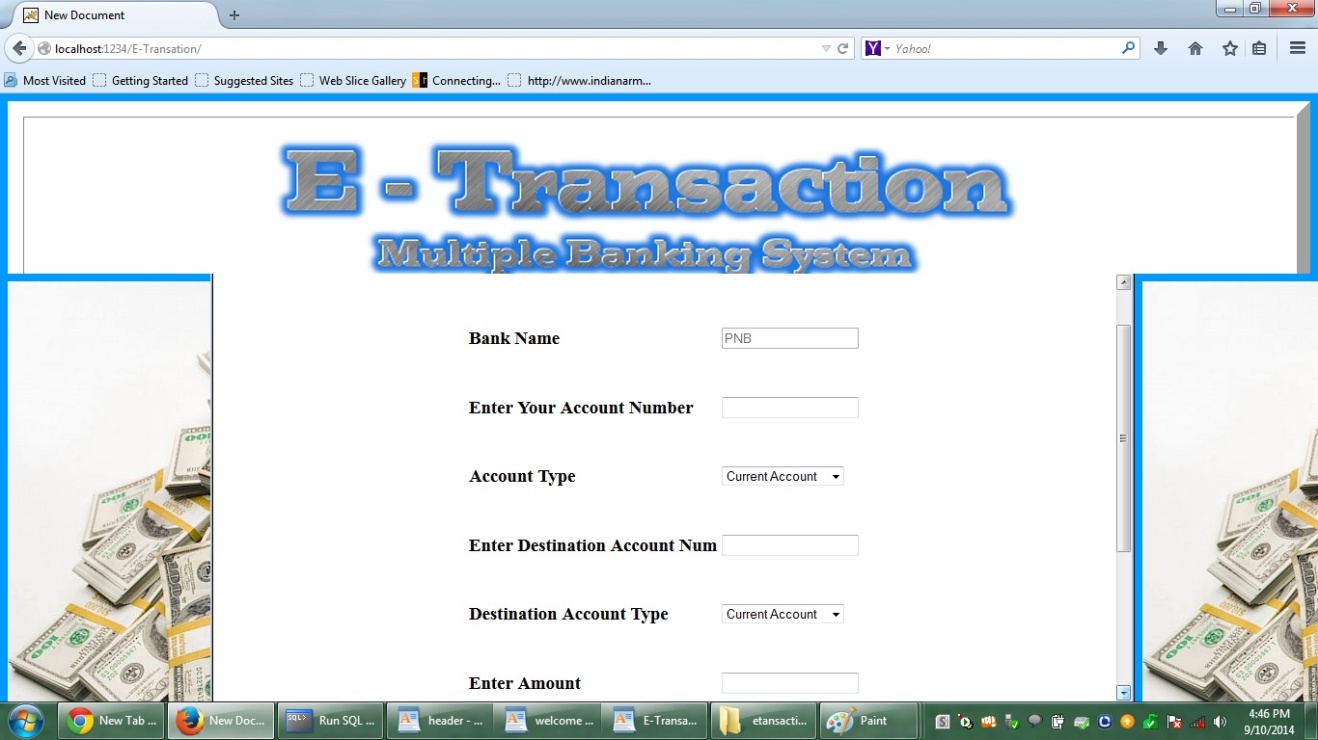
**Bank service**



**New user create account**



**Money transfer**



**Welcome.jsp**

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<HTML>

<HEAD>

<TITLE> New Document </TITLE>

<META NAME="Generator" CONTENT="EditPlus">

<META NAME="Author" CONTENT="">

<META NAME="Keywords" CONTENT="">

<META NAME="Description" CONTENT="">

</HEAD>

<frameset rows="30%,71%" cols="\*" border="" bordercolor="blue">

<frame name="header" src="header.jsp" scrolling="no"/>

<frame name="body" src="body.jsp" >

</frameset>

<noframes></noframes>

</HTML>

**Header.jsp**

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<HTML>

<HEAD>

<TITLE> New Document </TITLE>

<META NAME="Generator" CONTENT="EditPlus">

<META NAME="Author" CONTENT="">

<META NAME="Keywords" CONTENT="">

<META NAME="Description" CONTENT="">

<script language="javascript" type="text/javascript">

var timerid = 0;

var images = new Array("MES.jpg","E1.gif","E3.jpg","E41.jpg","E5.jpg","E6.jpg");

var countimages = 0;

function startTime()

{

if(timerid)

{

timerid = 0;

}

var tDate = new Date();

if(countimages == images.length)

{

countimages = 0;

}

if(tDate.getSeconds() % 2 == 0)

{

document.getElementById("img1").src = images[countimages];

}

countimages++;

timerid = setTimeout("startTime()",800);

}

</script>

</HEAD>

<body bgcolor="#0099FF">

<table width=1350 height=67 border=14 bordercolor="#FFFFFF" bgcolor="white">

<tr valign="top">

<th height="19">

<center>

<table width=900 height=20>

<tr><center> <img id="img1" src="a.png"/></center>

</tr>

</table>

</center></th>

</tr>

</table>

</body>

</HTML>

**Footer.jsp**

<%@ page import ="java.sql.\*,beans.\*" %>

<%@page import="com.et.beans.AbstractDataAccessObject"%>

<%

try{

//Connection con=DBConn.getConn();

Connection con=new AbstractDataAccessObject().getConnection();

//Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

float bal=200;int status=0;

//Connection con=DriverManager.getConnection("jdbc:odbc:amarDSN","system","amar");

String uid=(String)session.getAttribute("id");

String name=request.getParameter("name");

String pwd=request.getParameter("password");

String tpwd=request.getParameter("tpassword");

String accno=request.getParameter("accno");

String cid=request.getParameter("cid");

String bname=(String)session.getAttribute("bname");

String atype=request.getParameter("atype");

PreparedStatement st=con.prepareStatement("insert into customer values(?,?,?,?,?,?,?,?,?,?)");

st.setString(1,uid);

st.setString(2,cid);

st.setString(3,pwd);

st.setString(4,accno);

st.setString(5,atype);

st.setString(6,name);

st.setString(7,bname);

st.setFloat(8,bal);

st.setString(9,tpwd);

st.setInt(10,status);

response.sendRedirect("accsuccess.jsp");

con.close();

st.close();

}catch(Exception e){}

%>

**Laccounts.jsp;**

<%@page import="java.sql.\*,beans.\*"%>

<%@page import="com.et.beans.AbstractDataAccessObject"%>

<BODY bgcolor="white"><BR><BR><BR>

<CENTER>

<h2><font color="green" style="background-color: rgb(255, 255, 255)"><u>List of Accounts</u></font></H2><BR><BR><BR><BR>

<table align=center cellpadding=0 bgcolor="white">

<colgroup span=4 align=center width="20%">

<tr>

<td align=center><b><font color="black">A/C Number</font></b> </td>

<td align=center><b><font color="black">A/C Name</font></b> </td>

<td align=center><b><font color="black">Balance</font></b></td>

<td align=center><b><font color="black">A/C Type</font></td>

</tr>

<%

String bname=(String)session.getAttribute("bname");

System.out.println("bank name "+bname);

/\* Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Connection con=DriverManager.getConnection("jdbc:odbc:amarDSN","system","amar");\*/

try{

//Connection con=DBConn.getConn();

Connection con=new AbstractDataAccessObject().getConnection();

Statement st=con.createStatement();

ResultSet rs=st.executeQuery("select \* from customer where bname='"+bname+"'");

while(rs.next())

{

%>

<tr>

<td align=center><font color="blue"><%=rs.getString(4)%></font> </td>

<td align=center><font color="blue"><%=rs.getString(6)%></font> </td>

<td align=center><font color="blue"><%=rs.getFloat(8)%></font> </td>

<td align=center><font color="blue"><%=rs.getString(5)%></font> </td></tr>

<%

}

rs.close();

st.close();

con.close();

}catch(Exception e){}

%>

</table>

<br> <BR><BR>

<center><input type="submit" value=" OK ">

**Adminhome.jsp**

<html>

<head>

<title></title>

</head>

<body bgcolor="white">

<BR><BR><BR><BR><BR><center>

<table width="50%" align="center" border=0 bgcolor="white">

<tr>

<td align="center"><a href="pbreq.jsp"><h2><font color="green"> Pending Bankers Requests</font></h2></a></td></tr><tr></tr><tr></tr>

<tr align="center"><td><a href="pureq.jsp"><h2><font color="green">Pending User Requests</font></a></td></tr><tr></tr><tr></tr><br><br>

<tr align="center"><td><a href="home.jsp"><h2><font color="red">Signout</font></a></td></tr>

<tr></tr><tr></tr></table>

</body>

</html>

**Adminfailure.jsp**

<HTML>

<HEAD>

<TITLE> New Document </TITLE>

</head>

<BODY bgcolor="white">

<center><h2><font color="red" style="background-color: rgb(255, 255, 255)">Invalid user name and password</font></h2></center>

<jsp:include page="admin.jsp"/>

</BODY>

</HTML>

**Addamount .jsp**

<html>

<head>

<script>

function call() {

if (document.f.number.value == "") {

alert("Please Enter account number");

document.f.number.focus();

return false;

}

if(!(document.f.number.value.match(/^[0-9]+$/)))

{

alert("Invalid acc");

document.f.number.focus();

return false;

}

if (document.f.amount.value == "") {

alert("Please Enter amount");

document.f.amount.focus();

return false;

}

if (document.f.amt.value<=0)

{

alert("invalid");

document.f1.amt.focus();

return false;

}

if(!(document.f.amount.value.match(/^[0-9]+$/)))

{

alert("Invalid amount");

document.f.amount.focus();

return false;

}

}

</script>

</head>

<BODY background="img9.jpg">

<%String cid=request.getParameter("cid");

session.setAttribute("cid",cid);

%>

<center>

<form action="debitamount.jsp" name="f" onSubmit='return call()'>

<h1><font color="black" style="background-color: rgb(255, 255, 255)">Add Amount To Your Account</font></h1>

<table align="center" bgcolor="white">

<tr><td><font color="black">Enter Account Number</font></td>

<td><input type="text" name="number"></td>

</tr>

<tr><td><font color="black">Enter Amount</font></td>

<td><input type="text" name="amount"></td>

</tr>

<tr>

<td colspan="2" align="center"><input type ="submit" value="Add">

</td>

</tr>

<tr>

<td align="center"> <a href="lcustomer.jsp"><font color="black" style="background-color: rgb(255, 255, 255)"><h2>Back</h2></font></a>

</td>

</tr>

</table>

</form>

</center>

</body>

</html>

**Acceptuser.jsp**

<%@ page import ="java.sql.\*,beans.\*" %>

<%@page import="com.et.beans.AbstractDataAccessObject"%>

<%

// Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

/\* Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Connection con=DriverManager.getConnection("jdbc:odbc:amarDSN","system","amar");\*/

try{

//Connection con=DBConn.getConn();

Connection con=new AbstractDataAccessObject().getConnection();

int id=Integer.parseInt(request.getParameter("id"));

System.out.println(id);

session.setAttribute("id",id);

String bid="";

String pwd="";

int status=1;

// Connection con=DriverManager.getConnection("jdbc:odbc:amarDSN","system","amar");

PreparedStatement ps2=con.prepareStatement("select \* from clogin where id=?");

ps2.setInt(1,id);

ResultSet rs3=ps2.executeQuery();

if(rs3.next()){

bid=rs3.getString(2);

pwd=rs3.getString(3);

}

PreparedStatement ps=con.prepareStatement("update clogin set cid=?,pwd=?,status=? where id=?");

System.out.println("hi");

ps.setString(1,bid);

ps.setString(2,pwd);

ps.setInt(3,status);

ps.setInt(4,id);

ps.executeUpdate();

response.sendRedirect("acceptsuccess.jsp");

con.close();

}catch(Exception e){

System.out.println(e);

}

%>

**Debitamount.jsp**

<%@page import="java.sql.\*,beans.\*"%>

<%@page import="com.et.beans.AbstractDataAccessObject"%>

<html>

<BODY bgcolor="white">

<%

float amount=0;

String numb=request.getParameter("number");

String amt=request.getParameter("amount");

String cid=(String)session.getAttribute("cid");

System.out.println(cid);

float f=Float.parseFloat(amt);

/\* Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Connection con=DriverManager.getConnection("jdbc:odbc:amarDSN","system","amar");\*/

try{

//Connection con=DBConn.getConn();

Connection con=new AbstractDataAccessObject().getConnection();

Statement st=con.createStatement(ResultSet.TYPE\_SCROLL\_SENSITIVE,ResultSet.CONCUR\_UPDATABLE);

Statement st1=con.createStatement(ResultSet.TYPE\_SCROLL\_SENSITIVE,ResultSet.CONCUR\_UPDATABLE);

ResultSet rs=st.executeQuery("select bal from customer where cid='"+cid+"' and ACCNO="+numb);

if(rs.next())

{

float bal=rs.getFloat(1);

amount=bal+f;

int x=st1.executeUpdate("update customer set bal="+amount+" where cid='"+cid+"' and ACCNO="+numb);

if(x>0){

%>

<h1 align="center"><font color="green" style="background-color: rgb(255, 255, 255)">Amount Added Successfully</font></h1><br><br>

<h1 align="center"><a href="lcustomer.jsp"><font color="black">Back</font></a></h1>

<%}else{

%>

<h1 align="center" ><font color="red" style="background-color: rgb(255, 255, 255)">Amount Not Added</font></h1><br><br>

<h1 align="center"><a href="lcustomer.jsp"><font color="#80ffff">Back</font></a></h1>

<%}

%>

<%

}

}catch(Exception e){

e.printStackTrace();

}

%>

</body>

</html>

**Deleteuser.jsp**

<%@ page import ="java.sql.\*,beans.\*" %>

<%@page import="com.et.beans.AbstractDataAccessObject"%>

<%

/\*Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Connection con=DriverManager.getConnection("jdbc:odbc:amarDSN","system","amar");\*/

try{

//Connection con=DBConn.getConn();

Connection con=new AbstractDataAccessObject().getConnection();

int id=Integer.parseInt(request.getParameter("id"));

PreparedStatement ps=con.prepareStatement("delete from clogin where id=?");

System.out.println("hi");

ps.setInt(1,id);

ps.executeUpdate();

response.sendRedirect("deletesuccess.jsp");

con.close();

ps.close();

}catch(Exception e){}

%>

**Insaccount.jsp**

<%@ page import ="java.sql.\*,beans.\*" %>

<%@page import="com.et.beans.AbstractDataAccessObject"%>

<%

try{

//Connection con=DBConn.getConn();

Connection con=new AbstractDataAccessObject().getConnection();

//Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

float bal=200;int status=0;

//Connection con=DriverManager.getConnection("jdbc:odbc:amarDSN","system","amar");

String uid=(String)session.getAttribute("id");

String name=request.getParameter("name");

String pwd=request.getParameter("password");

String tpwd=request.getParameter("tpassword");

String accno=request.getParameter("accno");

String cid=request.getParameter("cid");

String bname=(String)session.getAttribute("bname");

String atype=request.getParameter("atype");

PreparedStatement st=con.prepareStatement("insert into customer values(?,?,?,?,?,?,?,?,?,?)");

st.setString(1,uid);

st.setString(2,cid);

st.setString(3,pwd);

st.setString(4,accno);

st.setString(5,atype);

st.setString(6,name);

st.setString(7,bname);

st.setFloat(8,bal);

st.setString(9,tpwd);

st.setInt(10,status);

response.sendRedirect("accsuccess.jsp");

con.close();

st.close();

}catch(Exception e){}

%>

**Insotheraccount.jsp**

<%@ page import ="java.sql.\*,beans.\*" %>

<%@ page import ="java.util.\*"%>

<%@page import="com.et.beans.AbstractDataAccessObject"%>

<%

try{

//Connection con=DBConn.getConn();

Connection con=new AbstractDataAccessObject().getConnection();

//Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Statement stt=null;

int ind=0;

// Connection con=DriverManager.getConnection("jdbc:odbc:amarDSN","system","amar");

String uid=(String)session.getAttribute("id");

String pwd=(String)session.getAttribute("pwd");

System.out.println(uid);

String name=request.getParameter("name");

System.out.println(name);

System.out.println(pwd);

String tpwd=request.getParameter("tpassword");

System.out.println(tpwd);

String city=request.getParameter("city");

System.out.println(city);

String cid=(String)session.getAttribute("cid");

System.out.println(cid);

String bname=request.getParameter("bname");

System.out.println(bname);

String atype=request.getParameter("atype");

System.out.println(atype);

String phone=request.getParameter("phone");

long ph=Long.parseLong(phone);

System.out.println(phone);

float bal=2000;int status=0;

PreparedStatement ps2=con.prepareStatement("select \* from customer where cid=? and bname=? and atype=?");

ps2.setString(1,cid);

ps2.setString(2,bname);

ps2.setString(3,atype);

ResultSet rs3=ps2.executeQuery();

if(rs3.next()){

response.sendRedirect("acexist.jsp");

rs3.close();

ps2.close();

con.close();

}

else{

String accno="91"+new Random().nextInt(100)+new Random().nextInt(999999);

session.setAttribute("accno",accno);

PreparedStatement st=con.prepareStatement("insert into customer values(?,?,?,?,?,?,?,?,?,?,?,?)");

st.setString(1,uid);

st.setString(2,cid);

st.setString(3,pwd);

st.setString(4,accno);

st.setString(5,atype);

st.setString(6,name);

st.setString(7,bname);

st.setFloat(8,bal);

st.setString(9,tpwd);

st.setInt(10,status);

st.setLong(11,ph);

st.setString(12,city);

System.out.println(st);

st.executeUpdate();

con.close();

response.sendRedirect("acprocess.jsp");

con.close();

st.close();

}

}catch(Exception e){

e.printStackTrace();

}

%>

**Bankprocess.jsp**

<%@ page import ="java.sql.\*,beans.\*" %>

<%@page import="com.et.beans.AbstractDataAccessObject"%>

<%

try{

//Connection con=DBConn.getConn();

Connection con=new AbstractDataAccessObject().getConnection();

int i=0;int ii=0;

//Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

float bal=200;

// Connection con=DriverManager.getConnection("jdbc:odbc:amarDSN","system","amar");

PreparedStatement st1=con.prepareStatement("select MAX(id) from blogin");

ResultSet rs1=st1.executeQuery();

if(rs1.next()){

String s=rs1.getString(1);

if(s==null){

i=1;}else{i=Integer.parseInt(s)+1;}

}

String bid=request.getParameter("id");

// String name=request.getParameter("name");

String pwd=request.getParameter("pwd");

String phone=request.getParameter("phone");

long ph=Long.parseLong(phone);

// String cid=request.getParameter("cid");

//String bname=(String)session.getAttribute("bname");

String bname=request.getParameter("bname");

int ind=0;

PreparedStatement ps2=con.prepareStatement("select \* from bank where bname=?");

ps2.setString(1,bname);

ResultSet rs3=ps2.executeQuery();

if(rs3.next()){

}else{

PreparedStatement ps1=con.prepareStatement("select MAX(id) from bank");

ResultSet rs2=ps1.executeQuery();

if(rs2.next()){

String s=rs2.getString(1);

if(s==null){

ii=1;}else{ii=Integer.parseInt(s)+1;}

}

PreparedStatement st3=con.prepareStatement("insert into bank values(?,?)");

st3.setInt(1,ii);

st3.setString(2,bname);

st3.executeUpdate();

}

PreparedStatement st=con.prepareStatement("insert into blogin values(?,?,?,?,?,?)");

st.setInt(1,i);

st.setString(2,bid);

st.setString(3,pwd);

st.setString(4,bname);

st.setInt(5,ind);

st.setLong(6,ph);

st.executeUpdate();

response.sendRedirect("process.jsp");

con.close();

}catch(Exception e){

System.out.println(e);

}

%>

**Customerprocess.jsp**

<%@ page import ="java.sql.\*,beans.\*" %>

<%@page import="com.et.beans.AbstractDataAccessObject"%>

<%

try{

//Connection con=DBConn.getConn();

Connection con=new AbstractDataAccessObject().getConnection();

// Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

float bal=200;

// Connection con=DriverManager.getConnection("jdbc:odbc:amitDSN","system","amit");

String id=request.getParameter("id");

int ii=0;

String pwd=request.getParameter("pwd");

String phone=request.getParameter("phone");

String email=request.getParameter("email");

long ph=Long.parseLong(phone);

PreparedStatement ps3=con.prepareStatement("select \* from clogin where cid=? ");

ps3.setString(1,id);

ResultSet rs3=ps3.executeQuery();

// System.out.println("hiii");

if(rs3.next()){System.out.println("hiii22");

response.sendRedirect("exist.jsp");

}

else{

PreparedStatement ps1=con.prepareStatement("select MAX(id) from clogin");

ResultSet rs2=ps1.executeQuery();

if(rs2.next()){

String s=rs2.getString(1);

if(s==null){

ii=1;}else{ii=Integer.parseInt(s)+1;}

}

int ind=0;

PreparedStatement st=con.prepareStatement("insert into clogin values(?,?,?,?,?,?)");

st.setInt(1,ii);

st.setString(2,id);

st.setString(3,pwd);

st.setInt(4,ind);

st.setLong(5,ph);

st.setString(6,email);

st.executeUpdate();

response.sendRedirect("process1.jsp");

}

//st.close();

con.close();

}

catch(Exception e){}

%>

**Accrej.jsp**

html>

<BODY bgcolor="white">

<center><h3><font color="red" style="background-color: rgb(255, 255, 255)">Your account request is rejected</font> </h3></center>

<br>

<center><h3>Sorry.... :-(</h3></center>

<center> <jsp:include page="user.jsp"/> </center>

</BODY>

</HTML>

**Admin.jsp**

<HTML>

<HEAD>

<TITLE></TITLE>

</HEAD>

<script>

function call() {

if (document.f.uname.value == "") {

alert("Please Enter Name");

document.f.name.focus();

return false;

}

if (document.f.pwd.value == "") {

alert("Please Enter Ur Password");

document.f.pwd.focus();

return false;

}

}

</script>

<BODY bgcolor="white">

<center><BR>

<BR><br><br>

<font size="+2" color="black" style="background-color: rgb(255, 255, 255)"><h2>

<b>Welcome To Multi Banking System</b>

</h2>

</font><br>

<table border=0 bgcolor="white">

<tr><td><br><center>

<form method=post action="admincheck.jsp" name="f"

onSubmit='return call()'>

<table bgcolor="white">

<tr><td>

<font color="black"><b>User Name:</b>

</font>

</td><td>

<font color="red"><input name="uname" type="text" tabindex="1">

</font></td>

</tr><tr>

<td>&nbsp;

</td></tr>

<tr> <td>

<b><font color="black"> Password :</font></b>

</td><td>

<input name="pwd" type="password" tabindex="2">

</td></tr>

<tr><td>&nbsp;

</td></tr>

<tr><td align="right">

<input type="submit" tabindex="3" value="Submit">

</td><td>

<input type="reset" tabindex="4" value="Reset">

</td>

<td width="120" align="left">

<a href="home.jsp"><font color="green" color="black" style="background-color: rgb(255, 255, 255)">HOME</font></a></td></tr></table>

<br>

<BR></form>

<br>

</center>

</td>

</tr>

</table>

</center>

</BODY>

</HTML>

**Bank Process.jsp**

<%@ page import ="java.sql.\*,beans.\*" %>

<%@page import="com.et.beans.AbstractDataAccessObject"%>

<%

try{

//Connection con=DBConn.getConn();

Connection con=new AbstractDataAccessObject().getConnection();

int i=0;int ii=0;

//Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

float bal=200;

// Connection con=DriverManager.getConnection("jdbc:odbc:amitDSN","system","amit");

PreparedStatement st1=con.prepareStatement("select MAX(id) from blogin");

ResultSet rs1=st1.executeQuery();

if(rs1.next()){

String s=rs1.getString(1);

if(s==null){

i=1;}else{i=Integer.parseInt(s)+1;}

}

String bid=request.getParameter("id");

// String name=request.getParameter("name");

String pwd=request.getParameter("pwd");

String phone=request.getParameter("phone");

long ph=Long.parseLong(phone);

// String cid=request.getParameter("cid");

//String bname=(String)session.getAttribute("bname");

String bname=request.getParameter("bname");

int ind=0;

PreparedStatement ps2=con.prepareStatement("select \* from bank where bname=?");

ps2.setString(1,bname);

ResultSet rs3=ps2.executeQuery();

if(rs3.next()){

}else{

PreparedStatement ps1=con.prepareStatement("select MAX(id) from bank");

ResultSet rs2=ps1.executeQuery();

if(rs2.next()){

String s=rs2.getString(1);

if(s==null){

ii=1;}else{ii=Integer.parseInt(s)+1;}

}

PreparedStatement st3=con.prepareStatement("insert into bank values(?,?)");

st3.setInt(1,ii);

st3.setString(2,bname);

st3.executeUpdate();

}

PreparedStatement st=con.prepareStatement("insert into blogin values(?,?,?,?,?,?)");

st.setInt(1,i);

st.setString(2,bid);

st.setString(3,pwd);

st.setString(4,bname);

st.setInt(5,ind);

st.setLong(6,ph);

st.executeUpdate();

response.sendRedirect("process.jsp");

con.close();

}catch(Exception e){

System.out.println(e);

}

%>

**Blogin.jsp**

<%@ page import="java.sql.\*,beans.\*"%>

<%@page import="com.et.beans.AbstractDataAccessObject"%>

<HTML>

<HEAD>

<TITLE> </TITLE>

</HEAD>

<script >

function call()

{

if(document.f.bid.value==""){

alert("Please Enter Ur Name");

document.f.bid.focus();

return false;

}

if(document.f.pwd.value==""){

alert("Please Enter Ur Password");

document.f.pwd.focus();

return false;

}

if((document.f.bid.value=="")||(document.f.pwd.value==""))

{

alert("Please Enter Username& password");

document.f.bid.focus();

return false;

}

}

</script>

<BODY bgcolor="white">

<br>

<center>

<BR>

<b><h2><u><font color="green" style="background-color: rgb(255, 255, 255)">Welcome To Multi Banking System</font></u></h2></b>

<br>

**Bank Admin Login**

<br>

<form method=post action="bloginch.jsp" name="f" onSubmit='return call()'>

<table width="40%" cellpadding="5" cellspacing="5" border=0 bgcolor="white">

<colgroup span=2 >

<tr>

<td><font color="black"><b>Username:</b></font></td><td><input type="text" name="bid" ></td>

</tr>

<tr>

<td><b><font color="black"> Password:</font></b></td><td><input type="password" name="pwd"></td>

<tr>

<td><b><font color="black"> Select Bank:</font></b></td><td><SELECT NAME="bl">

<%

/\* Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Connection con=DriverManager.getConnection("jdbc:odbc:amitDSN","system","amit");\*/

try{

//Connection con=DBConn.getConn();

Connection con=new AbstractDataAccessObject().getConnection();

Statement st=con.createStatement();

ResultSet rs=st.executeQuery("select \* from bank");

while(rs.next())

{

%>

<option><%=rs.getString(2)%></option>

<%

}

con.close();

}catch(Exception e){

System.out.println(e);

}

%>

</SELECT></td>

</tr>

</table>

<br> <input type="submit" value=" Login "> &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<input type="reset" value=" reset ">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<input type="button" name=back value="Home" onClick="window.location='home.jsp'"><BR><BR>

<center><h2><b><a href="newbankuser.jsp"><font color="black" size=5 color="black" style="background-color: rgb(255, 255, 255)"><BR>New User????? Click here</font> </a></b></h2></center>

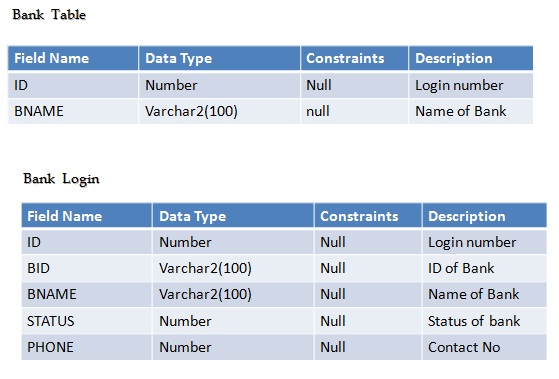
</form>

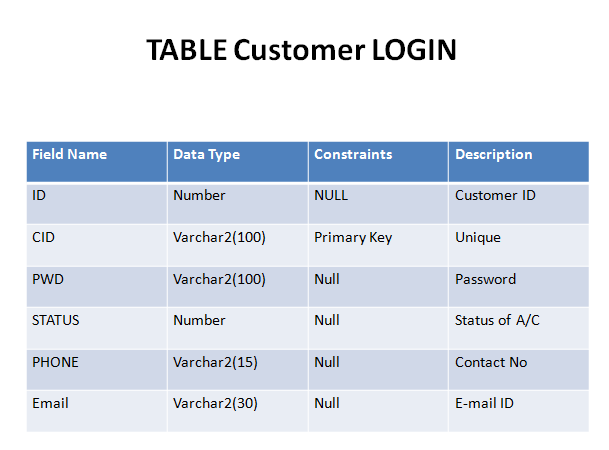
</center>

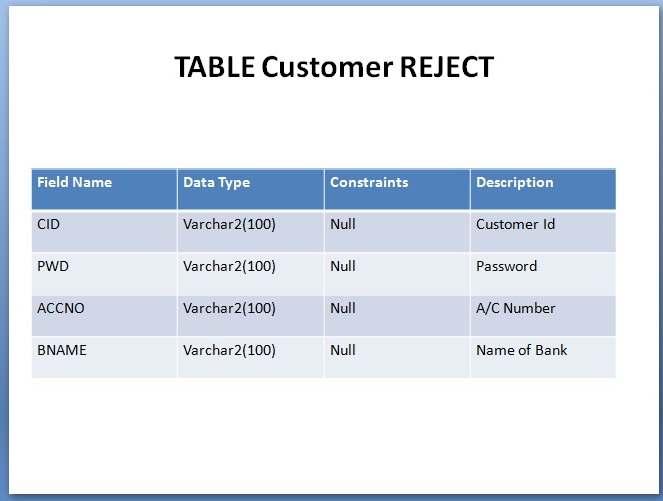
</BODY>

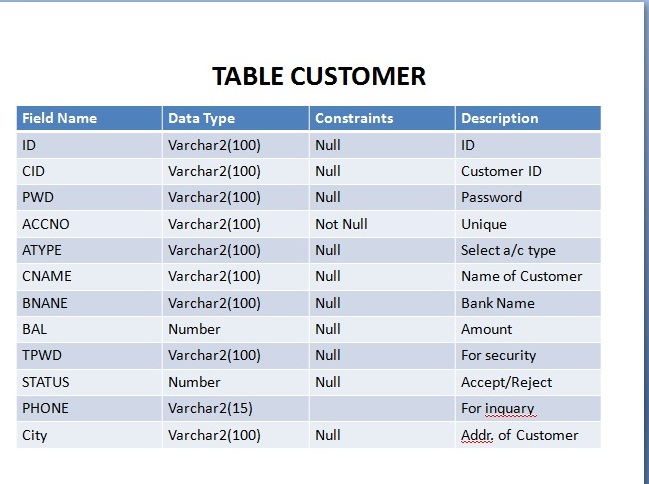
</HTML>

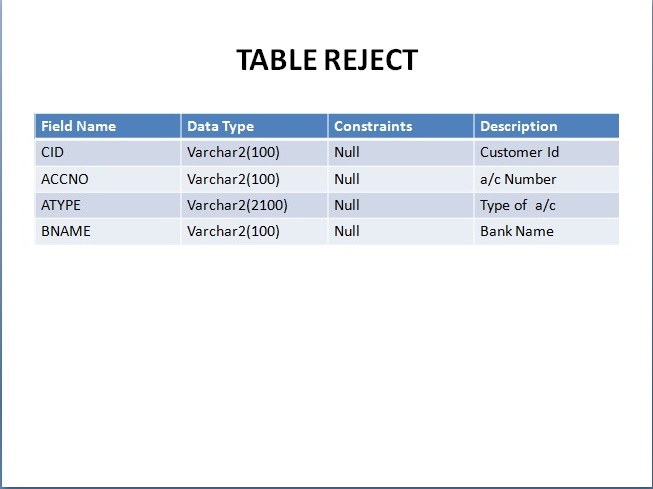
**Structure of Table**

****

****

****

****

****

**Code Efficiency**

The code used is very simple to understand, and applied according to algorithm. No too complex and confusing code is used. For software security reasons Only those data objects are declared global which are used in multiple modules. If the original developer leaves the project then it will not create too much problem for the new developer.

The queries wrote are very simple and requires less time to produce results. Printing option are well described. So, the code is very efficient for this project.

#### Debugging

Debugging occurs as a consequence of successful testing. That is, when a test case uncovers an error, debugging is the process that results in the removal of the error. Although debugging can and should be an orderly process, it is still very much an art. A software engineer, evaluating the results of a test, is often confronted with a “symptomatic “ indication of a software problem. That is the external manifestation of the error and the internal cause of the error may have no obvious relationship to one another.

Debug process always begins with the execution of a test case. The results are assessed and a lack of correspondence between the expected and actual outcome is encountered. In many cases the non-corresponding data are a symptom of an underlying cause that is as yet hidden. The debugging process attempts to match symptom with cause, thereby leading to error correction. The debugging process will always have one of two outcomes:

* The cause will be found corrected and removed, or
* The cause will not be found. In the latter case, the person performing debugging may suspect a cause, design a test case to help validate his suspicion and work toward error correction in an iterative fashion.

In general three categories for debugging approaches may be proposed

**Brute Force:**

The brute force category of debugging is probably the most common and least efficient method for isolating the cause of a software error. This technique is applied when all else fails. Using a “let the computer find the error “ philosophy, memory dumps are taken, run time traces are invoked, and the program is loaded with MSGBOX/PRINT statement .It is hoped that somewhere in the morass of information that is produced, we will find a clue that can lead us to the cause of an error . Although the mass of information produced may ultimately lead to success , it more frequently leads to wasted effort and time.

**Backtracking:**

This is a fairly common debugging approach that can be used successfully in small programs. Beginning at the site where a symptom has been uncovered,the source code is traced backward (manually) until the site of the cause is found. Unfortunately as thenumber of source linesincreases, the number of potential backward path may become unmanageably large.

**Cause Elimination:**

The third approach to debugging Cause Elimination is manifested by induction or deduction and introduces the concept of binary partitioning. Data related to the error occurrence are organized to isolate to potential causes. A “cause hypothesis” is devise and the above data are used to prove or disprove the hypothesis. Alternatively, a list of all possible causes is developed and tests are conducted to eliminate each. If Initial tests indicate that a particular cause hypothesis show promises the data are refined in an attempt to isolate the bug.

**Exception Handling:**

Comprehensive Exception Handling measures were taken in the tool so that all the exceptions thrown can be caught and analyzed (by use of on error go to label name).

**Validation checks**

Validation are necessary checks performed to make the program more reliable , efficient and strong when the user passes the values then the values would be checked to fall within the range. If values are incorrect or any data type error then an appropriate message would flashed. Validations build a relationship between program and data types. Since validation does not accept the wrong entries so the program becomes more dffective.

In this project the validations are performed through the following fields :-

* Primary key
* Foreign key
* Not null
* Limit value

**SYSTEM MAINTENANCE**

**Servlets, JSP, JDBC & HTML Technologies**

**SERVLETS**

**Introduction**

The Java web server is Java Soft's own web Server. The Java web server is just a part of a larger framework, intended to provide you not just with a web server, but also with tools. To build customized network servers for any Internet or Intranet client/server system. Servlets are to a web server, how applets are to the browser.

**About Servlets**

Servlets provide a Java-based solution used to address the problems currently associated with doing server-side programming, including inextensible scripting solutions, platform-specific APIs, and incomplete interfaces.

Servlets are objects that conform to a specific interface that can be plugged into a Java-based server. Servlets are to the server-side what applets are to the client-side - object byte codes that can be dynamically loaded off the net. They differ from applets in that they are faceless objects (without graphics or a GUI component). They serve as platform independent, dynamically loadable, plugable helper byte code objects on the server side that can be used to dynamically extend server-side functionality.

For example, an HTTP Servlets can be used to generate dynamic HTML content. When you use Servlets to do dynamic content you get the following advantages:

* They’re faster and cleaner than CGI scripts
* They use a standard API (the Servlets API)
* They provide all the advantages of Java (run on a variety of servers without needing to be rewritten).

**Attractiveness of Servlets:**

There are many features of Servlets that make them easy and attractive to use. These include:

* Easily configured using the GUI-based Admin tool
* Can be loaded and invoked from a local disk or remotely across the network.
* Can be linked together, or chained, so that one Servlets can call another Servlets, or several Servlets in sequence.
* Can be called dynamically from within HTML pages, using server-side include tags.
* Are secure - even when downloading across the network, the Servlets security model and Servlets sandbox protect your system from unfriendly behavior.

**Advantages of the Servlet API**

One of the great advantages of the Servlet API is protocol independence. It assumes nothing about:

* The protocol being used to transmit on the net
* How it is loaded
* The server environment it will be running in

These qualities are important, because it allows the Servlet API to be embedded in many different kinds of servers. There are other advantages to the Servlet API as well. These include:

* It’s extensible - you can inherit all your functionality from the base classes made available to you.
* It’s simple, small, and easy to use.

**Features of Servlets**

* Servlets are persistent. Servlet are loaded only by the web server and can maintain services between requests.
* Servlets are fast. Since Servlets only need to be loaded once, they offer much better performance over their CGI counterparts.
* Servlets are platform independent.
* Servlets are extensible. Java is a robust, object-oriented programming language, which easily can be extended to suit your needs
* Servlets are secure.
* Servlets can be used with a variety of clients.

**JavaScript**

JavaScript is a script-based programming language that was developed by Netscape Communication Corporation. JavaScript was originally called Live Script and renamed as JavaScript to indicate its relationship with Java. JavaScript supports the development of both client and server components of Web-based applications. On the client side, it can be used to write programs that are executed by a Web browser within the context of a Web page. On the server side, it can be used to write Web server programs that can process information submitted by a Web browser and then updates the browser’s display accordingly

Even though JavaScript supports both client and server Web programming, we prefer JavaScript at Client side programming since most of the browsers supports it. JavaScript is almost as easy to learn as HTML, and JavaScript statements can be included in HTML documents by enclosing the statements between a pair of scripting tags

<SCRIPTS>.. </SCRIPT>.

<SCRIPT LANGUAGE = “JavaScript”>

JavaScript statements

</SCRIPT>

Here are a few things we can do with JavaScript:

* Validate the contents of a form and make calculations.
* Add scrolling or changing messages to the Browser’s status line.
* Animate images or rotate images that change when we move the mouse over them.
* Detect the browser in use and display different content for different browsers.
* Detect installed plug-ins and notify the user if a plug-in is required.

We can do much more with JavaScript, including creating entire application.

**JavaScript Vs Java**

JavaScript and Java are entirely different languages. A few of the most glaring differences are:

* Java applets are generally displayed in a box within the web document; JavaScript can affect any part of the Web document itself.
* While JavaScript is best suited to simple applications and adding interactive features to Web pages; Java can be used for incredibly complex applications.

There are many other differences but the important thing to remember is that JavaScript and Java are separate languages. They are both useful for different things; in fact they can be used together to combine their advantages.

**Advantages**

* JavaScript can be used for Sever-side and Client-side scripting.
* It is more flexible than VBScript.
* JavaScript is the default scripting languages at Client-side since all the browsers supports it.

**Hyper Text Markup Language**

Hypertext Markup Language (HTML), the languages of the World Wide Web (WWW), allows users to produces Web pages that include text, graphics and pointer to other Web pages (Hyperlinks).

HTML is not a programming language but it is an application of ISO Standard 8879, SGML (Standard Generalized Markup Language), but specialized to hypertext and adapted to the Web. The idea behind Hypertext is that instead of reading text in rigid linear structure, we can easily jump from one point to another point. We can navigate through the information based on our interest and preference. A markup language is simply a series of elements, each delimited with special characters that define how text or other items enclosed within the elements should be displayed. Hyperlinks are underlined or emphasized works that load to other documents or some portions of the same document.

HTML can be used to display any type of document on the host computer, which can be geographically at a different location. It is a versatile language and can be used on any platform or desktop.

HTML provides tags (special codes) to make the document look attractive. HTML tags are not case-sensitive. Using graphics, fonts, different sizes, color, etc., can enhance the presentation of the document. Anything that is not a tag is part of the document itself.

**Basic HTML Tags**:

<! -- --> Specifies comments

<A>……….</A> Creates hypertext links

<B>……….</B> Formats text as bold

<BIG>……….</BIG> Formats text in large font.

<BODY>…</BODY> Contains all tags and text in the HTML document

<CENTER>...</CENTER> Creates text

<DD>…</DD> Definition of a term

<DL>...</DL> Creates definition list

<FONT>…</FONT> Formats text with a particular font

<FORM>...</FORM> Encloses a fill-out form

<FRAME>...</FRAME> Defines a particular frame in a set of frames

<H#>…</H#> Creates headings of different levels

<HEAD>...</HEAD> Contains tags that specify information about a document

<HR>...</HR> Creates a horizontal rule

<HTML>…</HTML> Contains all other HTML tags

<META>...</META> Provides meta-information about a document

<SCRIPT>…</SCRIPT> Contains client-side or server-side script

<TABLE>…</TABLE> Creates a table

<TD>…</TD> Indicates table data in a table

<TR>…</TR> Designates a table row

<TH>…</TH> Creates a heading in a table

**Advantages**

* A HTML document is small and hence easy to send over the net. It is small because it does not include formatted information.
* HTML is platform independent.
* HTML tags are not case-sensitive.

**Java Database Connectivity**

**What Is JDBC?**

JDBC is a Java API for executing SQL statements. (As a point of interest, JDBC is a trademarked name and is not an acronym; nevertheless, JDBC is often thought of as standing for Java Database Connectivity. It consists of a set of classes and interfaces written in the Java programming language. JDBC provides a standard API for tool/database developers and makes it possible to write database applications using a pure Java API.

Using JDBC, it is easy to send SQL statements to virtually any relational database.

**What Does JDBC Do?**

Simply put, JDBC makes it possible to do three things:

* Establish a connection with a database
* Send SQL statements
* Process the results.

JDBC versus ODBC and other APIs

At this point, Microsoft's ODBC (Open Database Connectivity) API is that probably the most widely used programming interface for accessing relational databases. It offers the ability to connect to almost all databases on almost all platforms.

So why not just use ODBC from Java? The answer is that you can use ODBC from Java, but this is best done with the help of JDBC in the form of the JDBC-ODBC Bridge, which we will cover shortly. The question now becomes "Why do you need JDBC?" There are several answers to this question:

1. ODBC is not appropriate for direct use from Java because it uses a C interface. Calls from Java to native C code have a number of drawbacks in the security, implementation, robustness, and automatic portability of applications.
2. A literal translation of the ODBC C API into a Java API would not be desirable. For example, Java has no pointers, and ODBC makes copious use of them, including the notoriously error-prone generic pointer "void \*". You can think of JDBC as ODBC translated into an object-oriented interface that is natural for Java programmers.
3. ODBC is hard to learn. It mixes simple and advanced features together, and it has complex options even for simple queries. JDBC, on the other hand, was designed to keep simple things simple while allowing more advanced capabilities where required.
4. A Java API like JDBC is needed in order to enable a "pure Java" solution. When ODBC is used, the ODBC driver manager and drivers must be manually installed on every client machine. When the JDBC driver is written completely in Java, however, JDBC code is automatically installable, portable, and secure on all Java platforms from network computers to mainframes.

**Two-tier and Three-tier Models:**

The JDBC API supports both two-tier and three-tier models for database access.

In the two-tier model, a Java applet or application talks directly to the database. This requires a JDBC driver that can communicate with the particular database management system being accessed. A user's SQL statements are delivered to the database, and the results of those statements are sent back to the user. The database may be located on another machine to which the user is connected via a network. This is referred to as a client/server configuration, with the user's machine as the client, and the machine housing the database as the server. The network can be an Intranet, which, for example, connects employees within a corporation, or it can be the Internet.

**JAVA**

**Application**

### JDBC

### DBMS

**Client machine**

**DBMS-proprietary protocol**

**Database server**

**Java applet or**

**Html browser**

**Application**

**Server (Java)**

**JDBC**

## DBMS

**Client machine (GUI)**

**HTTP, RMI, or CORBA calls**

**Server machine (business Logic)**

**DBMS-proprietary protocol**

**Database server**

In the three-tier model, commands are sent to a "middle tier" of services, which then send SQL statements to the database. The database processes the SQL statements and sends the results back to the middle tier, which then sends them to the user. MIS directors find the three-tier model very attractive because the middle tier makes it possible to maintain control over access and the kinds of updates that can be made to corporate data. Another advantage is that when there is a middle tier, the user can employ an easy-to-use higher-level API which is translated by the middle tier into the appropriate low-level calls. Finally, in many cases the three-tier architecture can provide performance advantages.

Until now the middle tier has typically been written in languages such as C or C++, which offer fast performance. However, with the introduction of optimizing compilers that translate Java byte code into efficient machine-specific code, it is becoming practical to implement the middle tier in Java. This is a big plus, making it possible to take advantage of Java's robustness, multithreading, and security features. JDBC is important to allow database access from a Java middle tier.

**JDBC Driver Types:**

The JDBC drivers that we are aware of at this time fit into one of four categories:

* JDBC-ODBC bridge plus ODBC driver
* Native-API partly-Java driver
* JDBC-Net pure Java driver
* Native-protocol pure Java driver

**JDBC-ODBC Bridge**

If possible, use a Pure Java JDBC driver instead of the Bridge and an ODBC driver. This completely eliminates the client configuration required by ODBC. It also eliminates the potential that the Java VM could be corrupted by an error in the native code brought in by the Bridge (that is, the Bridge native library, the ODBC driver manager library, the ODBC driver library, and the database client library).

**What Is the JDBC- ODBC Bridge?**

The JDBC-ODBC Bridge is a JDBC driver, which implements JDBC operations by translating them into ODBC operations. To ODBC it appears as a normal application program. The Bridge implements JDBC for any database for which an ODBC driver is available. The Bridge is implemented as the sun.jdbc.odbc Java package and contains a native library used to access ODBC. The Bridge is a joint development of Innersole and Java Soft.

**Java Server Pages (JSP)**

Java server Pages is a simple, yet powerful technology for creating and maintaining dynamic-content web pages. Based on the Java programming language, Java Server Pages offers proven portability, open standards, and mature re-usable component model .The Java Server Pages architecture enables the separation of content generation from content presentation. This separation not eases maintenance headaches; it also allows web team members to focus on their areas of expertise. Now, web page designer can concentrate on layout, and web application designers on programming, with minimal concern about impacting each other’s work.

**Features of JSP**

**Portability:**

Java Server Pages files can be run on any web server or web-enabled application server that provides support for them. Dubbed the JSP engine, this support involves recognition, translation, and management of the Java Server Page lifecycle and its interaction components.

**Components**

It was mentioned earlier that the Java Server Pages architecture can include reusable Java components. The architecture also allows for the embedding of a scripting language directly into the Java Server Pages file. The components current supported include Java Beans, and Servlets.

**Processing**

A Java Server Pages file is essentially an HTML document with JSP scripting or tags. The Java Server Pages file has a JSP extension to the server as a Java Server Pages file. Before the page is served, the Java Server Pages syntax is parsed and processed into a Servlet on the server side. The Servlet that is generated outputs real content in straight HTML for responding to the client.

**Access Models:**

A Java Server Pages file may be accessed in at least two different ways. A client’s request comes directly into a Java Server Page. In this scenario, suppose the page accesses reusable Java Bean components that perform particular well-defined computations like accessing a database. The result of the Beans computations, called result sets is stored within the Bean as properties. The page uses such Beans to generate dynamic content and present it back to the client.

In both of the above cases, the page could also contain any valid Java code. Java Server Pages architecture encourages separation of content from presentation.

**Steps in the execution of a JSP Application:**

1. The client sends a request to the web server for a JSP file by giving the name of the JSP file within the form tag of a HTML page.
2. This request is transferred to the JavaWebServer. At the server side JavaWebServer receives the request and if it is a request for a jsp file server gives this request to the JSP engine.
3. JSP engine is program which can understands the tags of the jsp and then it converts those tags into a Servlet program and it is stored at the server side. This Servlet is loaded in the memory and then it is executed and the result is given back to the JavaWebServer and then it is transferred back to the result is given back to the JavaWebServer and then it is transferred back to the client.

**JDBC connectivity**

The JDBC provides database-independent connectivity between the J2EE platform and a wide range of tabular data sources. JDBC technology allows an Application Component Provider to:

* Perform connection and authentication to a database server
* Manager transactions
* Move SQL statements to a database engine for preprocessing and execution
* Execute stored procedures
* Inspect and modify the results from Select statements

**Purpose**

The generated application is the first version upon the system. The overall system is planned to be in the formal of distributed architecture with homogeneous database platform.

**Conclusions & Recommendations**

The entire project has been developed and deployed as per the requirements stated by the user, it is found to be bug free as per the testing standards that are implemented. Any specification untraced errors will be concentrated in the coming versions, which are planned to be developed in near future. The system at present does not take care off the money payment methods, as the consolidated constructs need SSL standards and are critically to be initiated in the first face, the application of the credit card transactions is applied as a developmental phase in the coming days. The system needs more elaborative technicality for its inception and evolution.

**Testing & Debugging Strategies**

**Testing**

Testing is the process of detecting errors. Testing performs a very critical role for quality assurance and for ensuring the reliability of software. The results of testing are used later on during maintenance also.

**Psychology of Testing**

The aim of testing is often to demonstrate that a program works by showing that it has no errors. The basic purpose of testing phase is to detect the errors that may be present in the program. Hence one should not start testing with the intent of showing that a program works, but the intent should be to show that a program doesn’t work. Testing is the process of executing a program with the intent of finding errors.

Testing Objectives

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say,

* Testing is a process of executing a program with the intent of finding an error.
* A successful test is one that uncovers an as yet undiscovered error.
* A good test case is one that has a high probability of finding error, if it exists.
* The tests are inadequate to detect possibly present errors.
* The software more or less confirms to the quality and reliable standards.

**Levels of Testing**

In order to uncover the errors present in different phases we have the concept of levels of testing. The basic levels of testing are as shown below…

Acceptance Testing

System Testing

Integration Testing

Unit Testing

Client Needs

Requirements

Design

Code

**System Testing**

The philosophy behind testing is to find errors. Test cases are devised with this in mind. A strategy employed for system testing is code testing.

**Code Testing:**

This strategy examines the logic of the program. To follow this method we developed some test data that resulted in executing every instruction in the program and module i.e. every path is tested. Systems are not designed as entire nor are they tested as single systems. To ensure that the coding is perfect two types of testing is performed or for that matter is performed or that matter is performed or for that matter is performed on all systems.

**Types of Testing**

* Unit Testing
* Link Testing

**Unit Testing**

Unit testing focuses verification effort on the smallest unit of software i.e. the module. Using the detailed design and the process specifications testing is done to uncover errors within the boundary of the module. All modules must be successful in the unit test before the start of the integration testing begins.

In this project each service can be thought of a module. There are so many modules like Login, HWAdmin, MasterAdmin, Normal User, and PManager. Giving different sets of inputs has tested each module. When developing the module as well as finishing the development so that each module works without any error. The inputs are validated when accepting from the user.

In this application developer tests the programs up as system. Software units in a system are the modules and routines that are assembled and integrated to form a specific function. Unit testing is first done on modules, independent of one another to locate errors. This enables to detect errors. Through this errors resulting from interaction between modules initially avoided.

**Link Testing**

Link testing does not test software but rather the integration of each module in system. The primary concern is the compatibility of each module. The Programmer tests where modules are designed with different parameters, length, type etc.

Integration Testing

After the unit testing we have to perform integration testing. The goal here is to see if modules can be integrated properly, the emphasis being on testing interfaces between modules. This testing activity can be considered as testing the design and hence the emphasis on testing module interactions.

In this project integrating all the modules forms the main system. When integrating all the modules I have checked whether the integration effects working of any of the services by giving different combinations of inputs with which the two services run perfectly before Integration.

**System Testing**

Here the entire software system is tested. The reference document for this process is the requirements document, and the goal os to see if software meets its requirements.

Here entire ‘ATM’ has been tested against requirements of project and it is checked whether all requirements of project have been satisfied or not.

**Acceptance Testing**

Acceptance Test is performed with realistic data of the client to demonstrate that the software is working satisfactorily. Testing here is focused on external behavior of the system; the internal logic of program is not emphasized.

In this project ‘Network Management Of Database System’ I have collected some data and tested whether project is working correctly or not.

Test cases should be selected so that the largest number of attributes of an equivalence class is exercised at once. The testing phase is an important part of software development. It is the process of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied.

**White Box Testing**

This is a unit testing method where a unit will be taken at a time and tested thoroughly at a statement level to find the maximum possible errors. I tested step wise every piece of code, taking care that every statement in the code is executed at least once. The white box testing is also called Glass Box Testing.

I have generated a list of test cases, sample data, which is used to check all possible combinations of execution paths through the code at every module level.

**Black Box Testing**

This testing method considers a module as a single unit and checks the unit at interface and communication with other modules rather getting into details at statement level. Here the module will be treated as a block box that will take some input and generate output. Output for a given set of input combinations are forwarded to other modules.

Criteria Satisfied by Test Cases

Test cases that reduced by a count that is greater than one, the number of additional test cases that much be designed to achieve reasonable testing.

Test cases that tell us something about the presence or absence of classes of errors, rather than an error associated only with the specific test at hand.

**SYSTEM SECURITY MEASURES**

**Database/data security**

In addition, the DBMS could also require a number and password before allowing the user to perform any database operations.

The protection of data programs, both in primary and secondary memories. The OS to avoid direct access to the data in primary memory or to online files usually does this.

**Creation of User files and access rights**

The proper mechanisms for the identification and verification of users. Each user is assigned an account number and a password. The OS ensures that access to the system is denied unless the number and password are valid.

**COST ESTIMATION**

**U**sually cost of Project depends upon line of code (Kilo line of code). But in Visual Programming the cost estimation depends upon Number of forms, interfaces, tracking the cost of labors, goods and overhead is essential to determine whether a firm is performing in line with expectations-within budget. Besides it requires the following

Cost to be considered:

1. Systems implementation costs.
2. Computing platform costs
   1. Cost to acquire the initial system with finance charges
   2. Cost of additional programming and hardware if required

3. Cost to operate the computer facility

* 1. Power requirements
  2. Air conditioning
  3. Furniture and fixtures
  4. Supplies
  5. Staffing
  6. Maintenance contract fees
  7. Insurance

**Future Enhancements**

**FUTURE ENHANCEMENT:**

* With the spread in wireless technology this technology will be full fill the all requirement of smart card.
* A comprehensive user interface will be implemented.
* With comprehensive infrastructure we will implement it using TCP protocol to make it a more secure transactions mechanism.

**Bibliography:-**

1. References for the Project Development Were Taken From the following Books and Web Sites.
2. JAVA Technologies
3. JAVA Complete Reference by Herbert Sheildt and Patrick Naughton
4. Java Script Bible Gold Edition
5. Oracle 10g Books Online and Online Help support.
6. Web Enable Commercial Application Development Using… Java 2.0 by Ivan Bayross
7. Professional Java Server Programming Subhramanyam Allamaraju.
8. Inside Servlets.
9. Java.sun.com tutorials
10. Software Engineering by Roger Pressman
11. Object-Oriented Software Engineering Using UML, Pattern and Java Second Edition by Bernd Bruegge, Allen H. Duetoit.