**Chapter 1: INTRODUCTION**

Currently we are having lot of banks in the market and any person can do transactions of any individual bank either manually or in online. But no one can do all banks transactions in a single portal or in single bank. This is the main disadvantage in existing system to avoid this problem we are introducing “Multi-Banking system”.

Existing System proposed the user to go to Different banks or their respective sites in order to check balance, to do bill payments, do transaction. This leads to a lot of time wastage. People are really busy now-a-days who cannot afford such wastage of time. Multi-Banking System is a web-basedproject for number of banks who wants to make some revolution in the existing system. This web-application is providing the scope to provide reliable, scalable, low cost solutions to reach more users so that they can handle their multiple accounts very easily. This is a Web-application for its web-savvy users who expect 24 \* 7 information on demand over internet. This system is also very user friendly.

In the current scenario, there is a rat race in each and every professional field. It is true for the banking too. A multi-banking portal is a website dedicated for online information about different banking needs.

This multi-banking portal helps both the end user and bank management to keep track on the transaction. It is being developed in such a way that each record is stored sepertely without making any confusion.So, a multi-banking portal is the perfect online arena, where both the users and and the banks find their goal in the pursuit of finding easy way of doing transaction and keeping track on different functional areas.

**1.1 Multi-Banking System: What does it mean.**

The Multi-Banking System Interface is targeted to the future banking solution for the users who is having multiple bank accounts in 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, By using this portal any client who maintain accounts in various banks can directly log on to Multi-Banking System Interface and make any kind of transactions. In the backend, system will take care of the entire obligation required in order to carry on transaction smoothly.

In other words, Multi-Banking is the process of personnel banking using electronic resources, in particular the internet. Banks and user have moved much of their process online so as to improve the speed by which users can take advantage of various functionalities. Using database technologies, and search engines, users can now find schemes of different bank and choose the one suits them. Using an online multi-banking system may potentially save the user time.

The internet, which reaches a large number of people and can get immediate feedback, has become the major source of online banking.

Internet has radically changed the banking function from the user and bank perspective. Conventional methods of banking processes are readily acknowledged as being time-consuming with high costs and limited geographic reach. However, banking through World Wide Web (WWW) provides global coverage and ease. Likewise, the speedy integration of the internet into banking processes is primarily recognized due to the internet's unrivalled communications capabilities, which enable managers for written communications through e-mails, blogs and portals.

* 1. **OBJECTIVE OF THE SYSTEM**

The ‘**Multi-Banking System’** Interface is targeted to the future banking solution for the users who have multiple bank accounts in different banks. This interface integrates all existing banks and provides business solutions for both retail and corporate.

* This interface integrates all existing banks and provides business solutions for both retailers and corporate
* This system acts as a standard interface between the clients and the banks
* Users who have accounts in various banks can login here and can make any kind of transactions.
* In the backend, system will take care of the entire obligation required in order to carry on transaction smoothly.
  1. **JUSTIFICATION AND NEED FOR THE SYSTEM**

The system provides proper security and reduces the manual work. This application tries to eliminate the difficulties of the existing system and helps the user to reduce the workload and mental conflict. It is very user-friendly. Through this, the users can check their account, transaction and hence it becomes very easy for them.Thinking why multi-banking is so effective and is considered to be the most efficient way of banking? The following are the reasons:

• It is highly cost efficient and promises increased ROI

• Multi-Banking brings in an organized and proactive banking process

• Easy and efficient way of banking

• Reduced complexity, reduced paper work and streamlined workflow

• Establishes efficient communication channel between users and banks

• Helps in establishing a relationship between the user and the manager

• Dependable database applications available to support banking process

• One cannot ignore the efficiency that internet brings in

* 1. **ADVANTAGES OF THE SYSTEM**
* In this system the database is maintained in centralized manner.
* The user can access the entire account information
* This system is very fast because of the centralized database and accessing database will be very easy, when compared to the existing system.
* The user can also ask any query to support in the website, if any.

**Other advantages are:**

**Reach**

• As it is an online portal so it can be reached by every individual, moreover it depends how they use it.

**Speed**

• The speed of operation depends on network, though it works faster then offline mode and decreases paper work cost.

**Cost**

• Multi-Banking System is very cost effective and since it is done in planned way, it will be quite useful.

**Interaction**

• Online banking process provides interaction between user and bank. It is user friendly and also provides support face so user can ask their query also.

**Chapter 2: ASPECTS OF THE PROJECT**

* **Proposed Project: MULTI-BANKING SAYATEM**

A multi-banking portal helps both the users and banks to provide and find the best banking sevices . In the case of users, according to their banking needs and their preferences, the portal shows the desired result . And, to the banks, provides the user information and the transaction history.

So, a multi-banking portal is the perfect online arena, where both the users and and the banks find their goal in the pursuit of finding easy way of doing transaction and keeping track on different functional areas.

* **Proposed location: BANKING SECTOR**

In present scenario, usually a person has more then one account, multi-banking system will help them in managing it properly through one portal. It will bring a revolution in banking sector and brings in the necessary needs of individual.

* **Type of organization: Private and Public**
* **Oppertunities in the market:**

Due to the availability of large number of skilled staff working for lower pay rates than in the developed world, a few countries like India are front runners in setting up portals like this. This type of project demands advanced analysis of the problem and the knowledge about different computer technologies in a great deal. So, we are going to get the right blend of oppertunities and talent for our project.

* **Challenges:**
* high cost of setting up the website
* high cost of training for the newly recruited employees
* ensuring the security and confidentiality of information
* **Service description**:

We help both the user and banks. In case of users we provide them the best banking services with taking care of their security and, for the banks we provide them the services to manage their all work in an efficient way

* **For the user:**
* Create own profile
* Easy access to their accounts
* To make query with support team
* Long-term visibility of account history
* To make payments and transaction
* **For the banks:**
* Easy handling of accounts
* To enable or disable user
* Check out the required information
* **Market potentials:**
* **Present scenario:**

We are living in a modern generation where each banks provide online facility but, in order to save time, money we present you an online banking portal which enables you to handle and manage your accounts in different banks through one portal and one profile. It is hassle free and doesn’t require you to remember too many passwords.

* **Competition:**

As you are well aware, there are countless banking portals available online. It should be worth to create a portal, which would save time, reduce costs, and also assist you the schemes you need. Effective methods and technologies only can make the banking process easy.

* **Sources of revenue:**
* Revenue from banks for publishing their schemes
* Substantial revenue from the user for creating profile
* Online advertisement
* **Establishment Process:**

**Pre-establishment requirements:**Before we start giving the services to our users we have to first develop our portal site as our banking partner will provide the services over the internet.

* **Step by step process:**
* The service we provide is done under a step by step process
* We try to get a comprehensive idea about what the job seekers as well as the recruiters want
* We then provide the perfect solution to each party

**CHAPTER 3: SYSTEM ANALYSIS**

**3.1 STUDY OF THE SYSTEM**

To provide flexibility to the users, the interfaces have been developed that are accessible through a browser. The GUI’S at the top level have been categorized as

1. Administrative user interface

2. The operational or generic user interface

The ‘administrative user interface’ concentrates on the consistent information that is practically, part of the organizational activities and which needs proper authentication for the data collection. These interfaces help the administrators with all the transactional states like Data insertion, Data deletion and Date updating along with the extensive data search capabilities.

The ‘operational or generic user interface’ helps the end users of the system in transactions through the existing data and required services. The operational user interface also helps the ordinary users in managing their own information in a customized manner as per the included flexibilities.

**3.2 INPUT & OUTPOUT REPRESENTETION**

Input design is a part of overall system design. The main objective during the input design is as given below:

•To produce a cost-effective method of input.

•To achieve the highest possible level of accuracy.

•To ensure that the input is acceptable and understood by the user.

**INPUT STAGES:**

The main input stages can be listed as below:

•Data recording

•Data transcription

•Data conversion

•Data verification

•Data control

•Data transmission

•Data validation

•Data correction

**INPUT TYPES:**

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

•External inputs, which are prime inputs for the system.

•Internal inputs, which are user communications with the system.

•Operational, which are computer department’s communications to the system?

•Interactive, which are inputs entered during a dialogue.

**INPUT MEDIA:**

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

•Type of input

•Flexibility of format

•Speed

•Accuracy

•Verification methods

•Rejection rates

•Ease of correction

•Storage and handling requirements

•Security

•Easy to use

•Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

**OUTPUT DESIGN:**

In general are:

•External Outputs whose destination is outside the organization.

•Internal Outputs whose destination is within organization and they are the User’s main interface with the computer. Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs

•Operational outputs whose use is purely with in the computer department.

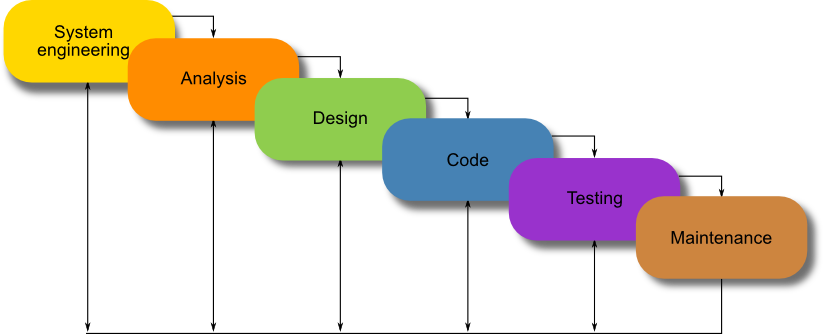
•Interface outputs, which involve the user in communicating directly with the system.

**3.3 PURPOSED MODEL: ITERATIVE MODEL**

An iterative lifecycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software for each cycle of the model.

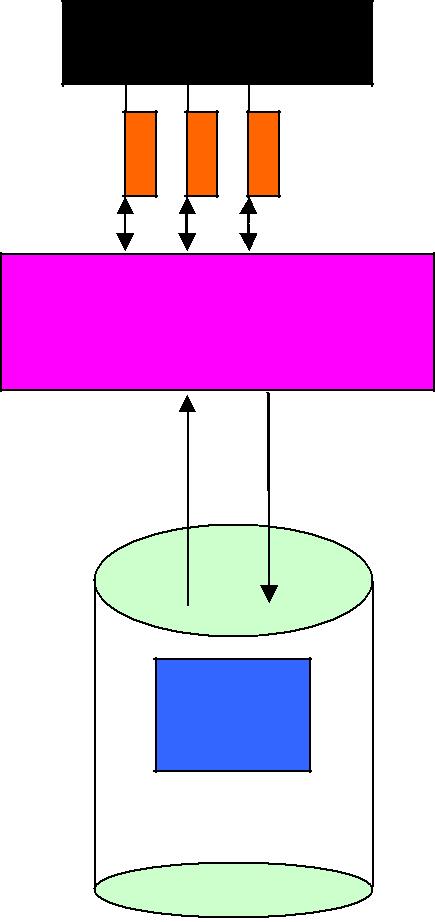
The iterative lifecycle model can be likened to producing software by successive approximation. Drawing an analogy with mathematical methods that use successive approximation to arrive at a final solution, the benefit of such methods depends on how rapidly they converge on a solution.

The key to successful use of an iterative software development lifecycle is rigorous validation of requirements, and verification (including testing) of each version of the software against those requirements within each cycle of the model. The first three phases of the example iterative model is in fact an abbreviated form of a sequential V or waterfall lifecycle model. Each cycle of the model produces software that requires testing at the unit level, for software integration, for system integration and for acceptance. As the software evolves through successive cycles, tests have to be repeated and extended to verify each version of the software.



**3.4 SYSTEM ARCHITECTURE**

**Architecture flow:**

Below architecture diagram represents mainly flow of requests from users to database through servers. In this scenario overall system is designed in three tires separately using three layers called presentation layer, business logic layer and data link layer. This project was developed using 3-tier architecture.

USER

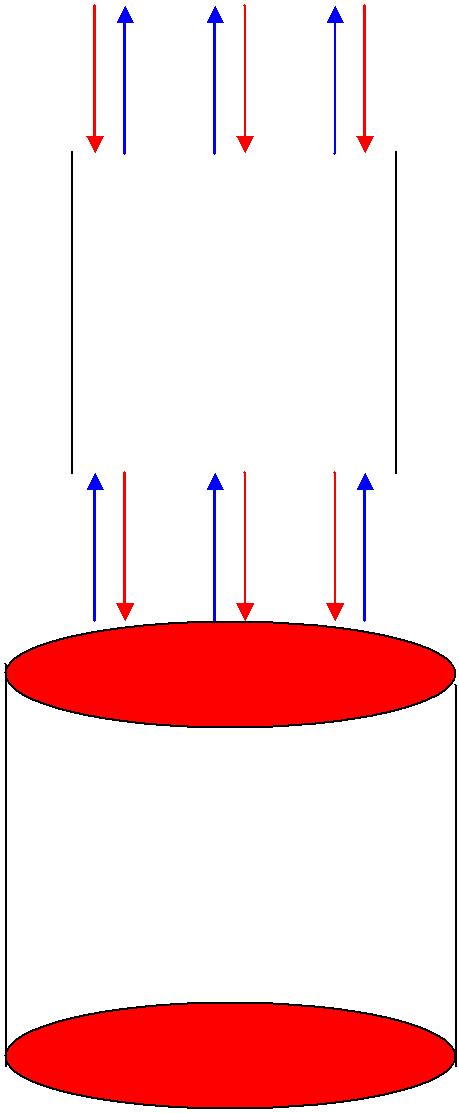
USER

SERVER

DATABASE

**URL Pattern:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Presentation |  |  |  |
|  |  | Layer |  |  |  |
| Response sent |  |  |  | URL Request |  |
|  |  |  |
| from the |  |  |  | sent through the |  |
| Servlet |  |  |  | browser |  |
|  |  |  |  |



SERVLETS

AT THE

SERVER

SIDE

|  |  |  |  |
| --- | --- | --- | --- |
| Reply from the |  | Verifying or |  |
|  |  |
|  | updating the |  |
| Database |  |  |
|  | database through a |  |
| according to the |  |  |
|  | statement |  |
| Statement |  |  |
|  |  |  |

DATABASE

URL pattern represents how the requests are flowing through one layer to another layer and how the responses are getting by other layers to presentation layer through server in architecture diagram.

**3.5 FEASIBILITY STUDY**

**Feasibility Study:**

Preliminary investigation examines project feasibility; the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All systems are feasible if they are given unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

* + Technical Feasibility
  + Economic Feasibility
  + Operation Feasibility

**3.5.1TECHNICAL FEASIBILITY**

The technical issue usually raised during the feasibility stage of the investigation includes the following:

* Does the necessary technology exist to do what is suggested?
* Do the proposed equipment’s have the technical capacity to hold the data required to use the new system?
* Will the proposed system provide adequate response to inquiries, regardless of

the number or location of users?

* Can the system be upgraded if developed?

Are there technical guarantees of accuracy, reliability, ease of access and data security?

**3.5.2. OPERATIONAL FEASIBILITY**

**User-friendly**

Customer will use the forms for their various transactions i.e. for adding new routes, viewing the routes details. Also the Customer wants the reports to view the various transactions based on the constraints. These forms and reports are generated as user-friendly to the Client.

**Reliability**

The package wills pick-up current transactions on line. Regarding the old transactions, User will enter them in to the system.

**Security**

The web server and database server should be protected from hacking, virus etc.

**Portability**

The application will be developed using standard open source software (Except Oracle) like Java, tomcat web server, Internet Explorer Browser etc. these software will work both on Windows and Linux operating system. Hence portability problems will not arise.

**Availability**

This software will be available always.

**Maintainability**

The system uses the 2-tier architecture. The 1st tier is the GUI, which is said to be front-end and the 2nd tier is the database, which uses **Oracle10g**, which is the back-end.

The front-end can be run on different systems (clients). The database will be running at the server. Users access these forms by using the user-ids and the passwords.

**3.5.3 ECONOMIC FEASILITY**

The computerized system takes care of the present existing system’s data flow and procedures completely and should generate all the reports of the manual system besides a host of other management reports.

It should be built as a web based application with separate web server and database server. This is required as the activities are spread throughout the organization customer wants a centralized database. Further some of the linked transactions take place in different locations.

Open source software like TOMCAT, JAVA and Oracle is used to minimize the cost for the Customer.

**CHAPTER 4: SOFTWARE REQUIREMENT SPECIFICATION**

Software Requirements Specification provides an overview of the entire project. It is a description of a software system to be developed, laying out functional and nonfunctional requirements. The software requirements specification document enlists enough and necessary requirements that are required for the project development. To derive the requirements we need to have clear and thorough understanding of the project to be developed. This is prepared after the detailed communication with project team and the customer.

**4.1 PURPOSE OF SRS**

* Communication between customer, analyst, system developers, maintainers
* Contract between Purchaser and Supplier
* Firm foundation for design phase
* Support system testing activities
* Support project management and control
* Controlling the evolution of the system

**4.2 USER CHARACTERISTICS**

* USER CUSTOMER AND MARKETING PERSONNEL- The goal of this set of audience in referring to the srs is to ensure that the system as described will met their needs
* SOFTWARE DEVELOPERS- It refers to the srs document to make sure that they are developing exactly what is required by the customer
* TEST ENGINEERS-Their goal is to ensure that requirement are understandable from a functionality point of view, so that they can test the software
* USER DOCUMENTATION WRITERS-their goal in reading srs is to ensure that they understand the feature of the product well enough to be able to write the user’s manual
* PROJECT MANAGERS- They want to ensure that they can estimate the cost of project
* MAINTENANCE ENGINEERS- The srs help maintenance engineers to understand the functionality of the system

**4.3 REQUIREMENTS SPECIFICATION**

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the design analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely with the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use. The requirement specification for any system can be broadly stated as given below:

**4.3.1. FUNCTIONAL REQUIREMENTS SPECIFICATION**

* Transformations (input, processing, output)
* Requirements for sequencing and parallelism(dynamic requirements)
* Data
* Stored data
* Transient data
* Exception handling

**Admin Module**

The admin module will be used by the administrator of this portal, admin can accept or reject the requests from the bank managers, and also admin can accept or reject the requests from the users. The requests are in the form of bank registration, customer registration. After accepting the request from bank managers, admin need to add the bank in the so that it is available for the user. This module is having following functionalities.

* **Pending User Requests**

By using this functionality admin give accesspermission to all users who are registered in this portal.

* **Pending Bank Manager Requests**

By using this functionality admin give accesspermission to all bank managers who are registered in this portal.

* **See Status**

By using this functionality admin see the status of user and manager in order to make change.

* **See list of banks**

By using this functionality admin can see all bank that are registered on the system

* **Add New Bank**

By using this functionality admin add the registered bank to the list to make it available for the user.

**User Module**

This module describes all about users, a user can do operations like add a new account, view the account information, transfer amount from one account to other account, update personal details, make bill payment, see the transaction and billing details. This module consists following functionalities:-

* **Add New Account**

By using it user add a new account and make an account request to the manager.

* **Delete Account**

By using this functionality user delete the account, if he/she does not require in future.

* **View Account Information**

By using this functionality user view all his/her accountdetails, this can be viewed by users who are having account in any bank.

* **Update Personal details**

By using this functionality user change their personal details, giving some new/updated data.

* **Change Password**

By using this functionality user change the login password and the transaction password in order to make his account more secured.

* **Transfer Amount**

By using this functionality user transfer money from his/heraccount to other accounts of same bank or other banks.

* **Bill Payment**

By using this functionality user make bill payment, by using the different bank accounts.

* **Transaction Enquires**

By using this functionality user is able to see the different transaction made. Each transaction is having a ‘transaction id’ along with the other details.

* **Billing Enquires**

By using this functionality user is able to see the different Bill payment made. Each transaction is having a ‘billing id’ along with the other details.

* **Product & Services and Contact details**

In any situation user can ask for help and product & services tab keep him/her updated about the different facilities of bank.

**Bank Admin Module:**

This module describe all about the bank manager. By using this module bank manager can view details of user, they can go for any transactions of their user and also they can give access permeations to all users of that bank. This module consists following functionalities:-

* **User Details**

By using this functionality bank manager get the entireuser list with its status and user id.

* **Accounts Details**

By using this functionality bank manager get the entire account list with the user detail of that account.

* **Transaction Details**

By using this functionality bank manager get the details of different transactions made by the user either to their accounts or by their accounts.

* **Billing Details**

By using this functionality bank manager get the details of different bill payment made by the user by their accounts.

* **New Account Request**

By using this functionality bank manager accept or reject account request, requested by the user.

**Software Engineering Methodology**

Object Oriented Analysis and Design (OOAD Standards)

**4.3.2 NON-FUNCTIONAL REQUIREMENTS**

* The system should be able to interface with the existing system
* The system should be accurate
* The system should be better than the existing system
* The system should be available for 24 X 7
* Flexible service based architecture will be highly desirable for future extension

**INTERFACES**

1. **USER INTERFACE**

* Login page (secured with password)
* For new user there will be the registration page to create an account
* New user must have an account in any of the bank linked
* In the first registration page new user has to give details regarding to account and has to choose one username
* In the second registration page new user has to give personal details
* In the last registration page user has to enter the password and transaction password

1. **BANK MANAGER INTERFACE**

* Login page (secured with password)
* For new bank there will be the registration page to create an account that will be handled by bank manager
* In the first registration page bank manager has to give one username and bank name
* In the second registration page bank manager has to give personal details along with password and security question.

1. **HARDWARE INTERFACES:**

The logical and physical characteristic of each interface between the software product and the hardware components of the system is same that require for any online project system.

1. **SOFTWARE INTERFACES:**

Software interface is very normal and easy so it requires less memory and requirement. There should be recovery for Data or important information.

1. **COMMUNICATIONS INTERFACES:**

The requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Communication standards that will be used, such as FTP or HTTP. Communication security or encryption issues will handle by using java scripts.

**CHAPTER 5: SYSTEM SPECIFICATIONS**

**5.1 HARDWARE REQUIREMENTS**

|  |  |  |
| --- | --- | --- |
| Processor | : | Pentium IV or above |
| Hard Disk | : | 40 GB |
| RAM | : | 512 MB |
| CPU Speed | : | 2.6 GHz |
| Monitor | : | EGA/SVGA (display), 800x600 |
| Mouse | : | PS2/ Optical Mouse |

|  |  |  |
| --- | --- | --- |
| Operating System | : | Windows XP/7/8 |
| IDE Tool | : | My Eclipse 8.6 |
| Framework | : | Struts 2 |
| Client Side Scripting | : | JavaScript, HTML, CSS |
| Server Side Scripting | : | Java(JDK 6.0) |
| Programming Language | : | Java(JDK 6.0) |
| Database | : | Oracle 10g XE |
| Web Requirements | : | JSP, Servlet |
| Web Server | : | Apache TomCat 6.x |
| Web Browser | : | Any Browser |

**5.2 SOFTWARE REQUIREMENTS**

**CHAPTER 6: TECHNOLOGICAL DETAILS**

**6.1 MyEclipse IDE:**

Welcome to MyEclipse Tomcat 6 Server. In MyEclipse 6.0 an embedded Tomcat 6 server was added to help developers get up and running immediately with their development work and not need to scour the net for an application server to install, download it, install it and configure it. Instead with MyEclipse 6.0 you’ll be able to immediately deploy your web project, run and debug it with no additional configuration.

System Requirement:

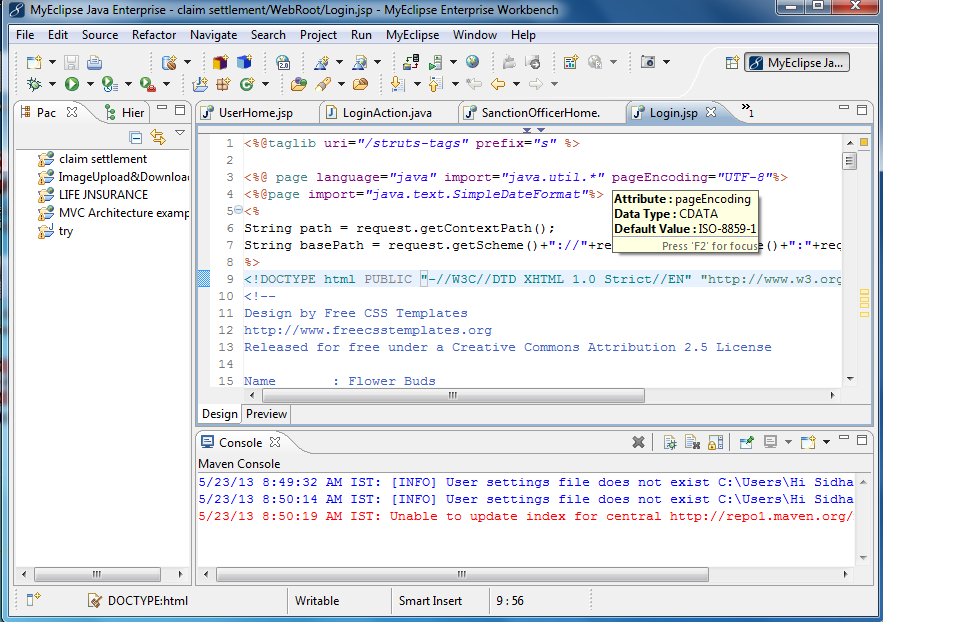
This tutorial was written using MyEclipse 6.0 GA(All in One installer).

**NOTE:** In MyEclipse 6.5 and beyond the JAX-WS reference implementation (Metro 1..1) libraries were added to the embedded MyEclipse Tomcat server to make development and testing of JAX-WS much easier. Deploying to an external Tomcat server or any other non-Java EE 5 compliant server can require additional adjustments to the build path of any JAX-WS project, please see the JAX-WS Tutorial for more information covering this.

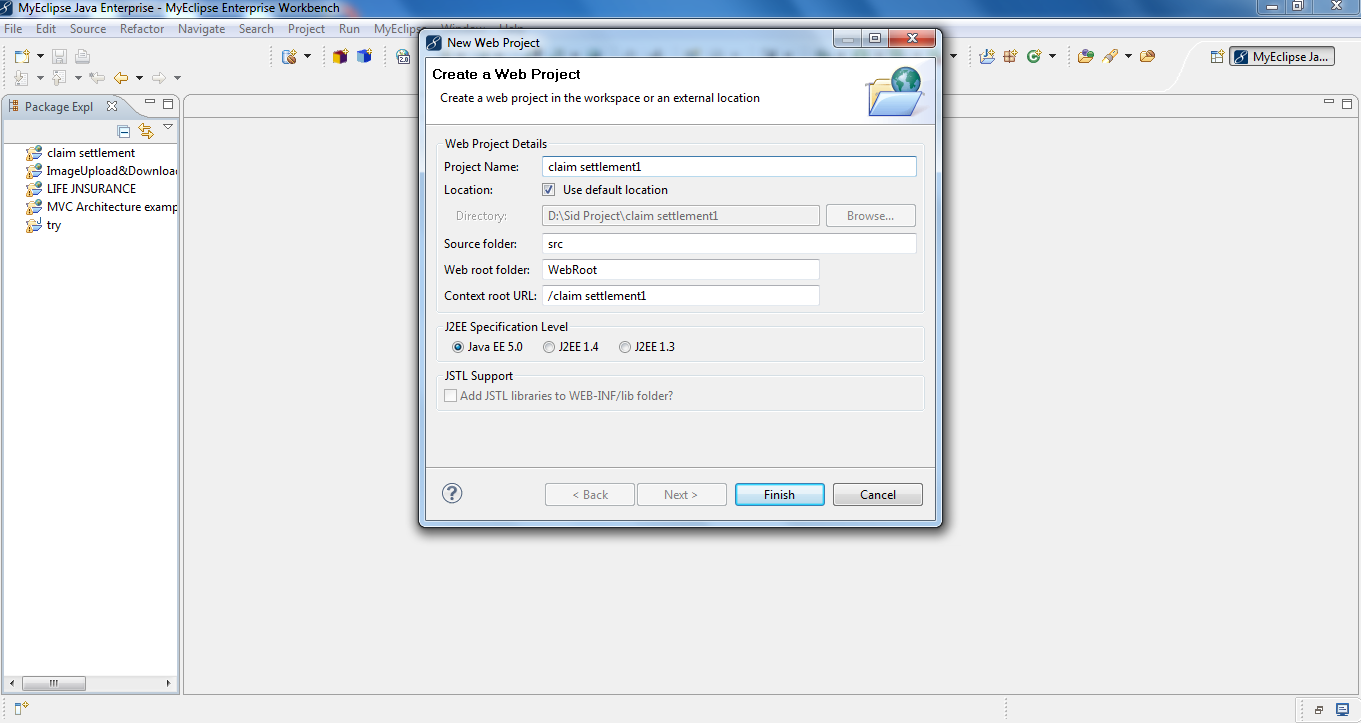
MyEclipse Tomcat will work with all Java 5 JDKs, however, if you wish to use a Java 6 JDK to run MyEclipse Tomcat, please use JDK 6 update 6 (1.6.0\_04) or above to avoid JAX-WS API compatibility issues.

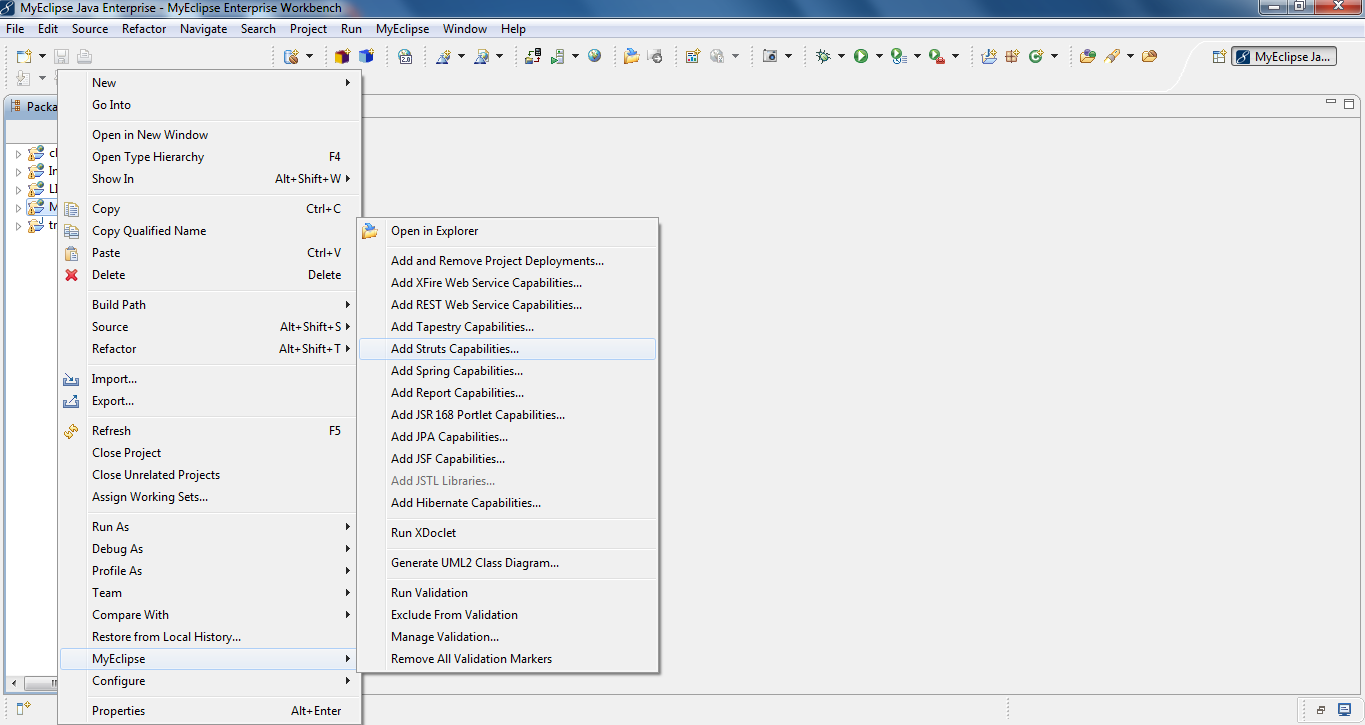
If you are using a newer version of MyEclipse and notice portions of this tutorial looking different than the screens you are seeing, please let us know and we will make sure to resolve any inconsistencies.

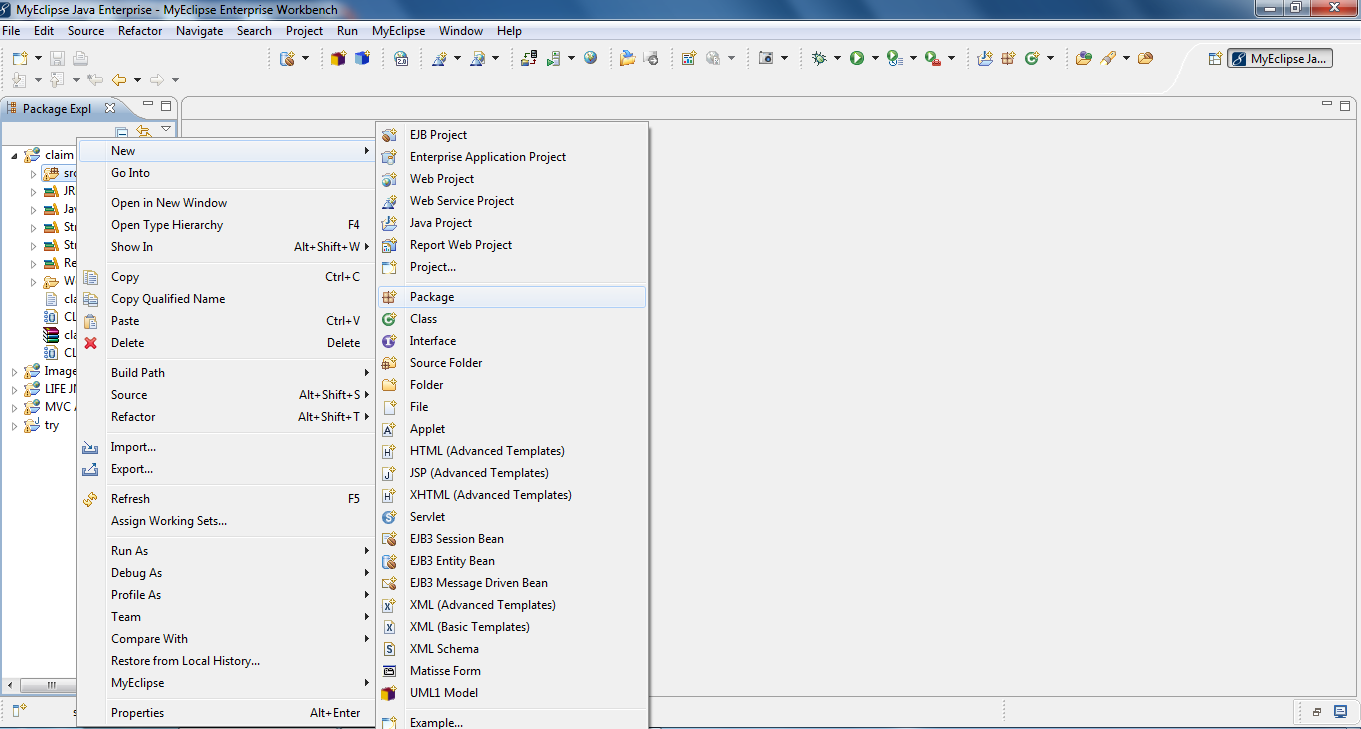
Getting Started

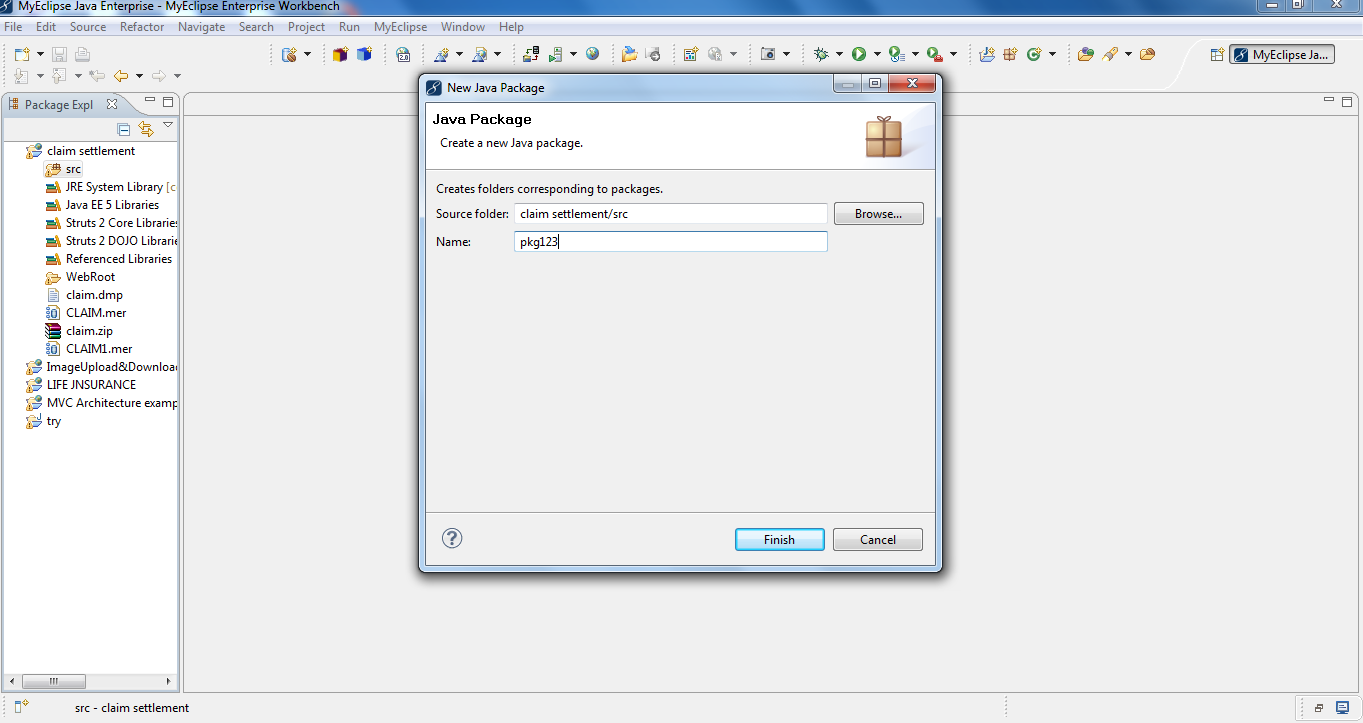


Creating a Web Project



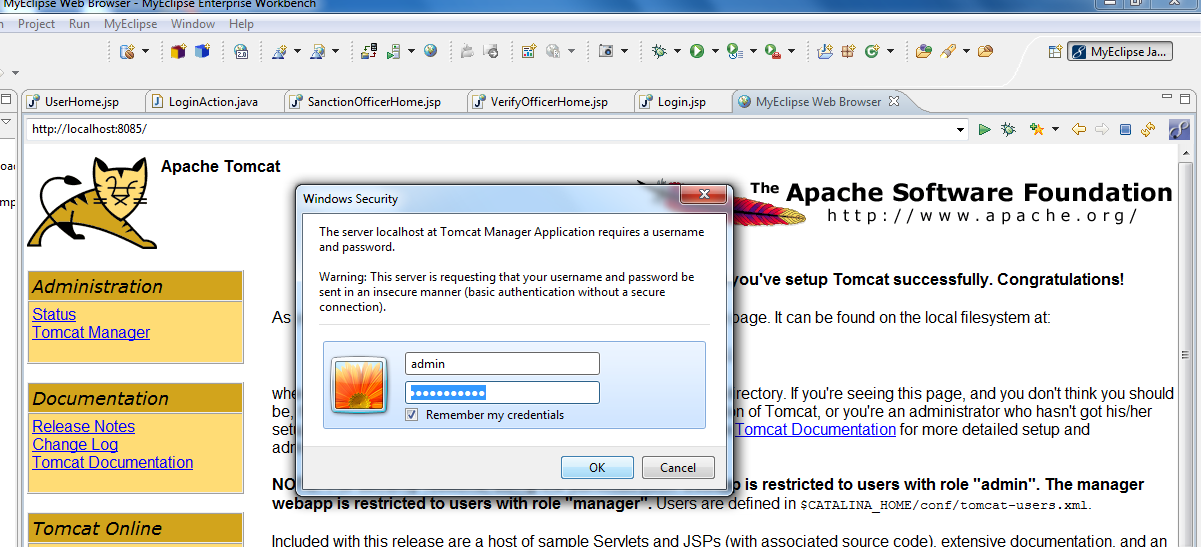






Deploying a Web Project

Running the Web Project



Conclusion:

Hopefully from this simple guide you can begin to get an idea of idea of how fast you can get up and running with the embedded Tomcat server that now ships with MyEclipse 6.0 and later when working on your own projects.

* 1. **ORACLE 10g XE:**

Oracle Database XE is easy to install and easy to manage. With Oracle Database XE, you use the Database Home Page, an intuitive browser-based interface, to administer the database; create tables, views, and other schema objects; import, export, and view table data; run queries and SQL scripts; and generate reports.

Oracle Database XE includes Oracle HTML DB2.1, a declarative, graphical development environment for creating database-centric Web applications. In addition to HTML DB 2.11, you can use popular Oracle and third-party languages and tools to develop your Oracle Database XE application.

* + 1. **Application Express Architecture :**

Oracle Application Express consists of a metadata repository that stores the definitions of applications and an engine (called the Application Express Engine) that renders and processes pages. It lives completely within your Oracle database. It is comprised of nothing more than that in tables and large amounts of PL/SQL code. The essence of Oracle Application Express is approximately 215 tables and 200 PL/SQL objects containing 3, 00, 000 + lines of codes.

The Application Express Engine performs:

* Session state management
* Authentication services
* Authorization services
* Page flow control
* Validations processing
* Rendering and Page processing

The asynchronous session state management architecture ensures the minimal CPU resources are consumed. The browser sends a URL request that is translated into the appropriate Oracle Application Express PL/SQL call. After the database processes PL/SQL, the results are relayed back to the browser as HTML. This cycle happens each time one either requests or submits a page. The session state is managed in the database and does not use a dedicated database connection. Each page view results in a new database session, thus database resources are only consumed when the Application ExpressEngine processes or renders a page.

The Application Express Engine is accessed from a Web browser through the Oracle HTTP Server (Apache) and mod\_plsql. Applications are rendered in real time from the metadata repository stored in database tables. Building or extending applications doesn’t cause code to be generated; instead metadata is created or modified.

Application Express Engine

Web Browser

mod\_pls

Apache

Oracle database

Oracle Application Express using the Oracle HTTP Server (Apache) and mod\_plsql with Oracle Database 11g, you can replace the Oracle HTTP Server (Apache) and mod\_plsql from the architecture with the Embedded PL/SQL Gateway (EPG). The EPG provides the Oracle Database with a web server and also the necessary infrastructure to create dynamic applications. The EPG runs in the XML Database HTTP Server, part of the Oracle Database and includes the core features of mod\_plsql, but doesn’t require the Oracle HTTP Server powered by Apache. Oracle Database 10g Express Edition (XE) also utilizes the EPG.

Embedded PL/SQL Gateway

Application ExpressEngine

Web Browser

EGP

OracleDatabase

Reporting is an important aspect, especially when it comes to customer satisfaction. The report created by this IDE can be conveniently manipulated. While sorting, searching, limiting the

records to a page and pagination are all possible. The user interface created has an aesthetically pleasing look.

There are some database terminologies are used they are as follows:

* 1. Tables
  2. Records
  3. Fields
  4. Primary key
  5. Foreign key
  6. Master table

**Description**:--

**Table:** - A database contains data, which are stored and arranged in tables. Tables are known as relations. Here data are arranged in row wise or column wise.

**Records:** - A table contains data arranged in rows every row is a separate record.

**Fields: -** The data in every row or record consists of several columns. These columns are called fields (while designing the database fields are referred as properties or attributes).

**Primary key:** - This is the field which contains the data that will uniquely define each record. The primary key values cannot be repeated across the table. They cannot be the blank.

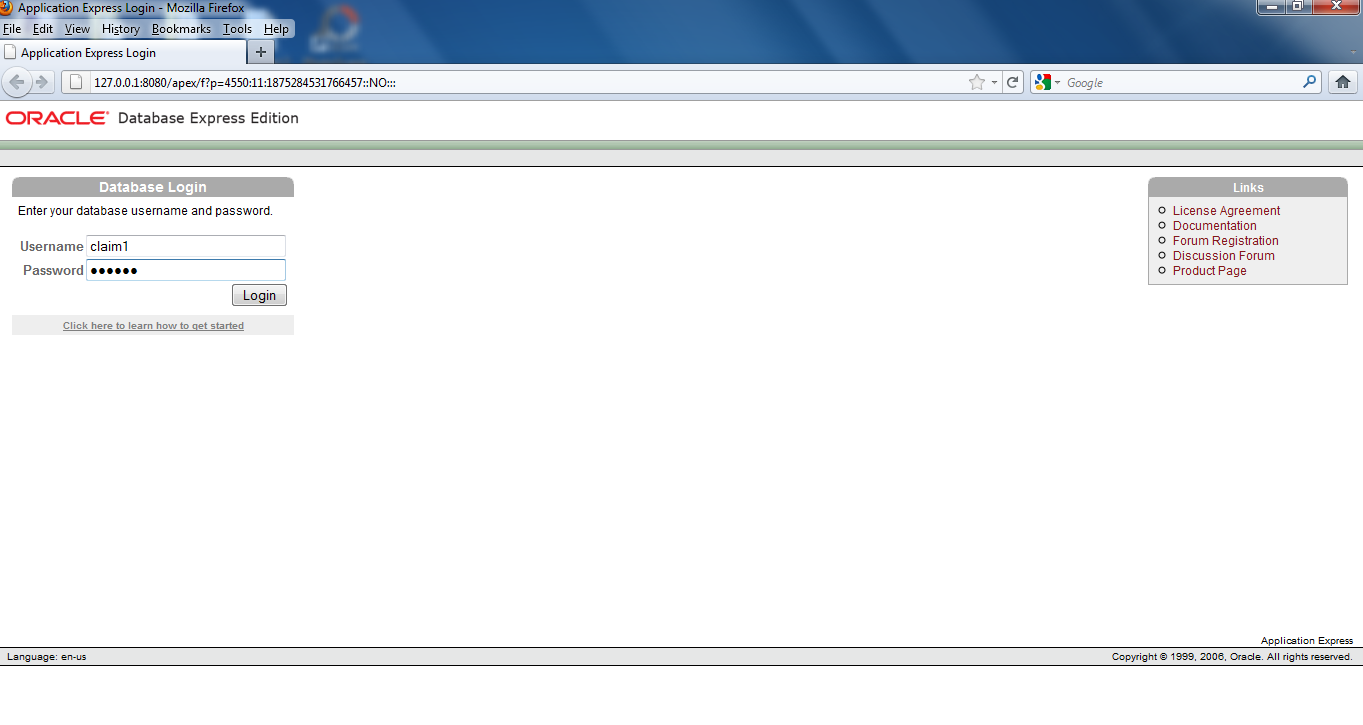
**Foreign key: -** This is the field which contains the data that exists in the primary key field of the master table. Any value inside the foreign key field must be taken from the corresponding primary key.

* + 1. **Getting into the XE IDE :**

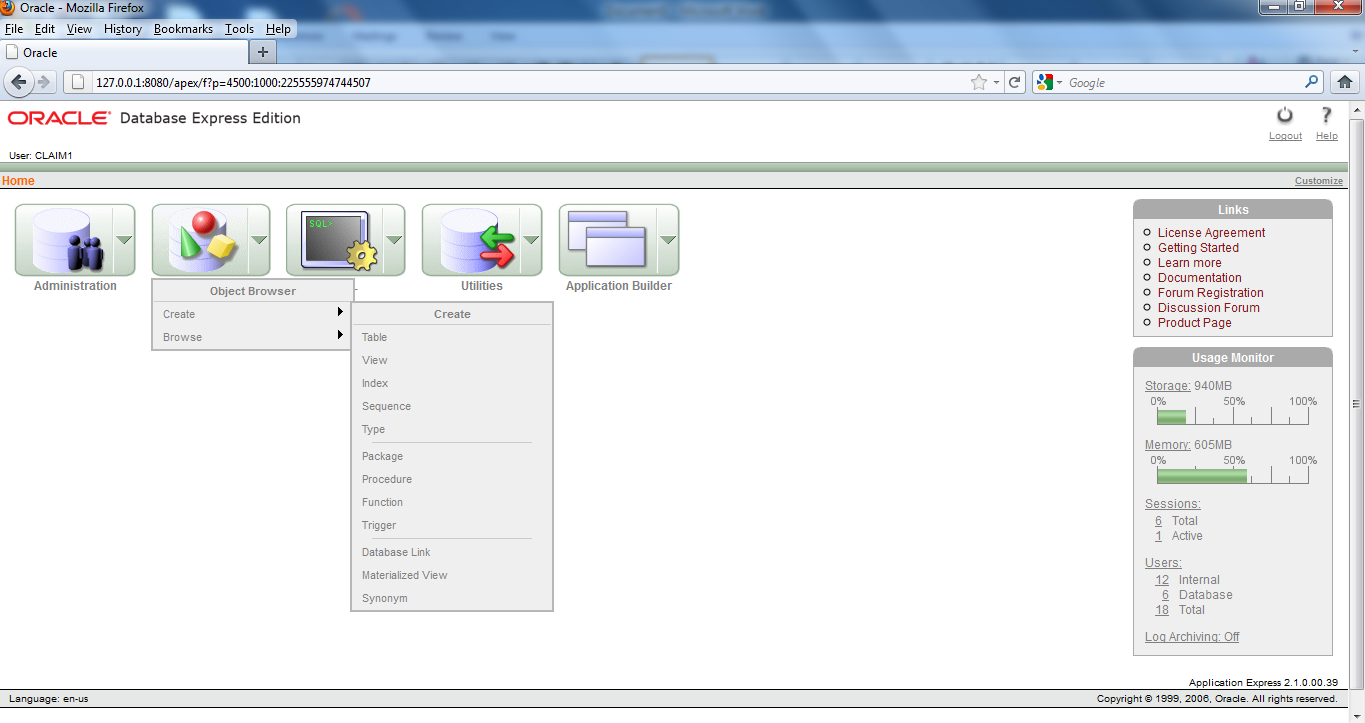
Unlock the vendor’s account and if you don’t have, please review the following tutorial. Go to the shortcut:

* Click on the Start menu
* Click on All Programs
* Click on Oracle 10g Express Edition
* Now click on “Go to Database Home Page”. which opens up the Database Login page.

Now you will login into the vendor’s database with the username: claim1 and password: claim1 as shown in the next picture.

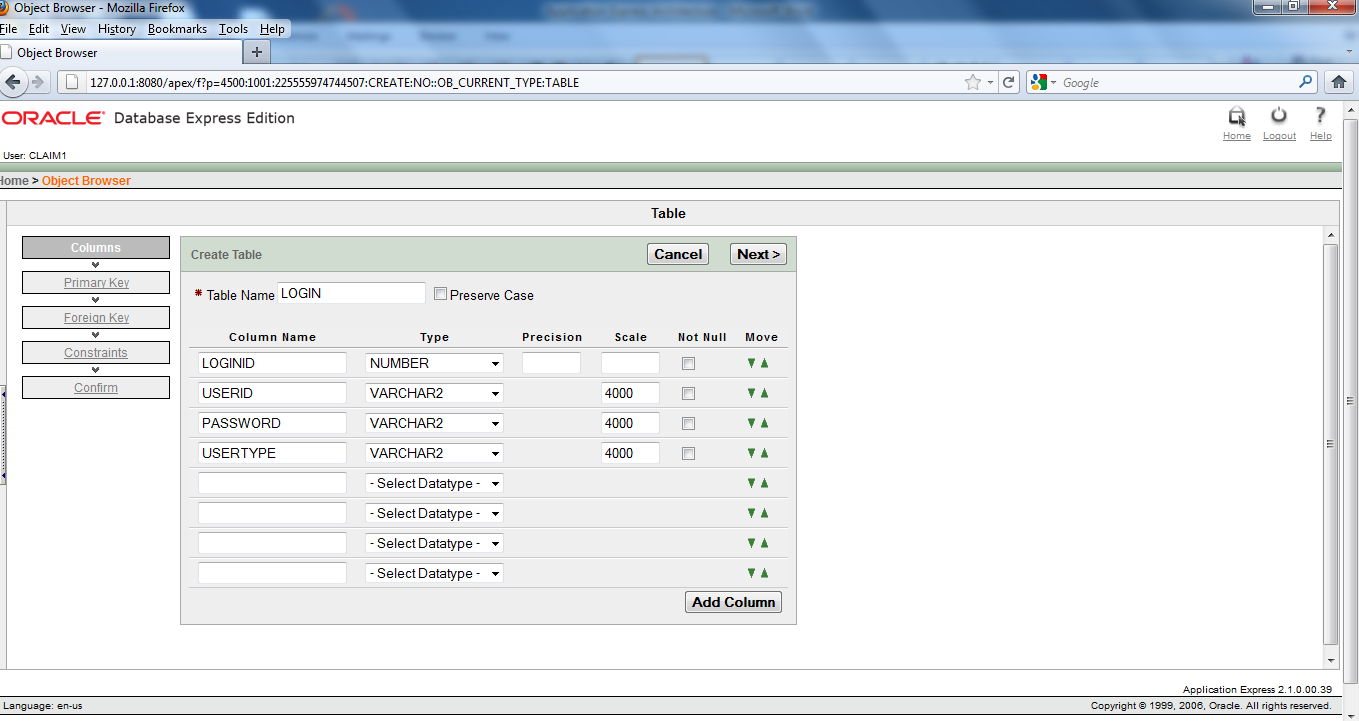


In the new window after login into the database, click on Object Browser.Click on the down arrow to reveal various object related items. You have an option to Create and Browse. Now click on Create arrow and then click on Table in order to create the tables required as per our project.

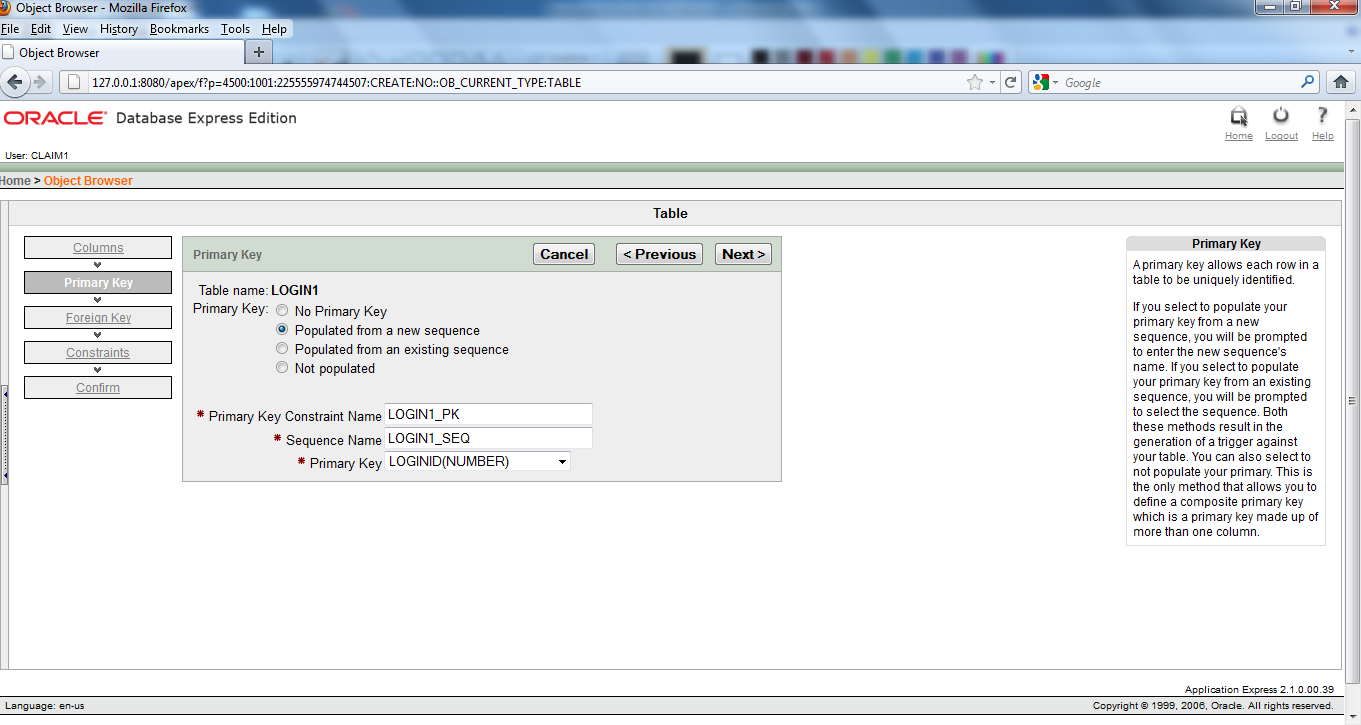


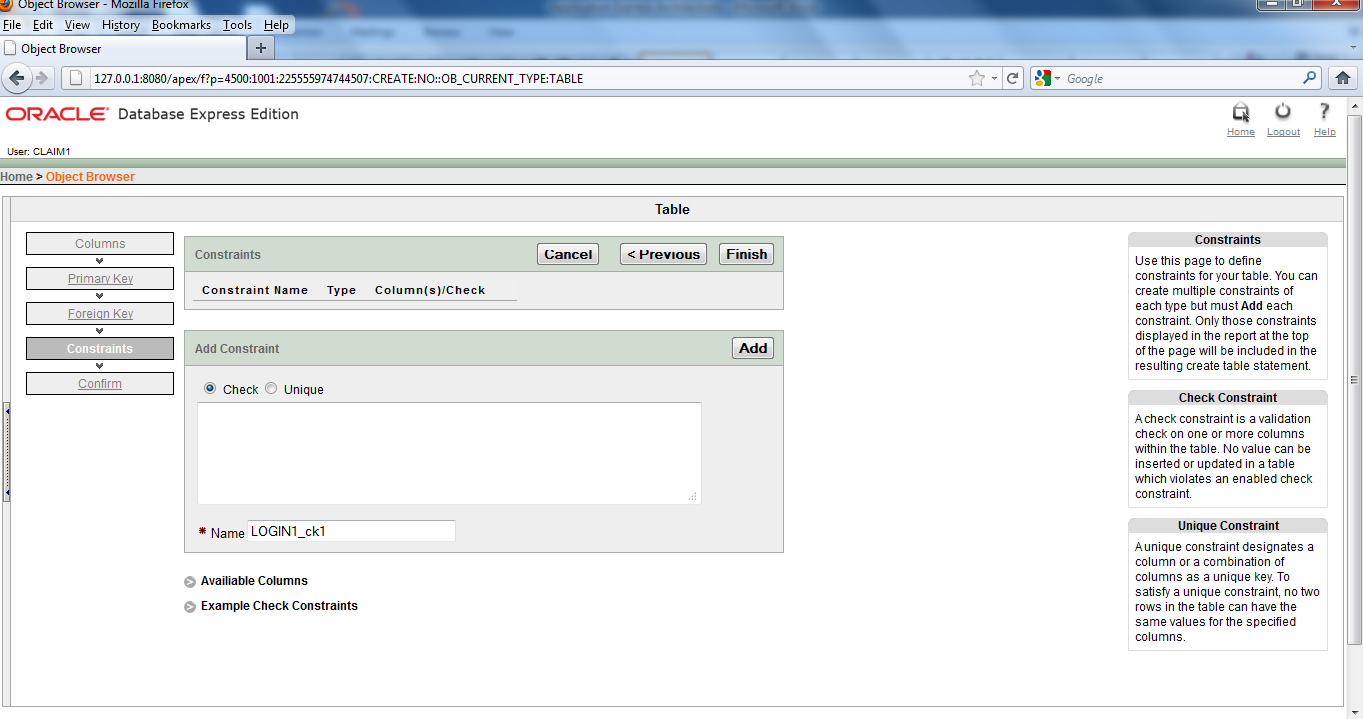
Now in the new window you have to provide your Table details by following procedure:

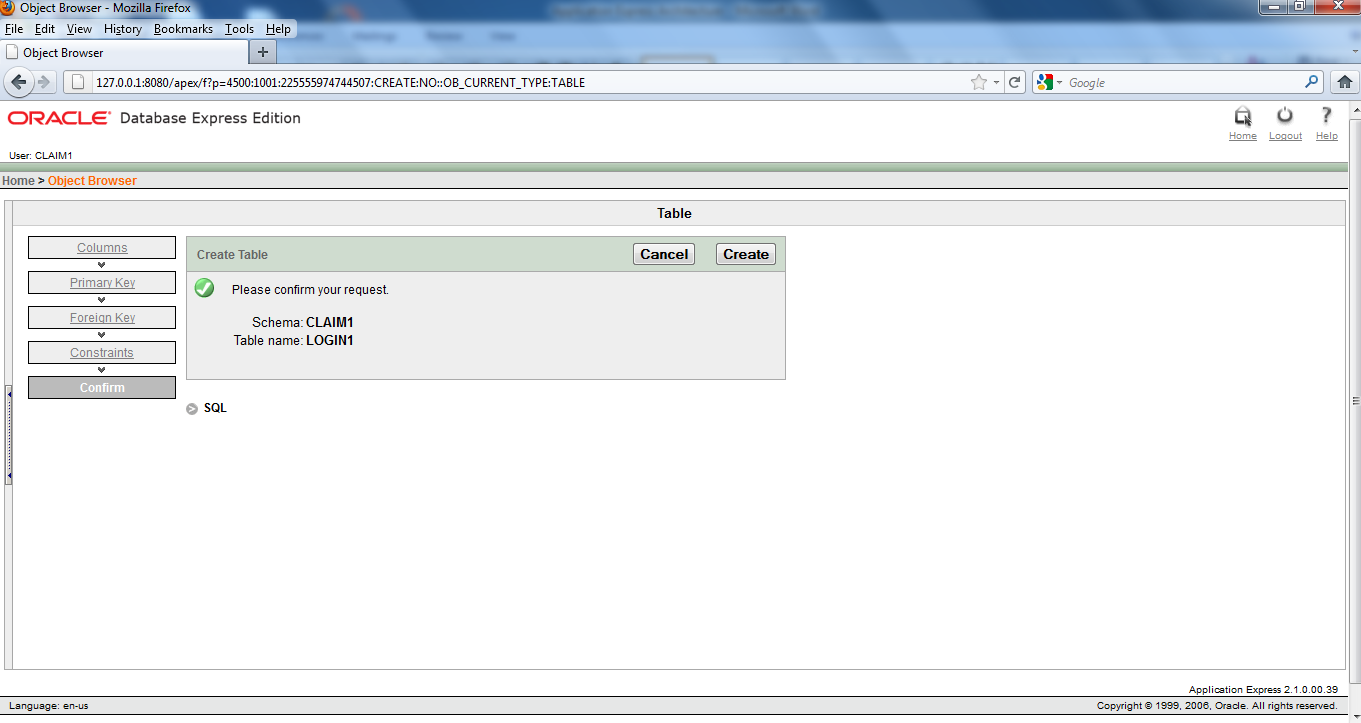
* Give your table name first (LOGIN)
* Now provide whatever columns and their type you want to have in your table
* Now click on Next after providing all the information

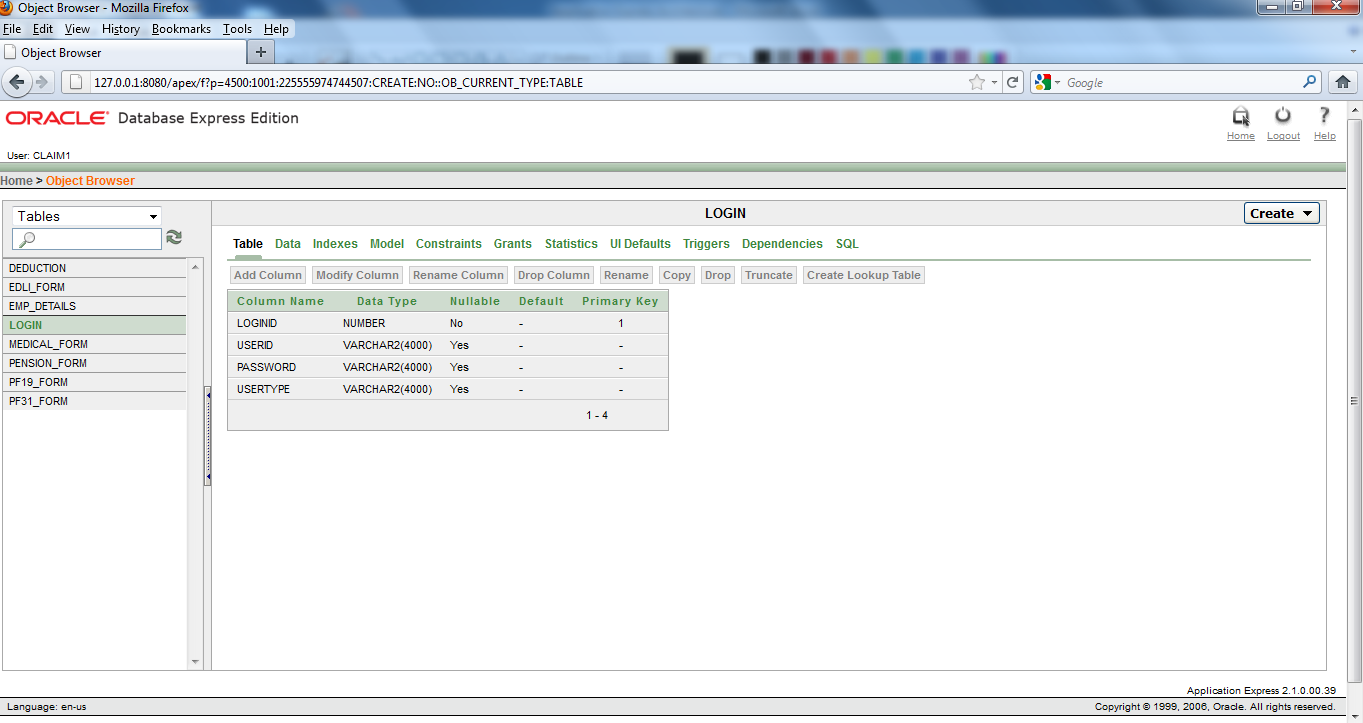


Now in the new window you have to choose Primary Key as” Populated from a new sequence” and give your \*Primary Key as LOGINID(NUMBER). Now click on ‘Next’ twice. Then click on Finish and then click on Create. After that in a new window you can see your table with its columns and column types in a good looking format.









In this way you have to create 8 tables one LOGIN table, one DEDUCTION table to show the deducted amount from the employee’s provident fund account, one EMP\_DETAILS table to show the details of the current employees and more 5 tables for the 5 claim forms such asPF19\_FORM,PF31\_FORM, MEDICAL\_FORM, PENSION\_FORM, EDLI\_FORM, as shown in the above figure.

* 1. **MVC**

The key motivation behind the MVC approach is the desire to separate the code that creates and manipulates the data from the code that presents data.

MVC- Model View Controller architecture is all about dividing application components into three different categories Model, View, Controller. Components of the MVC architecture have unique responsibility and each component is independent of the other component. Changes in one component will have no or less impact on other component.

Responsibilities of these components are:

MODEL: Model is responsible for providing data from the database and saving the data into the data store. All the business logics are implemented in the Model. Data entered by the user through View are checked in the Model before saving into the database. Data access, Data validation and the Data saving logic are part of the Model.

VIEW: View represents the user view of the application and is responsible for taking the input from the user, dispatching the request to the Controller and then receiving response from the Controller and displaying the result to the user. HTML, JSPs, Custom Tag Libraries and Resource files are the part of View component.

CONTROLLER: Controller is the intermediary between the Model and View. Controller is responsible for receiving the request from the client. Once the request is received from client it executes the appropriate business logic from the Model and then produces the output to the user using the View component. ActionServlet, Action, ActionForm and struts-config.xml are the part of Controller.

* + 1. **MVC Architecture**

**Model – 1 Architecture**

In Model- 1 architecture a request is made to a JSP or servlet and then that JSP or servlet handles all the responsibilities for request, including processing the request, validating data, handling the business logic, and generating a response.

**Request**

Request Processing

Data Validation

Business Logic

Data Manipulation

Response Generation

**BROWSER**

**Response**

**Database Services**

**Model – 2 Architecture**

In the MVC architecture, a central servlet, known as the Controller, receives all requests for the application. The Controller then processes the request and works with the Model to prepare any data needed by the view and forwards the response to the browser.

**BROWSER**

**Controller (Servlet)**

**Request Processing**

**Data Validation**

Dta Validation

**Model**

**Business Logic**

**Data Manipulation**

**View (JSP)**

**Report Generation**

Database Services

* 1. **STRUTS**

The remarkable thing about Struts is its early adoption, which is obviously a testament to both its quality and utility. The java community, both commercial and private, has really gotten behind Struts.

It is currently supported by all of the major application servers including BEA, Sun, Caucho, and of course Apaches Jakarta-Tomcat.

**FORMS**

Forms are the objects that expose properties, which define their appearance and methods and define the interaction with the user. By setting the properties of the form and writing JSP code to respond to its events, we customize the object to meet the requirement of the application.

## CONTROLS

Controls are objects that are contained within from objects. Each type of controls has its own set of properties, methods and events that makes it suitable for a particular purpose. Some of the controls are best suited for entering or displaying text. Other controls let you access to other application and process data as if the remote application was part of your code.

* + 1. Eeee

**6.4.1 Architecture of Struts**



**Process flow:**

**Web.xml :** Whenever the container gets start up the first work it does is to check the web.xml file and determine what struts action Servlets exist. The container is responsible for mapping all the file request to the correct action Servlet.

**A Request:** This is the second step performed by the container after checking the web.xml file. In this the user submits a form within a browser and the request is intercepted by the controller.

**The Controller**: This is the heart of the container. Most Struts application will have only one controller that is ActionServlet which is responsible for directing several Actions. The controller determines what action is required and sends the information to be processed by an action Bean. The key advantage of having a controller is its ability to control the flow of logic through the highly controlled, centralized points.

**Struts-config.xml** : Struts has a configuration file to store mappings of actions. By using this file there is no need to hard code the module which will be called within a component. The one more responsibility of the controller is to check the struts.config.xml file to determine which module to be called upon an action request. Struts only reads the struts.config.xml file upon start up.

**Model:** The model is basically a business logic part, which takes the response from the user and stores the result for the duration of the process. This is a great place to perform the preprocessing of the data received from request. It is possible to reuse the same model for many page requests. Struts provide the Action Form and the Action classes, which can be extended to create the model objects.

**View:** The view in struts framework is mainly a jsp page, which is responsible for producing the output to the user. Through the view Layer the user will interact with the System.

**Struts tag libraries:**These are struts components helps us to integrate the struts framework within the project's logic. These struts tag libraries are used within the JSP page. This means that the controller and the model part can't make use of the tag library but instead use the struts class library for strut process control.

**Property file:** It is used to store the messages that an object or page can use. Properties files can be used to store the titles and other string data. We can create many property files to handle different languages.  Business objects:  It is the place where the rules of the actual project exist. These are the modules that just regulate the day- to- day site activities.

**The Response:** This is the output of the View JSP object.

* 1. **JAVA Technology:**

Java has a profound effect on a network as Java expands the universe of objects that can move about freely in cyberspace. Java can be termed as any other object oriented language but the fundamental forces that necessitated the invention of java are security and portability, through other factors played an important role in modeling the final form of the language. Java can be described by the following buzzwords:-

1. Simple
2. Secure
3. Portable
4. Object-Oriented
5. Robust
6. Multithreaded
7. Architecture-neutral
8. Interpreted
9. High performance
10. Distributed
11. Dynamic
    * 1. **SERVLETS :-**

The Java web server is JavaSoft’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 confirm 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, pluggable 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:

1. They are faster and cleaner than CGI scripts
2. They use a standard API(the Servlet API)
3. They provide all the advantages of Java
   * 1. **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 ,JSP offers proven portability, open standards and a mature reusable component model. The JSP 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 :-**

1. **Portability:** JSP files can be run on any web server or web-enabled server that provides supports for them. Dubbed the JSP engine, this support involves recognition, translation and management of the JSP lifecycle and its interaction components.
2. **Components:** It was mentioned earlier that the JSP architecture can include reusable java components. The architecture also allows for the embedding of a scripting language directly into the JSP files. The components supported currently include Java Beans and Servlets.
3. **Processing:** A JSP file is essentially an HTML document with JSP scripting or tags. The JSP file has a JSP extension to the server as a JSP file. Before the page is saved, the JSP 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.
4. **Access models:** A JSP file may be accessed in atleast 2 different ways. A client’s request comes directly into a JSP 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 Java web server. At the server side Java Web Server receives the request and if it is a request for a JSP file server gives this request to the JSP engine.

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 Java web server and then it is transferred back to the result is given back to the Java web server and then it is transferred back to the client.

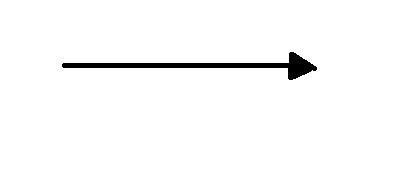
**CHAPTER 7: SYSTEM DESIGN**

**Systems design** is the process or art of defining the architecture,components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap and synergy with the disciplines of systems analysis, systems architecture and systems engineering.

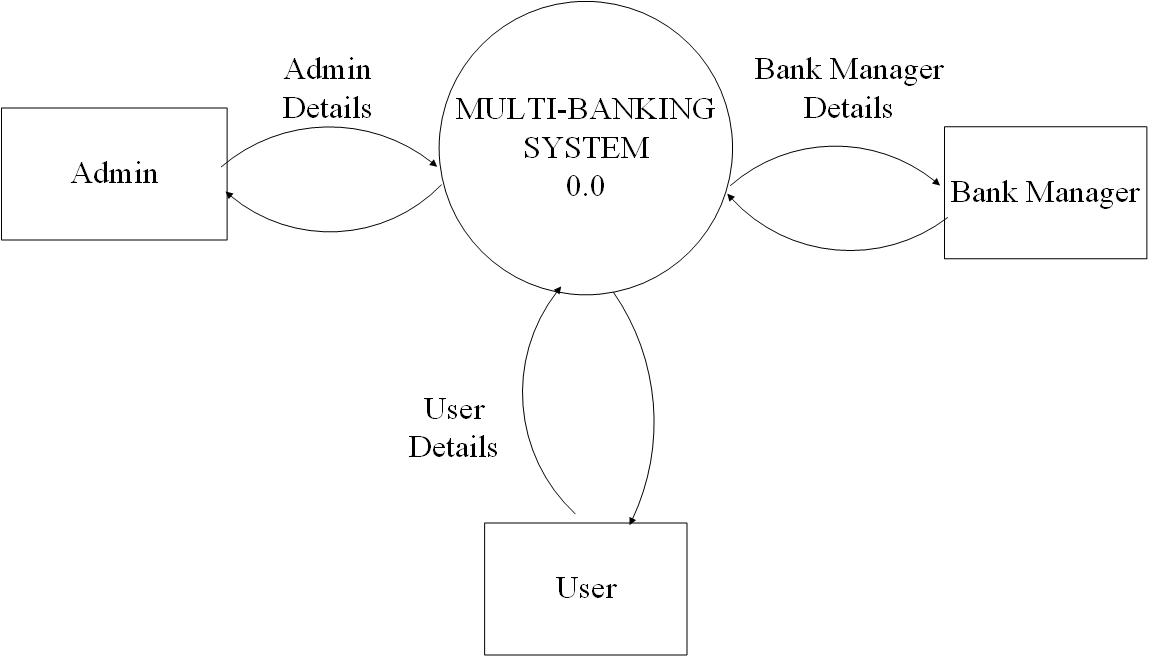
**7.1 DATA FLOW DIAGRAM**

Data Flow Diagram is a diagrammatic representation of data movement through a system- manual or automated- from inputs to ouputs through processing. The data flow diagram helps in analysis of the flow of data through a system and thus helps in identifying the system requirements.

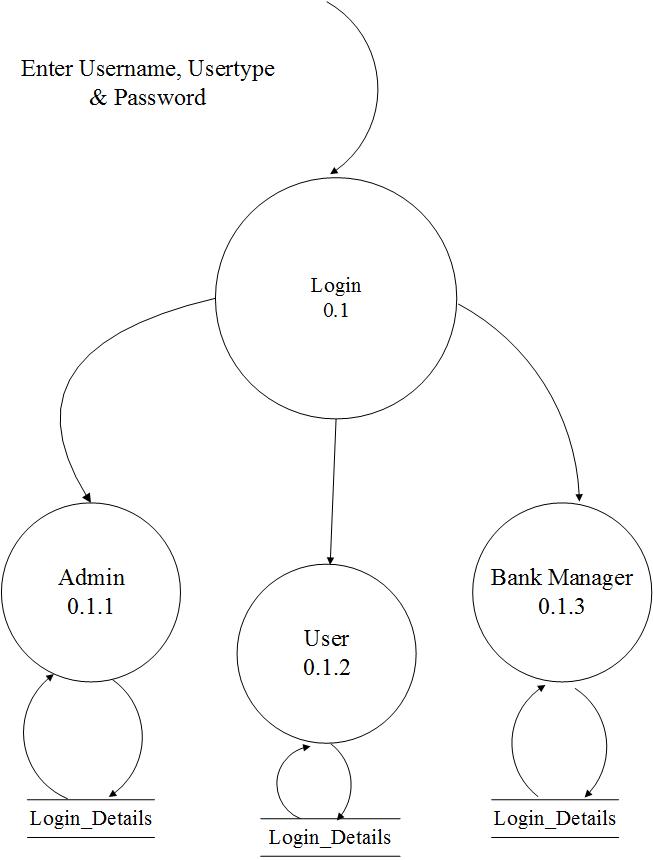
**7.1.1 NOTATIONS**

* Data flow shows the movement of data in a specific direction from the source to the destination. It represents a packet of data.
* Processes show the operations performed on data, which transform it from input to output.
* External entity is represented by a rectangle. The external entities are essentially those physical entities external to the software system which interact with the system by inputting data to the system or by consuming the data produced by the system.
* The output symbol is used when a hard copy is produced and the user of the copies cannot be clearly specified or there are several users of the output.
* Data stores are places where data are stored such as files and tables.

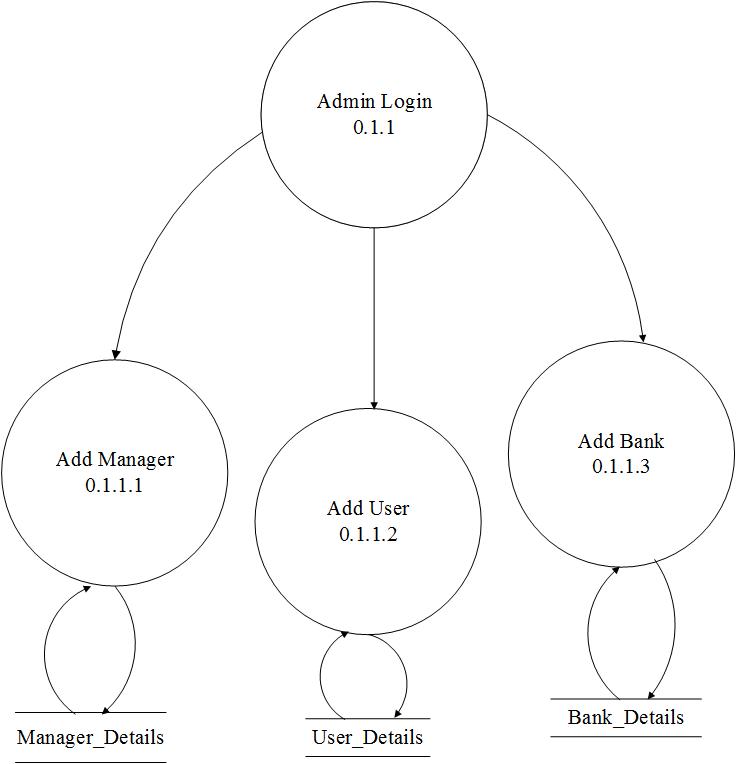
**LEVEL 0 (CONTEXT LEVEL DIAGRAM)**



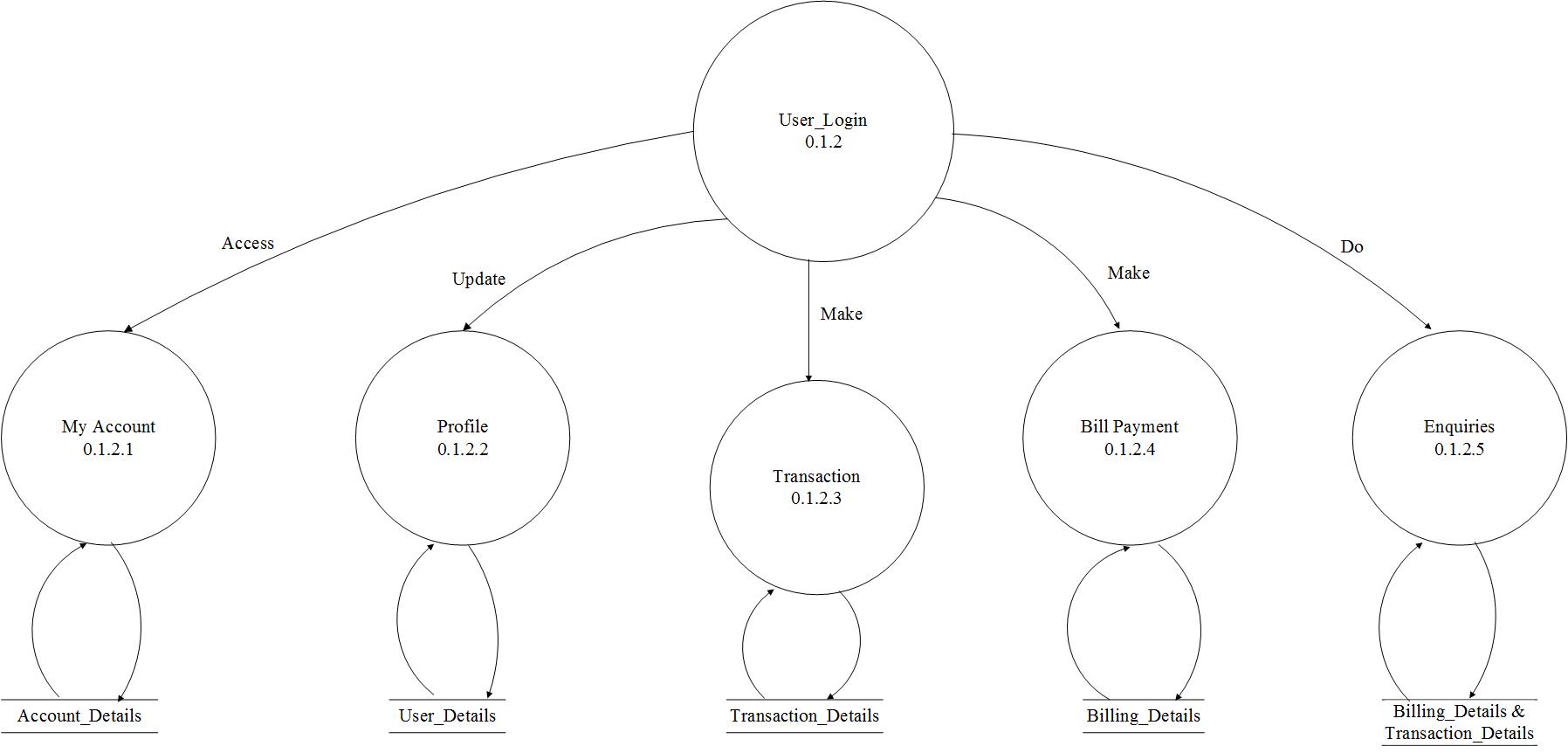
**LEVEL 1**



**LEVEL 2( ADMIN MODULE)**



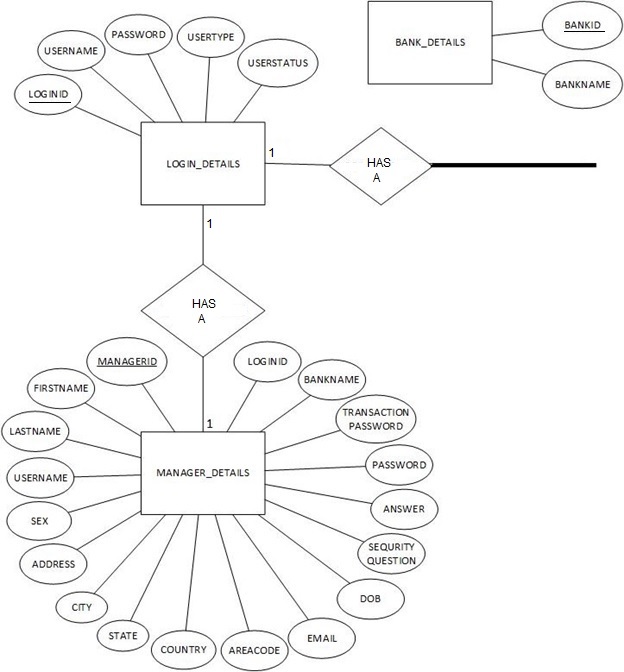
**LEVEL 2(USER MODULE)**

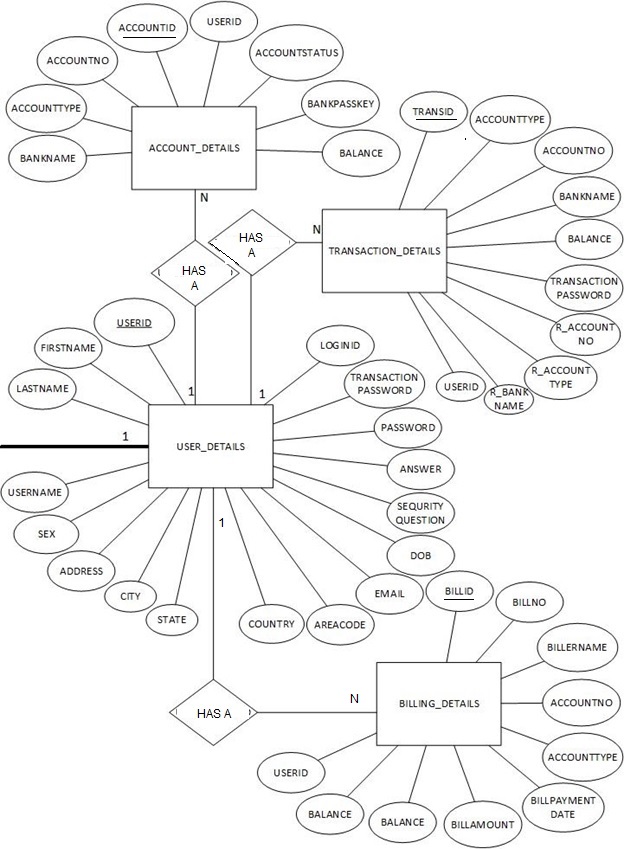


**LEVEL 2 (BANK MANAGER MODULE)**



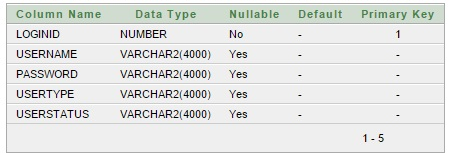
**7.2 ER DIAGRAM**





**CHAPTER 8: DATABASE DESIGN**

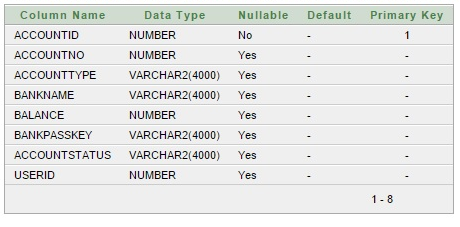
**TABLE- LOGIN\_DETAILS**



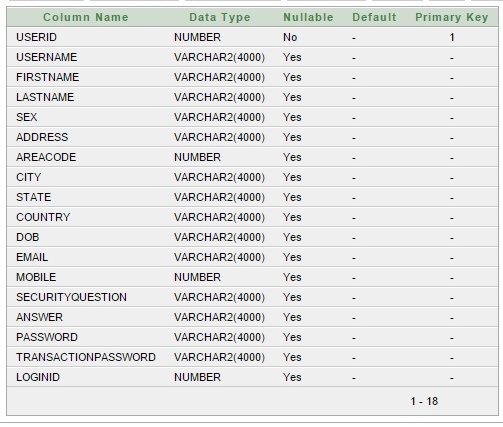
**TABLE- BANK\_DETAILS**



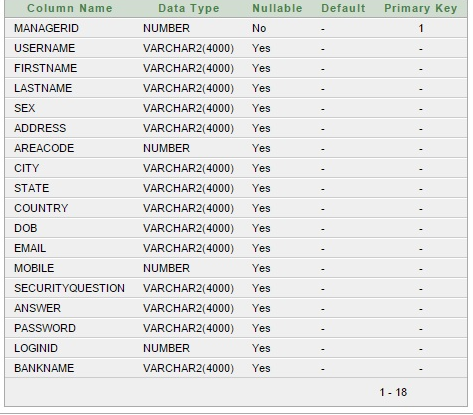
**TABLE- ACCOUNT\_DETAILS**



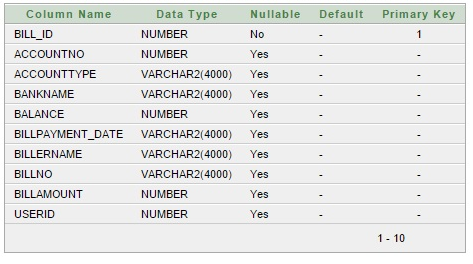
**TABLE- USER\_DETAILS**



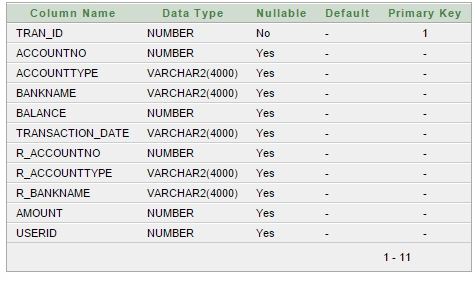
**TABLE- MANAGER\_DETAILS**



**TABLE- BILLING\_DETAILS**



**TABLE- TRANSACTION\_DETAILS**



**CHAPTER 9: SYSTEM TESTING**

Software Testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. Software Testing is one element of a broader topic that is often referred to as Verification and Validation. Verification refers to the set of activities that ensures that software correctly implements a specific function. Validation refers to a different set of activities that ensure that the software that has been built is able to meet customer requirement. Software testing is very much essential before actual implementation of the system.

**9.1 Testing Process:**

After successful completion of individual form, the whole system was run through a series of test to ensure the working of the system as a whole.The reliability of the system was systematically and carefully monitored at every stage of system development. The primary objective for test case design is to derive a set of tests that have the highest like hood for uncovering errors in the software. To accomplish these objective different categories of test case design techniques were used.The testing process, of the software, to uncover errors in the software, comprises of, testing techniques and testing strategies

The testing techniques applied to this software are

* White box Testing
* Black box Testing
* Testing GUIs

The Testing strategies adopted for this software are:

* + - Unit testing
    - Integration testing
    - Validation testing
    - System testing

**9.2Testing Objective**

Software testing is both a discipline and a process. It is a separate discipline from software development. Software development is the process of coding functionality to meet defined end-user needs. While Software testing tends to be considered a part of development, it is really its own discipline and should be tracked as its own project. Software testing, while working very closely with development, should be independent enough to be able to hold- p or slow product delivery if quality objectives are not met. The objective of software testing is to

find problems and fix them to improve quality. Software testing typically represents 40% of a software development budget.

**There are four main objectives of testing**:

**Demonstration**:

It shows that the system can be used with acceptable risk, demonstrate functions

under special conditions and show that products are ready for integration or use.

**Detection:**

It discovers defects, errors, and deficiencies. Determine system capabilities and limitations quality of components, work products, and the system.

**Prevention:**

It provides information to prevent or reduce the number of errors clarify system specifications and performance. Identify ways to avoid risks and problems in the future.

**Improving quality:**

By doing effective testing, we can minimize errors and hence improve the quality of software.

**9.3 Testing Strategies**

A strategy for software testing integrates software test case design techniques into a well-planned series of steps that results in the successful constructions of software .The testing strategy describes the steps to be conducted as a part of testing.The various testing strategies are:

* **Unit Testing:** The testing techniques were used to test the reliability of the individual modules that were developed separately. Since this system, is developed in a modular way, all the modules are tested separately to test the reliability.

The advantages of unit Testing are,

* The modules were developed separately, so their as a whole was difficult
* The error detection, isolation and correction were easy in Unit Testing
* The compilation time required to compile separate modules were less, so the progress was fast
* **Integration Testing:** After the different modules were tested individually now it was necessary to test their behavior after they were integrated into a single system. For this the different modules were first integrated. Necessary linking between different modules was done and the software was loaded at different workstations. The testing data of the different module were fed to the integrated system. The movement of data between the modules was noted at each step. The noted data manipulations and the trace table we studies carefully to ensure that the process worked as desired.The benefit of integrated testing that we have obtained during the project is the individual modules are error free we had to look for the errors in their integration only. This testing easily reflected the shortcoming of individual module, if there was any, so the module separately could be modified to suit the integrated requirement.
* **Validation Testing**: This Validation Testing comprises of two types of testing.
* **Alpha Testing:**The previous testing was not adequate to completely ensure the reliability of the developed software. The Alpha Testing was to be the most rigorous testing. For this samples of data were brought and entered into the developed system. This would make it free of all shortcomings before this system is ported into the real system.
* **Beta Testing:** The Beta Testing was conducted at one or more customer site by the end user(s) of the software. The Beta Testing is a “live “application of the software in an environment that cannot be controlled by the developer. The customer records all problems that are encountered

during Beta Testing and reports these to the developer, at regular interval. As a result of problems reported during Beta Test, the software developer makes modifications, rectifies errors and then prepares for the release of the software product to the entire customer base.

* **Performance/load testing:** Performance testing is in general testing performed to determine how a system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage. Load testing is the simplest form of performance testing. A load test is usually conducted to understand the behavior of the system under a specific expected load. This load can be the expected concurrent number of users on the application performing a specific number of transactions within the set duration. This test will give out the response times of all the important business critical transactions. If the database, application server, etc. are also monitored, then this simple test can itself point towards any bottlenecks in the application software. The execution of an application under customer expected configuration and customer expected load to estimate the speed in processing (performance) of the application is called "Load testing" .
* **Component testing:** Starting from the bottom the first test level is ―component testing‖ sometimes called Unit testing. It involves checking that each feature specified in the component design has been implemented in the component. In theory an independent tester should do this, but in practice the developer usually does it, as they are the only people who understand how a component works. The problem with a component is that it performs only a small part of the functionality of a system, and it relies on co-operating with other parts of the system, which may not have been built yet. To overcome this, the developer either builds, or uses special software to trick the component into believing it is working in a fully functional system.
* **System Testing:** A newly designed system should have all the parts or sub systems in working order, but in reality, each sub system works independently. In System Testing, we gather all the sub system into one pool, and test the whole system, to determine whether it meets the user’s entire requirement. This is the last chance to detect and correct errors before the system is implemented. System Testing is actually a series of different tests whose primary purpose is to fully exercise the computer-base systems.
* **Acceptance testing**: Acceptance testing checks the system against the Requirement. It is similar to system testing in that whole system is checked but the important difference is the change in focus:
* System testing checks that the system that was specified has been delivered.
* Acceptance testing checks that the system delivers what was requested.

The customer and not the developer should always do acceptance testing. The customer knows what is required from the system to achieve value in the business and is the only person qualified to make that judgment. The forms of the tests may follow those in system testing, but at all times they are informed by the business needs.

**CHAPTER 10: THE PROJECT CODING**

**10.1 The Source Code**

**10.1.1 Login Form Code**

**package**pkg;

**import**java.sql.ResultSet;

**import**java.sql.SQLException;

**import**java.text.SimpleDateFormat;

**import**java.util.\*;

**import**java.io.PrintWriter;

**import**javax.servlet.http.HttpServletRequest;

**import**javax.servlet.http.HttpServletResponse;

**import**javax.servlet.http.HttpSession;

Dblogicdl=**new**Dblogic();

**public** String LoginData()**throws**SQLException,ClassNotFoundException

{

ResultSetrs=dl.RetLogin(username,password,usertype);

**if**(rs.next())

{

**if**(usertype.equals("mbadmin"))

{

**return**"mbadmin";

}

**elseif**(usertype.equals("user"))

{

String status=rs.getString("userstatus");

**if**(status.equals("approved"))

{

bank.*s*=(rs.getString(1)).toString();

String name=(rs.getString("USERNAME")).toUpperCase();

ResultSet rs1=DbConnect.*getStatement*().executeQuery("select \* from user\_details where loginid='"+*s*+"'");

**while** (rs1.next())

{

addActionMessage("Please,remember it for further use,yourUserId is :" + rs1.getString(1));

bank.*s1*=(rs1.getString(1)).toString();

}

setMsg(name);

**return**"user";

}

}

**elseif**(usertype.equals("bankmanager"))

{

String status=rs.getString("userstatus");

**if**(status.equals("approved"))

{

bank.*m*=(rs.getString(1)).toString();

String name=(rs.getString("USERNAME")).toUpperCase();

ResultSet rs2=DbConnect.*getStatement*().executeQuery("select \* from MANAGER\_DETAILS where loginid='"+*m*+"'");

**while**(rs2.next())

{

bank.*m1*=(rs2.getLong("MANAGERID"));

bank.*m2*=(rs2.getString("BANKNAME")).toString();

}

setMsg(name);

**return**"bankmanager";

}

}

**else**

{

setMsg("Waiting for Approval");

**return**"loginfail";

}

}

setMsg("Check Your Login Details.");

**return**"loginfail";

}

**public** String CreateUser()**throws** Exception

{

Map session=ActionContext.*getContext*().getSession();

String Uname=(String)session.get("Username");

String Bname=(String)session.get("Bankname");

Long Accno=(Long)session.get("Accno");

String Acctype=(String)session.get("Acctype");

String Passkey=(String)session.get("Passkey");

String Fname=(String)session.get("Fname");

String Lname=(String)session.get("Lname");

String Sex=(String)session.get("Sex");

String Address=(String)session.get("Address");

Long Areacode=(Long)session.get("Areacode");

String City=(String)session.get("City");

String State=(String)session.get("State");

String Country=(String)session.get("Country");

String Dob=(String)session.get("Dob");

String Email=(String)session.get("Email");

Long Mobile=(Long)session.get("Mobile");

String Securityquestion=(String)session.get("Securityquestion");

String Answer=(String)session.get("Answer");

**int**i=0;

i=dl.Registeruser(Uname,Bname,Accno,Acctype,Passkey,Fname,Lname,Sex,Address,Areacode,City,State,Country,Dob,Email,Mobile,Securityquestion,Answer,getPassword(),getTransactionpassword());

**return**"registersuccess";

}

**public** String ManagerData()**throws** Exception

{

Map session=ActionContext.*getContext*().getSession();

session.put("Username", getUsername());

session.put("Bankname", getBankname().toUpperCase());

**return**"managersuccess";

}

**public** String ManagerData1()**throws** Exception

{

Map session=ActionContext.*getContext*().getSession();

String Uname=(String)session.get("Username");

String Bname=(String)session.get("Bankname");

**int**i;

i=dl.RegisterManager(Uname,Bname,getFirstname(),getLastname(),getSex(),getAddress(),getAreacode(),getCity(),getState(),getCountry(),getDob(),getEmail(),getMobile(),getSecurityquestion(),getAnswer(),getPassword());

**return**"managersuccess";

}

}

req.setAttribute("ARR",arlst);

**return**"request"

}

**public** String UserApproved()**throws**SQLException,ClassNotFoundException

{

HttpServletRequest request=(HttpServletRequest)ActionContext.*getContext*().get(ServletActionContext.*HTTP\_REQUEST*);

Long id=Long.*parseLong*(request.getParameter("id"));

System.*out*.println(id);

**int**i=dl.userApproved(id);

**if**(i>0)

{

**return**NewUserRequest();

}

**returnnull**;

}

**public** String UserReject()**throws**SQLException,ClassNotFoundException

{

HttpServletRequest request=(HttpServletRequest)ActionContext.*getContext*().get(ServletActionContext.*HTTP\_REQUEST*);

Long id=Long.*parseLong*(request.getParameter("id"));

System.*out*.println(id);

**int**i=dl.userReject(id);

**if**(i>0)

{

**return**NewUserRequest();

}

**returnnull**;

}

**public** String ManagerApproved()**throws**SQLException,ClassNotFoundException

{

HttpServletRequest request=(HttpServletRequest)ActionContext.*getContext*().get(ServletActionContext.*HTTP\_REQUEST*);

Long id=Long.*parseLong*(request.getParameter("id"));

System.*out*.println(id);

**int**i=dl.managerApproved(id);

**if**(i>0)

{

**return**NewManagerRequest();

}

**returnnull**;

}

**public** String ManagerReject()**throws**SQLException,ClassNotFoundException

{

HttpServletRequest request=(HttpServletRequest)ActionContext.*getContext*().get(ServletActionContext.*HTTP\_REQUEST*);

Long id=Long.*parseLong*(request.getParameter("id"));

System.*out*.println(id);

**int**i=dl.managerReject(id);

**if**(i>0)

{

**return**NewManagerRequest();

}

**returnnull**;

}

**public** String ListOfUser()**throws**SQLException,ClassNotFoundException

{

ResultSetrs=dl.userList();

ArrayList<bank>arlst=**new**ArrayList<bank>();

**while**(rs.next())

{

bank b=**new** bank();

b.setUsername(rs.getString("USERNAME"));

//System.out.println(rs.getString("LOGINID"));

b.setUserstatus(rs.getString("USERSTATUS"));

arlst.add(b);

}

req.setAttribute("ARR",arlst);

**return**"userlist";

}

**public** String ListOfManager()**throws**SQLException,ClassNotFoundException

{

ResultSetrs=dl.managerList();

ArrayList<bank>arlst=**new**ArrayList<bank>();

**while**(rs.next())

{

bank b=**new** bank();

b.setUsername(rs.getString("USERNAME"));

//System.out.println(rs.getString("LOGINID"));

b.setUserstatus(rs.getString("USERSTATUS"));

arlst.add(b);

}

req.setAttribute("ARR",arlst);

**return**"managerlist";

}

**public** String AddNewAccount()**throws** Exception

{

**int**i=0;

Long uid=Long.*parseLong*(*s1*);

Long bid=getBank\_id();

ResultSetrs=DbConnect.*getStatement*().executeQuery("select \* from bank\_details where bank\_id="+bid+"");

**if**(rs.next())

{

String bname=rs.getString(2).toUpperCase();

i=dl.addNewAccount(getUsername(),bname,getAccountno(),getAccounttype(),getBankpasskey(),uid);

}

**return**"accountadded";

}

**public** String Accountsummary() **throws** Exception

{

Long uid=Long.*parseLong*(*s1*);

ResultSet rs1=DbConnect.*getStatement*().executeQuery("select \* from ACCOUNT\_DETAILS where userid="+uid+" ");

ArrayList<bank>arlst=**new**ArrayList<bank>();

**while**(rs1.next())

{

bank b=**new** bank();

b.setAccountno(rs1.getLong("ACCOUNTNO"));

System.*out*.println(rs1.getLong("ACCOUNTNO"));

b.setAccounttype(rs1.getString("ACCOUNTTYPE"));

b.setBankname(rs1.getString("BANKNAME"));

b.setBalance(rs1.getLong("BALANCE"));

b.setAccountstatus(rs1.getString("ACCOUNTSTATUS"));

arlst.add(b);

}

req.setAttribute("ARR",arlst);

**return**"showaccounts";

}

**public** String deleteAccount()**throws**SQLException,ClassNotFoundException

{

HttpServletRequest request=(HttpServletRequest)ActionContext.*getContext*().get(ServletActionContext.*HTTP\_REQUEST*);

Long id=Long.*parseLong*(request.getParameter("id"));

System.*out*.println(id);

**int**i=dl.accountDelete(id);

**if**(i>0)

{

**return**DeleteAccounts();

}

**returnnull**;

}

**public** String PasswordChange() **throws** Exception

{

Long id1=Long.*parseLong*(*s*);

Long id2=Long.*parseLong*(*s1*);

**int**i=dl.passwordChange(getOldpassword(),getNewpassword(),id1,id2);

**return**"pswdchanged";

}

**10.1.2 DbLogic.java**

**package**pkg;

**import**java.sql.\*;

**publicclass**Dblogic

{

Connection con=**null**;

Statement st=**null**;

ResultSetrs=**null**;

**public**ResultSetRetLogin(String username,Stringpassword,Stringusertype)**throws**SQLException,ClassNotFoundException{

rs=DbConnect.*getStatement*().executeQuery("select \* from LOGIN\_DETAILS where username='"+username+"' and password='"+password+"' and usertype='"+usertype+"'");

**return**rs;

}

**publicint**Registeruser(String uname, String bname, Long accno,

String acctype, String passkey, String fname, String lname,

String sex, String address,

Long areacode, String city, String state, String country,

String dob, String email, Long mobile,Stringsecurityquestion,String answer,

String password, String transactionpassword) **throws**SQLException

{

**int**i=0;

**int**j,k;

**try**

{

i=DbConnect.*getStatement*().executeUpdate("insert into LOGIN\_DETAILS values(LOGIN\_DETAILS\_SEQ.nextval,'"+uname+"','"+password+"','user','notapproved')");

j=DbConnect.*getStatement*().executeUpdate("insert into USER\_DETAILS values(USER\_DETAILS\_SEQ.nextval,'"+uname+"','"+fname+"','"+lname+"','"+sex+"','"+address+"',"+areacode+",'"+city+"','"+state+"','"+country+"','"+dob+"','"+email+"',"+mobile+",'"+securityquestion+"','"+answer+"','"+password+"','"+transactionpassword+"',LOGIN\_DETAILS\_SEQ.nextval-1)");

k=DbConnect.*getStatement*().executeUpdate("insert into ACCOUNT\_DETAILS values(ACCOUNT\_DETAILS\_SEQ.nextval,"+accno+",'"+acctype+"','"+bname+"',5000,'"+passkey+"','notapproved',USER\_DETAILS\_SEQ.nextval-1)");

}

**catch**(Exception e)

{

System.*out*.println(e.getMessage());

}

**return**i;

}

**publicint**RegisterManager(String uname, String bname, String firstname,

String lastname, String sex, String address, Long areacode,

String city, String state, String country, String dob,

String email, Long mobile, String securityquestion, String answer,

String password) **throws**SQLException

{

**int**i=0;

**int** j=0;

**try**

{

i=DbConnect.*getStatement*().executeUpdate("insert into LOGIN\_DETAILS values(LOGIN\_DETAILS\_SEQ.nextval,'"+uname+"','"+password+"','bankmanager','notapproved')");

j=DbConnect.*getStatement*().executeUpdate("insert into MANAGER\_DETAILS values(MANAGER\_DETAILS\_SEQ.nextval,'"+uname+"','"+firstname+"','"+lastname+"','"+sex+"','"+address+"',"+areacode+",'"+city+"','"+state+"','"+country+"','"+dob+"','"+email+"',"+mobile+",'"+securityquestion+"','"+answer+"','"+password+"',LOGIN\_DETAILS\_SEQ.nextval-1,'"+bname+"')");

}

**catch**(Exception e)

{

System.*out*.println(e.getMessage());

}

**return**i+j;

}

}

**public**ResultSetViewprofile(Long id) **throws**SQLException

{

rs=DbConnect.*getStatement*().executeQuery("select \* from USER\_DETAILS where userid="+id+"");

**return**rs;

}

**public**ResultSettransactionAccount(Long accno, String acctype, String bname,

Long bal) **throws**SQLException

{

rs=DbConnect.*getStatement*().executeQuery("select \* from ACCOUNT\_DETAILS where accountno="+accno+" and accounttype='"+acctype+"' and bankname='"+bname+"' and balance="+bal+"");

**return**rs;

}

**publicint** bill(Long accno, Long b) **throws**SQLException

{

**int**i=0;

i=DbConnect.*getStatement*().executeUpdate("update ACCOUNT\_DETAILS set balance="+b+" where accountno="+accno+"");

**return**i;

}

**public**ResultSetbillLogin(String transactionpassword, Long id) **throws**SQLException,ClassNotFoundException

{

rs=DbConnect.*getStatement*().executeQuery("select \* from USER\_DETAILS where transactionpassword='"+transactionpassword+"' and userid="+id+"");

**return**rs;

}

**public**ResultSetbillAccount(Long accno, String acctype, String bname,

Long bal) **throws**SQLException

{

rs=DbConnect.*getStatement*().executeQuery("select \* from ACCOUNT\_DETAILS where accountno="+accno+" and accounttype='"+acctype+"' and bankname='"+bname+"' and balance="+bal+"");

**return**rs;

}

**public**ResultSetbillEnquire(Long accno, String bname, Long uid, String x, String y) **throws**SQLException,ClassNotFoundException

{

rs=DbConnect.*getStatement*().executeQuery("select bill\_id,billpayment\_date,billername,billno,billamount,accountno,balance from BILLING\_DETAILS where accountno="+accno+" and bankname='"+bname+"' and billpayment\_date BETWEEN '"+x+"' and '"+y+"' and userid="+uid+" order by bill\_id");

**return**rs;

}

**public**ResultSettransactionEnquire(Long accno, String bname, Long uid,

String x, String y)**throws**SQLException,ClassNotFoundException

{

rs=DbConnect.*getStatement*().executeQuery("select tran\_id,transaction\_date,r\_accountno,r\_bankname,amount,accountno,balance from TRANSACTION\_DETAILS where accountno="+accno+" and bankname='"+bname+"' and transaction\_date BETWEEN '"+x+"' and '"+y+"' and userid="+uid+" order by tran\_id");

**return**rs;

}

**10.1.3 DbConnect.java**

packagepkg;

importjava.sql.Connection;

importjava.sql.DriverManager;

importjava.sql.Statement;

public final class DbConnect

{

public static Statement getStatement()

{

Connection con=null;

Statement st=null;

try

{

Class.forName("oracle.jdbc.driver.OracleDriver");

con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","multibanking","multibanking");

st=con.createStatement();

}

catch(Exception e)

{

e.printStackTrace();

}

returnst;

}

}

**10.2 JSP PAGES**

**Login.jsp Page**

<%@pagelanguage=*"java"*import=*"java.util.\*"*pageEncoding=*"ISO-8859-1"*%>

<%@tagliburi=*"/struts-tags"*prefix=*"s"*%>

<%

String path = request.getContextPath();

String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";

%>

<!DOCTYPEHTMLPUBLIC"-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>

<metaname=*"keywords"*content=*""*/>

<metaname=*"description"*content=*""*/>

<metahttp-equiv=*"content-type"*content=*"text/html; charset=utf-8"*/>

<title>Multibanking</title>

<linkhref=*"http://fonts.googleapis.com/css?family=Abel"*rel=*"stylesheet"*type=*"text/css"*/>

<linkhref=*"style.css"*rel=*"stylesheet"*type=*"text/css"*media=*"screen"*/>

<scripttype=*"text/javascript"*src=*"jquery-1.7.1.min.js"*></script>

<scripttype=*"text/javascript"*src=*"jquery.slidertron-1.0.js"*></script>

<scripttype=*"text/javascript"*>

**functionValidateForm**(frm)

{

**var**pswd=frm.password.value;

**var**un=frm.username.value;

**if**(un.length==0)

{

alert("Username Must Not Be Blank...");

**returnfalse**;

}

**if**(pswd.length==0)

{

alert("Password Must Not Be Blank...");

**returnfalse**;

}

**if**(document.forms[0].usertype[0].checked==**false**&&document.forms[0].usertype[1].checked==**false**&&document.forms[0].usertype[2].checked==**false**)

{

alert("usertype cannot be null");

**returnfalse**;

}

**returntrue**;

}

</script>

</head>

<body><divid=*"wrapper"*>

<divid=*"header-wrapper"*>

<divid=*"header"*>

<divid=*"logo"*>

<h1>MULTIBANKING SYSTEM<br></h1>

<p><em><br></em></p>

</div>

</div>

</div>

<tablewidth=*"1500"*>

<tr><td><marqueescrollamount=*"7"*align=*"left"*><fontcolor=*"green"*size=*"8"*><b>Banks Linked:SBI,PNB,ICICI,INDIAN,IDBI</b></font></marquee></td></tr>

<s:formaction=*"LoginAction"*onsubmit=*"return ValidateForm(this)"*>

<tr><td><s:textfieldname=*"username"*label=*"User Name"*/></td></tr>

<tr><td><s:passwordname=*"password"*label=*"Password"*/></td></tr>

<tr><td><s:radioname=*"usertype"*label=*"User Type"*list=*"{'user','mbadmin','bankmanager'}"*/></td></tr>

<tr><tdcolspan=*"2"*align=*"right"*><s:submitvalue=*"Login"*></s:submit></td></tr>

<tr><td><s:propertyvalue=*"msg"*/></td></tr>

</s:form></table>

<br/>

&nbsp;&nbsp;<b><ahref=*"forgot password.jsp"*>forgot password?</a>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<ahref=*"signup.jsp"*>Register Now</a>

</b><br/><br/><hrcolor=*"red"*width=*"1500"*/>

</div>

</body>

</html>

**MyAccounts.jsp Page**

<%@pagelanguage=*"java"*import=*"java.util.\*"*pageEncoding=*"ISO-8859-1"*%>

<%@pageimport=*"pkg.bank"*%>

<%@tagliburi=*"/struts-tags"*prefix=*"s"*%>

<%

String path = request.getContextPath();

String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";

%>

<!DOCTYPEHTMLPUBLIC"-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>

<metaname=*"keywords"*content=*""*/>

<metaname=*"description"*content=*""*/>

<metahttp-equiv=*"content-type"*content=*"text/html; charset=utf-8"*/>

<title>Multibanking System</title>

<linkhref=*"http://fonts.googleapis.com/css?family=Abel"*rel=*"stylesheet"*type=*"text/css"*/>

<linkhref=*"style.css"*rel=*"stylesheet"*type=*"text/css"*media=*"screen"*/>

<scripttype=*"text/javascript"*src=*"jquery-1.7.1.min.js"*></script>

<scripttype=*"text/javascript"*src=*"jquery.slidertron-1.0.js"*></script>

</head>

<body><divid=*"wrapper"*>

<divid=*"header-wrapper"*>

<divid=*"header"*>

<divid=*"logo"*>

<h1>MULTIBANKING SYSTEM</h1>

<p><br></p>

</div>

</div>

</div><hrwidth=*"1500"*/>

<!-- end #header -->

<divid=*"menu"*>

<center><ul>

<liclass=*"current\_page\_item"*><br></li>

<li><br></li>

<li><ahref=*"User Home.jsp"*">Home</a></li>

<li><ahref=*"profile.jsp"*>Profile</a></li>

<li><ahref=*"Transaction Login.jsp"*>Transaction</a></li>

<li><ahref=*"Bill Login.jsp"*>Bill Payment</a></li>

<li><ahref=*"Product and Services.jsp"*>Product & Services</a></li>

<li><ahref=*"enquire type.jsp"*>Enquires</a></li>

<li><ahref=*"Contact.jsp"*>Contact</a></li>

<li><ahref=*"Logout"*>Logout</a></li>

</ul></center>

</div>

<!-- end #menu --><divid=*"slider"*>&nbsp;

<br/><divclass=*"viewer"*>

<divclass=*"reel"*>

<divclass=*"slide"*><imgsrc=*"images/3.jpg"*width=*"1500"*height=*"300"*alt=*""*/></div>

<divclass=*"slide"*><imgsrc=*"images/img01.jpg"*width=*"1500"*height=*"300"*alt=*""*/></div>

<divclass=*"slide"*><imgsrc=*"images/1.jpg"*width=*"1500"*height=*"300"*alt=*""*/></div>

</div>

</div>

</div>

<scripttype=*"text/javascript"*>

$('#slider').slidertron({

viewerSelector:'.viewer',

reelSelector:'.viewer .reel',

slidesSelector:'.viewer .reel .slide',

advanceDelay:3000,

speed:'slow'

});

</script>

<br/><br/><br/>

<center><ul>

<b><li><ahref=*"AccountSummary"*>Account Summary</a></li>

<li><ahref=*"ANA"*>Add New Account</a></li>

<li><ahref=*"DeleteAccount"*>Delete Account</a></li></b>

</ul></center>

<br/><br/><br/><hrcolor=*"red"*width=*"1500"*/>

</div>

</body>

</html>

**CHAPTER 11: OUTPUT SCREEN**

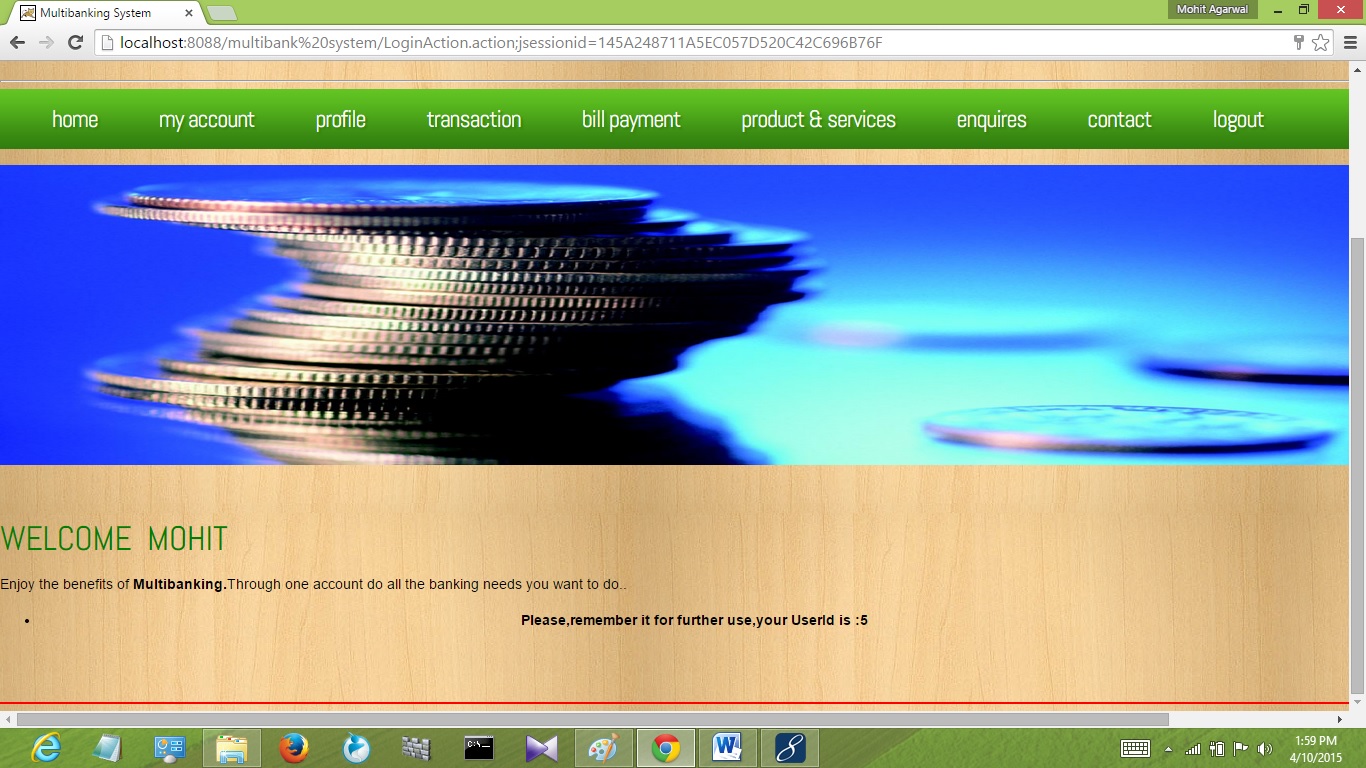
**HOME PAGE**



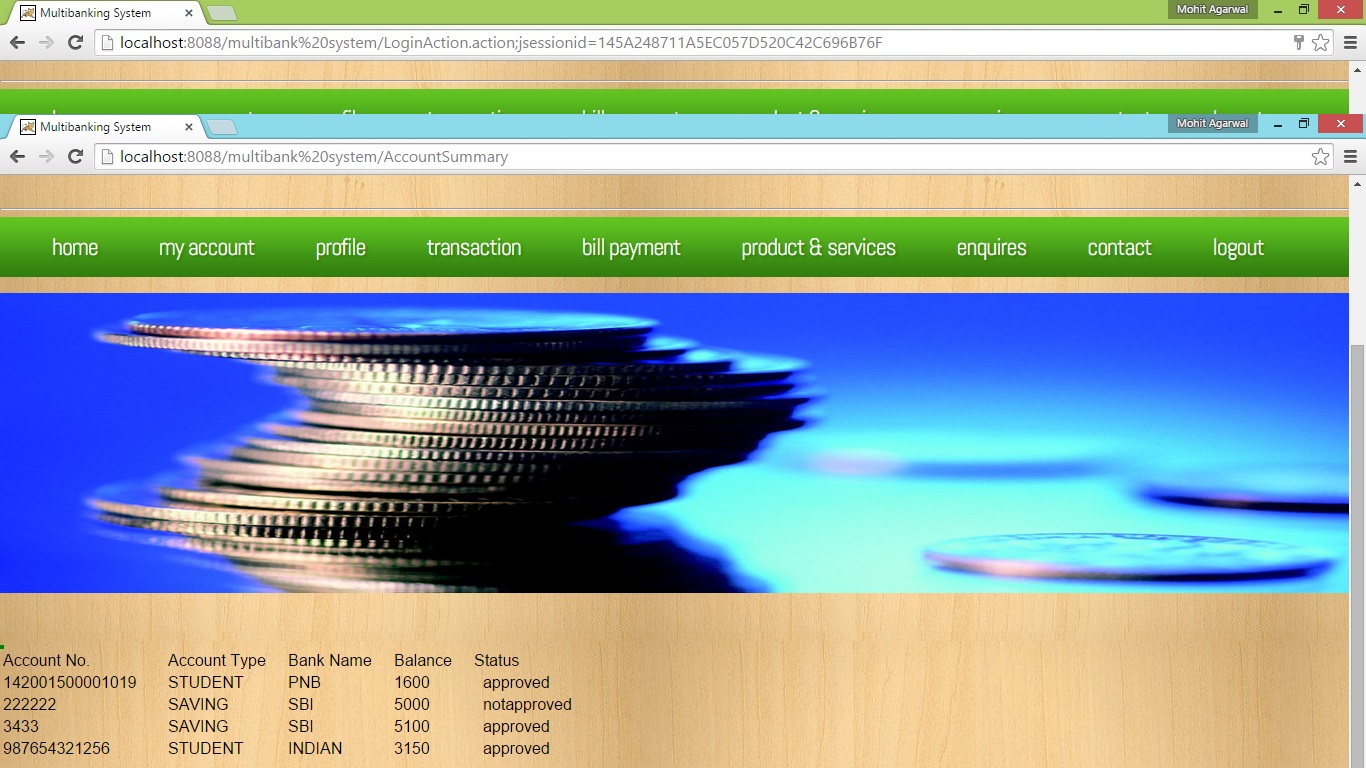
**LOGIN PAGE**

****

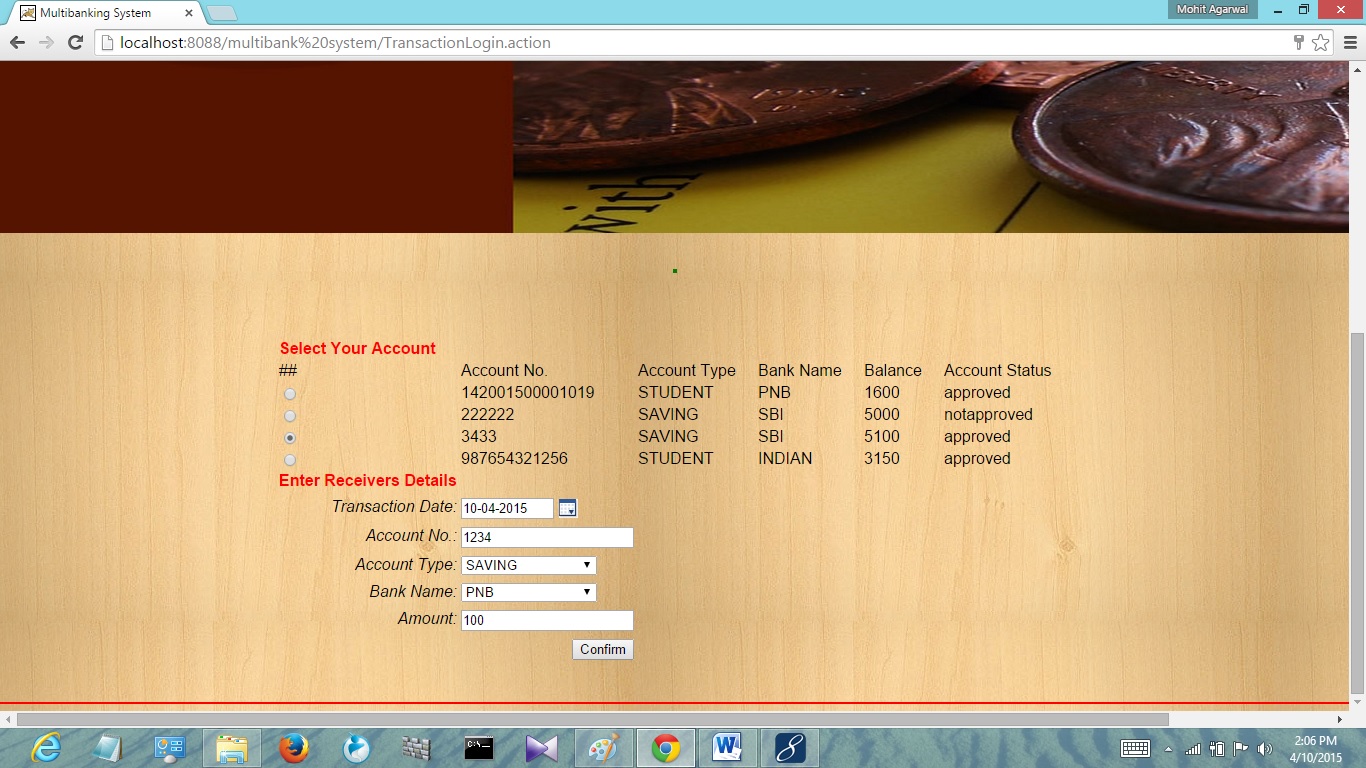
**USER HOMEPAGE**

****

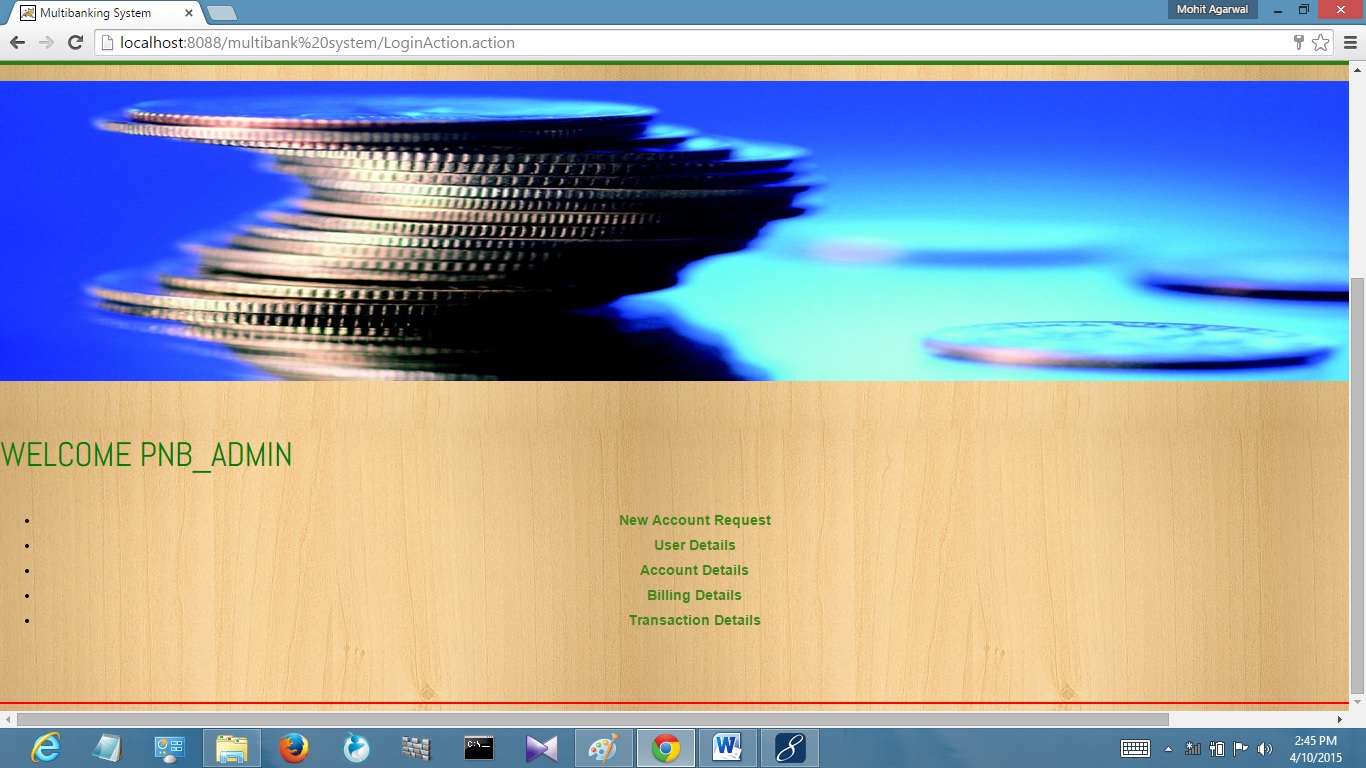
**ACCOUNT SUMMARY PAGE**

****

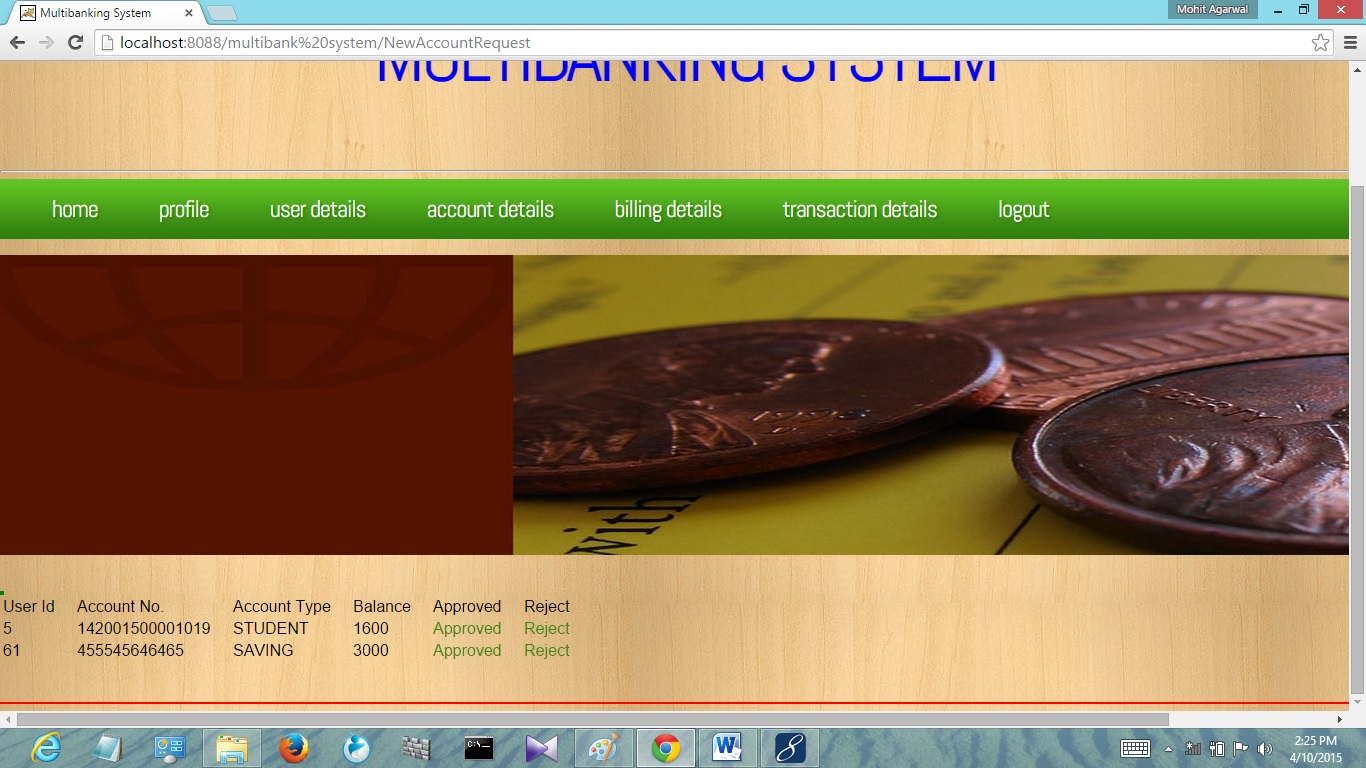
**TRANSACTION PAGE**

****

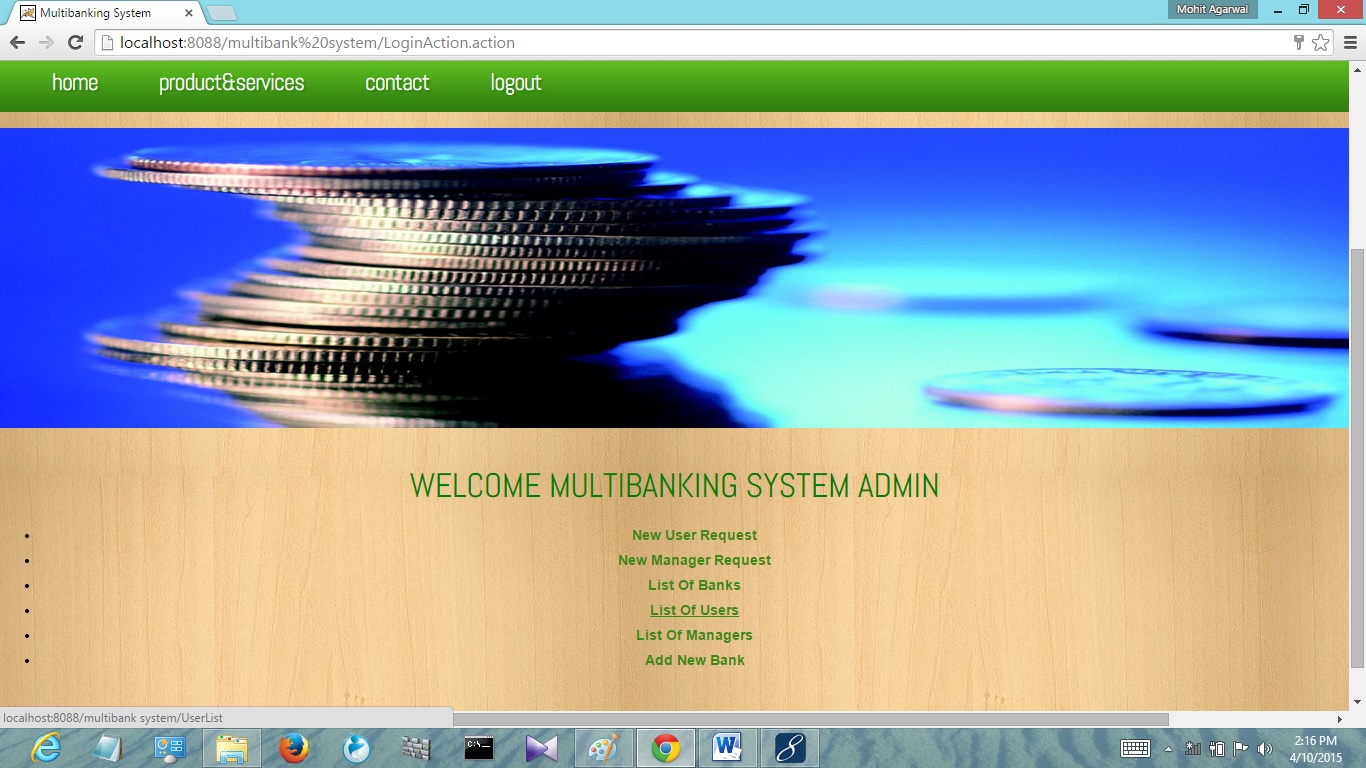
**BANK MANAGER HOME PAGE**

****

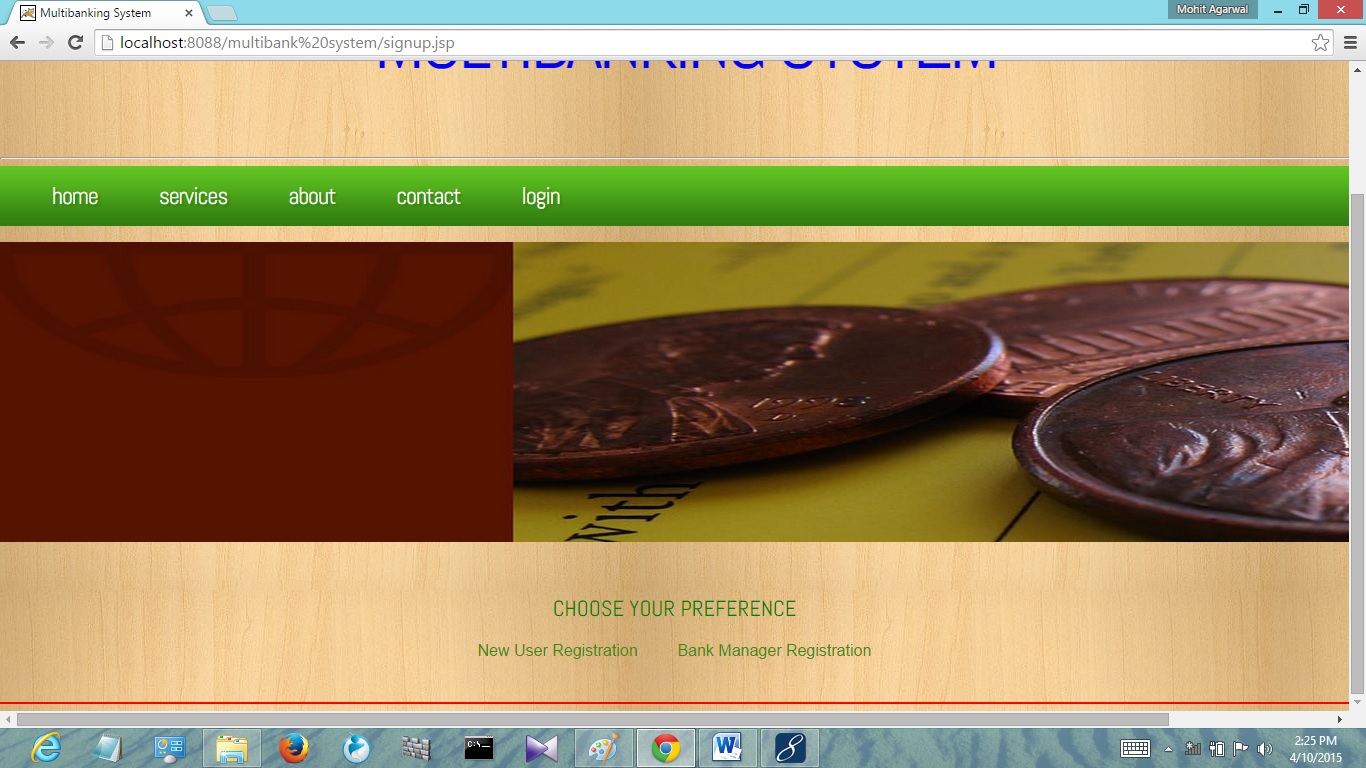
**ACCOUNT REQUEST PAGE**

****

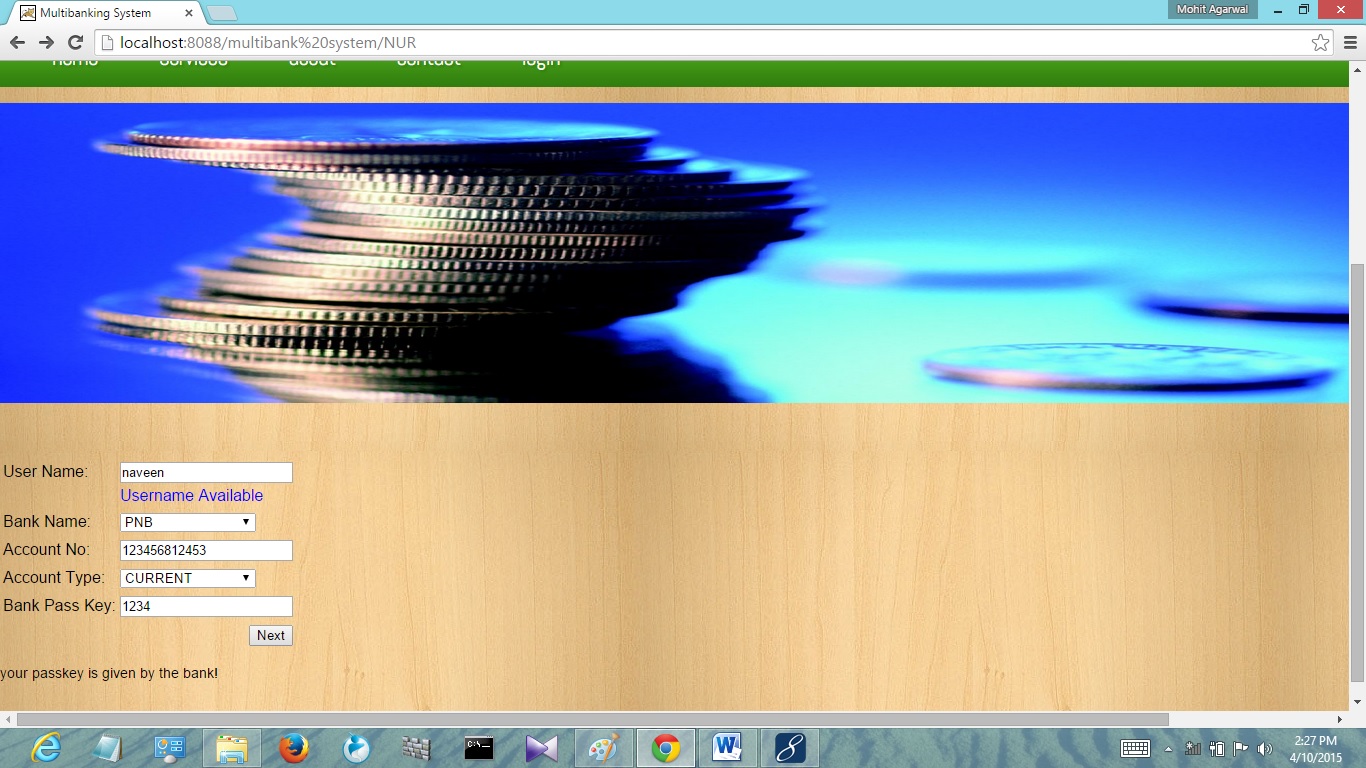
**MBADMIN HOME PAGE**

****

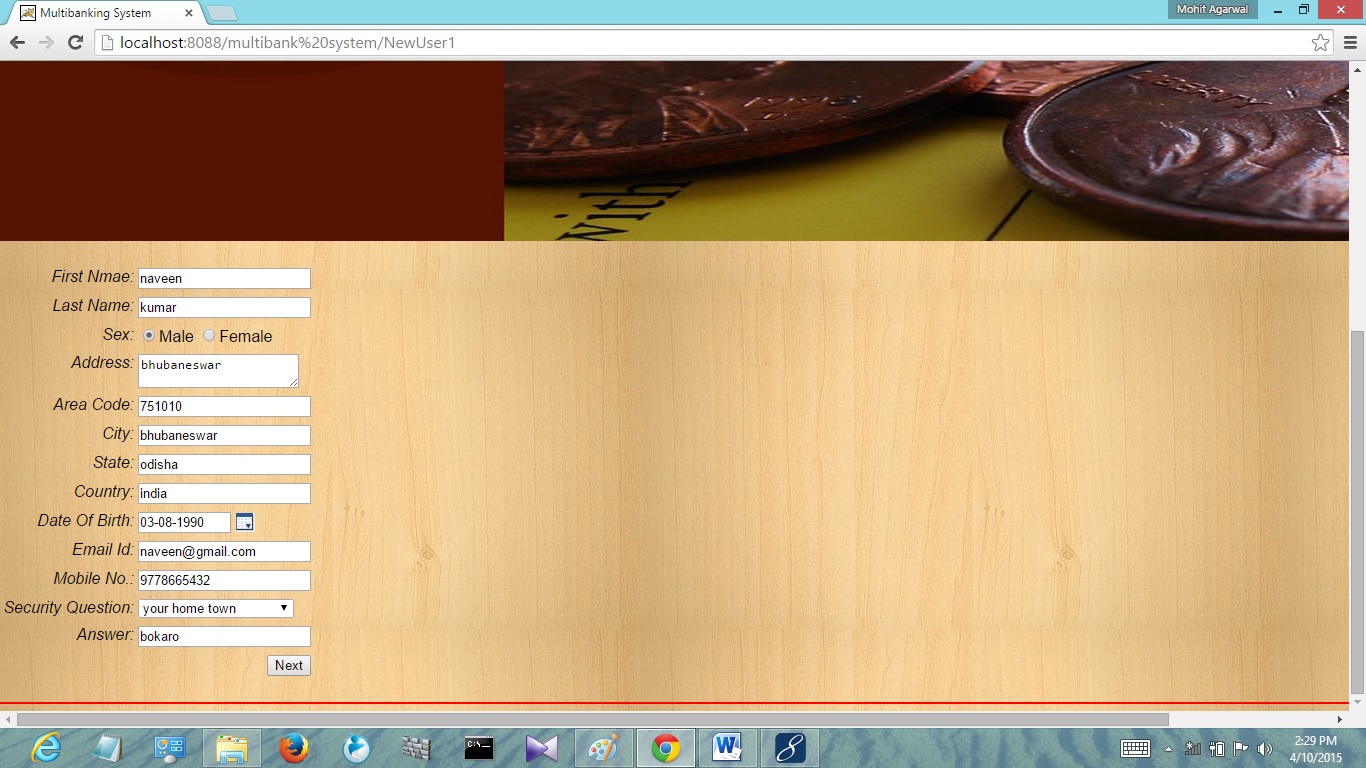
**REGISTERATION PAGE**

****

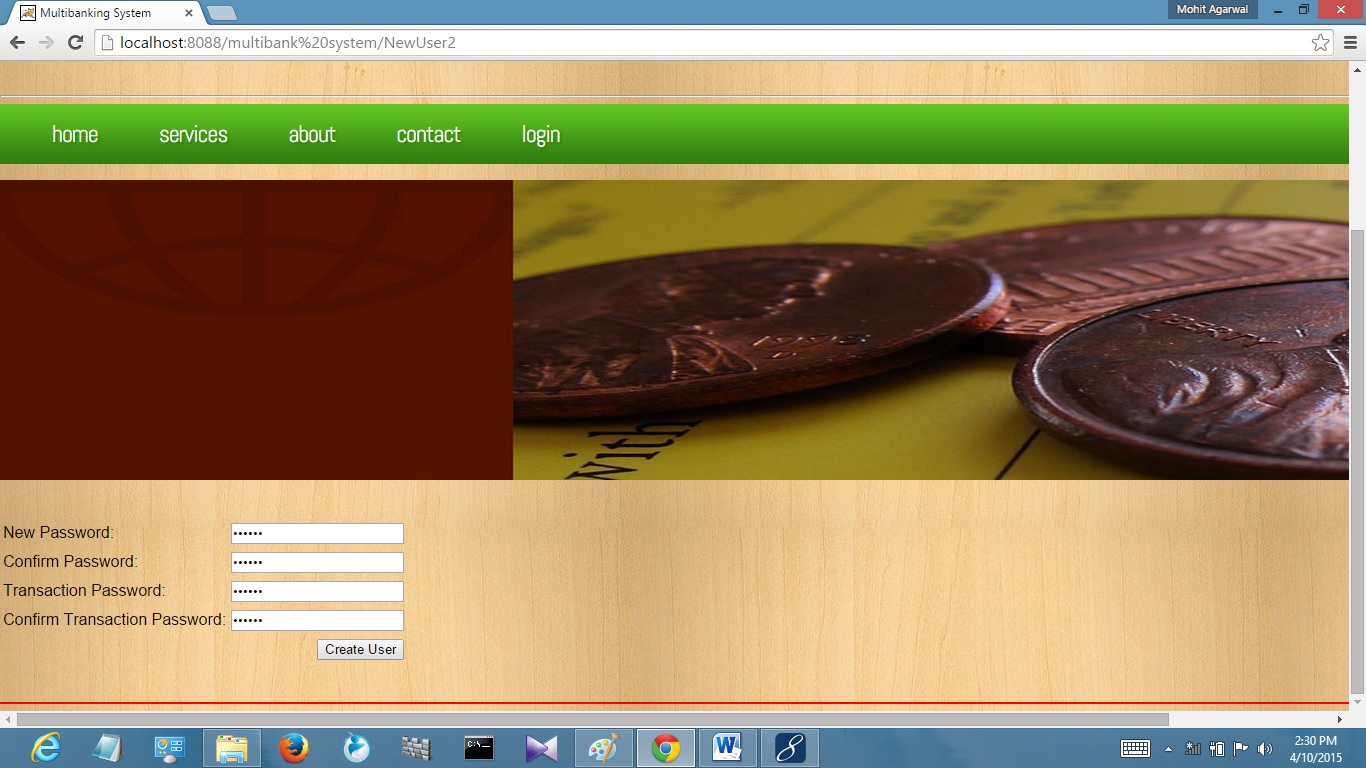
**USER REGISTERATION PAGE PART- 1**

****

**USER REGISTERATION PAGE PART-2**

****

**USER REGISTERATION PAGE PART- 3**

****

**CHAPTER 12: CONCLUSION**

The project “Multi-Banking System” has been developed as per the requirement specification. This Project has several forms that are validated properly and provide facilities to the user to know more about their accounts and banks. Here giving the user id provide complete security and authentication to each department so that unauthenticated user could not hack the system. So it is a secured and better performanceoriented project.

It has been developed properly and the complete system is thoroughly tested with the availability data and throughput reports which are prepared manually. This design is so flexible that any new modules can be incorporated easily. The concept and the technology used in the project make it robust and user friendly with 100% accuracy output. The cost of this system is also less and the deployment sector does not have to make any additional investment for implementing this project. So, it is economically feasible.

In this growing industry, it is very important to meet the growingrequirements and to save time of the users. A multi-banking portal just fulfils the tough demand of the corporate world. It is one of the fastest growing industries. There are more and more opportunities to come for the portals in the time to come,there is no doubt about it.

**CHAPTER 13: BIBLIOGRAPHY**

**References for the Project Development Were Taken From the following**

Books and Web Sites.

[www.tutorialspoints.com](http://www.tutorialspoints.com)

[www.roseindia.com](http://www.roseindia.com)

[www.publib.boulder.ibm.com](http://www.publib.boulder.ibm.com)

Head First EJB Sierra Bates

J2EE Professional by Shadab Siddiqui

Java Database Programming with JDBC by Patel moss.

Software Engineering by Roger Pressman.

Software Engineering by Rajib Mall.

STRUTS2 by P.K. Dixit.