

1. INTRODUCTION

Education and development are closely related, as education provides inputs for economic growth among which knowledge is an important one. Modern economic growth, as seen in recent years, is driven by knowledge, for which knowledge creation and utilization acquires significance. Development of knowledge is based on the quantity and quality of education system available, particularly of higher education, in a country. Therefore, higher education assumes significance as creator and supplier of knowledge. Access to and availability of higher education is crucial for creation, spread and application of knowledge for development. Importance of higher education can be understood from its functions like (i) creation and dissemination of knowledge, (ii) supply of manpower, particularly knowledge workers, (iii) attitudinal changes for modernization and social transformation, (iv) formation of strong nation-state, and (v) promotion of higher quality individual and social life (GoI 2005). Hence, development of higher education becomes significance. Development of higher education depends on various factors, among which finance plays a major role. Financing higher education has attracted serious attention of policy makers and educational thinkers as higher education system is facing financial crunch (Varghese 2009, Rani 2009, GoI 2005) in recent years. Alternative ways of financing higher education are being explored and implemented to overcome the problem of deficit finance and cost recovery. Among them educational loan is increasingly seen as an important source of finance. It is necessary to note that education loan has become a way transferring financial burden from government to consumers of higher education. In this background an analysis of pattern of educational loan or student loan is necessary to understand the existing scenario.

Loans designed to help students pursue an education recognize that students should spend their school time studying, not working to repay a loan. For this reason, many loans created for students allow students to pay back their debt vary gradually.

Financing Higher Education Loan

There are various kinds of courses which are eligible for different kinds of bank student loans.

- Loans are available for various graduation, post graduation, computer education and diploma courses offered by recognized universities of government statutory bodies in India.
- Education loan is also provided for technical and professional courses, regular diploma and degree courses as well as courses conducted reputed foreign universities in India.
- In case of studies abroad the institutes offering the courses should be approved by the complaint authority of the nation.
- The loan amounts that are offered to students may vary. If the student will stay in India to complete their studies, they could get up to Rs.3lakhs. If the student is studying abroad, they may receive a loan which could be as high as Rs.5.00lakhs. The interest rates for these loans will range from 12 to 14%. Rebates may sometime be offered depending on the circumstances.

1.1 Objective and Scope of the Project

Objective:

If you want to study abroad in India, it is possible to obtain education loans. While getting these loans was difficult in the past, it has now become easier than ever before. A large number of national banks in India have created education loans. The terms of these loans will vary with each institution, so it is important to do your research before using them.

Scope:

With the cost of various professional courses going up rapidly the demand for educational loans has gone up substantially in the recent time. The education loan industry is growing and the current outstanding of education loan is about Rs43, 000 crore. Apart from banks, Credila is the only specialized player in the educational loans market what banks are offering is the very basic product, which has been in existence for around eight years now. This enables us to come up the new products and process that suit the changing needs of all stakeholders. Education loan is a seasonal business and a sudden surge in applications during the admissions time makes it challenging for banks to ensure that students are given the decision on their applications before they have to pay the fees.

2. System Analysis:

2.1 Existing System

In Current system Student Should apply for His/her Higher Education loan to the Banks manually. Its huge process to apply a bank along with student should apply optional banks because student will not depend particular bank not only that but this system takes more energy, cost, time.

Disadvantages:

The following are the disadvantages of the existing system

- Indefinite results or out put
- Lengthy loan Process
- Student not aware about banks rules, regulation and loan interest etc.
- Student should apply loan each or every bank manually.

2.2 Proposed System

- The major purpose of this loan is to give financial help to the students for further studies.
- Interest rate is less as compared to other types of loans.
- Repaying the loan amount is only after completion of the study.
- It helps the students to fulfill his/her dreams by availing this loan.
- The banks offer good services to the society by providing Educational loan for the students who are unable to continue their higher education due to lack of cash.

3. SYSTEM REQUIREMENTS:

Software Requirements:

- *Operating System* : Windows XP/Windows 7/Windows 8
- *Web Server* : Apache Tomcat Server 5.5
- *IDE* : Micro media Dreamweaver
- *Browser* : Firefox, Chrome etc
- *Database* : My SQL
- *Designing Technology* : HTML
- *Server side Technology* : JSP
- *Animation* : Flash

Hardware Requirements:

- *Processor* : Pentium 4 or Above
- *RAM* : 512 MB or Above
- *Storage Capacity* : 10 GB free space in hard disk

4. MODULES:

1. Admin-Module:

Admin can perform the following activities:

- She/he can insert bank details
- View the user registration details
- View the loan details

2. Bank-Module:

Bank can perform the following activities:

- Inserting the loan details, which loan will be required in student purpose
- View the apply loan details
- The loan will be approved only when the correct document will be submitted.
- When a documents are invalid then loan will be rejected.

3. Student-Module:

Student can perform the following activities:

- View the bank details
- View the loan details
- View the apply loan details, the loan will approved or pending will be display
- Insert the my documents details

5. FEASIBILITY STUDY:

Feasibility study is conducted once the problem is clearly understood. Feasibility study is a high level capsule version of the entire system analysis and design process. The objective is to determine quickly at a minimum expense how to solve a problem. The purpose of feasibility is not to solve the problem but to determine if the problem is worth solving.

The system has been tested for feasibility in the following points.

1. Technical feasibility
2. Economical feasibility
3. Operational feasibility
4. Schedule feasibility

1. Technical Feasibility:

The project entitles "Financing Higher Education Loan" is technically feasibility because of the below mentioned feature. The project was developed in Java which Graphical User Interface. It provides the high level of reliability, availability and compatibility. All these make Java an appropriate language for this project. Thus the existing software Java is a powerful language.

2. Economical Feasibility:

The computerized system will help in automate the selection leading the profits and details of the organization. With this software, the machine and manpower utilization are expected to go up by 80-90% approximately. The costs incurred of not creating the system are set to be great, because precious time can be wanted by manually.

3. Operational Feasibility:

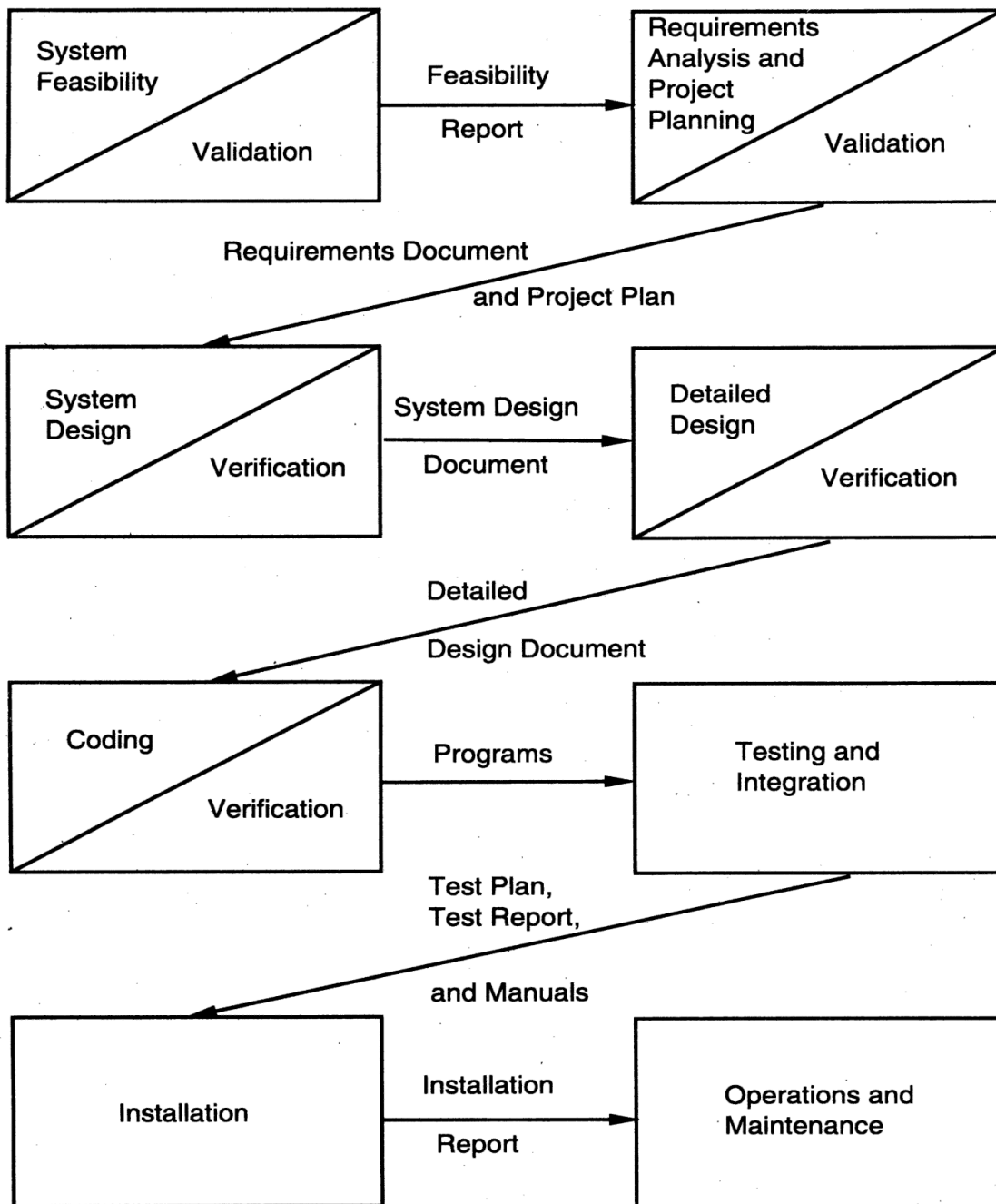
In this project, the management will know the details of each project where he may be presented and the data will be maintained as decentralized and if any inquires for that particular contract can be known as per their requirements and necessities.

4. Schedule Feasibility:

Time evaluation is the most important consideration in the development of project. The time schedule required for the developed of this project is very important since more development time effect machine time, cost and cause delay in the development of other systems.

A reliable Financing Higher Education System can be developed in the considerable amount of time.

6.METHODOLOGY ADAPTED:

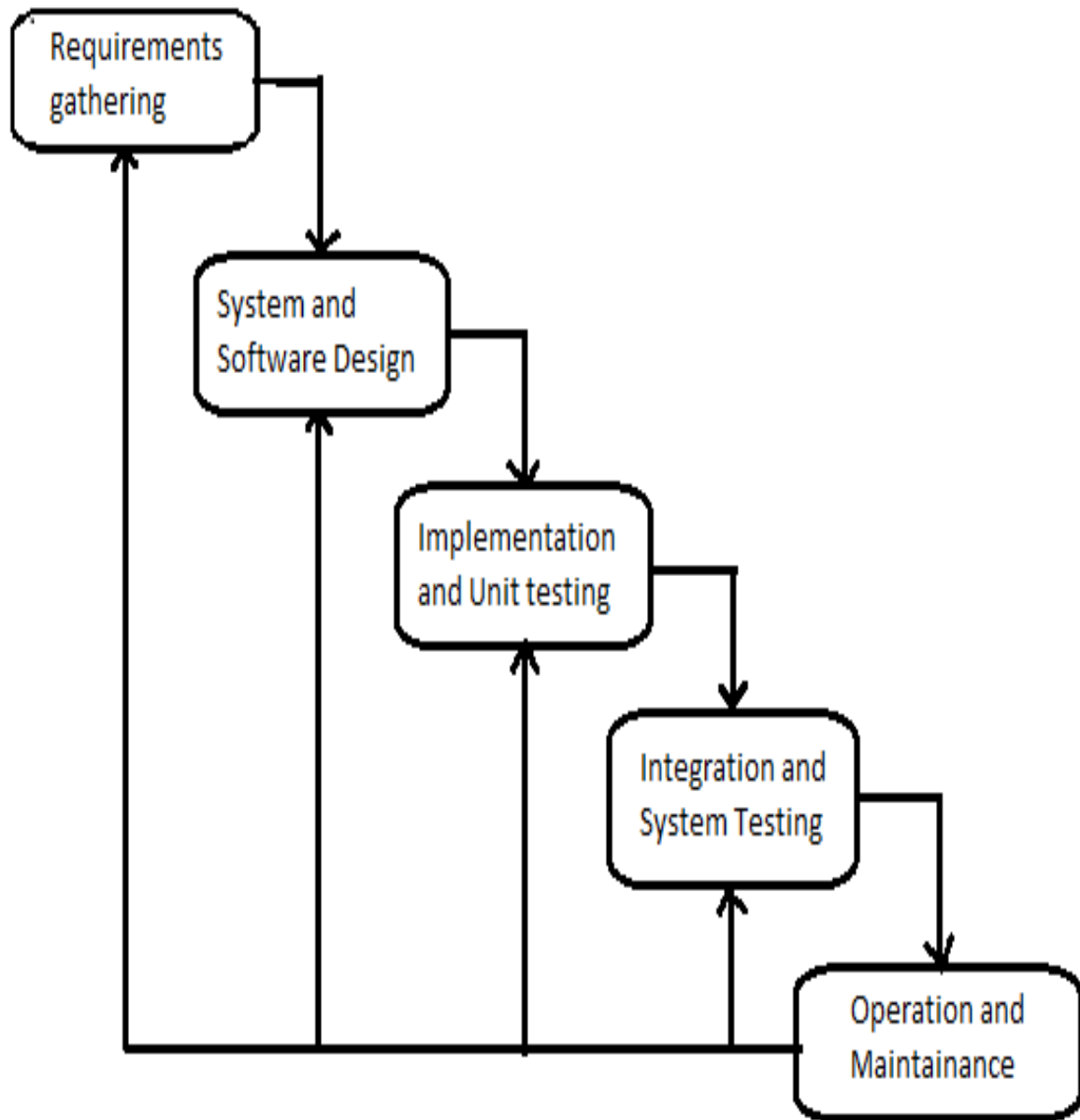


6.1 WATERFALL MODEL:

The Waterfall model was first process model to be introduced. It is also referred to as a linear-sequential life cycle model. In a waterfall model, each phase must be completed fully before the next phase can begin.

The principal stages of the waterfall model directly reflect the fundamental development activities.

1. Requirements gathering: The system's services, constraints and goals are established by consultation with system users. They are then defined in detail and serve as a system specification
2. System and Software design: The systems design process allocates the requirements to either hardware or software systems by establishing an overall system architecture. Software design involves identifying and describing the fundamental software system abstractions and their relationships.
3. Implementation and unit testing: During this stage, the software design is realized as a set of programs or program units. Unit testing involves verifying that each unit meets its specification.
4. Integration and system testing: The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer.
5. Operation and maintained: Normally, this is the longest life cycle phase. The system is installed and put into practical use. Maintenance involves correcting errors which were not discovered in earlier stages of the life cycle, improving the implementation of system units and enhancing the system's services as new requirements are discovered.



7. SYSTEM IMPLEMENTATION:

The goal of the coding phase is to translate the design into code in the given programming language. The coding steps translate the detailed design of the system into programming language. The translation process continues when compiler accepts source code as input and produces machine dependent object code as output. Linking of object files are done to produce the machine code. Internal documentation is another important factor, to facilitate other to understand the code and the logic.

Code review and walk through:

Both reviews and walk through used to deliver the correct codes. The code review is done as soon as code is ready to be executed, this is to reduce syntax errors and also check the coding standard.

Module Specification:

The modules specified in the design are implemented using various “.html”, “.jsp” and “.class” files. These files in the source code shares the common routines and share the data structure, to establish the hierarchical relationship

Compilation and Building the executable:

The source code for the system organized in various files is compiled using the “java” utility provided in the JAVA. The application is made to run in web browser the address as “http://localhost:8081/staff” present in ROOT directory of Tomcat Server.

Running the package:

The following steps are undertaken to execute this application:

- First start Tomcat Server.
- Open any Web Browser like Google Chrome, Mozilla Firefox.
- Type the following address in address bar of Web browser.
- http://localhost:8080/staff
- Now the Home page of application is displayed on screen. If new registration is there then register or sign up first and use the application. If already registered then directly logon into the system.

8. THEORETICAL BACKGROUND:

Java:

Java is a computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java applications are typically compiled to byte code (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture. Java is, as of 2014, one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems(which has since merged into Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1991 and first released in 1995.As of May 2007, in compliance with the specifications of the Java Community Process; Sun relicensed most of its Java technologies under the GNU. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java (byte code compiler), GNU Class path (standard libraries),and Iced Tea-Web (browser plug in for applets).

Advantages of java:

- Purely Object oriented.
- Platform independent.
- It is dynamic, simple and robust.
- Easy to learn.
- Multithreaded.

Financing Higher Education Loan

- Secure.
- Wide variety of Application Programmer Interfaces (APIs).
- Excellent networking capability.

The java Platform:

The Java platform is the name given to the computing platform from Oracle that helps users to run and develop Java applications. The platform does not just enable a user to run and develop Java application, but also features a wide variety of tools that can help developers work efficiently with the Java programming language.

The platform consists of two essential software's:

- The Java Runtime Environment (JRE), which is needed to run Java applications and applets.
- The Java Development Kit (JDK), which is needed to develop those Java applications and applets. If you have installed the JDK, you should know that it comes equipped with a JRE as well. So, for all the purposes of this book, you would only require the JDK.

Java Components:

- Java has two components those are
 - 1. Java virtual machine (JVM).
 - 2. Java Application Programmers Interface (API).

JVM:

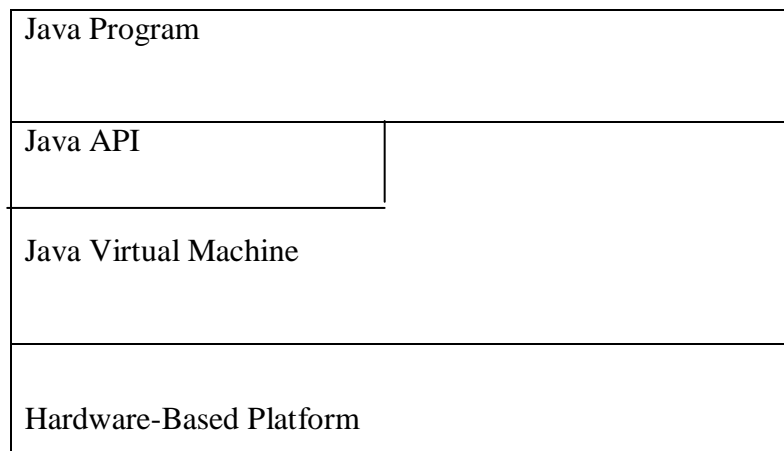
A Java virtual machine (JVM) interprets compiled Java binary code (called byte code) for a computer's processor (or "hardware platform") so that it can perform a Java program's instructions. Java was designed to allow application programs to be built that could be run on any platform without having to be rewritten or recompiled by the programmer for each

Financing Higher Education Loan

separate platform. A Java virtual machine makes this possible because it is aware of the specific instruction lengths and other particularities of the platform

API:

An application programming interface (API) is a library of functions that Java provides for programmers for common tasks like file transfer, networking, and data structures.



Java in web:

Java covers the whole application from server to client and back again, it provides many powerful technologies, it can be used to extend the browser, and it provides good security system.

HTML:

HTML stands for hypertext markup Language. It is very useful to make web pages and very easy to learn. Hypertext Markup file is a text file containing small markup tags.

These marks up tags tell the browser how to display a web page. It has two types of

Financing Higher Education Loan

extensions one is .html and second is .htm but both are used for html web pages. For hypertext markup language you can use the simple text editor for example; use notepad for writing your HTML code in the windows. If you are using Mac you can use simple text editor.

HTML uses approach of what you see is what you get. You can also use to write tags other software that is FrontPage and Dreamweaver. In HTML character are surrounded by the tags. HTML tags come in pair. The beauty of this language is that it is not case sensitive. Every web page need HTML with it you cannot make the good web pages. And it is the base for every web page and used to display the text in the web pages there are some other latest version of HTML like DHTML which stands for dynamic html and is used to make the web pages more interactive

Features of HTML:

- It is simple to understand and implement.
- HTML constructs are very easy to comprehend, and can be used effectively by anybody.
- HTML syntax is a worldwide standard.
- The methodology used by HTML to markup information is independent of its representation on a particular hardware or software architecture.
- It is not a programming language.
- And it is also not a description language

Java Server Pages (JSP):

Java Server Pages (JSP) lets you separate the dynamic part of your pages from the static HTML. We simply write the regular html in the normal manner, using whatever Web-page-building tools you normally use. We then enclose the code for the c parts in special tags, most of which start with "<%" and end with "%>". We normally give your file a .jsp extension, and typically install it in any ace you could place a normal Web page. Although what you write often looks more a regular html file than a servlet, behind the scenes, the JSP

page just gets converted to a normal servlet, with the static html simply being printed to the output stream associated with the servlet's service method. This is normally done the first time the page is requested, and developers can simply request the page themselves when first installing it if they want to be sure that the first real user doesn't get a momentary' delay when the JSP page is translated to a servlet and the servlet is compiled and loaded. Many Web servers let you define aliases that so that a URL that appears to reference an html file really points to a servlet or JSP page

Advantages of jsp:

- Separation of static from dynamic content.
- Write Once Run Anywhere.
- Recommended Web access layer for n-tier architecture.
- //Completely leverages the Servlet API.
- Platform independent.
- Reuse of components and tag libraries.
- Encapsulation of functionality.
- They have a better performance and scalability than ordinary CGI scripts, because they are persistent in memory and multi-threaded.
- They have built in support for HTTP sessions, which makes application Programming possible.
- They have full access to Java Technology-Network awareness, threads and Database connectivity-without the limitations of client side application applets.
- They are automatically recompiled when necessary.
- They exist in the ordinary Web server document space, no special URL mapping is required to address them.

How jsp works?

JSP pages exist in 3 forms or versions-

- JSP source code consists of text file with an extension of .jsp and contains a mix of HTML template code, Java language statements and JSP directives and actions that describe how to generate a web page to service a particular request.
- Java source code: the jsp container translates the jsp source code into the source code for an equivalent Java Servlet as needed.
- Compiled Java class: Like any other Java class, the generated servlet code is compiled into byte-codes in a .class file ready to be loaded and executed.

Java Script:

What is Java Script?

- Java Script is embedded into html.
- It is browser dependent.
- JavaScript depends on the web browser to support it. If the browser doesn't support it, JavaScript code will be ignored. Internet Explorer 3.0 and Netscape Navigator 2.0 onwards support JavaScript.
- It is an interpreted language, loosely typed, object based language.
- Java script is not Java

Data Validation:

JavaScript provides the means for basic data validation before it is sent to the server. Whether the values entered are correct or not or whether all the fields in a form are filled out or not can be checked before sending data to web server, if JavaScript is not used then data is sent to web server, and the web server would response with a message that the data sent to it is incorrect or incomplete Thus JavaScript ensures data validation and also reduces the network traffic. .

Financing Higher Education Loan

Reasons for Back end:

As Oracle 8.0 is object oriented and robust RDBMS & a number of tools available to work with it make it quite attractive as a backend. In a market place populated by computer companies with “proprietary” hardware, “proprietary” operating systems, “proprietary” databases and “proprietary” applications, ORACLE gives business users systems departments’ new control over their lives and futures. They are no longer bound to the database product of a single hardware vendor. ORACLE can run on nearly every kind of computer they can own. This is a basic revolution in the workplace and in application development, with consequences that will extend far into the future.

Java Data Base Connectivity (JDBC):-

In an enterprise computing which is largely the black art of managing huge databases? People associated with the enterprise need to be able to use and update the data easily, quickly and securely. The powerful Java Data Base Connectivity (JDBC) suit the java.sql.* package realizes java’s promise as a serious business programming tools. Java Data Base Connectivity is a standard SQL database access interface providing uniform access to a wide range of relational databases. It also provides a common base on which higher level tools and interfaces can be built. This comes with an “ODBC Bridge”. That bridge is a library, which implements JDBC in terms of ODBC standard API

There are many types of drivers used in connecting such as Native API Partly Java driver, a net protocol all java driver. The driver used here is JDBC-ODBC Bridge. JDBC-ODBC bridge plus ODBC driver –This is the crudest possible solution. Applets access data base using a combination of the JDBC-ODBC Bridge and an ODBC driver. This requires both drivers to be installed on the user’s computer- A very cumbersome solution for both internet and intranet users.

JDBCTM is a Java Tm 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

Financing Higher Education Loan

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. In other words, with the JDBC API, it isn't necessary to write one program to access a Sybase database, another program to access an MySQL database, another program to access an Informix database, and so on. One can write a single program using the JDBC API, and the program will be able to send SQL statements to the appropriate database. And, with an application written in the Java programming language, one also doesn't have to worry about writing different applications to run on different platforms. The combination of Java and JDBC lets a programmer write it once and run it anywhere. Java being robust, secure, easy to use, easy to understand, and automatically downloadable on a network, is an excellent language basis for database

Applications:

What is needed is a way for Java applications to talk to a variety of different databases. JDBC is the mechanism for doing this. JDBC extends what can be done in Java. For example, with Java and the JDBC API, it is possible to publish a web page containing an applet that uses information obtained from a remote database. Or an enterprise can use JDBC to connect all its employees (even if they are using a conglomeration of Windows, Macintosh, and UNIX machines) to one or more internal databases via an intranet. With more and more programmers using the Java programming language, the need for easy database access from Java is continuing to grow.

MIS managers like the combination of Java and JDBC because it makes disseminating information easy and economical. Businesses can continue to use their installed databases and access information easily even if it is stored on different database management systems. Development time for new applications is short. Installation and version control are greatly simplified. A programmer can write an application or an update once, put it on the server, and everybody has access to the latest version. And for businesses selling information services, Java and JDBC offer a better way of getting out information updates to external customers.

JDBC does the following things:

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

The following code fragment gives a basic example of these three steps:

```
Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");  
  
Connection con=DriverManager.getConnection("Jdbc: Odbc: dsname");  
  
Statement stmt=con.createStatement();
```

Connection:-

A connection object represents a connection with a database. A connection session includes the SQL statements that are executed and the results that are returned over the connection. A single application can have one or more connections with a Single database or it can have connections with many different databases.

Opening a Connection:

The standard way to establish a connection with a database is to call the method `DriverManager.getConnection`. This method takes a string containing a URL. The `DriverManager` class, referred to as the JDBC management layer, attempts to locate a driver that can connect to the database represented by the Driver classes, and when the method `getConnection` is called, it checks with each driver in the list until it finds one that can connect using this URL to actually establish the connection.

Sending Statement:

Financing Higher Education Loan

Once a connection is established, it is used to pass SQL statements to its underlying database. JDBC does not put any restrictions on the kinds of SQL statements that can be sent; this provides a great deal of flexibility, allowing the use of database-specific statements or even Non-SQL statements. It requires, however, that the user be responsible for making sure that the underlying database can process the SQL statements being sent and suffer the consequences if it cannot.

Driver Manager:

The Driver Manager class is the management layer of JDBC, working between the user and the drivers. It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver. In addition, the driver manager class attends to things like driver login time limits and the printing of log and tracing messages. The only method in this class that a general programmer needs to use directly is `DriverManager.getConnection`. As its name implies, this method establishes a connection to a database.

Why we need JDBC?

- ODBC is not appropriate for direct use from Java because it uses a C interfaces.
- ODBC is hard to learn. It mixes simple and advanced features together, and it has Complex options even for simple queries.

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.

SESSION:

This is the HttpSessionobject associated with the request. Recall that sessions are created automatically, so this variable is bound even if there was no incoming session reference. The one exception is if you use the session attribute of the page directive to turn sessions off, in which case attempts to reference the session variable cause errors at the time the JSP page is translated into a servlet.

Apache Tomcat:

Apache Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and Java Server Pages technologies. The Java Servlet and Java Server Pages specifications are developed by Sun under the Java Community Process. Tomcat is a web server that supports servlets and JSPs.

Tomcat comes with the Jasper compiler that compiles JSPs into servlets. Tomcat is available for commercial use under the ASF license from the Apache web site in both binary and source versions.

Apache Tomcat is developed in an open and participatory environment and released under the Apache Software License. Apache Tomcat powers numerous large scale, mission-critical web applications across a diverse range of industries and organizations. Different versions of Apache Tomcat are available for different versions of the Servlet and JSP specifications.

The Tomcat servlet engine is often used in combination with an Apache web server or other web servers. Tomcat can also function as an independent web server. Earlier in its development, the perception existed that standalone Tomcat was only suitable for development environments and other environments with minimal requirements for speed and transaction handling. However, that perception no longer exists; Tomcat is increasingly used as a standalone web server in high-traffic, high availability environments. Since its developers wrote Tomcat in Java, it runs on any operating system that has a JVM

Usage:

Java Web Applications, Java Mobile Applications using J2ME

Access:

Create a Java Web Application or Mobile Application using Net Beans Build and run project, this will automatically launch the Apache Tomcat as default.

Viewing Web Applications:

`http://localhost: 8080/WebpageName.jsp`

Directory Structure:

- The typical and default directory hierarchy of a Tomcat installation comprises the following: bin - startup, shutdown and other scripts and executable.
- common - common classes that Catalina and web applications can use.
- conf - XML files and related DTDs to configure Tomcat.
- logs - Catalina and application logs.
- server - classes used only by Catalina.
- shared - classes shared by all web applications.
- webapps - directory containing the web applications.
- work - temporary storage for files and directories.

A web application is basically a web site that:

- "Knows who you are"--it doesn't just give you static pages, it interacts with you.
- Can permanently change data (such as in a database).

Financing Higher Education Loan

- A web application can consist of multiple pieces.
- Static web pages (possibly containing forms).
- Servlets.
- JSP.

Tomcat organizes all these parts into a single directory structure for each web application.

The flow that takes place is:

- The user submits an HTML form.
- Tomcat finds the servlet based on the URL and the deployment descriptor.
- (web.xml) and passes the request to the servlet
- The servlet computes a response.
- The servlet writes an HTML page containing the response.
- The servlet forwards the response to the JSP.
- The JSP embeds the response in an HTML page
- Tomcat returns the HTML page to the user.

Status:

Tomcat is available at the Jakarta binary downloads page. The Tomcat server is a Java based Web Application container that was created to run Servlets and Java Server Pages (JSP) in Web applications. As part of Apache's open source Jakarta project, it has nearly become the industry accepted standard reference.

Implementation for both the Servlets and JSP API. Written by expert Servlets and JSP software architect and author James Goodwill, this column will feature introductory Web application development issues, Tomcat installation and configuration, deploying Web applications onto Tomcat, Struts and much more.

Apache Tomcat (formerly under the Apache Jakarta Project; Tomcat is now a top level project) is a web container developed at the Apache Software Foundation. Tomcat implements the servlet and the Java Server Pages (JSP) specifications from Sun

Financing Higher Education Loan

Microsystems, providing an environment for Java code to run in cooperation with a web server. It adds tools for configuration and management but can also be configured by editing configuration files that are normally XML-formatted. Because Tomcat includes its own HTTP server internally, it is also considered a standalone web server.

Tomcat is a web server that supports servlets and JSPs. Tomcat comes with the Jasper compiler that compiles JSPs into servlets. The Tomcat servlet engine is often used in combination with an Apache webserver or other web servers. Tomcat can also function as an independent web server

Earlier in its development, the perception existed that standalone Tomcat was only suitable for development environments and other environments with minimal requirements for speed and transaction handling. However, that perception no longer exists;

Tomcat is increasingly used as a standalone web server in high-traffic, high availability environments. Since its developers wrote Tomcat in Java, it runs on any operating system that has a JVM

Bean:

It's used to achieve reusability in java it is introduced to overcome the drawbacks of the traditional inheritance and object relations in JavaBean. The reusability is achieved using bean class. Bean is a component equivalent to ActiveX component of visual

Basic To create beans we must follow design pattern rules

- The class must be placed inside the package
- Class must be defined as public
- Variables in the class are defined as private to avoid direct accessibility of variables
- Variables must be defined in lower case
- A bean class can have public variable also, for every private variable there must be Set and Get methods. Set takes parameter which is used to assign the values for the variable. Get is used to retrieve the values. Every bean class should have implicit constructor (default constructor).

8.1 SYSTEM MAINTENANCE:

The term “software maintenance” is used to describe the software engineering activities that occur following delivery of a software product to the customer. The maintenance phase of the software life cycle is the time period in which a software product performs useful work. Maintenance activities involve making enhancement to software products, adapting products to new environments and correcting problems. Software product enhancement may involve providing new functional capabilities, improving user display and modes of interaction, and upgrading external documents. Adaptation of software to a new environment may involve moving the software to a different machine. Problem correction involves modification and revalidation of software to correct errors. The enhancement of this project can be accomplished easily. That is, any new functional capabilities can be added to the project by simply including the new module in the homepage and giving a hyperlink to that module. Adaptation of this project to a new environment is also performed easily. Even with the best quality assurance activities, it is likely that the customer will uncover defects in the software. Corrective maintenance changes the software to correct defects.

8.2 **COST AND BENEFIT ANALYSIS:**

Cost–benefit analysis (CBA), sometimes called benefit–cost analysis (BCA), is a systematic approach to estimating the strengths and weaknesses of alternatives that satisfy transactions, activities or functional requirements for a business. It is a technique that is used to determine options that provide the best approach for the adoption and practice in terms of benefits in labor, time and cost savings etc. The CBA is also defined as a systematic process for calculating and comparing benefits and costs of a project, decision government policy (hereafter, "project").

Broadly, CBA has two purposes:

1. To determine if it is a sound investment/decision (justification/feasibility),
2. To provide a basis for comparing projects. It involves comparing the total expected cost of each option against the total expected benefits, to see whether the benefits outweigh the costs, and by how much.

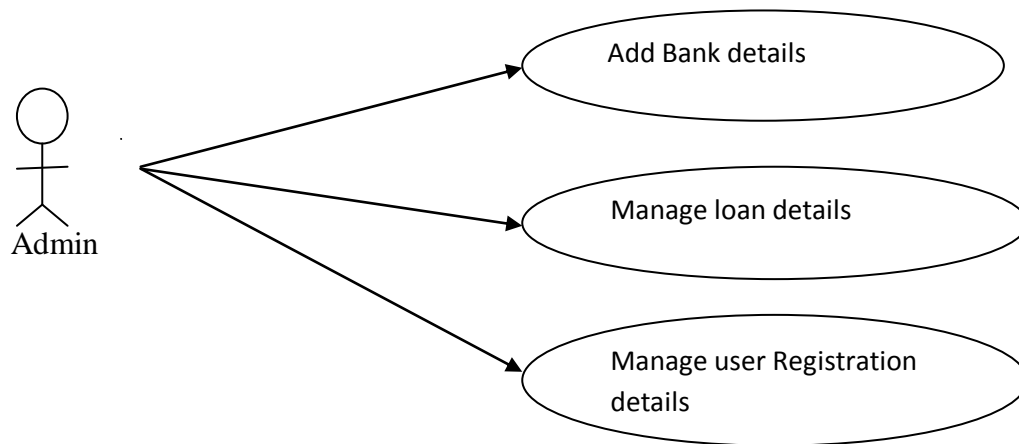
CBA is related to, but distinct from cost-effectiveness analysis. In CBA, benefits and costs are expressed in monetary terms, and are adjusted for the time value of money, so that all flows of benefits and flows of project costs over time (which tend to occur at different points in time) are expressed on a common basis in terms of their "net present value."

Closely related, but slightly different, formal techniques include cost-effectiveness analysis, cost–utility analysis, risk–benefit analysis, economic impact analysis, fiscal impact analysis, and Social return on investment (SROI) analysis.

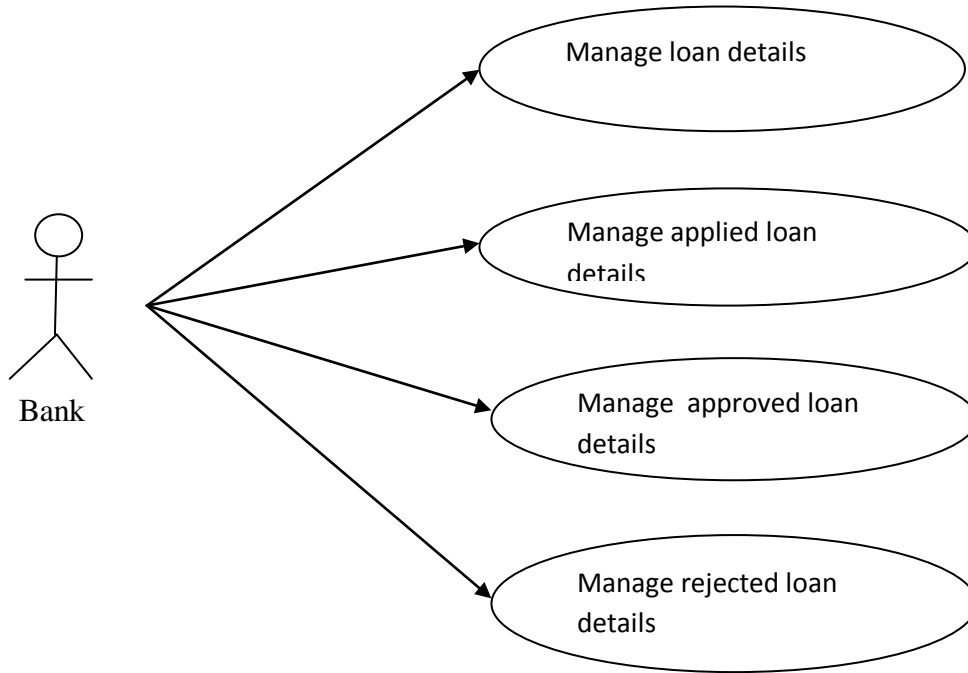
9. System Design:

9.1 Use case Diagram

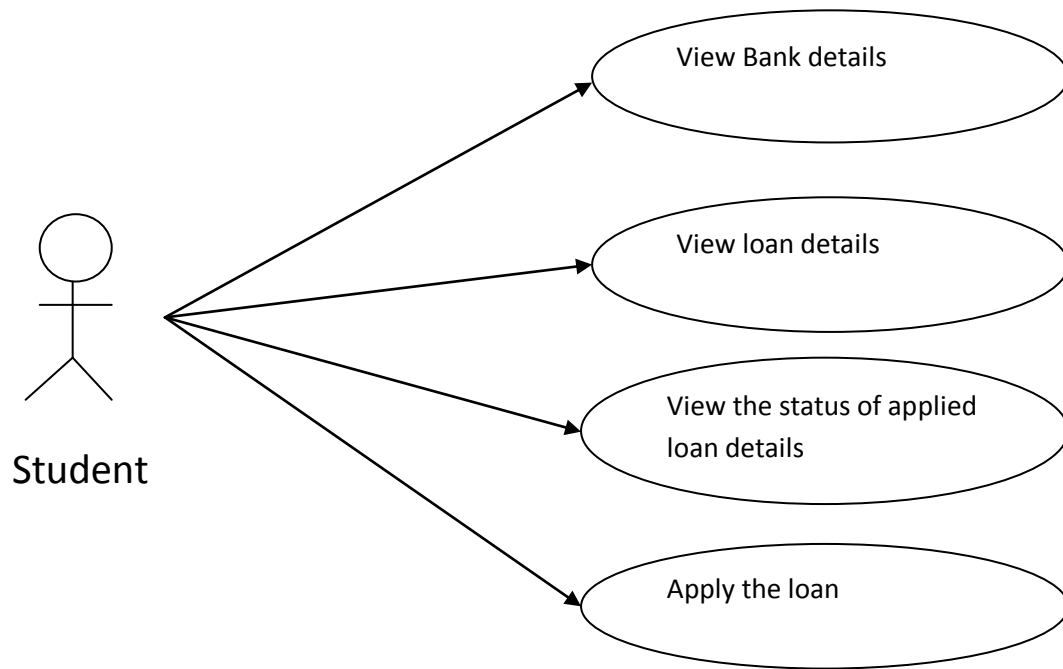
Admin Model:



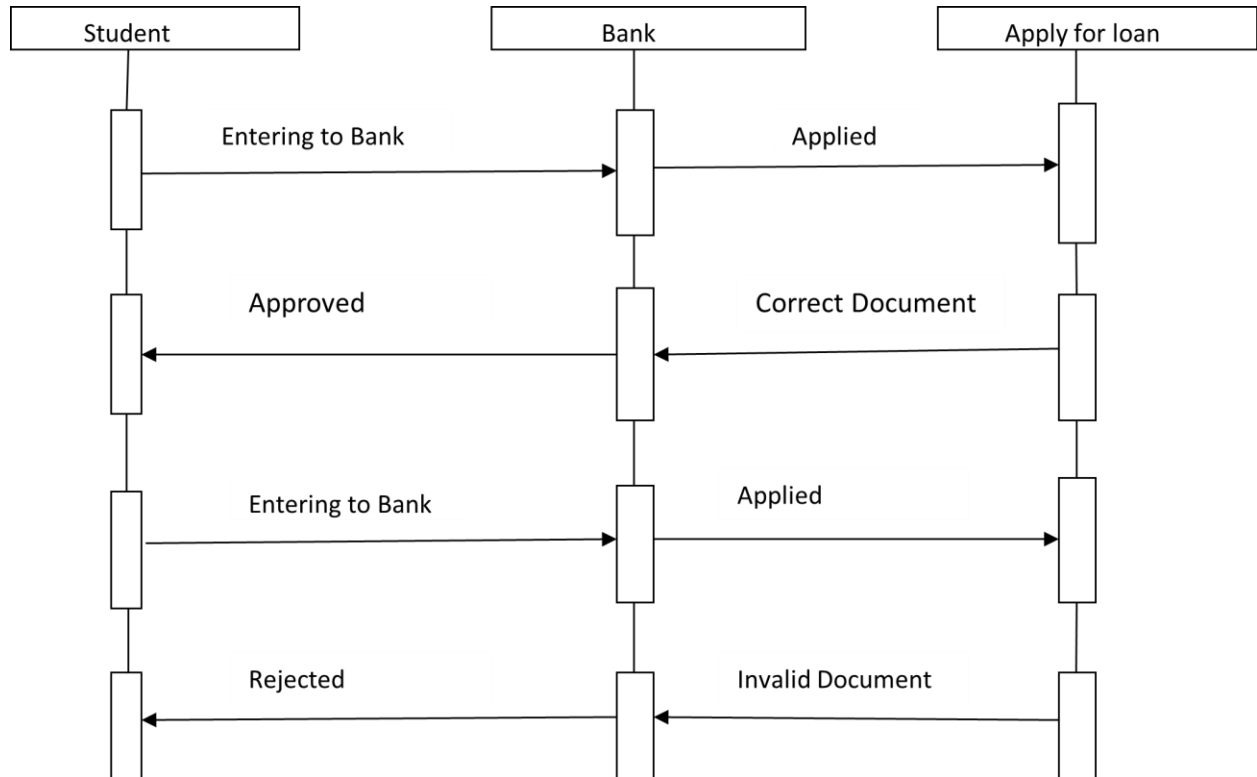
Bank Model:



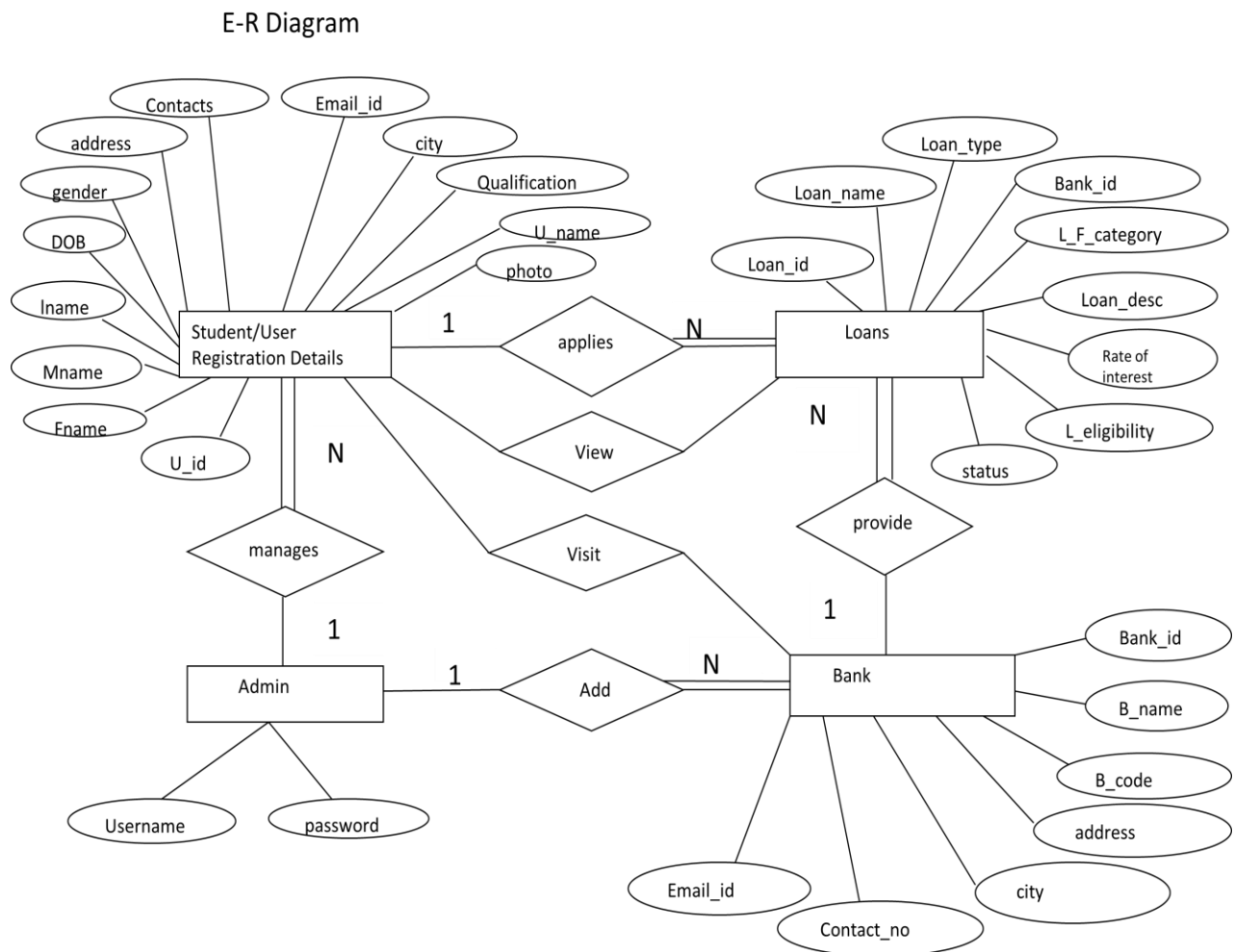
Student Model:



9.2 Sequence Diagram:

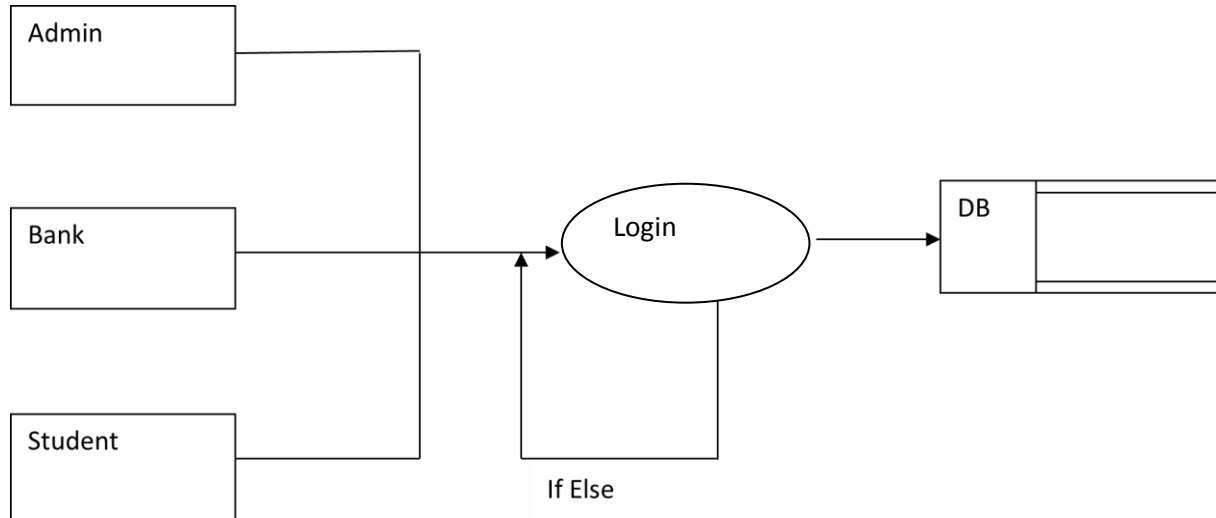


9.3 ER Diagram:

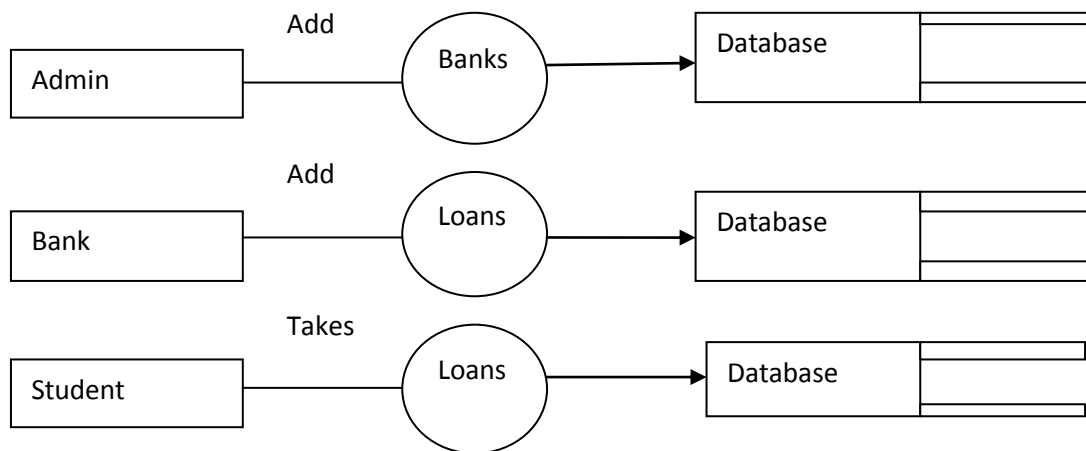


9.4 Data Flow Diagram :

0th Level Diagram:

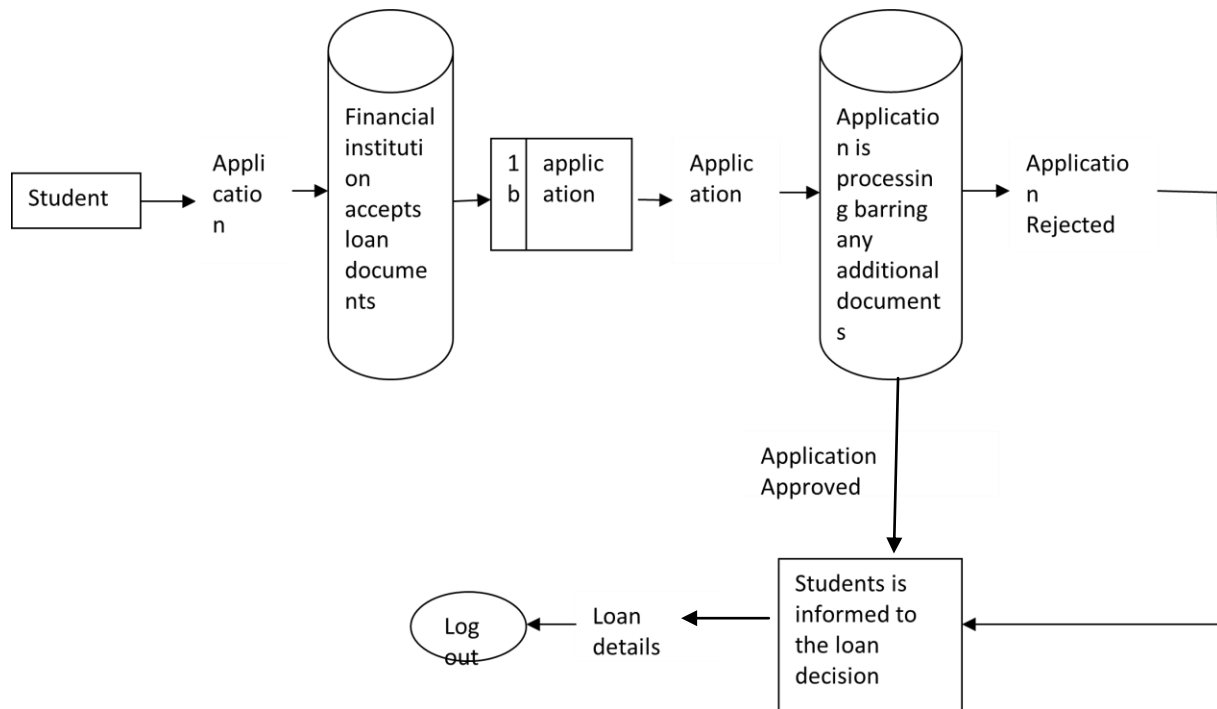


1st Level Diagram:







































Financing Higher Education Loan

2nd Level Diagram:













































10. Database Table:

1. Apply Loan Details:

	Field	Type	Collation	Attributes	Null	Default	Extra	Action					
<input type="checkbox"/>	<u>applyloan_id</u>	int(100)			No		auto_increment						
<input type="checkbox"/>	user_id	int(100)			No								
<input type="checkbox"/>	loan_id	int(100)			No								
<input type="checkbox"/>	desc	varchar(200)	latin1_swedish_ci		No								
<input type="checkbox"/>	status	varchar(200)	latin1_swedish_ci		No								
<input type="checkbox"/>	apply_date	varchar(100)	latin1_swedish_ci		No								

2. Bank Details:

	Field	Type	Collation	Attributes	Null	Default	Extra	Action					
<input type="checkbox"/>	<u>bank_id</u>	int(100)			No		auto_increment						
<input type="checkbox"/>	bank_name	varchar(100)	latin1_swedish_ci		No								
<input type="checkbox"/>	bank_code	varchar(100)	latin1_swedish_ci		No								
<input type="checkbox"/>	address	varchar(200)	latin1_swedish_ci		No								
<input type="checkbox"/>	city	varchar(100)	latin1_swedish_ci		No								
<input type="checkbox"/>	contact_no	varchar(20)	latin1_swedish_ci		No								
<input type="checkbox"/>	email_id	varchar(20)	latin1_swedish_ci		No								

Financing Higher Education Loan

3. Loan details:









































	Field	Type	Collation	Attributes	Null	Default	Extra	Action							
<input type="checkbox"/>	<u>loan_id</u>	int(100)			No		auto_increment								
<input type="checkbox"/>	loan_name	varchar(100)	latin1_swedish_ci		No										
<input type="checkbox"/>	loan_type	varchar(100)	latin1_swedish_ci		No										
<input type="checkbox"/>	bank_id	int(100)			No										
<input type="checkbox"/>	loan_for_cate	varchar(200)	latin1_swedish_ci		No										
<input type="checkbox"/>	loan_desc	varchar(200)	latin1_swedish_ci		No										
<input type="checkbox"/>	rate_of_interest	int(100)			No										
<input type="checkbox"/>	loan_amount	int(11)			No										
<input type="checkbox"/>	loan_eligibility	varchar(100)	latin1_swedish_ci		No										
<input type="checkbox"/>	status	varchar(200)	latin1_swedish_ci		No										

4. Login:









































































































	Field	Type	Collation	Attributes	Null	Default	Extra	Action							
<input type="checkbox"/>	user_name	varchar(100)	latin1_swedish_ci		No										
<input type="checkbox"/>	password	varchar(100)	latin1_swedish_ci		No										
<input type="checkbox"/>	type	varchar(100)	latin1_swedish_ci		No										
<input type="checkbox"/>	hint_qst	varchar(100)	latin1_swedish_ci		No										
<input type="checkbox"/>	hint_ans	varchar(100)	latin1_swedish_ci		No										
<input type="checkbox"/>	status	varchar(100)	latin1_swedish_ci		No										

Financing Higher Education Loan

5. Reject loan:









































	Field	Type	Collation	Attributes	Null	Default	Extra	Action							
<input type="checkbox"/>	<u>rejectloan_id</u>	int(40)			No		auto_increment								
<input type="checkbox"/>	applyloan_id	int(40)			No										
<input type="checkbox"/>	reason	varchar(200)	latin1_swedish_ci		No										
<input type="checkbox"/>	bank_id	int(40)			No										
<input type="checkbox"/>	reject_date	varchar(100)	latin1_swedish_ci		No										

6. User details:

	Field	Type	Collation	Attributes	Null	Default	Extra	Action							
<input type="checkbox"/>	<u>user_id</u>	int(100)			No		auto_increment								
<input type="checkbox"/>	first_name	varchar(100)	latin1_swedish_ci		No										
<input type="checkbox"/>	mid_name	varchar(100)	latin1_swedish_ci		No										
<input type="checkbox"/>	last_name	varchar(100)	latin1_swedish_ci		No										
<input type="checkbox"/>	DOB	varchar(100)	latin1_swedish_ci		No										
<input type="checkbox"/>	gender	varchar(20)	latin1_swedish_ci		No										
<input type="checkbox"/>	address	varchar(200)	latin1_swedish_ci		No										
<input type="checkbox"/>	contact_no	varchar(20)	latin1_swedish_ci		No										
<input type="checkbox"/>	email_id	varchar(20)	latin1_swedish_ci		No										
<input type="checkbox"/>	city	varchar(100)	latin1_swedish_ci		No										
<input type="checkbox"/>	qualification	varchar(100)	latin1_swedish_ci		No										
<input type="checkbox"/>	u_name	varchar(100)	latin1_swedish_ci		No										
<input type="checkbox"/>	photo	varchar(200)	latin1_swedish_ci		No										

Financing Higher Education Loan

7. User Document:

	Field	Type	Collation	Attributes	Null	Default	Extra	Action						
	<u>doc_id</u>	int(40)			No		auto_increment							
	user_id	int(100)			No									
	doc_name	varchar(100)	latin1_swedish_ci		No									
	doc	varchar(100)	latin1_swedish_ci		No									
	upload_date	varchar(100)	latin1_swedish_ci		No									

11. CODING:

Sample Code:

Bean Program:

Bean Program:

```
package loan;
import java.sql.*;
public class education
{
    private Connection con;
    public Statement stmt;
    public String getconn()
    {
        try
        {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            con=DriverManager.getConnection("Jdbc:Odbc:loan");
            stmt=con.createStatement();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
        return " ";
    }
}
```


Financing Higher Education Loan

Form:

```
<jsp:include page="header.jsp"></jsp:include>

<body>

    <div id="wrapper">

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

        <!-- /. NAV TOP -->

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

        <!-- /. NAV SIDE -->

        <div id="page-wrapper">

            <div id="page-inner">

                <div class="row">

                    <div class="col-md-12">

                        <h1 class="page-head-line">User Details Form</h1>

                    </div>

                </div>

                <!-- /. ROW -->

            <div class="row">

                <div class="col-md-6 col-sm-6 col-xs-12">

                    <div class="panel panel-info">

                        <div class="panel-body">

<form action="user_insert.jsp" method="post" name="form1" id="formID">

                            <p>&nbsp;</p>

                            <p>&nbsp;</p>

                        </div>

                    </div>

                </div>

            </div>

        </div>

    </div>

</body>
```

Financing Higher Education Loan

```
<table width="431" border="0" align="center">

<tr>

<td height="31" colspan="2"><div align="center">User details</div></td>

</tr>

<tr>

<td width="105" height="31">First name </td>

<td width="316"><input name="fname" type="text" id="fname"
class="validate[required,custom[onlyLetter]]"></td>

</tr>

<tr>

<td height="33">Middle name </td>

<td><input name="mname" type="text" id="mname" class="validate[required,custom[onlyLetter]]"></td>

</tr>

<tr>

<td height="36">Last name </td>

<td><input name="lname" type="text" id="lname" class="validate[required,custom[onlyLetter]]"></td>

</tr>

<tr>

<td height="32">DOB</td>

<td><input name="dob" type="text" id="dob" class="validate[required,custom[date]]"></td>

</tr>

<tr>

<td height="41">Gender</td>

<td><input name="gender" id="gender" type="radio" value="male">
```

Financing Higher Education Loan

Male

☐

Female</td>

</tr>

<tr>

<td>Address</td>

<td><textarea name="u_ad" id="u_ad" class="validate[required]"></textarea></td>

</tr>

<tr>

<td height="35">Contact No </td>

<td><input name="u_cn" type="text" id="u_cn" class="validate[required,custom[mobile]]"></td>

</tr>

<tr>

<td height="35">Email ID </td>

<td><input name="u_eid" type="text" id="u_eid" class="validate[required,custom[email]]"></td>

</tr>

<tr>

<td height="38">City</td>

<td><input name="u_city" type="text" id="u_city" class="validate[required]"></td>

</tr>

<tr>

<td height="41">Qualification</td>

<td><input name="qual" type="text" id="qual" class="validate[required]"></td>

</tr>

Financing Higher Education Loan

```
<tr>

<td height="46">U name</td>

<td><input name="uname" type="text" id="uname" class="validate[required,custom[onlyLetter]]"></td>

</tr>

<tr>

<td height="50">Photo</td>

<td><input type="file" name="file"></td>

</tr>

<tr>

<td height="70" colspan="2"><div align="center">

<input type="submit" name="Submit" value="Submit" class="btn btn-primary">

<input type="reset" name="Reset" value="Reset" class="btn btn-danger">

s  </div></td>

</tr>

</table>

<p>&nbsp;</p>

</form>

</div>

</div> </div> </div>

<!--/.ROW-->

</div>

<!-- /. PAGE INNER -->

</div>

<!-- /. PAGE WRAPPER -->
```

Financing Higher Education Loan

</div>

<!-- /. WRAPPER -->

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

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

</body>

</html>

Financing Higher Education Loan

Insert:

```
<% @page import="java.sql.*"%>

<% @page import="financing.dbconnect"%>

<jsp:useBean id="s" class="financing.dbconnect"/>

<jsp:getProperty name="s" property="conn"/>

<%

String f_name=request.getParameter("fname");

String m_name=request.getParameter("mname");

String l_name=request.getParameter("lname");

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

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

String address=request.getParameter("u_ad");

String cont_no=request.getParameter("u_cn");

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

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

String qualification=request.getParameter("qual");

String u_name=request.getParameter("uname");

String photo=request.getParameter("file");

intk=s.stmt.executeUpdate("insertintouser_detailsvalues(null,'" +f_name+"','"+m_name+"','"+l_name+"','"+dob+"',
'"+gender+"','"+address+"','"+cont_no+"','"+email+"','"+city+"','"+qualification+"','"+ u_name+"','"+photo+"')");

%>

<script>

    alert("User details inserted.....");

    document.location="user_view.jsp";

</script>
```

Financing Higher Education Loan

View:

```
<jsp:include page="header.jsp"></jsp:include>

<body>

    <div id="wrapper">

        <link href="../../temp/css/bootstrap.min.css" rel="stylesheet" />

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

        <!-- /. NAV TOP -->

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

        <!-- /. NAV SIDE -->

        <div id="page-wrapper">

            <div id="page-inner">

                <div class="row">

                    <div class="col-md-12">

                        <h1 class="page-head-line">User View Details</h1>

                    </div>

                </div>

                <!-- /. ROW -->

                <div class="row">

                    <div class="col-md-8">

                        <div class="table-responsive">

                            <p>&nbsp;</p>

                            <table class="table table-striped table-bordered table-hover" id="sample_1">

                                <thead>

                                    <tr>

                                        <th width="58" height="34">User id</th>

                                        <th width="85">First name</th>

                                        <th width="73">Mid name</th>
```

Financing Higher Education Loan

```
<th width="84">Last name</th>

<th width="66">DOB</th>

<th width="54">Gender</th>

<th width="63">Address</th>

<th width="87">Contact no</th>

<th width="70">Email id</th>

<th width="49">City</th>

<th width="100">Qualification</th>

<th width="62">U name</th>

<th width="78">Photo</th>

</tr>

</thead>

</tbody>

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

<% @page import="financing.dbconnect"%>

<jsp:useBean id="s" class="financing.dbconnect"/>

<jsp:getProperty name="s" property="conn"/>

<%

ResultSet rs=s.stmt.executeQuery("select * from user_details");

while(rs.next())

{

int uid=rs.getInt("user_id");

%>

<tr>

<td height="57">&nbsp;<%=uid%></td>

<td>&nbsp;<%=rs.getString("first_name")%></td>

<td>&nbsp;<%=rs.getString("mid_name")%></td>

<td>&nbsp;<%=rs.getString("last_name")%></td>
```


Financing Higher Education Loan

```
<td>&nbsp;<%=rs.getString("DOB")%></td>
<td>&nbsp;<%=rs.getString("gender")%></td>
<td>&nbsp;<%=rs.getString("address")%></td>
<td>&nbsp;<%=rs.getString("contact_no")%></td>
<td>&nbsp;<%=rs.getString("email_id")%></td>
<td>&nbsp;<%=rs.getString("city")%></td>
<td>&nbsp;<%=rs.getString("qualification")%></td>
<td>&nbsp;<%=rs.getString("u_name")%></td>
<td>&nbsp;" width="100" height="150"></td>
</tr>
<%
}
%>
</tbody>
</table>
</div> </div> </div>
<!--/.Row-->
<!--/.ROW-->
</div>
<!-- /. PAGE INNER -->
</div>
<!-- /. PAGE WRAPPER -->
</div>
<!-- /. WRAPPER -->
<script src="../temp/js/jquery-1.8.3.min.js"></script>
<script type="text/javascript" src="../temp/js/jquery.dataTables.js"></script>
<script type="text/javascript" src="../temp/js/DT_bootstrap.js"></script>
<script src="../temp/js/dynamic-table.js"></script>
<jsp:include page="footer.jsp"></jsp:include>
```

Delete:

```
<% @page import="java.sql.*"%>

<% @page import="financing.dbconnect"%>

<jsp:useBean id="s" class="financing.dbconnect"/>

<jsp:getProperty name="s" property="conn"/>

<%

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

int k=s.stmt.executeUpdate("delete from user_details where user_id="+uid+"");

%>

<script>

    alert("Deleted.....");

    document.location="user_view.jsp";

</script>
```

Update:

Update1:

```
<% @page import="java.sql.*"%>

<% @page import="financing.dbconnect"%>

<jsp:useBean id="s" class="financing.dbconnect"/>

<jsp:getProperty name="s" property="conn"/>

<%

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

ResultSet rs=s.stmt.executeQuery("select * from user_details where user_id="+uid+" ");

rs.next();

%>

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

<body>

    <div id="wrapper">

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

        <!-- /. NAV TOP -->

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

        <!-- /. NAV SIDE -->

        <div id="page-wrapper">

            <div id="page-inner">

                <div class="row">

                    <div class="col-md-12">
```

Financing Higher Education Loan

```
<h1 class="page-head-line">User Details Form</h1>

</div>

</div>

<!-- /. ROW -->

<div class="row">

<div class="col-md-6 col-sm-6 col-xs-12">

<div class="panel panel-info">

<div class="panel-body">

<form action="user_edit_2.jsp" method="post" name="form1" id="formID">

<p>&nbsp;</p>

<p>&nbsp;</p>

<table width="431" border="0" align="center">

<tr>

<td height="33" colspan="2"><div align="center">User details</div></td>

</tr>

<tr>

<td width="98" height="39">User id </td>

<td width="323"><input name="uid" type="text" id="uid" value="<%=uid%>" readonly=""
class="validate[required,custom[onlyNumber]]"></td>

</tr>

<tr>

<td height="34">First name </td>
```

Financing Higher Education Loan

<td><input name="fname" type="text" id="fname" value="<%=rs.getString("first_name")%>" class="validate[required,custom[onlyLetter]]"></td>

</tr>

<tr>

<td height="42">Middle name </td>

<td><input name="mname" type="text" id="mname" value="<%=rs.getString("mid_name")%>" class="validate[required,custom[onlyLetter]]"></td>

</tr>

<tr>

<td height="38">Last name </td>

<td><input name="lname" type="text" id="lname" value="<%=rs.getString("last_name")%>" class="validate[required,custom[onlyLetter]]"></td>

</tr>

<tr>

<td height="35">DOB</td>

<td><input name="dob" type="text" id="dob" value="<%=rs.getString("DOB")%>" class="validate[required,custom[date]]"></td>

</tr>

<tr>

<td height="38">Gender</td>

<td><input name="gender" id="gender" type="radio" value="male">

Male

<input name="gender" id="gender" type="radio" value="female">

Female</td>

Financing Higher Education Loan

</tr>

<tr>

<td height="87">Address</td>

<td><textarea name="u_ad" id="u_ad" class="validate[required]"><%=rs.getString("address")%></textarea></td>

</tr>

<tr>

<td height="36">Contact No </td>

<td><input name="u_cn" type="text" id="u_cn" value="<%=rs.getString("contact_no")%>" class="validate[required,custom[mobile]]"></td>

</tr>

<tr>

<td height="46">Email ID </td>

<td><input name="u_eid" type="text" id="u_eid" value="<%=rs.getString("email_id")%>" class="validate[required,custom[email]]"></td>

</tr>

<tr>

<td height="40">City</td>

<td><input name="u_city" type="text" id="u_city" value="<%=rs.getString("city")%>" class="validate[required,custom[onlyLetter]]"></td>

</tr>

<tr>

<td height="41">Qualification</td>

Financing Higher Education Loan

```
<td><input    name="qual"    type="text"    id="qual"    value="<%=rs.getString("qualification")%>"
class="validate[required,custom[onlyLetter]]"></td>
```

```
</tr>
```

```
<tr>
```

```
<td height="58">U name</td>
```

```
<td><input    name="uname"    type="text"    id="uname"    value="<%=rs.getString("u_name")%>"
class="validate[required,custom[onlyLetter]]"></td>
```

```
</tr>
```

```
<tr>
```

```
<td height="60">Photo</td>
```

```
<td><input type="file" name="file" value="<%=rs.getString("photo")%>"></td>
```

```
</tr>
```

```
<tr>
```

```
<td height="58" colspan="2"><div align="center">
```

```
<input type="submit" name="Submit" value="Submit" class="btn btn-primary">
```

```
<input type="reset" name="Reset" value="Reset" class="btn btn-danger">
```

```
</div></td>
```

```
</tr>
```

```
</table>
```

```
<p>&nbsp;</p>
```

```
</form>
```

```
</div> </div>
```

```
</div> </div>
```

Financing Higher Education Loan

<!--/.ROW--> </div>

<!-- /. PAGE INNER --> </div>

<!-- /. PAGE WRAPPER --> </div>

<!-- /. WRAPPER -->

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

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

</body>

</html>

Financing Higher Education Loan

Update2:

```
<% @page import="java.sql.*"%>

<% @page import="financing.dbconnect"%>

<jsp:useBean id="s" class="financing.dbconnect"/>

<jsp:getProperty name="s" property="conn"/>

<%

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

String f_name=request.getParameter("fname");

String m_name=request.getParameter("mname");

String l_name=request.getParameter("lname");

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

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

String address=request.getParameter("u_ad");

String cont_no=request.getParameter("u_cn");

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

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

String qualification=request.getParameter("qual");

String u_name=request.getParameter("uname");

String photo=request.getParameter("file");

intk=s.stmt.executeUpdate("Updateuser_detailsssetfirst_name='"+f_name+"',mid_name='"+m_name+"',l
ast_name='"+l_name+"',DOB='"+dob+"',gender='"+gender+"',address='"+address+"',contact_no='"+con
t_no+"',email_id='"+email+"',city='"+city+"',qualification='"+qualification+"',u_name='"+
u_name+"',photo='"+photo+"' where user_id='"+uid+" " ");

%>

<script>
```

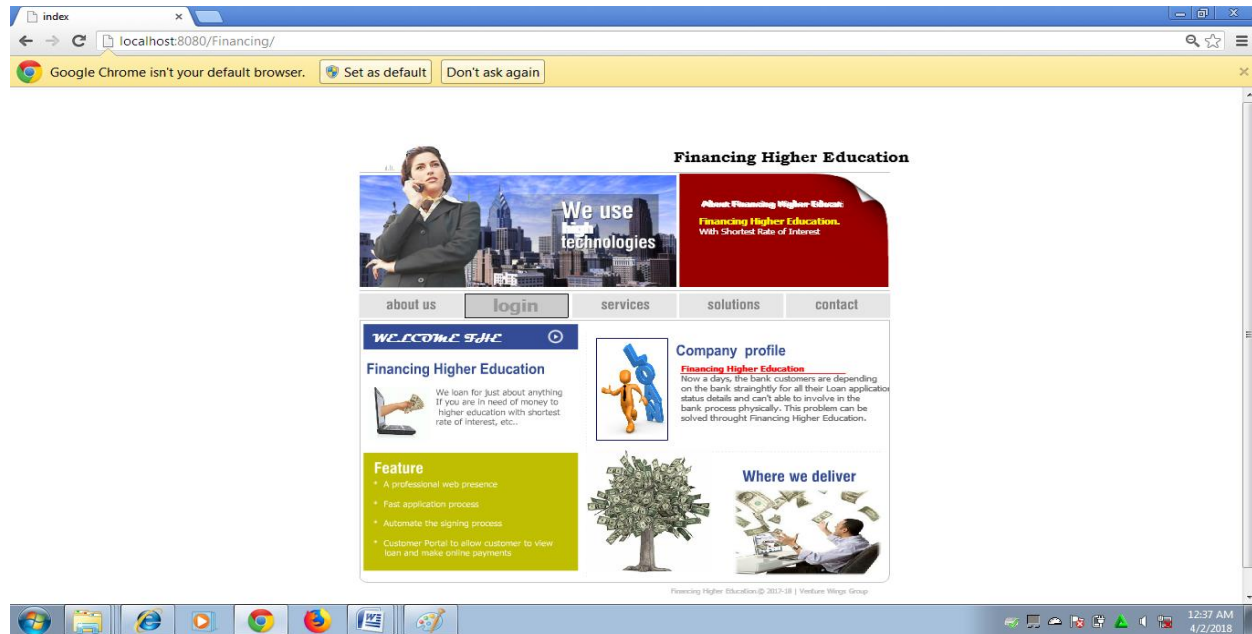
Financing Higher Education Loan

```
alert("User details edited.....");  
  
document.location="user_view.jsp";  
  
</script>
```

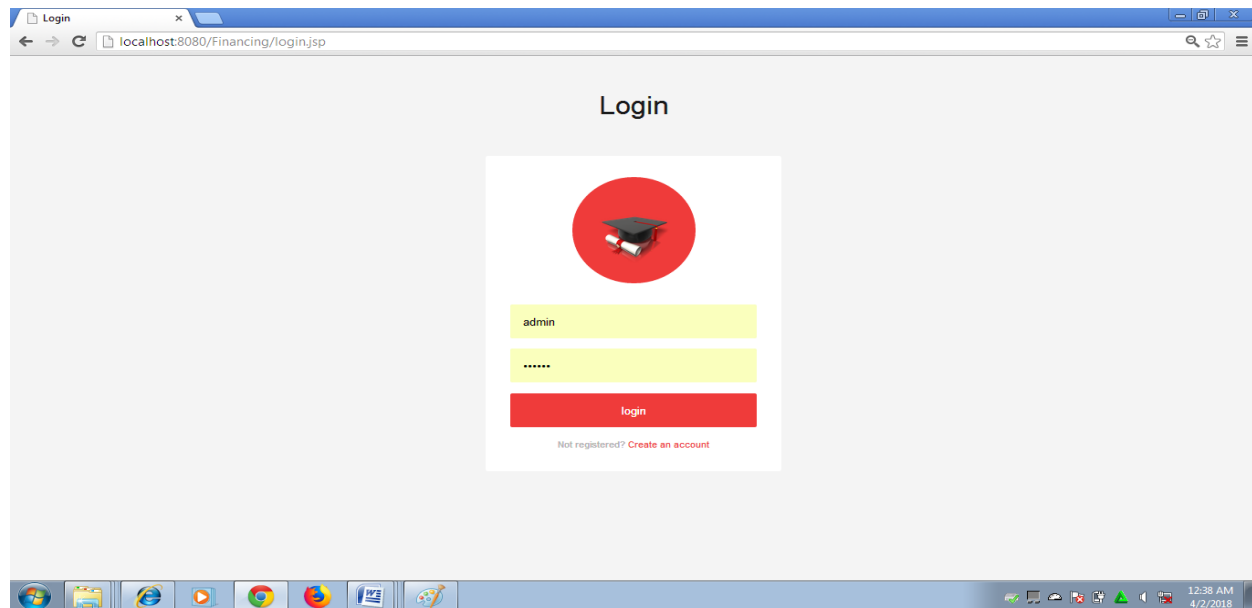
Financing Higher Education Loan

12. OUTPUT SCREEN:

Home Page:

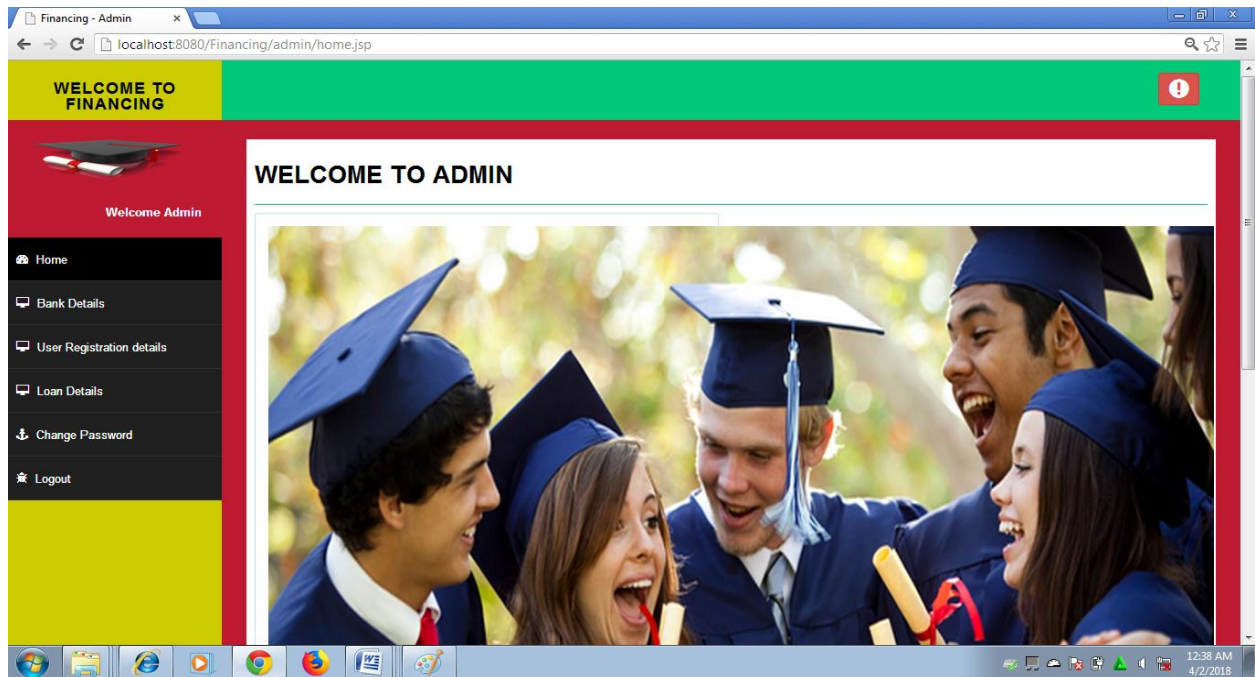


LoginPage:

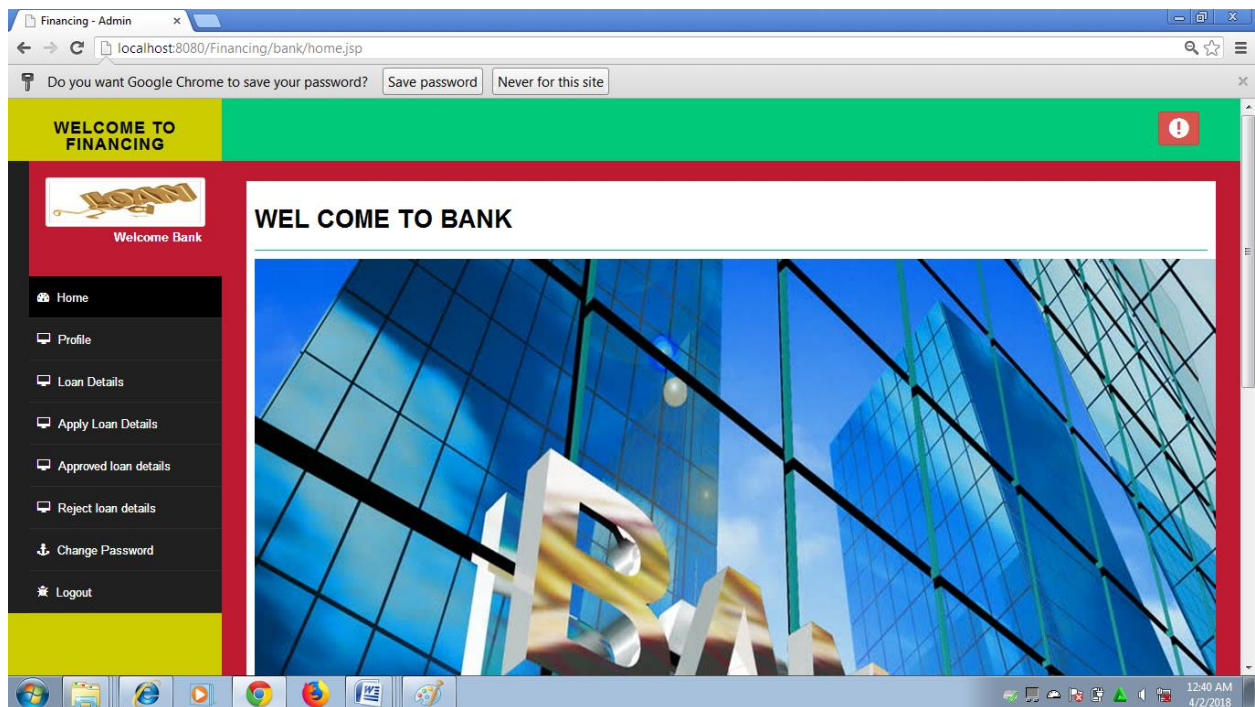


Financing Higher Education Loan

Admin Module:

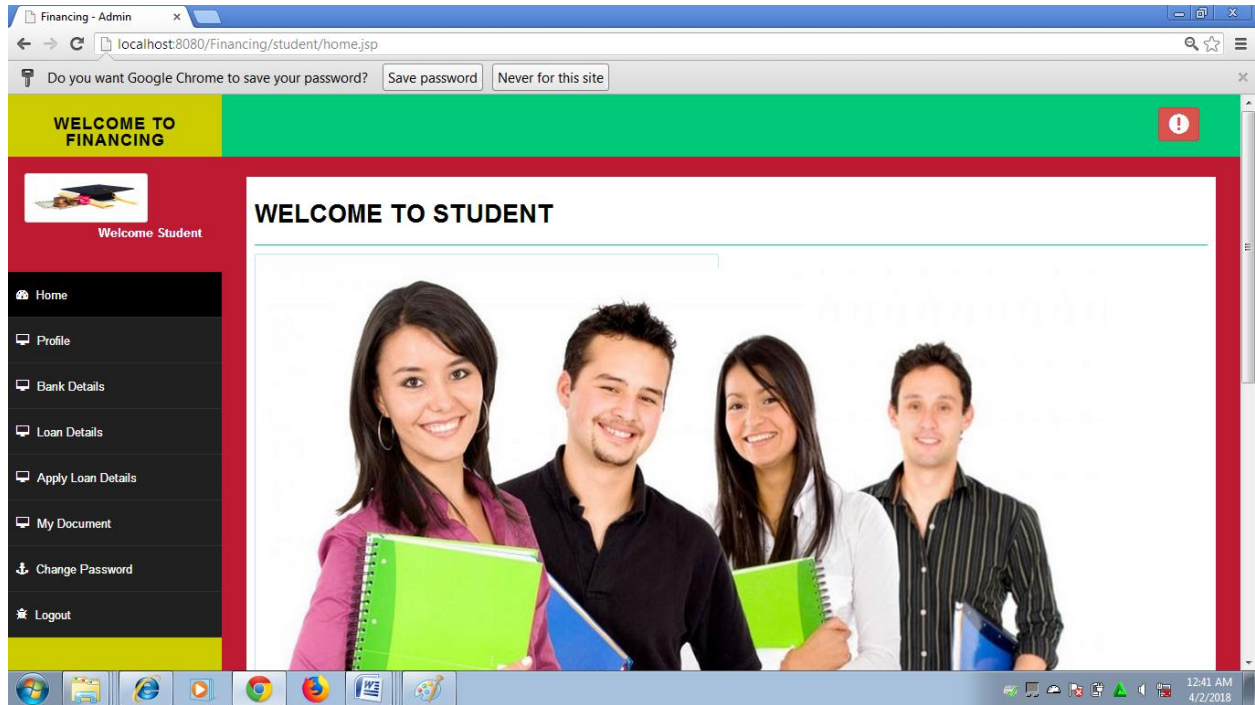


Bank Module:



Financing Higher Education Loan

Student Module:



13. TESTING:

Testing is a process of executing a program with the intent of finding an error. Testing is a crucial element of software quality assurance and presents ultimate review of specification, design and coding.

System Testing is an important phase. Testing represents an interesting anomaly for the software. Thus a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

A good test case is one that has a high probability of finding an as undiscovered error. A successful test is one that uncovers an as undiscovered error.

Testing Objectives:

- Testing is a process of executing a program with the intent of finding an error
- A good test case is one that has a probability of finding an as yet undiscovered error
- A successful test is one that uncovers an undiscovered error

Testing Principles:

- All tests should be traceable to end user requirements
- Tests should be planned long before testing begins
- Testing should begin on a small scale and progress towards testing in large Exhaustive testing is not possible
- To be most effective testing should be conducted by a independent third party

The primary objective for test case design is to derive a set of tests that has the highest likelihood for uncovering defects in software. To accomplish this objective two different categories of test case design techniques are used. They are

- White box testing.
- Black box testing.

White-box testing:

White box testing focus on the program control structure. Test cases are derived to ensure that all statements in the program have been executed at least once during testing and that all logical conditions have been executed.

Block-box testing:

Black box testing is designed to validate functional requirements without regard to the internal workings of a program. Black box testing mainly focuses on the information domain of the software, deriving test cases by partitioning input and output in a manner that provides thorough test coverage. Incorrect and missing functions, interface errors, errors in data structures, error in functional logic are the errors falling in this category.

Testing strategies:

A strategy for software testing must accommodate low-level tests that are necessary to verify that all small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements.

Testing Information flow:

Information flow for testing flows the pattern. Two class of input provided to test the process. The software configuration includes a software requirements specification, a design specification and source code.

Unit testing:

Unit testing is essential for the verification of the code produced during the coding phase and hence the goal is to test the internal logic of the modules. Using the detailed design description as a guide, important paths are tested to uncover errors within the boundary of the modules. These tests were carried out during the programming stage itself. All units of Vienna SQL were successfully tested.

Integration testing:

Integration testing focuses on unit tested modules and build the program structure that is dictated by the design phase.

System testing:

System testing tests the integration of each module in the system. It also tests to find discrepancies between the system and it's original objective, current specification and system documentation. The primary concern is the compatibility of individual modules. Entire system is working properly or not will be tested here, and specified path ODBC connection will correct or not, and giving output or not are tested here these verifications and validations are done by giving input values to the system and by comparing with expected output. Top-down testing implementing here.

Acceptance Testing:

This testing is done to verify the readiness of the system for the implementation. Acceptance testing begins when the system is complete. Its purpose is to provide the end user with the confidence that the system is ready for use. It involves planning and execution of functional tests, performance tests and stress tests in order to demonstrate that the implemented system satisfies its requirements.

Tools to special importance during acceptance testing include:

- Test coverage Analyzer – records the control paths followed for each test case.
- Timing Analyzer – also called a profiler, reports the time spent in various regions of the code are areas to concentrate on to improve system performance.
- Coding standards – static analyzers and standard checkers are used to inspect code for deviations from standards and guidelines.

Financing Higher Education Loan

Test Report:

	<u>Test case</u>	<u>Expected Output</u>	<u>Observed Output</u>	<u>Remark</u>
1	Login, By blank Username.	User name should not be blank.	The user name should not be blank.	OK
2	Login, By blank Password.	Password should not be blank.	The password should not be blank.	OK
3	User name and password both blank.	User name and password should not be blank.	The User name and password should not be blank.	OK
4	Wrong entry of user name and password.	Invalid user name and password.	The user name and password is invalid.	OK
5	Wrong entry of name.	Characters only.	Enter Characters only.	OK
6	Wrong entry of Mobile no.	Numbers only.	Enter Numbers only, minimum and maximum 10 digits.	OK
7	NULL value for ID (primary key)	Null not allowed.	ID should not be null, Enter ID	OK
8	Wrong entry of email id.	Invalid email address.	Invalid email address.	OK

14. FUTURE ENHANCEMENT:

- Online through deposit the amount

15. Conclusion

- Faster processing
- Time saving
- Since data is available at one place, can be accessed any where any time.

16. **Bibliography:**

- The Complete Reference.....Helbert schildt published
fifth edition
- Server side programming withAptech J2EE book
JSP & Servlet

Websites:

1. <http://Java.sun.com/j2ee/faq/html>
2. <http://Java.sun.com/products/jdbc/reference/faqs/index.html>
3. <http://www.apptech.com/~hall/java/servlet-Tutorial>