1. **INTRODUCTION**

In finance, a **loan** is the lending of money from one individual, organization or entity to another individual, organization or entity. A loan is a debt provided by an organization or individual to another entity at an interest rate, and evidenced by a promissory note which specifies, among other things, the principal amount of money borrowed, the interest rate the lender is charging, and date of repayment. The loan is generally provided at a cost, referred to as interest on the debt, which provides an incentive for the lender to engage in the loan. In a legal loan, each of these obligations and restrictions is enforced by contract, which can also place the borrower under additional restrictions known as loan covenants.

* 1. **Types of loans:**

There are many types of loans. Some of them are as follows:-

* ***Student loan***:-

Student loan is money owed to pay for educational expenses.

* ***Auto loan***:-

Auto loan is money owed to purchase automobile (Car loan is a type of automobile loan).

* ***Home loan***:-

Home loan is money owed to buy a house.

* ***Small business loan***:-

Small business loan is money owed to start a new small business.

* ***Personal loan***:-

Personal loan is money owed by the person to fulfil his personal needs. It is based on the borrower’s credit history and ability to repay the loan.

* ***Gold loan***:-

Gold loan is the money which is provided by lender against the gold. By pledging gold ornaments, coins, biscuits, bars, etc. the lender provides borrower with liquidity at predetermined rate of interest.

1. **CURRENT SYSTEM**

The call for loans is increasing day by day. According to a survey, the major portion of population applying for loan belongs to the middle class families. There are a lot of difficulties and hardships faced by people. These includes:

* **Paper work and long procedure:**

A lot of paper work and many long procedures don’t allow the appliers to fulfil their ardent need for money. For instance, it takes a period of few days for the loan to approve and sanction.

* **Bank fraud:**

A pressing concern for the banking regulator is the increased number of fraudulent transactions at Indian banks. What’s adding to the concerns is that banks often seem reluctant to report these cases. In the last five years, the volume of bank fraud has increased by 19.6% to 5,064 cases.

* **Records manipulation:**

Apart from the cost of funds, the other factors include overheads such as the cost of running branches and employee wages, the so-called negative carry on the cash reserve ratio and the level of bad loans. There is no cap on the spread between the base rate and the actual rate that the bank would charge its customer as long as they could justify the spread.

* **High rate of interest:**

For an instance, a person is repaying a home loan and the repayment time is scheduled for 10 years. The first few years of the repayment follows only paying the rate of interest and later repaying the principle amount.

1. **PROJECT PROPOSAL**

The following is the Proposal for the project submitted to the project stakeholders.

**EMPLOYEE LOAN**

**PROJECT PROPOSAL**

**DATE:**

**PREPARED BY:** Anubhav Singh

Yashika Sharma

Mahima Yogi

(IIPS, DAVV, Indore)

1. **AIM OF PROJECT**

This project is aimed to develop web application automate in managing the employee profile and their loans in an organisation. The aim of the project is to maintain the employee, department, designation, information. It also provide a structured method of managing employee’s loan type and billing details etc. A web based solution will ensure that the management with the intranet access will be able to use the system without installing additional software.

* 1. **OBJECTIVES OF THE PROJECT**

The main objectives of the project are as follows:

* ***Reduced Risk:***

It provides a comprehensive depth analysis and recommendation for the decision maker. This eliminates human error and reduces the risk of providing loans which can later turn out to be Bad debts. Personal Risk Management facility evaluates applicants for Liquidity, Collateral and Credit History. Commercial Risk Assessment assists users in analyzing financial data, financial ratios and comparing them with industry standards, lender standards and loan product standards.

* ***Optimized Processing Time:***

This enables the flow of loan applications through various stages seamlessly within time frames provided for each flow point. This workflow integrated with Standardized and up-to-date application forms, smart links to policies and procedures, management control tools, Document Scanning, risk analysis and recommendation reports enable at least 70% reduction in processing time.

* ***Mobility of Users:***

Users can interact with the system even staying outside the office through getting task alert, SMS, Email in handheld.

* ***Reduced Lending Cost:***

The decrease in processing time and making it a paperless solution reduces labour cost, paper and printing cost, telephone cost etc.

* ***Management Control:***

It assists the management team to monitor and control loan service operations, customer performance, business quality (branch performance, staff performance, loan product and renewal management) and provides outstanding reporting capabilities which facilitate performance review of loans & personnel across various levels & structures of the organization.

* ***Customer Satisfaction:***

Customers dealing with guaranteed quick quotes, faster processing time, better customer service, easier to apply as a repeat customer and can be guaranteed of better one to one experience.

* 1. **MODULES IN BRIEF**

Briefly the module of Employee loan can be classified in following points:

* **Agreement Management**

Product life cycle starts with an agreement registration. This project supports different financial products like Leasing, Term Finance, Hire Purchase, Sale & Lease Back etc. Various products with structured payment schedule, delinquent charging and Grace Period options are available. Agreement Documentation checking, security checking for receipt of PDC, Sanction Amount and etc are included as well.

* **Application Form For Loans**

The application form captures all the details typically collected regarding the loan product, interest rate, amount requested and other items such as the duration of the loan, principal loan amount, and the financial standards against which the borrowers standing will be compared.

* **Risk Assessment**

This module is designed to capture the required financial data in detail of current assets, fixed assets, current liabilities and long-term liabilities to enable the system to perform a thorough risk analysis of the borrower against preset ratios and deliver to the reviewer a comprehensive picture, which leads to a well-informed decision.

* **Loan Pricing**

The loan pricing module provides a comprehensive and detailed analysis of each loan. Project will recommend pricing based on the lender's management policies for each loan type or classification. The system presents a comprehensive analysis of cost and revenue components, such as: Minimum Return, Cost of Borrowing, Borrower Risk Rating, Lending Personnel Cost, Overhead Cost, Debt Recovery Losses and Cost of Unused Funds. All these components are used to make a timely and informed decision for pricing Commercial Loans. Interest rates can be determined either manually or with the pricing module.

* **Work Flow**

The most critical feature of this project is its Workflow management tool, which allows an application to be initiated, routed to the appropriate staff in a pre-defined sequence, processed, tracked, and eventually completed and filed. This module facilitates electronic processing of the loan application between several "work flow points" which can be set up to meet the unique needs of every organization depending on their size, structure and review/approval methodology. The system assures that there are no delays in the processing of the loan. Efficiency reports can be generated to reveal any recurring delays at specific "flow points" so that corrective action can be taken.

* **Document Archiving/Management**

This module gives lending institutions the ability to digitally capture and store various documents that may be required as part of the loan package. It becomes easier to transfer the loan package to the next point in the process without the need for stacks of paper work to accompany the application.

* **Loan Committee**

The Loan Committee Module can efficiently control information flow to decision makers for loan approvals. Loan Committee members can be in different locations, but still be able to view all or any part of the complete loan documentation. This module will allow the lender to assign multiple committees based on loan amount or loan classification.

* **Monitoring**

This module uses the amortization schedule of the loan and compares it to actual remittances. It is also designed to generate alerts in the event a payment is not made on time or for the appropriate amount.

* **Reports**

One very valuable feature is the number and variety of data and reports that can be generated for review and analysis. From product popularity and processing efficiency, to application status, there are large numbers of reports available for each level of the organization. These reports enable different types of analysis for strategic management and presentation of data for review by senior management.

1. **METHODOLOGY**

The implementation of the whole system is proposed to be done in a completely step-by-step manner. The entire project is divided in the following phases:

**Phase 1: Analysis Phase:**

In this phase the requirements are first gathered from the customer and the requirement analysis starts. Web sites providing loans and related sites are visited. Some existing systems are also compared and the feasibility of the requirements is checked. A plan to the interaction of the end user with the proposed system is made which tells all the basic requirements of the end user, their informational needs along with their problems. Study of the complete functional system is done. Proposed system is formed as the problems in the existing system found. Study of the concepts of database design and develop this system is done.

**Phase 2: Project Planning Phase:**

This phase involves plan for the project that we have is prepared, which describes all the section of project management. Plan is prepared with the help of Project Plan, Schedule, Assigning Roles and Responsibilities, Communication Plan, Quality Assurance Plan, Project Cost Estimate and Risk Plan.

**Phase 3: Conceptual Design Phase:**

In this phase of the project system is shown in the modular form.

* Identification of Entities and their Relationships:

After completion of the analysis phase modelling phase starts. This is based on the analysis done. In modelling with the concepts of DBMS high level entities are found and the relationship between the entities is shown.

* Development of ER and EER model for the proposed system:

Entity Relationship diagram and extended ER diagram is made to show relationship between the entities. It makes easy to learn the system.

* Identification of classes and their relationship:

Classes represent entities with common characteristics or features. These features include attributes, operations and associations.

* Development of the class Diagram:

Class diagram show the class of the system, their interrelationship (including inheritance, aggression and association), and the operations and attributes of the classes.

**Phase 4: Logical Design Phrase:**

Logical design phase is for preparing for the structure of the database, interface forms. This phase is the conversion of the conceptual design of the modelling phase. This design is just before the implementation phase.

**Phase 5: Implementation and Testing:**

This phase is for the implementation of the design development in the logical design phase. In this phase programs are written, physical database designing is done and after implementing all the function of user requirements test of implemented functions with various test cases is done.

**Phase 6: User Interface:**

User interface is very essential part of the any product through which all type interacts thus user interface must be user friendly. User friendly environment enables user to get information efficiently without any problem in interacting with the system.

1. **APPLICATION DEVELOPMENT**

**N-Tier Applications:**

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

The N-Tier Application provide specific advantage that are vital to the business continuity to the enterprise. Typical features of a real life N-Tier may include the following:

* Security
* Availability and Scalability
* Management
* Easy Maintenance
* Data Abstraction

The above mentioned points are some of the key design goals of a successful N-Tier application that intends to provide a good business solution. An N-Tier application helps us distribute the overall functionality into various tiers or layers:

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

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

This is the one of the biggest advantages of the N-Tier application. Each layer can potentially treat the other layer as a ‘Block-Box’. In other words, each layer does not care how other layer processes the data as long as it sends the right data in a correct format.

Presentation layer

Business rules layer

Data access layer

Services

Services

DB

**Fig. 6.1 Data flown through different layers**

* **PRESENTATION LAYER**

This layer is also called as the Client layer comprises of components that are dedicated to presenting the data to the user. For example: Windows/WEB forms and bottoms, Edit boxes, Text boxes, Labels, Grids, etc.

* **BUSINESS RULES LAYER**

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

* **DATA ACCESS LAYER**

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

* **DATABASE LAYER**

This layer comprises of the Database Components such as DB Files, Tables, Views, etc. The Actual database could be created using SQL Server, Oracle, Flat files, etc. In an N-Tier application, the entire application can be implemented in such a way that it is independent of the actual database. For instance, you could change the Database Location with minimal changes to Data Access Layer. The rest of the Application should remain unaffected.

**PERFORMANCE REQUIREMENTS**

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the 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 in the part of 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 during the initial stages so that the system can be designed 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:

* 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 existing system is completely dependent on the user to perform all the duties.

1. **SOFTWARE REQUIREMENT AND HARDWARE REQUIREMENT**

**SOFTWARE REQUIREMENTS**

* Operating System : Windows 10 pro
* User Interface : HTML
* Client-side Scripting : JavaScript
* Programming Language : Java
* Web Technologies : JDBC, JSP
* IDE/Workbench : Eclipse oxygen 3.0
* Database : MySQL
* Server Deployment : Apache Tomcat 9.0.7

**HARDWARE REQUIREMENTS**

* Processor : Pentium i3
* Hard Disk : 40 GB or more
* RAM : 1 GB or more
* System Type : 32 Bit OS

1. **SYSTEM DESIGN DATAFLOW DIAGRAM**
   1. **DATAFLOW DIAGRAMS:**

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

DFDs are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system’s structure charts.

**Context level DFD:-**

Employee

Administrator

Register employee

View employee

Departments

Loan type

Designation

Designation loan type

Loan payment

GUI interface

User Interface

Data output stage

Data base

Data input stage

Data output stage

Data input stage

Data output stage

**Figure 8.1 Context level of DFD**

**DFD level 1 for Administrator:-**

Administrator

Register Employee

Add Department

Add Designation

Add Loan Types

Add Employee Loans

Add Designation Loans

Add Loan Payments

GUI Interface

User interface

Data output stages

Data base

Data input stage

Data output stages

Data output stages

**Figure 8.2 DFD level 1 for Administrator**

**DFD Level 2 for Employee:-**

Administrator

View profile

View Designation

View Department

View Designation Loan

View Designation Loans

Loan Payments

GUI Interface

User interface

Data output stages

Data base

Data input stage

Data output stages

Data output stages

**Figure 8.3 DFD level 2 for Employee**

**8.2 ER DIAGRAMS:**

An **entity–relationship model** (**ER model** for short) describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types. An ER model is typically implemented as a [database](https://en.wikipedia.org/wiki/Database). In a simple relational database implementation, each row of a table represents one instance of an entity type, and each field in a table represents an attribute type. In a [relational database](https://en.wikipedia.org/wiki/Relational_database) a relationship between entities is implemented by storing the primary key of one entity as a pointer or "foreign key" in the table of another entity. Entity–relationship diagrams don't show single entities or single instances of relations. Rather, they show entity sets(all entities of the same entity type) and relationship sets(all relationships of the same relationship type).

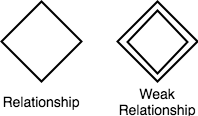
Entity

Simple rectangular box represents an Entity.

Entity in ER diagram

Relationships between Entities - Weak and Strong

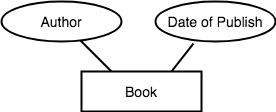
Rhombus is used to setup relationships between two or more entities.



Attributes for any Entity

Ellipse is used to represent attributes of any entity. It is connected to the

entity.



1.

Employee master

Works in

Department master

**ER diagram 8.1 Relation b/w Employee & Department**

* *Employee works in many Departments*.
* *Each Department has Employees.*

Employee master Department master

Employee master

# Emp ID

# Emp Fname

# Emp Lname

# Emp Address

# DOB

# DOJ

***PK*** # Emp DeptID

# Emp MailID

# Emp Image

# Emp DesignID

# Emp Sal

# Emp Password

Department master

***FK*** # Department ID

# Department Name

**Table 1 Relation b/w Employee & Department**

2.

Emp salary fixed

Is given to

Employee master

**ER diagram 8.2 Relation b/w Salary & Employee**

* *Salary is given to employee.*
* *Employee receives salary.*

Emp salary fixed Employee master

Emp salary fixed

# Salary Fixed ID

# Salary Fixed Date

# Salary End Date

***PK*** # Employee ID

Employee master

***FK*** # Emp ID

# Emp Fname

# Emp Lname

# Emp Address

# DOB

# DOJ

# Emp DeptID

# Emp MailID

# Emp Image

# Emp DesignID

# Emp Sal

# Emp Password

**Table 2** **Relation b/w Salary & Employee**

3.

Designation master

Is assigned to

Employee master

**ER diagram 8.3 Relation b/w Employee & its Designation**

* *Designation is assigned to employee.*
* *Each employee has a designation.*

Designation master Employee master

Designation master

# Design Name

***PK*** # Design ID

# Supervisor Design ID

Employee master

# Emp ID

# Emp Fname

# Emp Lname

# Emp Address

# DOB

# DOJ

# Emp DeptID

# Emp MailID

# Emp Image

***FK*** # Emp DesignID

# Emp Sal

# Emp Password

**Table 3 Relation b/w Employee & its Designation**

4.

Financial advisor master

Serves

Department master

**ER diagram 8.4 Relation b/w Financial Advisor & Department**

* *Financial advisor serves department.*
* *Every department have financial advisor.*

Financial advisor master Department master

Financial advisor master

# Financial Advisor Eid

***PK*** # Department ID

# Financial Advisor ID

Department master

***FK*** # Department ID

# Department Name

**Table 4 Relation b/w Financial Advisor & Department**

5.

Designation master

decides

Loan type details

**ER diagram 8.5 Relation b/w Designation & Loan type**

* *Designation master decides loan type details.*
* *Loan type details is based on designation master.*

Designation master Loan type details

Designation master

***PK*** # Design Name

# Supervisor Design ID

# Design ID

Loan type details

# Loan Type ID

# Loan Min. Range

***FK*** # Designation ID

# Loan Max. Range

**Table 5 Relation b/w Designation & Loan type**

6.

Loan payment master

contains

Employee loan master

**ER diagram 8.6 Relation b/w Loan payment & Employee loan**

* *Loan payment master contains employee master.*
* *Employee loan details are contained in loan payment.*

Loan payment master Employee loan master

Loan payment master

# Loan Payment ID

# Loan Payment Date

# Amount Paid

***PK*** # Emp Loan ID

# Next Payment Date

# Amount Balance

Employee master

# Loan Type ID

# Emp ID

# Amount Taken

***FK*** # Emp Loan ID

# Loan Taken Date

# Loan End Date

**Table 6 Relation b/w Loan payment & Employee loan**

**7. Final ER Model with cardinalities :**

Auth details

# User ID

# User Name

# Employee ID

# Employee Name

Loan payment master

# Loan Payment ID

#Loan Payment Date

# Amount Paid

# Emp Loan ID

# Next Payment Date

# Amount Balance

Loan type detail

# Loan Type ID

# Loan Min. Range

# Designation ID

# Loan Max. Range

Designation master

# Design Name

# Supervisor Design ID

# Design ID

Department master

# Department Name

# Department ID

Financial advisor name

# Financial Advisor Eid

# Department ID

# Financial Advisor ID

Employee master

# Emp ID

# Emp Fname

# Emp Lname

# Emp Address

# DOB

# DOJ

# Emp Dept ID

# Emp Mail ID

# Emp Image

# Emp Design ID

# Emp Sal

# Emp Password

Employee loan master

# Loan Type ID

# Emp ID

# Amount Taken

# Emp Loan ID

# Loan Taken Date

# Loan End Date

Loan info

# Loan Type ID

# Loan Type Name

#Loan Min range

# Loan Max Range

# Loan Repayment Period Min

# Loan Repayment Period Max

# Interest

Employee salary fixed

# Salary Fixed ID

# Salary Fixed Date

# Salary End Date

# Employee ID

Is given to

Contains

Based on

Is assigned to

Works in

Serves

**ER diagram 8.7 ER Model with all entities, relations and attributes with their cardinalities**

1. **TECHNOLOGY DESCRIPTION**
   1. **HTML**

HTML (Hypertext Mark-up Language) is a predominant mark-up language for web pages. It provides a means to describe the structure of text- based information in a document – by denoting certain text as headings, paragraphs, lists and so on, and to supplement that text with interactive forms, embedded images, and other objects. HTML is written in the form of labels (known as tags), surrounded by angle brackets. HTML can also describe, to some degree, the appearance and semantics of a document, and can include embedded scripting language code which can affect the behaviour of the web browsers and other HTML processors.

HTML is also often used to refer to content of MIME type text/html or even more broadly as a generic term for HTML whether in its XML descended form (such as XHTML 1.0) or its form descended directly from SGML.

**HYPERTEXT MARKUP LANGUAGE (HTML),** the languages of the World Wide Web (WWW), allows users to produce Web pages that include text, graphics and pointer to other Web pages (Hyperlinks).

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

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

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

**ATTRIBUTES**

The attributes of an element are name-value pairs, separated by “=”, and written within the start label of an element, after the element’s name. the value should be enclosed in single or double quotes, although values consisting of certain characters can be left unquoted in HTML (but not XHTML). Leaving attribute values unquoted is considered unsafe.

Most elements take any of several common attributes: id, class, style and title. Most also take language-related attributes: lang. and dir. The id attribute provides a document-wide unique identifier for an element. This can be used by style sheets to provide presentational properties, by browsers to focus attention on the specific element or by scripts to alter the contents or presentation of an element. The class attribute provides a way of classifying similar elements for presentation purposes. For example, an HTML document (or a set of documents) may use the designation class=”notation” to indicate that all elements with this class value are all subordinate to the main text of the document (or documents). Such notation classes of elements might be gathered together and presented as footnotes on a page, rather than appearing in the place where they appear in the source HTML.

An author may use the style non-attributal codes presentational properties to a particular element. It is considered better practise to use an element’s son-id page and select the element with a style sheet, though sometimes this can be too cumbersome for a simple ad hoc application of styled properties. The title is used to attach sub textual explanation to an element. In most browsers this title attribute is displayed as what is often referred to as a tooltip. The generic inline span element can be used to demonstrate these various non-attributes.

The preceding displays as HTML (pointing the cursor at the abbreviation should display the title text in the most browsers).

**ADVANTAGES**

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

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

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

<SCRIPTS>.. </SCRIPT>.

<SCRIPT LANGUAGE= “JavaScript”>

JavaScript statements

</SCRIPT>

Here are a few things we can do with JavaScript:

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

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

*JAVASCRIPT Vs JAVA*

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

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

There are many other differences but the important thing to remember is that:

JavaScript and Java are separate languages. They are both useful for different things; In fact they can be used together to combine their advantages.

**Advantages:**

* JavaScript can be used for Server-site and Client-site scripting.
* It is more flexible than VBScript.
* JavaScript is the default scripting languages at Client-site since all the browsers support it.
  1. **JAVA TECHNOLOGY**

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

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

**Importance of Java to the Internet**

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

Java can be used to create two types of programs:

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

**Features of Java Security**

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

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

**Portability**

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

**The Byte Code**

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

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

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

**Java Virtual Machine (JVM)**

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

*Overall Description*

Java Source

Java Byte Code

JVM

**Fig. 9.1 Development process of JAVA program**

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

**Java Architecture**

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

**Compilation of code**

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

Compiling and Interpreting Java Source Code

Source Code

……………..

……………..

………………

………………

PC Compiler(PC)

Java Interpreter (PC)

Java Interpreter (MC)

Java Interpreter (SPARC)

JAVA

BYTE CODE

(PLATFORM INDEPENDENT)

SPARC

Macintosh Compiler(MC)

**Fig. 9.2 Compiling & Interpreting of Java source code**

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

**SIMPLE**

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

**Object-Oriented**

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

**Robust**

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

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

**9.4 Java Database Connectivity**

What is JDBC?

JDBC is a Java API for executing SQL statements. It consists of set of classes and interfaces written in the Java programming language. JDBC provides a standard API for tool/database developers and makes it possible to write database applications using a pure Java API.

Using JDBC, it is easy to send SQL statements to virtually ant relational database. One can write a single program using JDBC API, and the program will be able to send SQL statements to the appropriate database. The combinations of Java and JDBC lets a programmer write it once and run it anywhere.

What does JDBC do?

JDBC makes it possible to do three things:

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

JDBC versus ODBC and other APIs

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

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

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

**9.5 Two -Tier and Three-Tier Models**

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

JAVA

Application

JDBC

DBMS

Client Machine

DBMS-Propriety Protocol

Database Server

**Fig 9.3 JDBC API two-tier model for database access**

Java Applet or HTML Browser

Application Server (Java) JDBC

DBMS

Client Machine (GUI)

Server Machine (Business)

DBMS-Propriety Protocol

Database Server

**Fig 9.4 JDBC API three-tier model for database access**

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

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

JBDC is important to allow database access from a Java middle tier.

Database Server

Database Specific API’s

Workstation

**Fig. 9.5 Database access from Java middle tier**

* Presentation, business and data model processing logic into client application.
* Server is typically a database server.
* Client sends SQL statements, retrieves raw data.

**JDBC Driver Types**

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

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

**JDBC-ODBC Bridge**

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

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

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

**JDBC Connectivity**

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

* Perform connection and authentication to a database server.
* Manager transactions.
* Move SQL statements to a database engine for pre-processing and execution.
* Execute stored procedures.
* Inspect and modify the results from Select statements.
  1. **Database**

A database management system (DBMS) is computer software designed for the purpose of managing databases, a large set of structured data, and run operations on the data requested by numerous users. Typical examples of DBMS include Oracle, DB2, Microsoft Access, Microsoft SQL Server, Firebird, MySQL, SQLite, FileMaker and Sybase Adaptive Server Enterprise. DBMS are typically used by Database administrators in the creation of Database systems. Typical examples of DBMS use include accounting, human resources and customer support systems.

Originally found only in large companies with the computer hardware needed to support large data sets. DBMS have more recently emerged as a fairly standard part of any company back office.

**Description**

A DBMS is a complex set of software programs that controls the organisation, storage, management and retrieval of data in a database. A DBMS includes:

* A modelling language to define the schema of each database hosted in the DBMS, according to the DBMS data model.
* The four most common types of organisations are the hierarchical, network, relational and object models. Inverted lists and other methods are also used. A given database management system may provide one or more of the four models. The optimal structure depends on the natural organization of the application’s requirements (which include transaction rate (speed), reliability, maintainability, scalability and cost) only available when a set of application programs are customized for each data entry and updating function.
* A transaction mechanism that ideally would guarantee the ACID properties, in order to ensure data integrity, despite concurrent user accesses (concurrency control), and faults (fault tolerance).
* It also maintains the integrity of the data in the database.
* The DBMS can maintain the integrity of the database by not allowing more than one user to update the same record at the same time. The DBMS can help prevent duplicate records vis unique index constraints; for example, no two customers with the same customer numbers (key fields) can be entered into the database. See ACID properties for more information (Redundancy avoidance).

The DBMS accepts requests for data from the application program and instructs the operating system to transfer the appropriate data.

When a DBMS is used, information systems can be changed much more easily as the organisation’s information requirements change. New categories of data can be added to the database without disruption to the existing system.

Organisations may use one kind of DBMS for daily transaction processing and then move the detail onto another computer that uses another DBMS better suited for random inquires and analysis. Overall systems design decisions are performed by data administrations and system analysts. Detailed database design is performed by database administrators.

Database servers are specially designed computers that hold the actual databases and run only the DBMS and related software.

Database servers are usually multiprocessor computers, with RAID disk arrays used for stable storage. Connected to one or more servers via a high-speed channel, hardware database accelerators are also used in large volume transaction processing environments.

DBMS are found at the heart of most database applications.

Sometimes DBMS are built around a private multitasking kernel with built-in networking support although nowadays these functions are left to the operating system.

* 1. **SQL**

Structured Query Language (SQL) is the language used to manipulate relational databases. SQL is tied very closely with the relational model.

In relational model, data is stored in structures called relations or tables.

SQL statements are issued for the purpose of:

**Data Definition:**

Defining tables and structures in the database. (DDL used to create, alter and drop schema objects such as tables and indexes)

**Data Manipulation:**

Used to manipulate the data within those schema objects. (DML inserting, updating, deleting the data and querying the database)

A schema is a collection of database objects that can include: tables, views, indexes and sequences.

List of SQL statements that can be issued against an oracle database schema are:

* **ALTER-**

Change an existing table, view or index definition (DDL)

* **AUDIT-**

Track the changes made to a table (DDL)

* **COMMENT-**

Add a comment to a table or column in a table (DDL)

* **COMMIT-**

Make all recent changes permanent (DML-transactional)

* **CREATE-**

Create new database objects such as tables or views (DDL)

* **DELETE-**

Delete rows from a database table (DML)

* **DROP-**

Drop a database object such as a table, view or indexes (DDL)

* **GRANT-**

Allow another user to access database objects such as tables or views (DDL)

* **INSERT-**

Insert new data into database table (DML)

* **NO AUDIT-**

Turn off the auditing function (DDL)

* **REVOKE-**

Disallow a user access to database objects such as tables and views (DDL)

* **ROLLBACK-**

Undo any recent changes to the database (DML-transactional)

* **SELECT-**

Retrieve data from a database table (DML)

* **TRUNCATE-**

Delete all rows from a database table (can’t be rolled back) (DML)

* **UPDATE-**

Change the values of some data items in a database table (DML)

* 1. **SERVLETS**

**INTRODUCTION**

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 it’s tools. To build customized network servers for any Internet client/server system. Servlets are to a web server, how applets are to the browser.

**ABOUT**

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

Servlets are objects that conform to a specific interface that can be plugged into a Java-based server. Servlets are to the server-side what applets are to the client-side- object byte codes that can be dynamically loaded off the net. They differ from applets in that they faceless objects (without graphics or a GUI component). They serve as a 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 Servlet can be used to generate dynamic HTML content. When you use Servlets to do dynamic content you get the following advantages:

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

**WHY SERVLETS?**

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

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

**Advantages of the Servlet API**

One of the greatest advantage of the Servlet API is protocol independence. It assumes nothing about:

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

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

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

Web Components

* Servlets or JSP pages
* JavaBeans(optional)

**Web Browser, Web Pages, Applets and Optional JavaBeans components**

**Application Client and Optional JavaBeans Components**

**JavaBeans Components (Optional**)

Web Tier

J2EE Server

**Fig. 9.6 Web Components**

**Features of Servlets:**

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

**Loading Servlets:**

Servlets can be loaded from three places-

1. From a directory that is on the CLASSPATH. The CLASSPATH of the Java Web Server includes service root/classes which is where the system classes reside.
2. From the <SERVICE\_ROOT /Servlets/directory. This is not in the server’s class path. A class loader is used to create Servlets from this directory. New Servlets can be added, existing Servlets can be recompiled and the server will notice these changes.
3. From a remote location, for this a code base like http://nine.eng/classes/foo/ is required in addition to the Servlets class name. Refer to the admin GUI docs on Servlet section to see how to set this up.

**Loading Remote Servlets**

Remote Servlets can be loaded by:

1. Configuring the Admin Tool to setup automatic loading of remote Servlets.
2. Setting up server side include tags in.shtml files.
3. Defining a filter chain configuration.

**Invoking Servlets**

A Servlet invoker is a Servlet that invokes the “service” method on a named Servlet. If the Servlet is not loaded in the server, then the invoker first loads the Servlet (either from local disk or disk from the network) and then invokes the “service” method. Also like applets, local Servlets in the server can be identified by just the class name. In other words, if a Servlet name is not absolute, it is treated as local. A client can invoke Servlets in the following ways:

* The client can ask for a document that is served by the Servlet.
* The client (browser) can invoke the Servlet directly using a URL, once it has been mapped using the Servlet Aliases section of the admin GUI.
* The Servlet can be invoked through server side include tags.
* The Servlet can be invoked by placing it in the Servlets or directory.
* The Servlet can be invoked by using it in a filter chain.
  1. **Java Server Pages (JSP)**

Java server page is a simple, yet powerful technology for creating and maintaining dynamic- content web pages. Based on the Java programming language, Java Server Page offer proven portability, open standards and a mature re-usable component model. The Java Server Page architecture enables the separation of the content generation from the content presentation. This separation not only eases maintenance headaches but 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:**

* ***Portability***

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

* ***Components***

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

* ***Processing***

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

* ***Access Models***

A Java Server Page file may be accessed in at least two different ways. A client’s request comes directly into Java Server Page. In this scenario, suppose the page accesses reusable Java Bean components that perform particular well-defined computations like accessing a database. The result of the Beans computations, called result sets is stored within the Bean 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 page 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.
3. JSP engine is program which can understand 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.
   1. **Eclipse IDE**

Eclipse is an open-source software framework written primarily in Java. In its default form it is an Integrated Development Environment (IDE) for Java developers, consisting of the Java Development Tools (JDT) and the Eclipse Compiler for Java (ECJ). Users can extend its capabilities by installing plug-ins written for the Eclipse software framework, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. Language packs are available for over a dozen languages.

**Architecture**

The basis for Eclipse is the Rich Client Platform (RCP). The following components constitute the rich client platform:

* OSGi – a standard bundling framework.
* Core platform – boot Eclipse, run plug-ins.
* The Standard Widget Toolkit (SWT) – a portable widget toolkit.
* JFace – viewer classes to bring model view controller programming to SWT, file buffers, text handling and text editors.
* The Eclipse Workbench – views, editors, perspectives, wizards.

Eclipse’s widgets are implemented by a widget toolkit for Java called SWT, unlike most Java applications, which use the Java standard Abstract Window Toolkit (AWT) or Swing. Eclipse’s user interface also leverages an intermediate GUI layer called JFace, which simplifies the construction of applications based on SWT.

Eclipse employs plug-ins in order to provide all of its functionality on top of (and including) the rich client platform, in contrast to some other applications where functionality is typically hard coded. This plug-in mechanism is a light weighted software componentry framework. In addition to allowing Eclipse to be extended using other programming languages such as C and Python, the plug-in framework allows Eclipse to work with typesetting languages like LateX, networking applications such as telnet, and database management systems. The plug-in architecture supports writing any desired extension to the environment, such as for configuration management. Java and CVS support is provided in the Eclipse SDK.

The key to the seamless integration of tools with Eclipse is the plugin. With the exception of a small run-time kernel, everything in Eclipse is a plug-in. this means that a plug-in you develop integrates with Eclipse in exactly the same way as other plug-ins; in this respect, all features are created are created equal.

The eclipse SDK includes the Eclipse Java Development Tools, offering an IDE with a built-in incremental Java compiler and a full model of the Java source files. This allows for advanced refactoring techniques and code analysis. The IDE also makes use of a workspace, in this case a set of metadata over a flat file space allowing external file modifications as long as the corresponding workspace “resource” is refreshed afterwards. The Visual Editor project allows interfaces to be created interactively, hence allowing Eclipse to be used as a RAD tool.

1. **TESTING**

Testing is defined in several ways, some of them are as follows:

* The process of executing a system with the intent of finding an error is known as testing.
* Testing is defined as, the process in which defects are identified, isolated, subjected for rectification and ensure that product is defect free in order to produce the quality product and to satisfy customer.
* Quality is defined as justification of the requirements.
* Defect is nothing but deviation from the requirements.
* Defect is nothing but bug.
* Testing can demonstrate the presence of bugs, but not their absence.
* Debugging and Testing are not the same thing.
* Testing is a systematic attempt to break a program.
* Debugging is the art or method of uncovering why the program did not executed properly.
  1. **TESTING METHODOLOGIES**

Testing methodologies includes the types of testing, which are as follows:

1. ***Black box testing:***

Black box testing is the process in which tester can perform on an application without having any internal structural knowledge of application.

Usually Test Engineers are involved in the black box testing.

1. ***White box testing:***

White box testing is the process in which tester can perform testing on an application with having internal structural knowledge.

Usually Developers are involved in white box testing.

1. ***Gray box testing:***

Gray box testing is the process in which the combination of black box and white box tonic’s are used.

* 1. **STLC (Software Testing Life Cycle)**

**Test Planning:**

1. Test Plan is defined as a strategic document which describes the procedure how to perform various testing on the total application in the most efficient way.
2. This document involves the scope of testing.
3. Objective of testing.
4. Areas that need to be tested.
5. Areas that should not be tested.
6. Scheduling Resource Planning.
7. Areas to be automated, various testing tools used.

**Test Development**

1. Test Case Development (Checklist).
2. Test Procedure Preparation (Description of the test cases).
3. Implementation of test cases (Observing the result).

**Result Analysis:**

1. Expected value; is nothing but expected behaviour of application.
2. Actual Value; is nothing but actual behaviour of application.

**Bug Tracing:** collect all the failed cases, prepare documents.

**Reporting:** prepare documents (status of the application).

**Types of Testing**

* + **Smoke Testing:**

It is the process of initial testing in which tester looks for availability of all the functionality of the application in order to perform detailed testing on them. (Main check is for available forms).

* + **Sanity Testing:**

It is a type of testing that is conducted on an application initially to check for the proper behaviour of an application that is to check all the functionality are available before the detailed testing is conducted by on them.

* + **Regression Testing:**

It is one of the best and important testing. Regression testing is the process in which the functionality, which is already tested before, is once again tested whenever some new change is added in order to check whether the existing functionality remains same.

* + **Re-Testing:**

It is the process in which testing is performed on some functionality which is already tested before to make sure that the defects are reproducible and to rule out the environments issues if at all any defects are there.

* + **Static Testing:**

It is testing, which is performed on an application when it is not been executed. Ex: GUI, Document Testing.

* + **Dynamic Testing:**

It is testing which is performed on an application when it is being executed. Ex: Functional testing.

* + **Alpha Testing:**

It is a type of user acceptance testing, which is conducted on an application when it is just before released to the customer.

* + **Beta Testing:**

It is a type of UAT that is conducted on an application when it is released to the customer, when deployed in to the real time environment and being accessed by the real time users.

* + **Monkey Testing:**

It is the process in which abnormal operations, beyond capacity operations are done on the application to check the stability of it in spite of users abnormal behaviour.

* + **Compatibility Testing:**

It is the testing process in which usually the products are tested on the environments with different combinations of databases (application servers, browsers, etc). In order to check how far the product is compatible with all these environments platform combination.

* + **Installation Testing:**

It is the process of testing in which the tester try to install or try to deploy the module into the corresponding environment by following the guidelines produced in the deployment document and check whether the installation is successful or not.

* + **Adhoc Testing:**

It is the process of testing in which unlike the formal testing where in test case document is used, without that test case document testing can be done of an application, to cover that testing of the future which are not covered in that test case document. Also it is intended to perform GUI testing which may involve the cosmotic issues.

* 1. **TCD (Test Case Document):**

**Test Case Document Contains**

* **Test Scope or Test Objective**
* **Test Scenario**
* **Test Procedure**
* **Test case**

This is the sample test case document for the advertise details of Blazon Agency Project:

**Test Scope:**

* Test coverage is provided for the screen “Login check” form of a Administration module of Blazon Agency application.
* Areas of the application to be tested.

**Test Scenario:**

* When the office personals use this screen for the payment, tariff and company registration and web advertisements on basis of requirements and quit the form.

**Test Procedure:**

* The procedure for testing this screen is planned in such a way that the data entry, status calculation functionality, saving and quitting operations are testing in terms of GUI testing. Positive testing, Negative testing using the corresponding GUI test cases, Positive test cases, Negative test cases respectively.
  1. **TEST CASES:**
* **Template for Test Case**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T.C. No | Description | Exp | Act | Result |
|  |  |  |  |  |

* **Guidelines For Test Cases:**

1. **GUI Test Cases:**

* Total no. of features that need to be checked.
* Look & Feel.
* Look for Default values if at all any (data & time, if at all any require)
* Look for spell check.

**Example:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T.C. No | Description | Expected  Value | Actual  value | Result |
| 1. | Check for all the features in the screen. | The screen must contain all the features. |  |  |
| 2. | Check for the alignment of the objects as per the validations. | The alignment should be in proper way. |  |  |

1. **Positive Test Cases:**

* The positive flow of the functionality must be considered.
* Valid inputs must be used for testing.
* Must have the positive perception to verify whether the requirements are justified.

**Example:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T.C. No | Description | Expected  Value | Actual  Value | Result |
| 1. |  |  |  |  |
| 2. |  |  |  |  |

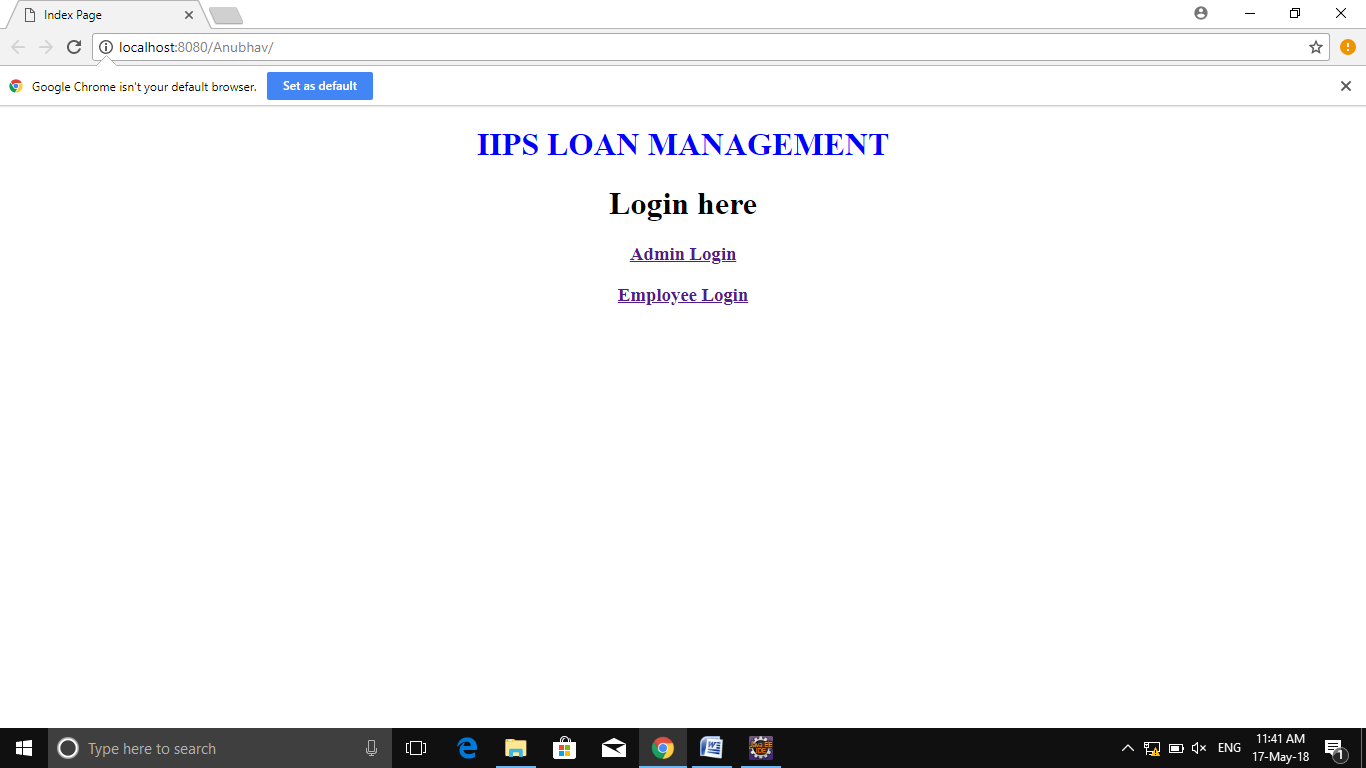
1. **Negative Test Cases:**

* Must have negative perception.
* Invalid inputs must be used for test.

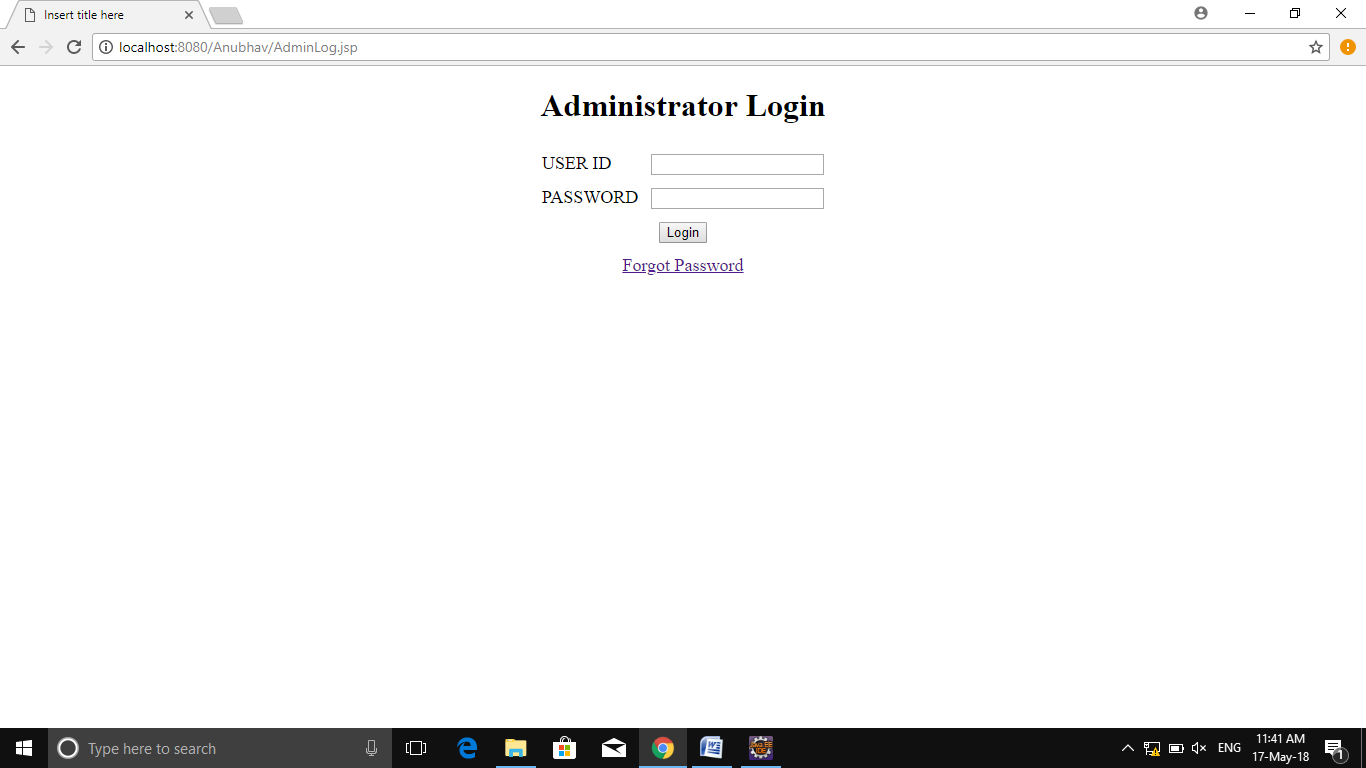
**Example:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T.C. No | Description | Expected  Value | Actual value | Result |
| 1. | Try to modify the information in date and time. | Modification should not be allowed. |  |  |
| 2. |  |  |  |  |

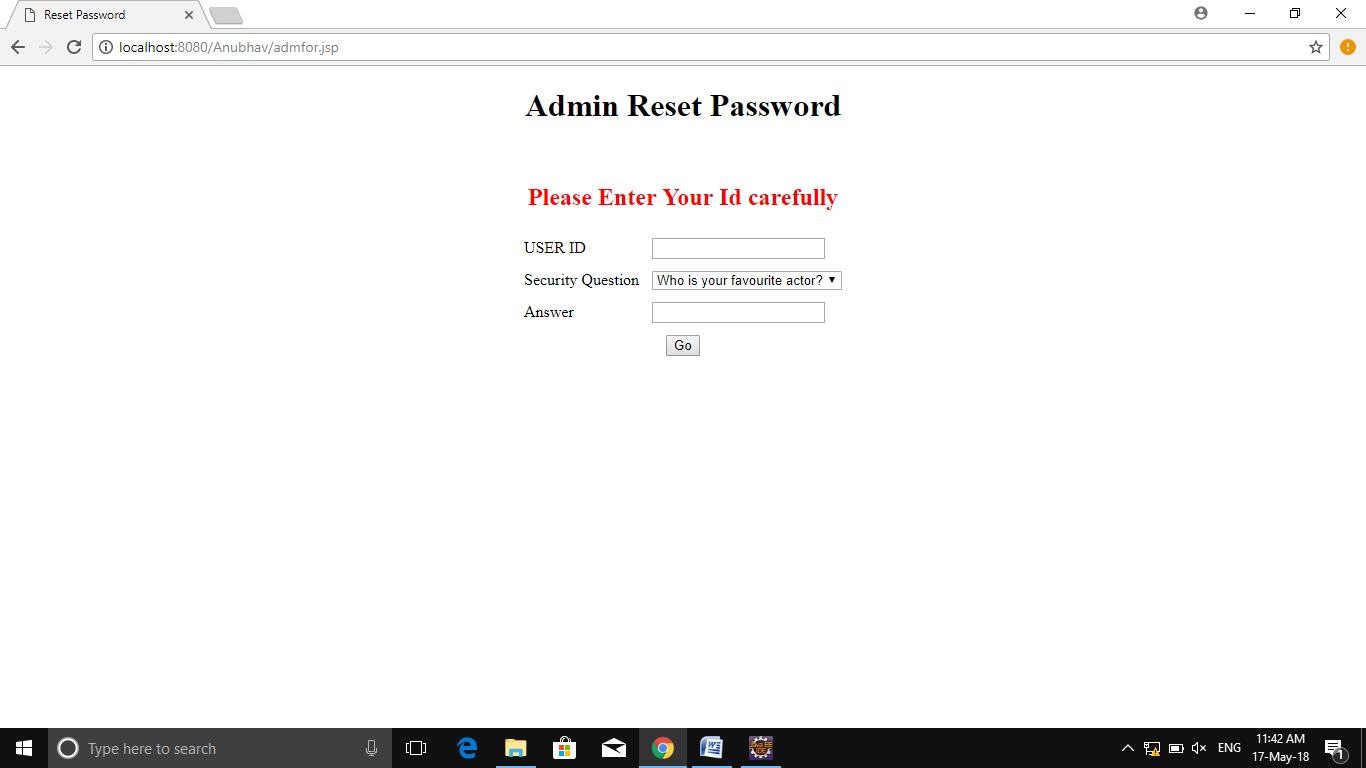
1. **Screens**
2. Login Screen



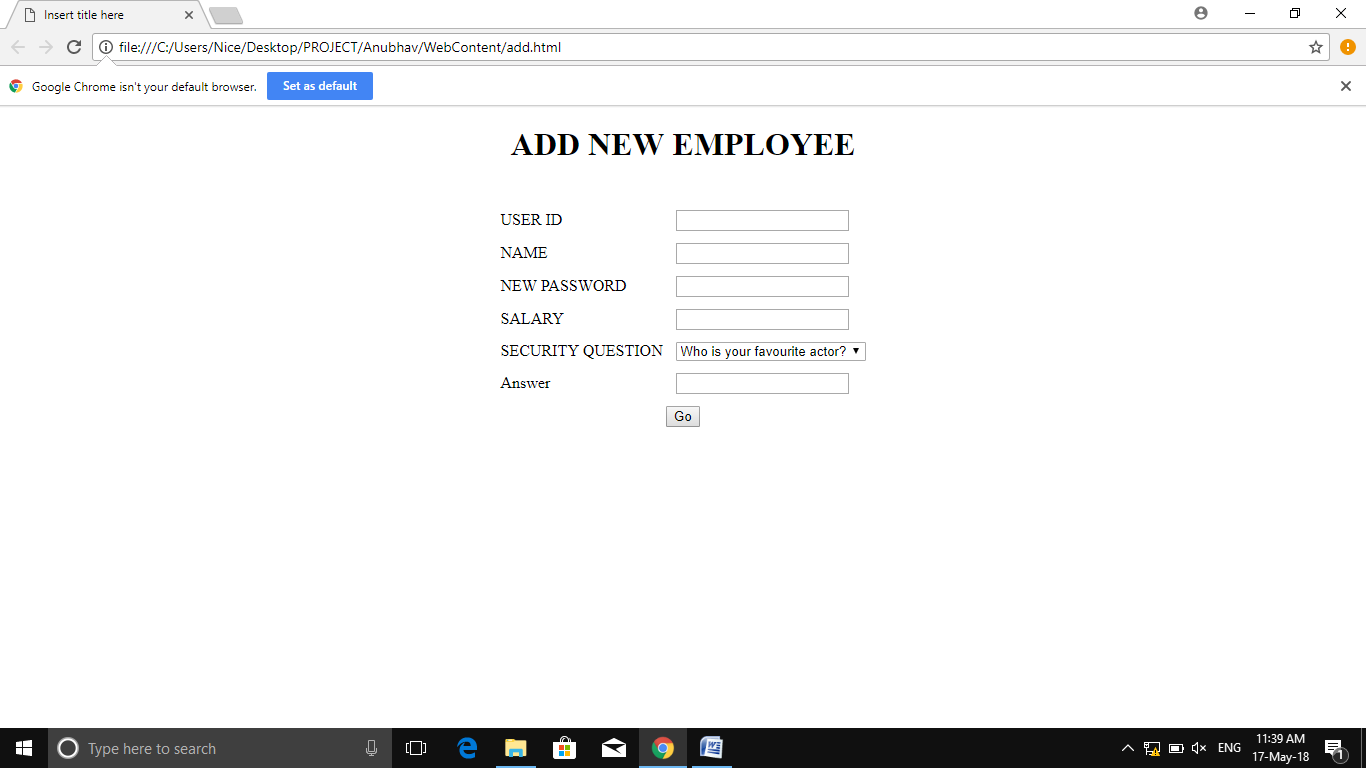
1. Administrator Login



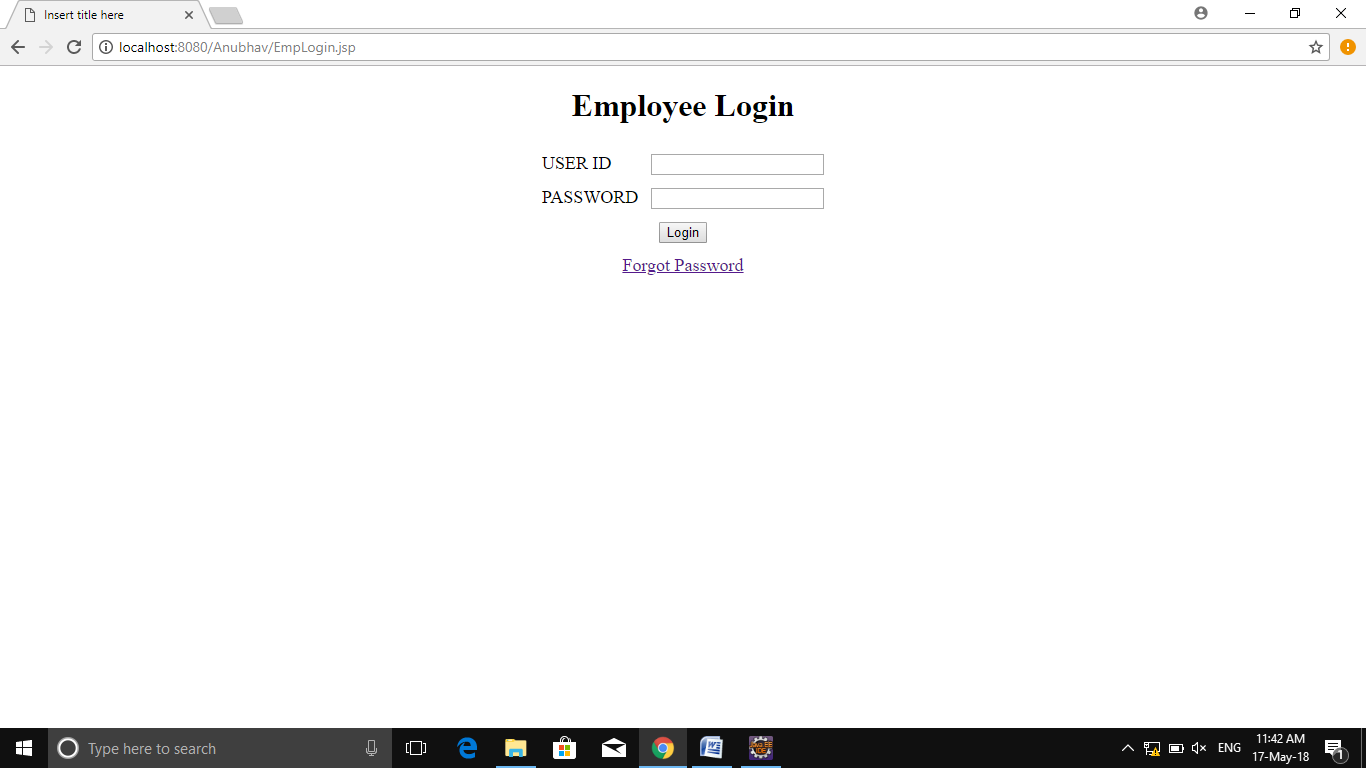
1. Admin Reset Password



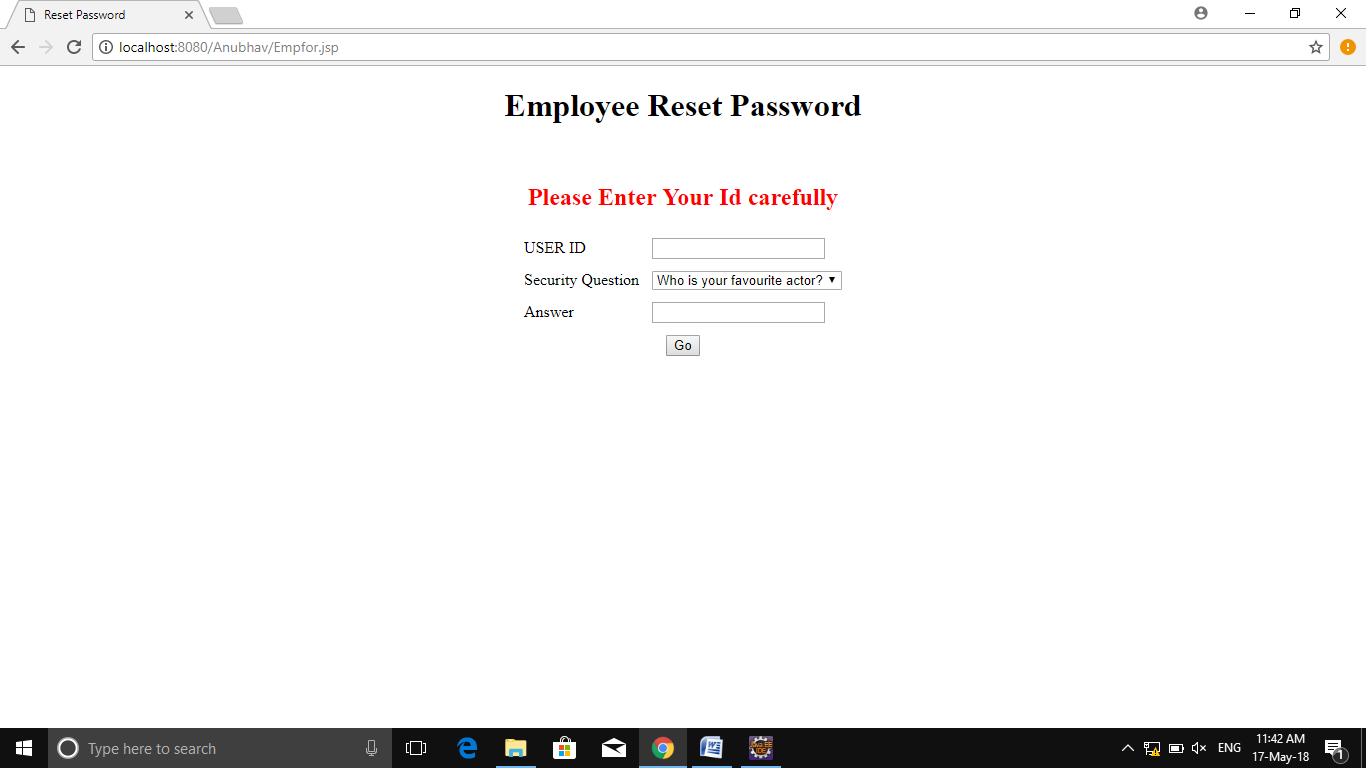
1. Add new Employee



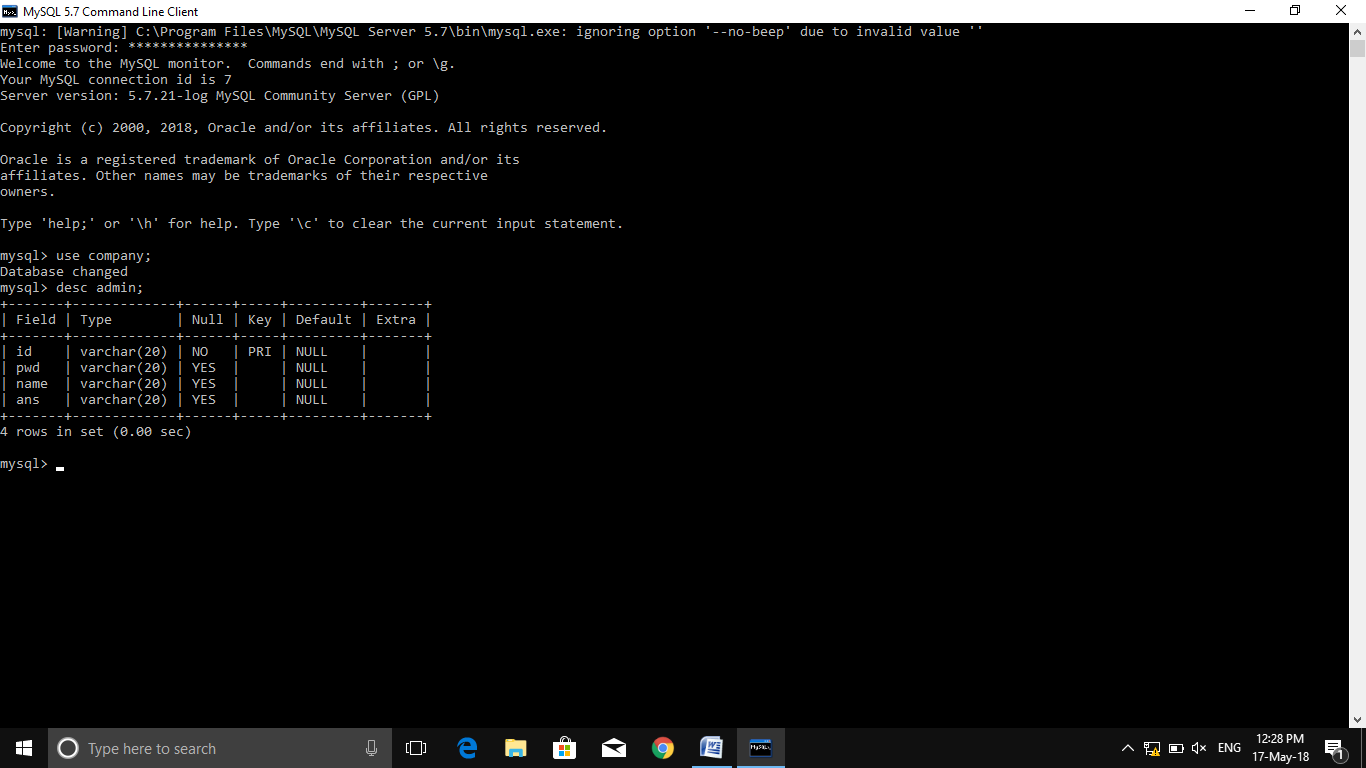
1. Employee Login



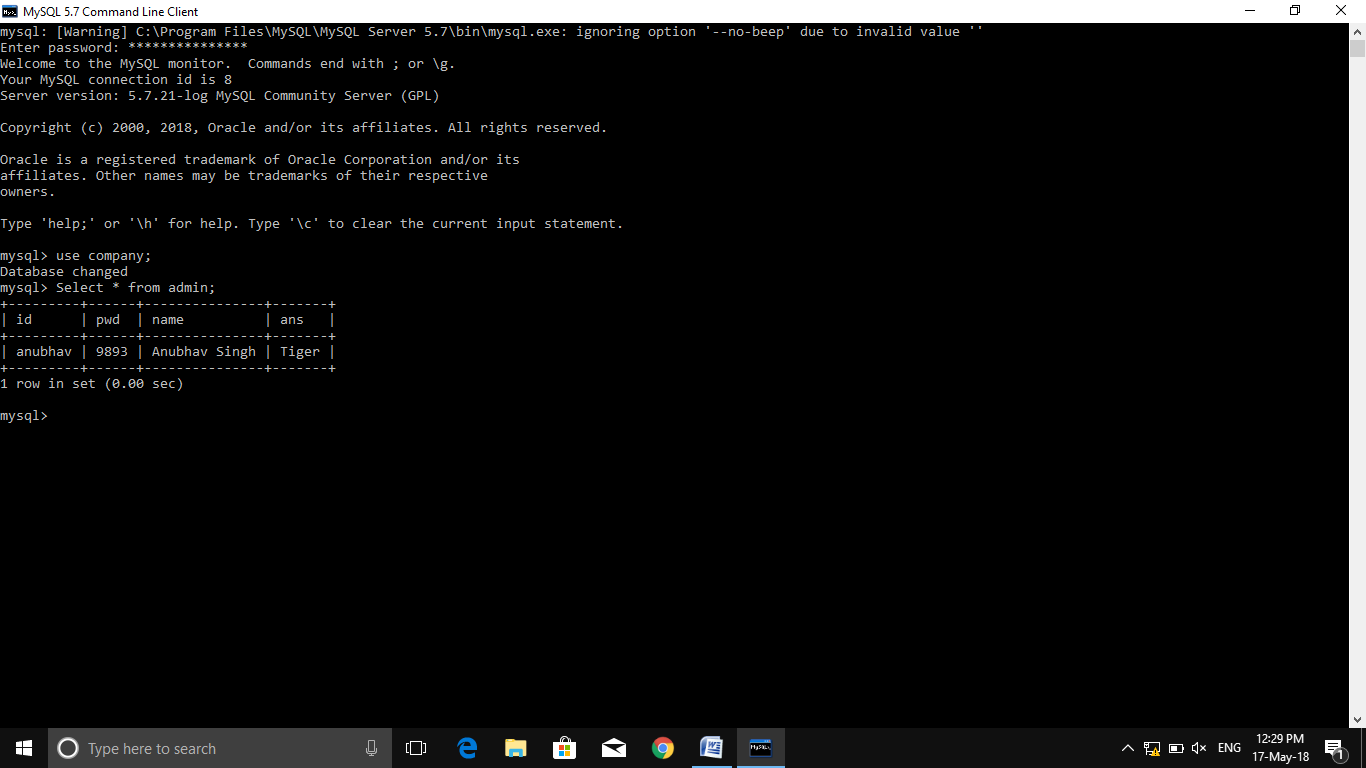
1. Employee reset password



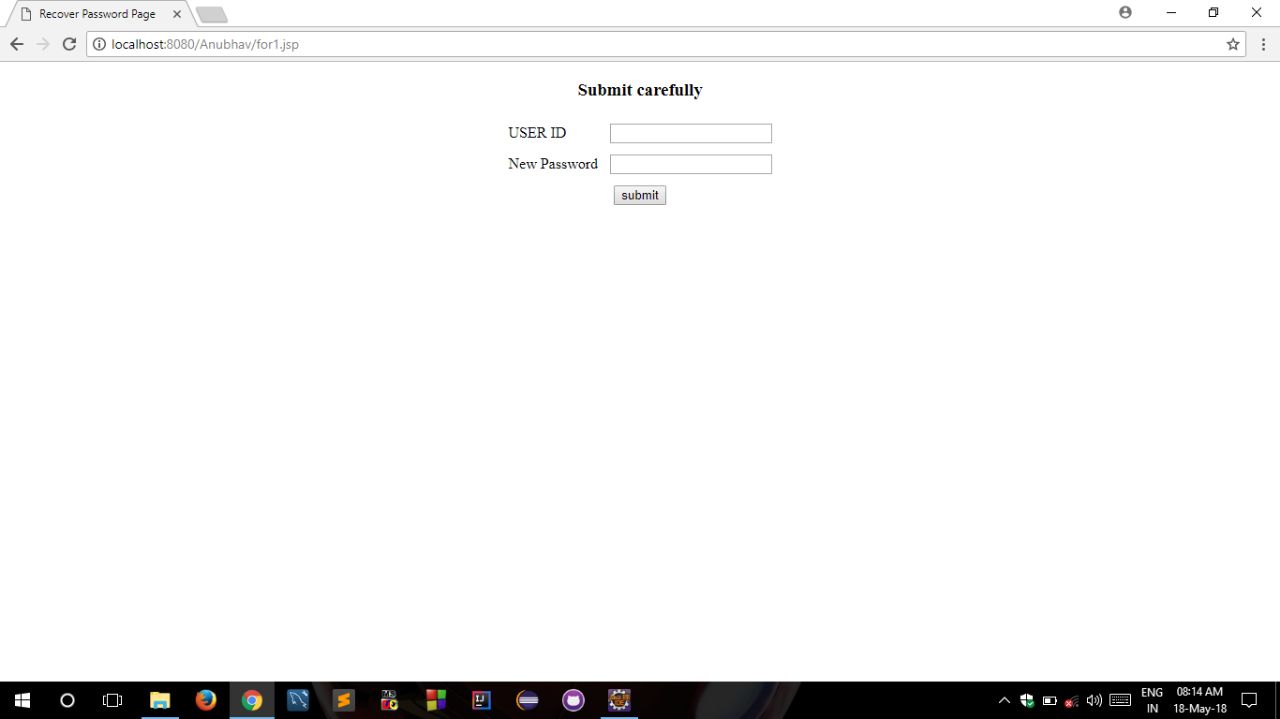
1. Admin database columns



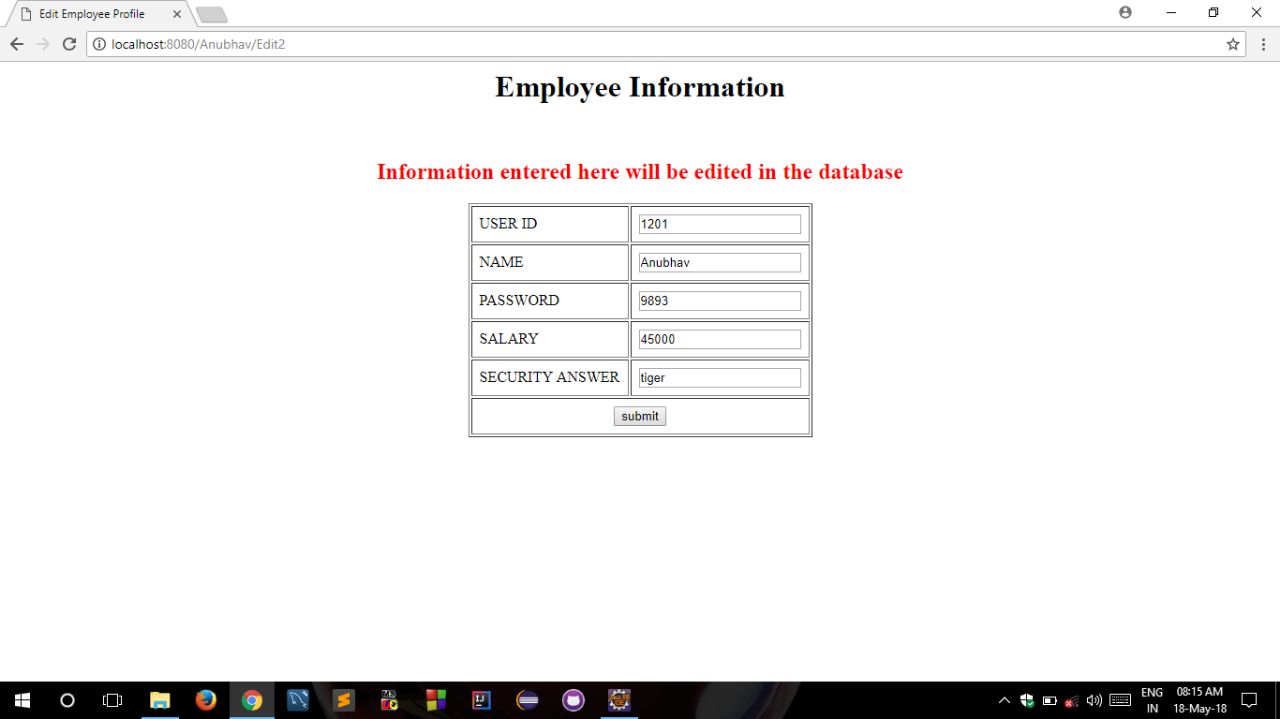
1. Admin login Database



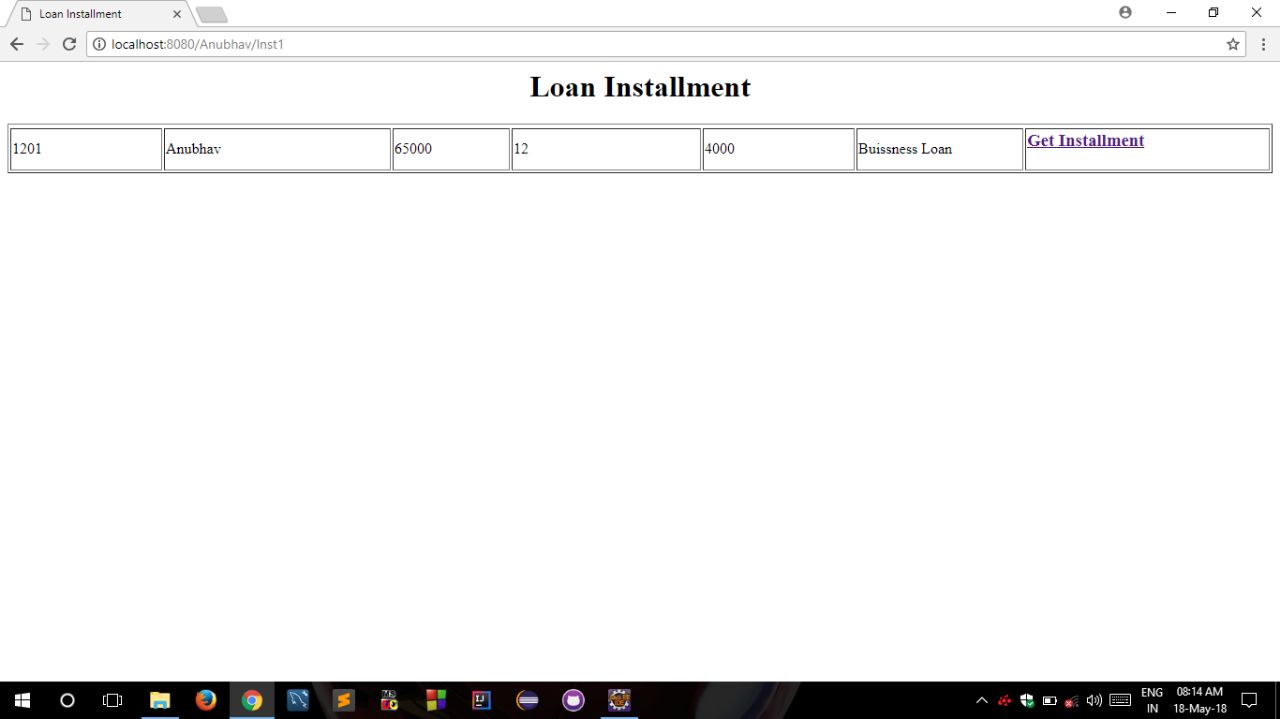
1. Recovery of Password



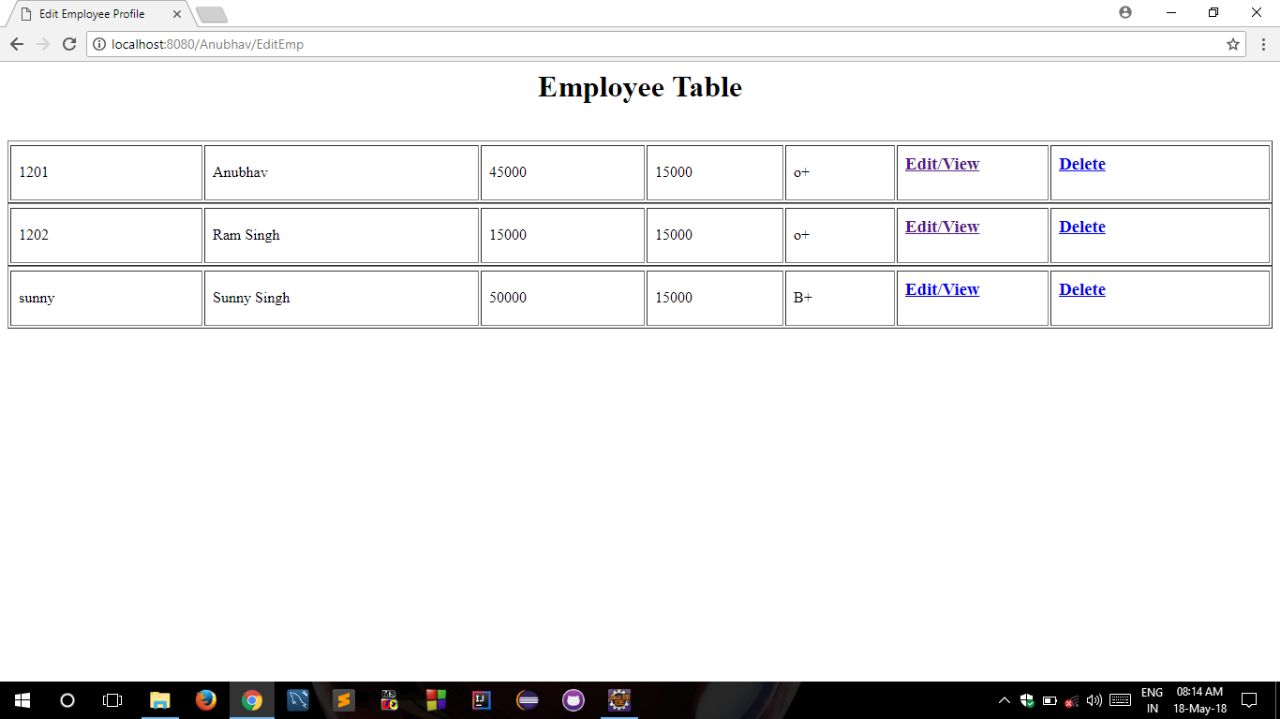
1. Employee information



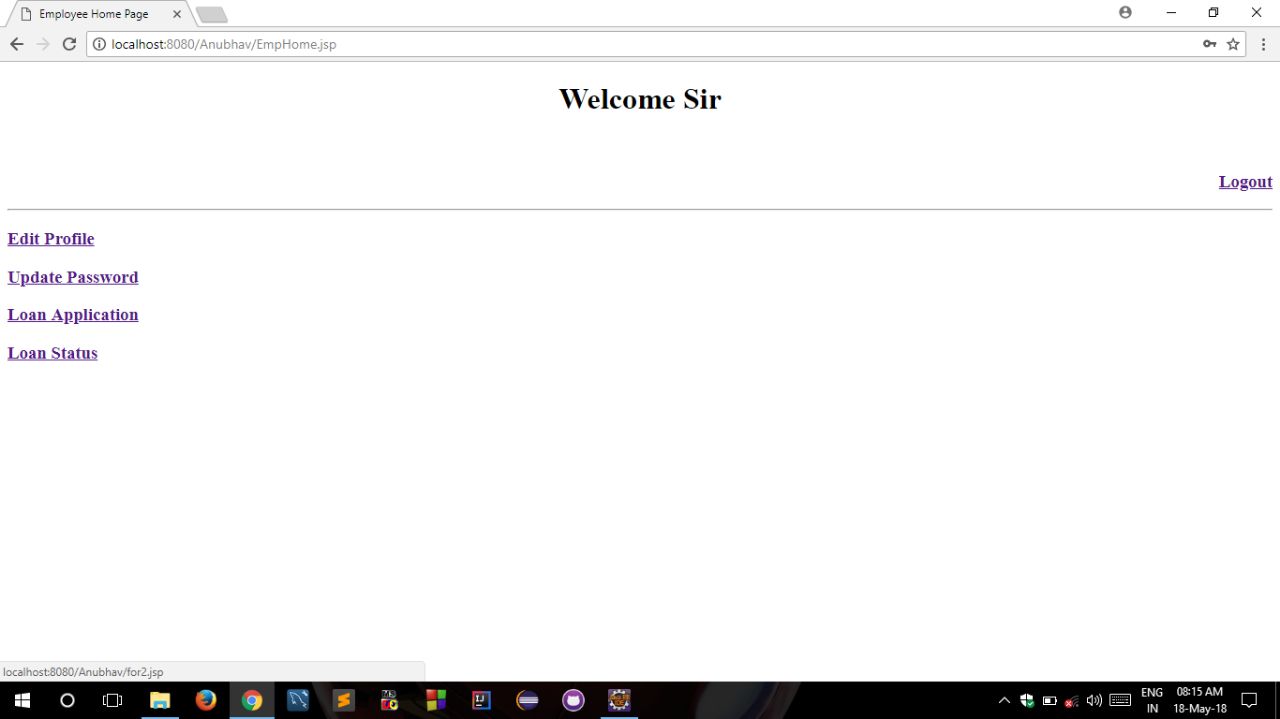
1. Loan instalment details



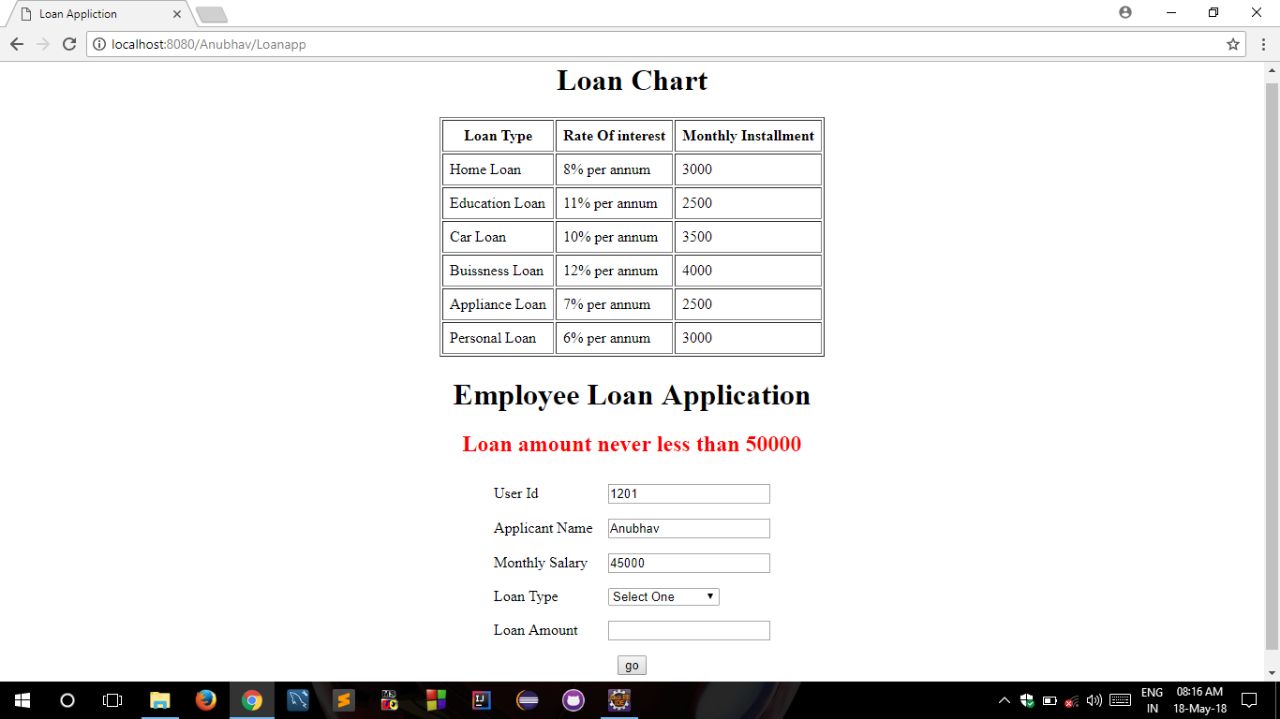
1. Employee Table Records



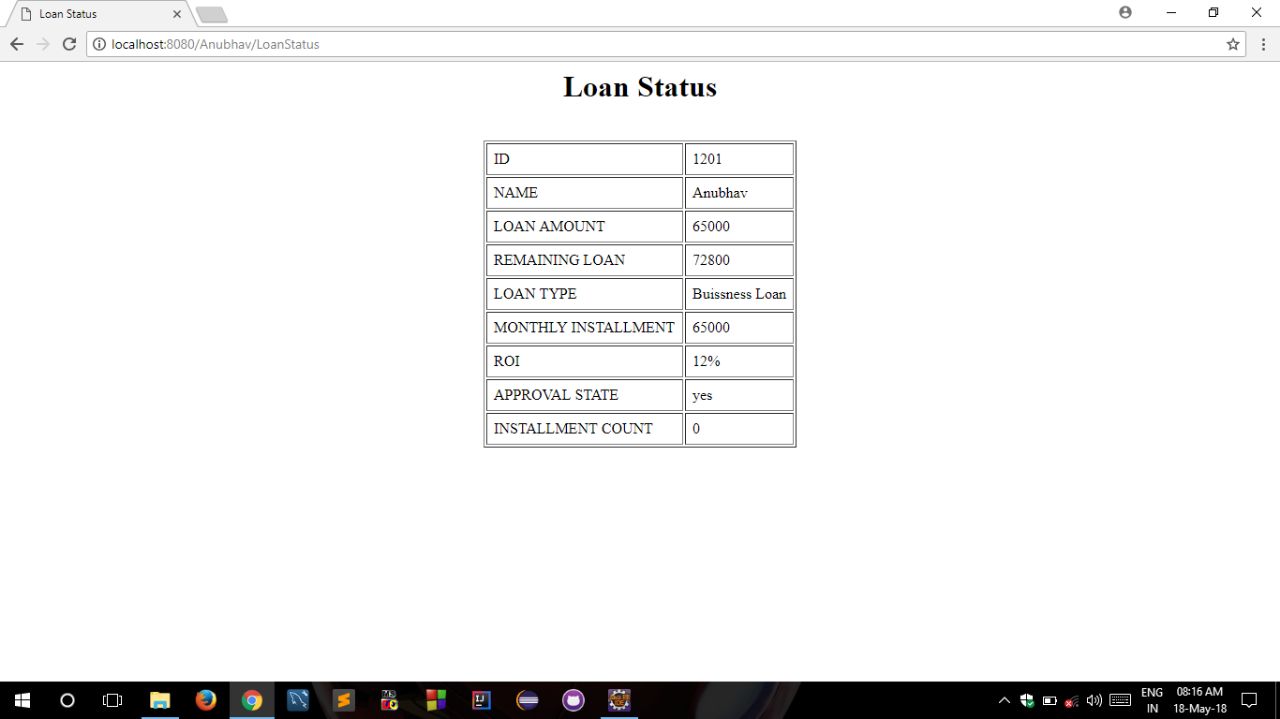
1. Welcome Screen for Employee



1. Loan Chart



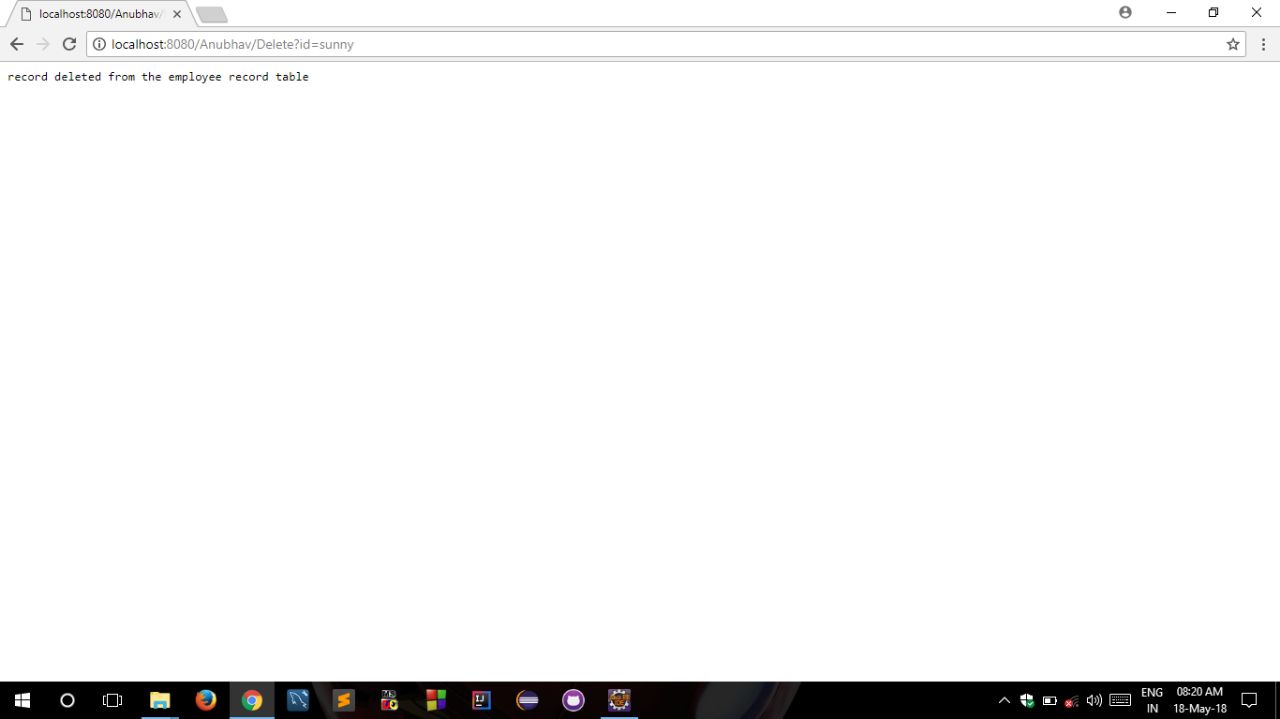
1. Loan Status



1. Loan Request



1. Record Deleted Screen



1. **CODING**

* **Admin login**

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Insert title here</title>

</head>

<body>

<h1 align=*"center"*>ADD NEW EMPLOYEE</h1><br>

<form action=*"Add"*>

<table align=*"center"* cellpadding=*"5"*>

<tr><td>USER ID</td><td><input type=*"text"* name=*"id"*></td></tr>

<tr><td>NAME</td><td><input type=*"text"* name=*"name"*></td></tr>

<tr><td >NEW PASSWORD</td><td><input type=*"password"* name=*"pwd"*></td></tr>

<tr><td>SALARY</td><td><input type=*"text"* name=*"salary"* ></td></tr>

<tr><td>SECURITY QUESTION</td><td>

<select>

<option>Who is your favourite actor?</option>

<option>What's your first pet name ?</option>

<option>What's your nick name ?</option>

<option>Who is your best friend ?</option>

</select>

</td></tr>

<tr><td>Answer</td><td><input type=*"text"* name=*"ans"*></td></tr>

<tr><td colspan=*"2"* align=*"center"*><input type=*"submit"* value=*"Go"* ></td></tr>

</table>

</form>

</body>

</html>

* Add new Employee

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Insert title here</title>

</head>

<body>

<%@ include file=*"db.jsp"* %>

<%

String id=(String) session.getAttribute("k1");

**if**(id==**null**)

{

response.sendRedirect("index.jsp");

}

%>

<h1 align=*"center"*>ADD NEW EMPLOYEE</h1><br>

<form action=*"Add"*>

<table align=*"center"* cellpadding=*"5"*>

<tr><td>USER ID</td><td><input type=*"text"* name=*"id"*></td></tr>

<tr><td>NAME</td><td><input type=*"text"* name=*"name"*></td></tr>

<tr><td>EMAIL ID</td><td><input type=*"text"* name=*"email"*></td></tr>

<tr><td >BLOOD GROUP</td><td>

<select name=*"blood"*><option>-</option>

<option>A+</option><option>A-</option>

<option>B+</option><option>B-</option>

<option>O+</option><option>O-</option>

<option>AB+</option><option>AB-</option>

</select>

</td></tr>

<tr><td >NEW PASSWORD</td><td><input type=*"password"* name=*"pwd"*></td></tr>

<tr><td>SALARY</td><td><input type=*"text"* name=*"salary"* ></td></tr>

<tr><td>ACCOUNT BALANCE</td><td><input type=*"text"* name=*"balance"* ></td></tr>

<tr><td>SECURITY QUESTION</td><td>

<select><option> - </option>

<option>Who is your favourite actor?</option>

<option>What's your first pet name ?</option>

<option>What's your nick name ?</option>

<option>Who is your best friend ?</option>

</select>

</td></tr>

<tr><td>ANSWER</td><td><input type=*"text"* name=*"ans"*></td></tr>

<tr><td colspan=*"2"* align=*"center"*><input type=*"submit"* value=*"Go"* ></td></tr>

</table>

</form>

</body>

</html>

* Admin Home Screen

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Admin Home Page</title>

</head>

<body>

<%@ include file=*"db.jsp"* %>

<%

String id=(String) session.getAttribute("k1");

**if**(id==**null**)

{

response.sendRedirect("index.jsp");

}

%>

<h1 align=*"center"*> Welcome Sir</h1><br><a href=*"Logout"*><h3 align=*"right"*>Logout</h3></a><hr>

<a href=*"add.jsp"*><h3>Add Employee Record</h3></a>

<a href=*"EditEmp"*><h3>Edit Employee Record</h3></a>

<a href=*"for1.jsp"*><h3>Update Password</h3></a>

<a href=*"Loan2"*><h3>Loan Request</h3></a>

<a href=*Inst1*><h3>Loan Installment</h3></a>

</body>

</html>

* Admin Reset Password

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Reset Password</title>

</head>

<body>

<h1 align=*"center"*>Admin Reset Password</h1><br>

<h2 align=*"center"* style="color: *red*">Please Enter Your Id carefully </h2>

<form action=*"Admfor"*>

<table align=*"center"* cellpadding=*"5"*>

<tr><td>USER ID </td><td><input type=*"text"* name=*"id"*></td></tr>

<tr><td>Security Question</td><td>

<select>

<option>Who is your favourite actor?</option>

<option>What's your first pet name ?</option>

<option>What's your nick name ?</option>

<option>Who is your best friend ?</option>

</select>

</td></tr>

<tr><td>Answer</td><td><input type=*"text"* name=*"ans"*></td></tr>

<tr><td colspan=*"2"* align=*"center"*><input type=*"submit"* value=*"Go"*></td></tr>

</table>

</form>

</body>

</html>

* Employee home screen

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Edit Employee</title>

<%@ include file=*"db.jsp"* %>

<%

**try**

{

PreparedStatement ps=con.prepareStatement("select \* from emp");

ResultSet rs=ps.executeQuery();

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

{

**do**

{

response.setContentType("html/text");

out.println("<html>");

out.println("<body>");

out.println("<h1 align=center> </h1><br>");

out.println("<table cellpadding=5><tr><th>ID</th><th>NAME</th><th>SALARY</th><th>EDIT</th><th>DELETE</th</tr>");

out.println("<tr><td>"+rs.getString("id")+"</td><td>"+rs.getString("name")+"</td><td>"+rs.getString("salary")+"</td><td><a href=>S#</a></td><td><a href=>S!</a></td></tr>");

out.println("<html>");

out.println("</table>");

out.println("</body>");

out.println("</html>");

}

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

}

**else**

{

out.println("not registered");

}

con.close();

}

**catch**(Exception e)

{

out.println(e);

}

%>

</head>

<body>

</body>

</html>

* Employee login

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Insert title here</title>

</head>

<body>

<%@ include file=*"db.jsp"* %>

<%

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

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

**try**

{

PreparedStatement ps=con.prepareStatement("select \* from emp where id=? and pwd=?");

ps.setString(1, id);

ps.setString(2, pwd);

ResultSet rs=ps.executeQuery();

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

{

session.setAttribute("k2", id);

response.sendRedirect("EmpHome.jsp");

//out.println("successfully login ");

}

**else**

{

out.println("not registered");

}

con.close();

}**catch**(Exception e)

{

out.println(e);

}

%>

</body>

</html>

* Employee reset password

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Reset Password</title>

</head>

<body>

<h1 align=*"center"*>Employee Reset Password</h1><br>

<h2 align=*"center"* style="color: *red*">Please Enter Your Id carefully </h2>

<form action=*"Empfor"*>

<table align=*"center"* cellpadding=*"5"*>

<tr><td>USER ID </td><td><input type=*"text"* name=*"id"*></td></tr>

<tr><td>Security Question</td><td>

<select>

<option>Who is your favourite actor?</option>

<option>What's your first pet name ?</option>

<option>What's your nick name ?</option>

<option>Who is your best friend ?</option>

</select>

</td></tr>

<tr><td>Answer</td><td><input type=*"text"* name=*"ans"*></td></tr>

<tr><td colspan=*"2"* align=*"center"*><input type=*"submit"* value=*"Go"*></td></tr>

</table>

</form>

</body>

</html>

* Employee home screen

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Employee Home Page</title>

</head>

<body>

<%@ include file=*"db.jsp"* %>

<%

String id=(String) session.getAttribute("k2");

**if**(id==**null**)

{

response.sendRedirect("index.jsp");

}

%>

<h1 align=*"center"*> Welcome Sir</h1><br><a href=*"Logout"*><h3 align=*"right"*>Logout</h3></a><hr>

<a href=*"Edit2"*><h3>Edit Profile</h3></a>

<a href=*"for2.jsp"*><h3>Update Password</h3></a>

<a href=*"Loanapp"*><h3>Loan Application</h3></a>

<a href=*"LoanStatus"*><h3>Loan Status</h3></a>

</body>

</html>

* Employee Login Screen

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Insert title here</title>

<style type=*"text/css"*>

*#tb*{font-size: *large*;}

</style>

</head>

<body>

<h1 align=*"center"*>Employee Login</h1>

<form action=*"Emp.jsp"*>

<table align=*"center"* id=*"tb"* cellpadding=*"5"*>

<tr><td>USER ID </td><td><input type=*"text"* name=*"id"*></td></tr>

<tr><td>PASSWORD </td><td><input type=*"password"* name=*"pwd"*></td></tr>

<tr><td colspan=*"2"* align=*"center"*><input type=*"submit"* value=*"Login"*></td></tr>

<tr><td colspan=*"2"* align=*"center"*><a href=*"Empfor.jsp"*>Forgot Password</a></td></tr>

</table>

</form>

</body>

</html>

* Recover Password Page

<!DOCTYPE html>

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Recover Password Page</title>

</head>

<body>

<form action=*"S1"*>

<h3 align=*"center"*> Submit carefully</h3>

<table cellpadding=*"5"* align=*"center"*>

<tr><td>USER ID</td><td><input type=*"text"* name=*"id"*></td></tr>

<tr><td>New Password</td><td><input type=*"password"* name=*"pwd"*></td></tr>

<tr><td colspan=*"2"* align=*"center"*><input type=*"submit"* value=*"submit"*></td></tr>

</table>

</form>

</body>

</html>

* Index page

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Index Page</title>

</head>

<body>

<h1 align=*"center"* style="color: *blue*">IIPS LOAN MANAGEMENT</h1>

<div class=*"for-img"*></div>

<h1 align=*"center"*>Login here</h1>

<table align=*"center"*>

<tr><a href=*"AdminLog.jsp"*> <h3 align=*"center"*>Admin Login</h3></a></tr>

<tr><a href=*"EmpLogin.jsp"*><h3 align=*"center"*> Employee Login</h3></a></tr>

</table>

</body>

</html>

1. **FUTURE ENHANCEMENTS**

**Future Enhancements:**

It is not possible to develop a system that makes all the requirements of the user. User requirements keep changing as the system is being used. Some of the future enhancements that can be done to this system are:

* As the technology emerges, it is possible to upgrade the system and can be adaptable to desired environment.
* It is based on object-oriented design, any further changes can be easily adaptable.
* Based on the future security issues, security can be improved using emerging technologies.
* Sub admin module can be added.

1. **CONCLUSION**

The Employee Loan Management System has been computed successfully and was also tested successfully by taking “test cases”. It is user friendly and has required options, which can be utilized by the user to perform the desired operations.

The software is developed using Java as front end and Database connective as back end in Windows environment. The goals that are achieved by software are:

* Optimum utilization of resources.
* Efficient management of records.
* Simplification of the operations.
* Less processing time and getting required information.
* User friendly.
* Portable and flexible for future enhancement.

**ABBRIVIATIONS**

ER Entity Relationship

EER Extended Entity Relationship

SQL Standard Query Language

JDBC Java Database Connectivity

JSP Java Server Page

HTML Hyper Text Markup Language

SGML Standard Generalized Markup Language

DFD Data Flow Diagram

JVM Java Virtual Machine

ODBC Open Database Connectivity

GUI Guide User Interface

IDE Integrated Development Environment

API Application Programming Interface

AWT Abstract Window Toolkit

SDK Software Development Kit

STLC Software Testing Life Cycle

TCD Test Case Document