A

Seminar report

On

EMPLOYEE MANAGEMENT SYSTEM

Submitted in partial fulfillment of the requirement for the award of degree
Of MBA

SUBMITTED TO:
www.studymafia.org

SUBMITTED BY:
www.studymafia.org

Acknowledgement

I would like to thank respected Mr…….. and Mr. ……..for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present a seminar report. It helped me a lot to realize of what we study for.

Secondly, I would like to thank my parents who patiently helped me as i went through my work and helped to modify and eliminate some of the irrelevant or un-necessary stuffs.

Thirdly, I would like to thank my friends who helped me to make my work more organized and well-stacked till the end.

Next, I would thank Microsoft for developing such a wonderful tool like MS Word. It helped my work a lot to remain error-free.

Last but clearly not the least, I would thank The Almighty for giving me strength to complete my report on time.

Preface

I have made this report file on the topic; **EMPLOYEE MANAGEMENT SYSTEM** ; I have tried my best to elucidate all the relevant detail to the topic to be included in the report. While in the beginning I have tried to give a general view about this topic.

My efforts and wholehearted co-corporation of each and everyone has ended on a successful note. I express my sincere gratitude to ……..who assisting me throughout the preparation of this topic. I thank him for providing me the reinforcement, confidence and most importantly the track for the topic whenever I needed it.

ABSTRACT

Employee Management System is a distributed application, developed to maintain the details of employees working in any organization. It maintains the information about the personal details of their employees, also the details about the payroll system which enable to generate the payslip. The application is actually a suite of applications developed using Java.

It is simple to understand and can be used by anyone who is not even familiar with simple employees system. It is user friendly and just asks the user to follow step by step operations by giving him few options. It is fast and can perform many operations of a company.

This software package has been developed using the powerful coding tools of JAVA at Front End and Microsoft Sql Server at Back End.The software is very user friendly. The package contains different modules like Employee details. This version of the software has multi-user approach. For further enhancement or development of the package, user's feedback will be considered.

CONTENTS

INTRODUCTION TO THE PROJECT

Employee Management system is an application that enables users to create and store Employee Records. The application also provides facilities of a payroll system which enables user to generate Pay slips too. This application is helpful to department of the organization which maintains data of employees related to an organization .

Java is a platform independent language. Its created applications can be used on a standalone machine as well as on distributed network. More over applications developed in java can be extended to Internet based applications.

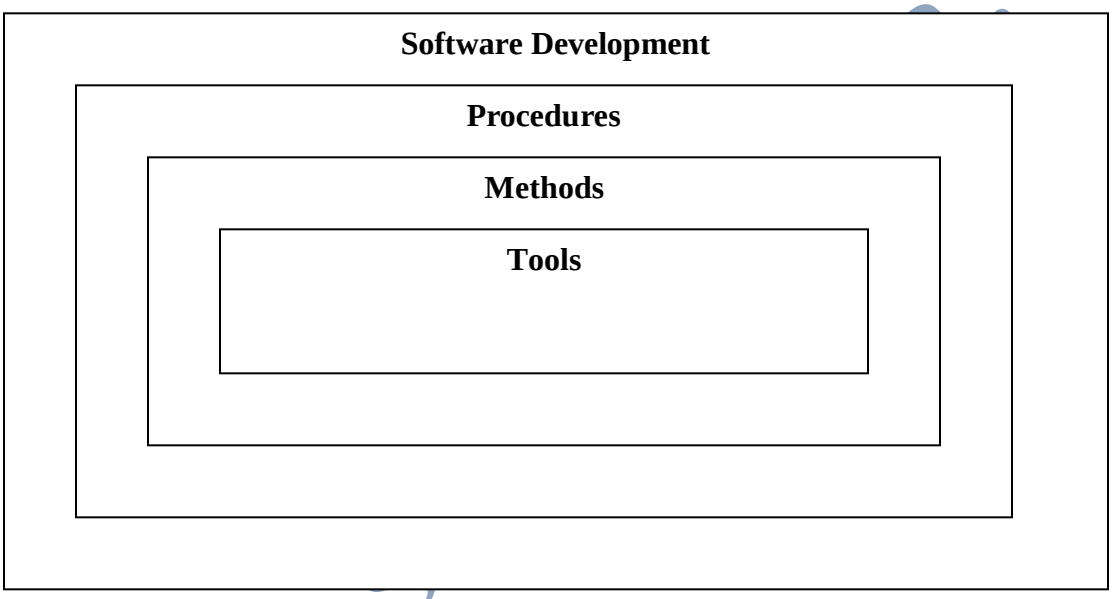
Thus java was chosen as background to design this application.

OBJECTIVE OF THE PROJECT

In this world of growing technologies everything has been computerized. With large number of work opportunities the Human workforce has increased. Thus there is a need of a system which can handle the data of such a large number of Employees in an organization. This project simplifies the task of maintain records because of its user friendly nature.

SYSTEM SPECIFICATIONS

Software Engineers have been trying various tools, methods and procedures to control the process of software development in order to build high quality software with high productivity. This method provides "how it is" for building the software while the tools provide automated or semi automated support for the methods. They are used in all stages of software development process, namely, planning, analysis, design, development and maintenance. The software development procedure integrates the methods and tools together and enables rational and timely development of the software system.



They provide the guidelines as how to apply these methods and tools, how to produce the deliverable at each stage, what controls to apply, and what milestones to use to assess the performance of the program. There exist a number of software development paradigms each using a different set of methods and tools. The selection of a particular paradigm depends on the nature of application of the programming language used for the controls and the deliverables required. The development of such successful systems depends not only on the use of appropriate methods and techniques but also the developers' commitment to the objective of the system.

A successful system must: -

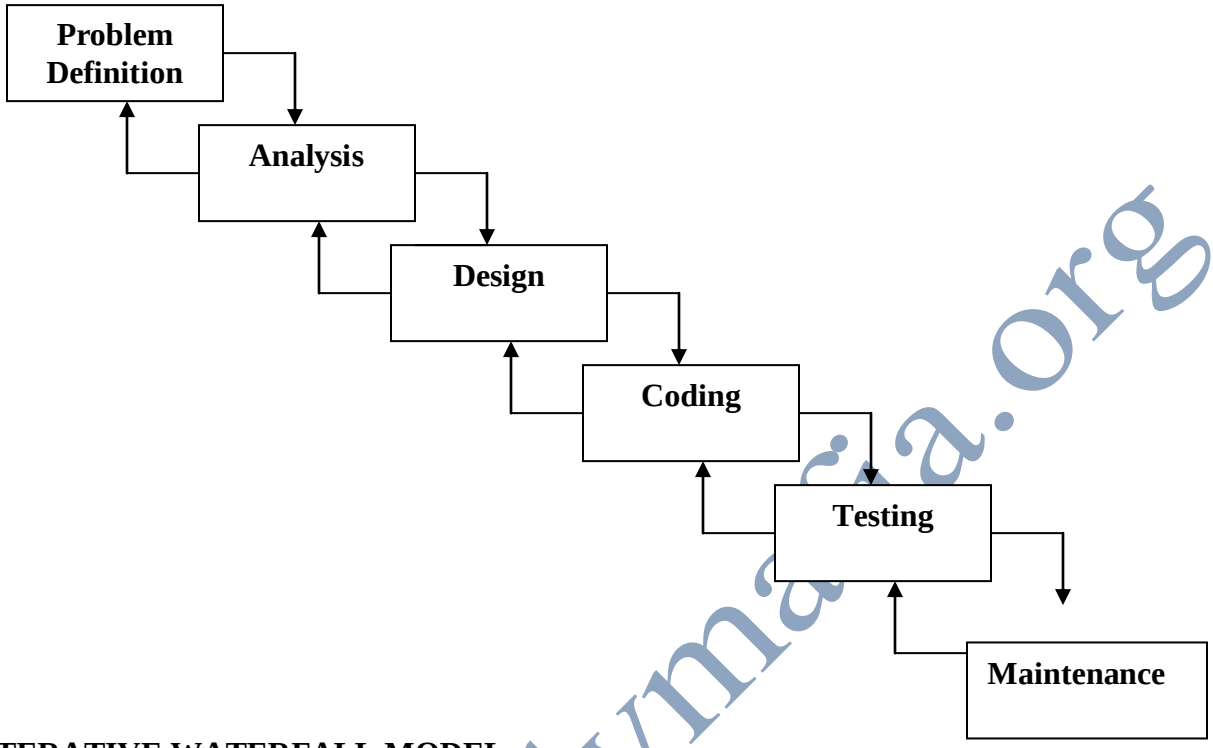
1. Satisfy the user requirements
2. Be easy to understand by user and operator
3. Be easy to operate
4. Have a good user interface
5. Be easy to modify
6. Be expandable
7. Have adequate security control against the misuse of data
8. Handle the errors and exceptions satisfactorily
9. Be delivered on schedule within the budget

SOFTWARE LIFE CYCLE MODEL

The series of stages depicting the various tasks involved in development process is called Software Life Cycle Model. The model implemented by us is Waterfall Model, which is most widely used in procedure-oriented development. This model attempts to break up the identifiable activities into series of actions, each of which must be completed before the next begins. The activities include problem definition, requirement analysis, design, coding, testing, maintenance. Further refinements to this model include iteration back to previous stages in order to incorporate any changes or missing links.

The phases and the outputs of the waterfall model at each phase are summarized below in tabular format.

PHASE	**OUTPUT**
Problem Description	Feasibility Report
Analysis	SRS
Design	Detailed Design Report
Coding	Complete Source Code
Testing	Test plan, report and manual
Installation	Installation Report



ITERATIVE WATERFALL MODEL

Software development life cycle process specifies a method of developing the software. Each software development projects starts with some needs and ends with some software that satisfies those needs. A software development life cycle specifies the set of activities that should be preformed to go from user needs to final products. There are different models of SDLC process and each model specifies the activities and the order in which they should be preformed. Depending on the nature of project, a suitable model is chosen and the entire process of software requirement analysis, design, coding, testing and maintenance is preformed accordingly.

Various life cycle models are present, but our system is based on WATER FALL MODEL, which is most widely used in procedure oriented development. This model attempts to break up the identifiable activities into series of action, each of which must be completed before the next begins. The activities include Problem definition, Requirement Analysis, Design, Coding, Testing and maintenance.

An initial investigation culminates in a proposal that determines whether a system is feasible or not. It determines its workability, impact on the organization, ability to meet user needs, and effective user resources. The objective of feasibility study is not solve the problem but to acquire a sense of its scope. During the study, the problem definition is crystallized and aspects of the problem to be included in the system are determined. Consequently, cost and benefits are estimated with greater accuracy at this stage. This is a bridge in between the User Requirements and the output that he can avail under a set of given constraints, inputs and outputs.

The main steps are :

. Statement of constraints

. Identification of specific system objectives
. Description of outputs

SOFTWARE REQUIREMENT SPECIFICATION (SRS)

The aim of the system is to develope **"EMPLOYEE MANAGEMENT SYSTEM"** software, which should automate the process to create and store employee details . The system is supposed to be used as a subsystem in a large office system, which could be manual system or a computerized one. Therefore, the proposed system must be able to function under both circumstances.

The proposed system is not a freeware and due to the usage of swings, becomes user interactive.

- The project demand a page of employee details that include:
- Employees personel detail.
- Employees salary, allowances, deductions.

HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements

- ✓ **MEMORY SPACE:**
  - Minimum - 32 MB
  - Recommended- - 64 MB
- ✓ HDD - To install the software at least 2 GB and the data storage is depending upon the organizational setup.
- ✓ PROCESSOR - Intel Pentium IV, 1GHZ or above
- ✓ RAM - 256MB or above
- ✓ VIDEO - 1024x768, 24-bit colors
- ✓ KEYBOARD - Standard 104 Keys(QWERTY)

Blank writable CD to keep the backup of the Package.

Software Requirements

- ✓ OPERATING SYSTEM - WindowsXP Professional
- ✓ DEVELOPING LANGUAGE - JAVA(jdk-5)
- ✓ DATABASE - MICROSOFT SQL SERVER 2005

TOOLS, PLATFORM AND LANGUAGE USED

FRONT END:

The programming has been done using the language Java. It is Sun Microsystem's strategic language for platform independent programming. It is easy to use, efficient and flexible. This language is preferred because one can build a program using this object oriented and platform independent programming with less effort than with any other programming language. It's a natural language for building database applications, owing to the level and sophistication of the tools included with the language.

BACK END:

Microsoft Sql Server is one of the leading database management systems available on the market today. It is easy to use and administer, and it comes with tools and wizards that make it easy to develop applications. The database itself has been redesigned to automatically perform many tuning functions, leaving you free to focus on most important tasks.

PLATFORM USED:

The Acconts Automation System is targeted at Microsoft Windows platforms.

WINDOWS XP PROFESSIONAL

Windows (Operating System)

a) Window derived from its name from the on-screen "WINDOWS" that it used to display information's.

b) Windows XP Professional is a multi-user operating system. It has been designed and developed by Microsoft Co-operation.

c) When a computer is switch on for working, it needs the operating system because all the activities of a system are supervised by the operating system.

The features of the operating system are:-

1. Provides an interactive environment.

2. Graphical user interfaces.

   i. The Commands are displayed on the screen and we don't have to remember all commands.
   ii. GUI makes it easy to work with disks and directories. It can display tree like diagram of directories, directories sub-0 directories on hard disk.

3. Point and Click.

4. User friendly.

***LANGUAGE USED – JAVA*:**

Java is a general computer programming language developed by Sun Microsystems. Originally called "Oak", by its inventor James Gosling, **Java** was designed with several innovative features. These include a language that is entirely object oriented, and the ability to write an application once and move it to virtually any platform.

INNOVATIONS

The benefits of object-oriented programming are not necessarily obvious to non-programmers. Essentially what this means to managers is that code written in **Java** will be easier to maintain and reuse in the long run. The downside is that object-oriented programming requires better planning from the beginning of a project, and may increase the amount of initial development time a project requires. For this reason alone, **Java** is rarely a good choice for small projects requiring a fast turnaround time.

The second innovation that **Java** provides, platform neutrality, is perhaps the greatest reason for its wide acceptance. The fact that **Java** was originally intended a language for writing device controllers for items such as garage openers and microwave ovens is the key reason for this. In practice, however, this ability has been more useful in writing enterprise class business applications, where mission critical software may be required to run on a variety of platforms over its lifetime. Theoretically at least, once compiled, a **Java** binary should be able to run on any machine that also has a piece of software called a **Java** Virtual Machine. In reality, this is not always the case. However, more often than not **Java** does succeed in this regards, whereas this is impossible with an application written in a language such as C++.

JAVA IN WEB DEVELOPMENT

In terms of web development, **Java** is frequently used in two ways. Most commonly **Java** is used to write server-side web applications using two technologies: JSPs and servlets. Using **Java** in this capacity is a good choice for complex applications, that will have large numbers (~1000+) of concurrent users, and will be developed by a team of programmers. Less complex projects, with fewer concurrent users may have better outcomes when developed in procedural scripting languages such as PHP or PERL.

The second way in which **Java** is used, is to create special, browser embeddable, programs called Applets. While applets had a brief period of acceptance, their use is becoming increasingly rare, being replaced by a number of technologies such a Flash and JavaScript, which are more effective at providing design enhancements such as animation and rollovers. Using applets for these purposes is a mistake frequently made by beginning developers. Still, Applets do have a place in writing specialized browser-based applications that cannot be accomplished by these other technologies. Careful consideration should be made before approving any project that uses Applets.

ORIGIN AND GROWTH

Like many recently developed computer languages, **Java** borrows much of its language design from C/C++. For that reason, many programmers who are proficient in those languages have leaned **Java**, and provide a large pool of qualified developers. **Java** has gained additional ground as a first language, as it is generally simpler to master than C++, another commonly used programming language.

CLASSES AND OBJECTS

CLASSES

In the real world, you'll often find many individual objects all of the same kind. There may be thousands of other bicycles in existence, all of the same make and model. Each bicycle was built from the same set of blueprints and therefore contains the same components. In object-oriented terms, we say that your bicycle is an instance of the class of objects known as bicycles. A class is the blueprint from which individual objects are created.

OBJECTS

Objects are key to understanding object-oriented technology. Look around right now and you'll find many examples of real-world objects: your dog, your desk, your television set, your bicycle.

Real-world objects share two characteristics: They all have state and behavior. Dogs have state (name, color, breed, hungry) and behavior (barking, fetching, wagging tail). Bicycles also have state (current gear, current pedal cadence, current speed) and behavior (changing gear, changing pedal cadence, applying brakes). Identifying the state and behavior for real-world objects is a great way to begin thinking in terms of object-oriented programming.

Take a minute right now to observe the real-world objects that are in your immediate area. For each object that you see, ask yourself two questions: "What possible states can this object be in?" and "What possible behavior can this object perform?". Make sure to write down your observations. As you do, you'll notice that real-world objects vary in complexity; your desktop lamp may have only two possible states (on and off) and two possible behaviors (turn on, turn off), but your desktop radio might have additional states (on, off, current volume, current station) and behavior (turn on, turn off, increase volume, decrease volume, seek, scan, and tune). You may also notice that some objects, in turn, will also contain other objects. These real-world observations all translate into the world of object-oriented programming.

Software objects are conceptually similar to real-world objects: they too consist of state and related behavior. An object stores its state in fields (variables in some programming languages) and exposes its behavior through methods (functions in some programming languages). Methods

operate on an object's internal state and serve as the primary mechanism for object-to-object communication. Hiding internal state and requiring all interaction to be performed through an object's methods is known as data encapsulation — a fundamental principle of object-oriented programming.

By attributing state (current speed, current pedal cadence, and current gear) and providing methods for changing that state, the object remains in control of how the outside world is allowed to use it. For example, if the bicycle only has 6 gears, a method to change gears could reject any value that is less than 1 or greater than 6.

Bundling code into individual software objects provides a number of benefits, including:

1. **Modularity:** The source code for an object can be written and maintained independently of the source code for other objects. Once created, an object can be easily passed around inside the system.

2. **Information-hiding:** By interacting only with an object's methods, the details of its internal implementation remain hidden from the outside world.

3. **Code re-use:** If an object already exists (perhaps written by another software developer), you can use that object in your program. This allows specialists to implement/test/debug complex, task-specific objects, which you can then trust to run in your own code.

4. **Pluggability and debugging ease:** If a particular object turns out to be problematic, you can simply remove it from your application and plug in a different object as its replacement. This is analogous to fixing mechanical problems in the real world. If a bolt breaks, you replace it, not the entire machine.

INHERITANCE

Different kinds of objects often have a certain amount in common with each other. Mountain bikes, road bikes, and tandem bikes, for example, all share the characteristics of bicycles (current speed, current pedal cadence, current gear). Yet each also defines additional features that make them different: tandem bicycles have two seats and two sets of handlebars; road bikes have drop handlebars; some mountain bikes have an additional chain ring, giving them a lower gear ratio.

Object-oriented programming allows classes to inherit commonly used state and behavior from other classes. For example, `Bicycle` becomes the superclass of `MountainBike`, `RoadBike`, and `TandemBike`. In the Java programming language, each class is allowed to have one direct superclass, and each superclass has the potential for an unlimited number of subclasses.

INTERFACE

As you've already learned, objects define their interaction with the outside world through the methods that they expose. Methods form the object's interface with the outside world; the buttons on the front of your television set, for example, are the interface between you and the electrical wiring on the other side of its plastic casing. You press the "power" button to turn the television on and off.

In its most common form, an interface is a group of related methods with empty bodies. A bicycle's behavior, if specified as an interface, might appear as follows:

```
interface Bicycle {
                                    void changeCadence(int newValue);

                                    void changeGear(int newValue);

                                   void speedUp(int increment);

                                  void applyBrakes(int decrement);
                          }
```

To implement this interface, the name of your class would change (to `ACMEBicycle`, for example), and you'd use the `implements` keyword in the class declaration:

```
class ACMEBicycle implements Bicycle {

   // remainder of this class implemented as before

}
```

Implementing an interface allows a class to become more formal about the behavior it promises to provide. Interfaces form a contract between the class and the outside world, and this contract is enforced at build time by the compiler. If your class claims to implement an interface, all methods defined
by that interface must appear in its source code before the class will successfully compile.

PACKAGE

A package is a namespace that organizes a set of related classes and interfaces. Conceptually you can think of packages as being similar to different folders on your computer. You might keep HTML pages in one folder, images in another, and scripts or applications in yet another. Because software written in the Java programming language can be composed of hundreds or thousands of individual classes, it makes sense to keep things organized by placing related classes and interfaces into packages.

The Java platform provides an enormous class library (a set of packages) suitable for use in your own applications. This library is known as the "Application Programming Interface", or "API" for short. Its packages represent the tasks most commonly associated with general-purpose

programming. For example, a `String` object contains state and behavior for character strings; a `File` object allows a programmer to easily create, delete, inspect, compare, or modify a file on the filesystem; a `Socket` object allows for the creation and use of network sockets; various GUI objects control buttons and checkboxes and anything else related to graphical user interfaces. There are literally thousands of classes to choose from. This allows you, the programmer, to focus on the design of your particular application, rather than the infrastructure required to make it work.

The Java Platform API Specification contains the complete listing for all packages, interfaces, classes, fields, and methods supplied by the Java Platform 6, Standard Edition. Load the page in your browser and bookmark it. As a programmer, it will become your single most important piece of reference documentation.

GUI SWING WIDGETS

A ***Graphical User Interface (GUI)*** is a visual paradigm which allows a user to communicate with a program in an intuitive way. Its main features are ***widgets*** (aka controls) and ***event driven activities***. Clients expect a graphical interface in an application.

Java has two GUI packages, the original ***Abstract Windows Toolkit (AWT)*** and the newer ***Swing***. AWT uses the native operating system's window routines and therefore the visual effect is dependent on the run-time system platform. But this is contrary to the concept of having a virtual model. ***Swing*** allows three modes: a unified 'Java' look and feel [the default], the native platform look, or a specific platform's look. Swing is built on the original objects and framework of AWT. Swing components have the prefix ***J*** to distinguish them from the original AWT ones (eg JFrame instead of Frame). To include Swing components and methods in your project you must import the java.awt.*, java.awt.event.*, and javax.swing.* packages.

CONTAINERS, FRAMES AND CONTENT PANES

Containers are ***widgets*** (GUI controls) that are used to hold and group other widgets such as text fields and checkboxes. Displayable ***frames*** are top-level containers such as JFrame, JWindow, JDialog, JApplet and JInternalFrame which interface to the operating system's window manager. Non-displaying ***content panes*** are intermediate containers such as JPanel, JOptionPane, JScrollPane, JLayeredPane, JSplitPane and JTabbedPane which organize the layout structure when ***multiple*** controls are being used.

JWindow is an unadorned container that has been superceded for the most part by ***JDialog***. However it does provide a useful container for a splash screen.

COMMON WIDGET METHODS

Several methods can be used with many different controls. These include: add(), requestFocus(), setToolTipText().

JFRAME AND JPANEL

JFrame is the most commonly used top-level container. It adds basic functionality such as minimize, maximize, close, title and border to basic frames and windows. Some important JFrame methods are: setBounds(x,y,w,h), setLocation(x,y), setSize(w,h), setResizable(bool), setTitle(str), setVisible(bool), isResizable() and getTitle(). The setDefaultCloseOperation(constant) method controls the action that occurs when the close icon is clicked. Normally the constant used is ***JFrame.EXIT_ON_CLOSE.***

JPanel is the most commonly used content pane. An instance of the pane is created and then added to a frame. The ***add()*** method allows widgets (GUI components) to be added to the pane. The way they are added is controlled by the current layout manager.

LABELS, ICONS AND BUTTONS

Labels are non-interactive text objects most commonly used as prompts. They are created using the ***JLabel()*** constructor with the required text as the first parameter. Another parameter can be added using a SwingConstant value to set horizontal alignment. Vertical alignment is through the ***setVerticalAlignment()*** method. The contents of a label can be changed with the ***setText()*** method.

Icons can be easily added to labels or other controls either to brand, dress up, or aid accessibility. Icons are constructed from the ***ImageIcon*** class and then added as a parameter to the label (or other) control. An extra parameter can be used to control the ***position*** of the text relative to the icon. It must use one of the ***SwingConstants*** values.

Simple ***buttons*** are used to start operations. They are created with the ***JButton()*** constructor. They can be deactivated with the ***setEnabled(false)*** method and tested with the ***isEnabled()*** method. One useful button method is ***setMnemonic(char)*** which allows a hot key to be associated with the button

Simple buttons require an ***ActionEvent*** event listener that reacts to the button click.
Toggle buttons are a visual push on - push off mechanism. They are created with the JToggleButton() constructor. The ***isSelected()*** method returns the state of the button. And in addition to ***ActionEvent***, the ***ChangeEvent*** is triggered.

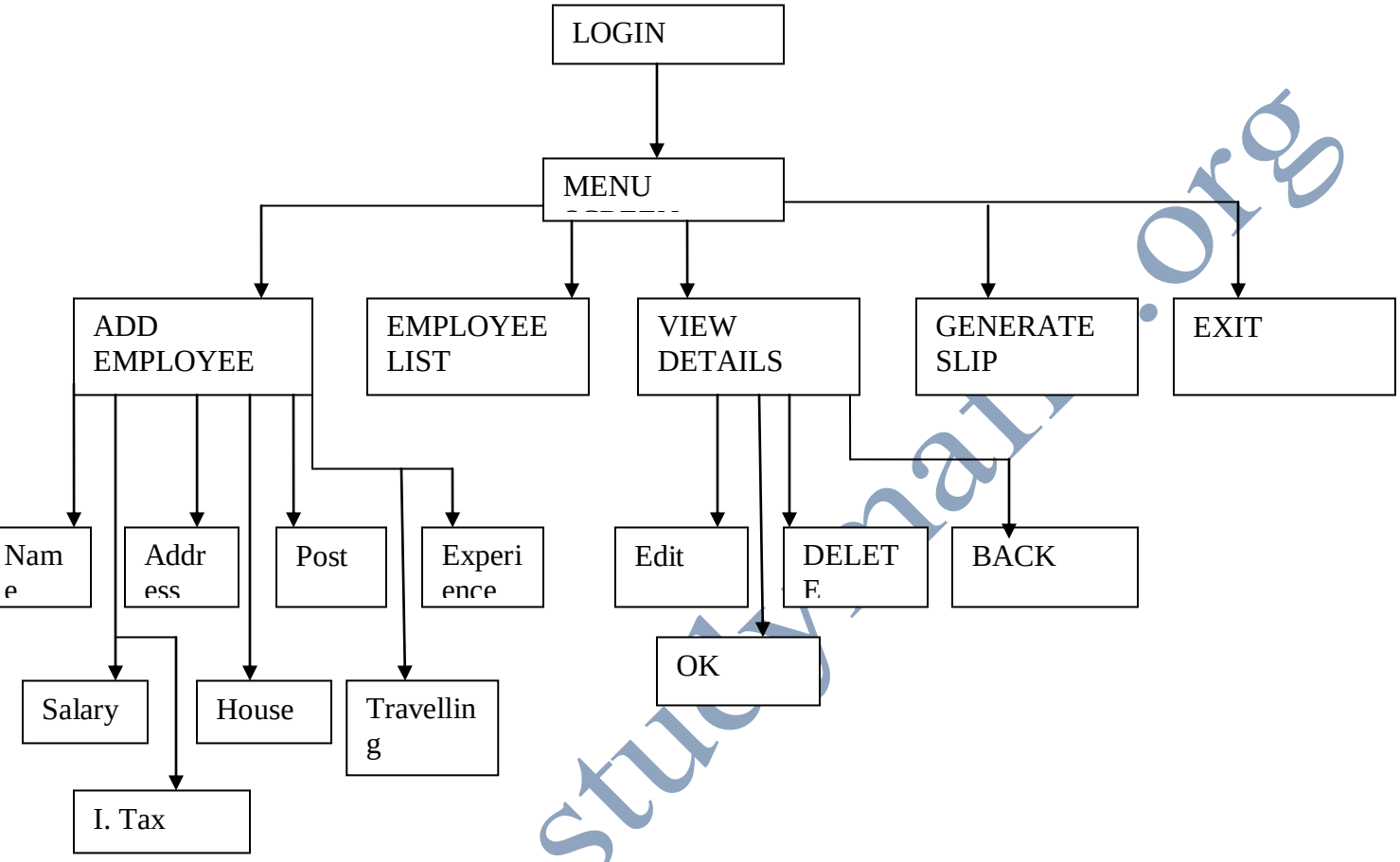
EVENT LISTENERS

GUIs are ***event-based***. That is they respond to buttons, keyboard input or mouse activities. Java uses ***event listeners*** to monitor activity on specified objects and react to specific conditions. For a listing of useful event listeners check the appendix. For techniques on organizing many different events in larger projects, view advanced event listeners.

The first step in adding a basic ***button push event*** handler to the above example is to import ***awt.event.**** which contains all of the event classes. Next add the phrase ***implements ActionListener*** to the class header. Register event listeners for each button widget using the ***addActionListener(this)*** method.

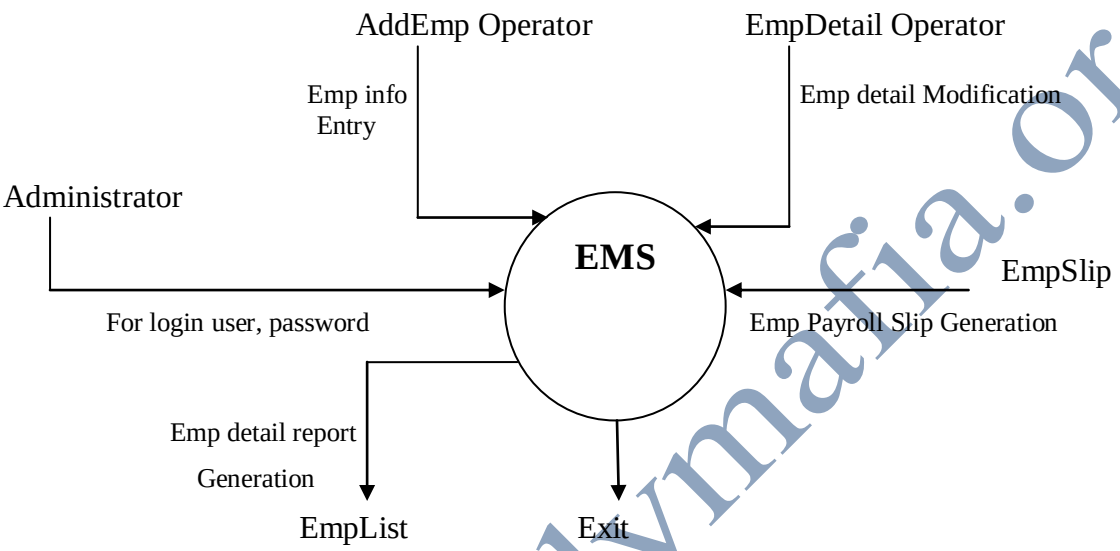
SYSTEM DESIGN

Process Diagram

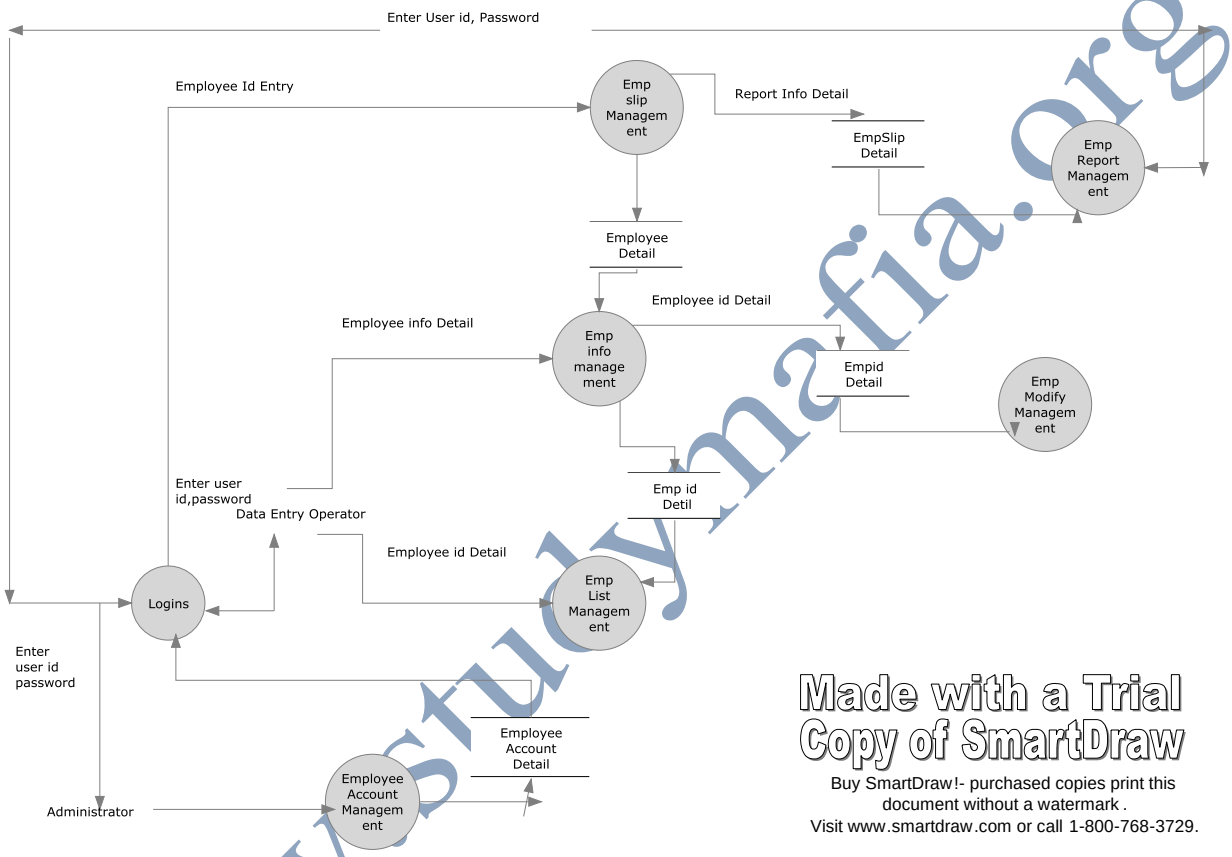


DATA FLOW DIAGRAMS

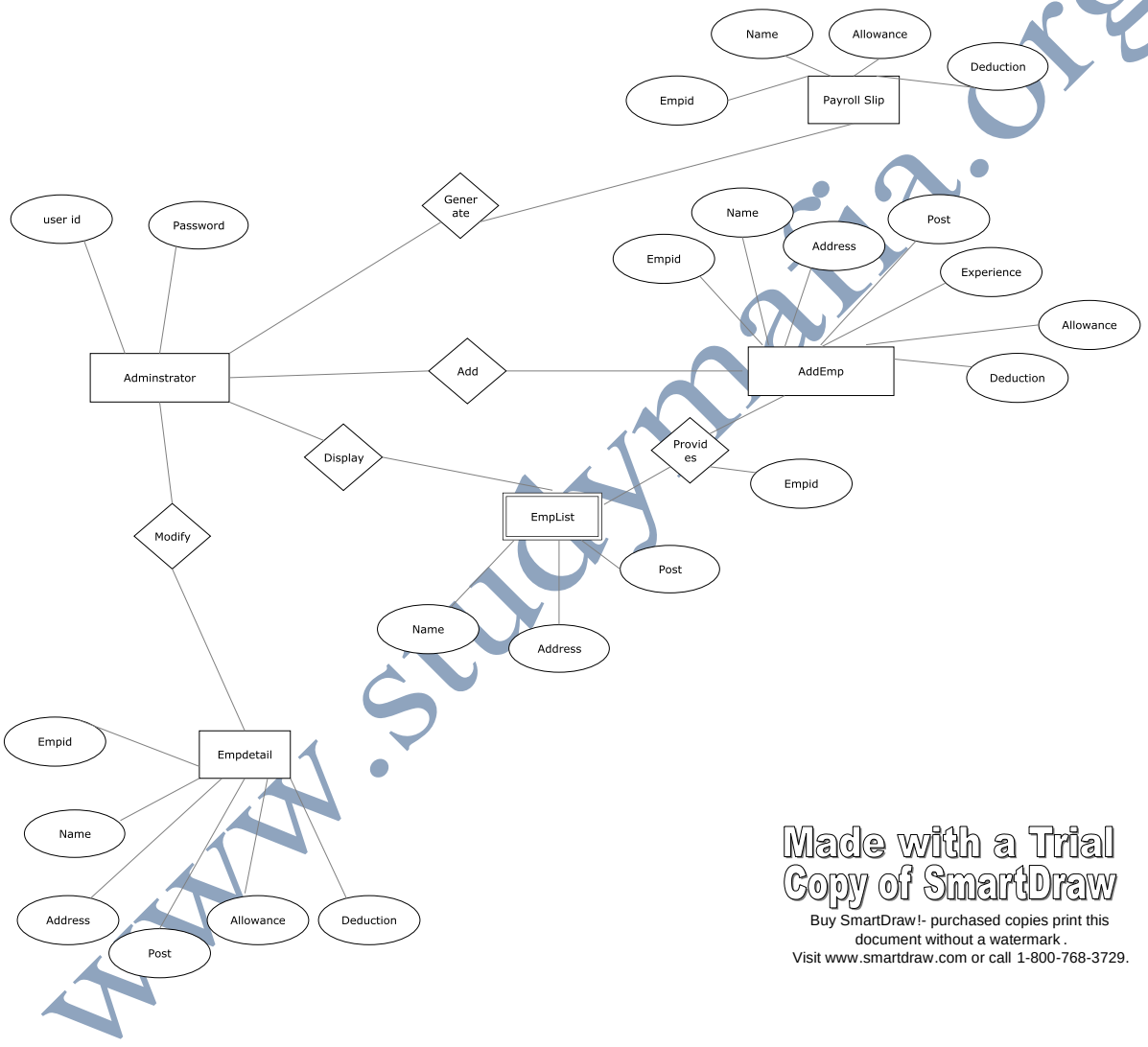
LEVEL 0 DFD:



LEVEL 1 DFD:



ENTITY RELATIONSHIP DIAGRAM:

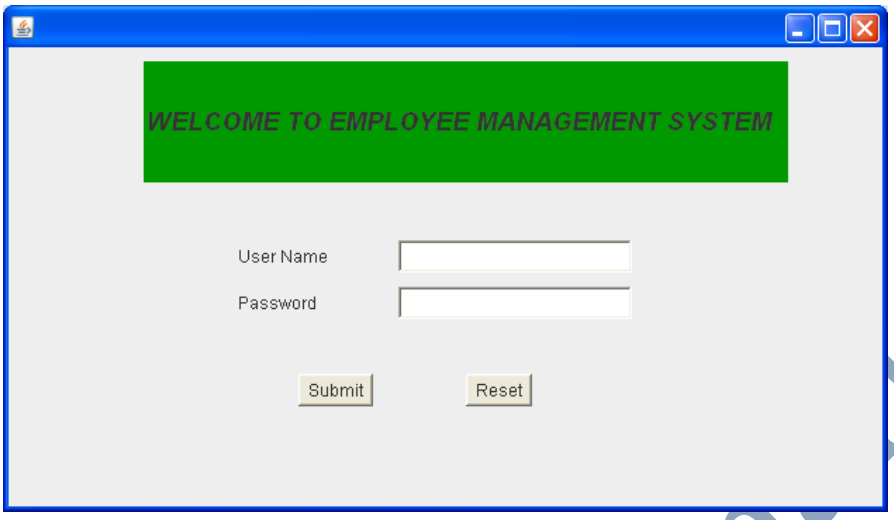


FORMS LAYOUT

Package employeeManagement

Class Summary	
AddEmp	This class provides code for Addition of Employee to the Firm
EmpDetail	Provides code to view Employee Details and also to Delete/Modify Accounts
EmpList	Provides the code to show the List of employees in the Firm
EmpSlip	Provides codes for Salary Slip Generation
Intro	Contains code for Welcome/Login Page of the project
Main	The Base class which has the code to show the First/Login Screen
Menuscrn	This class has code which represents the Screen which shows the Buttons for Selection of Different Tasks

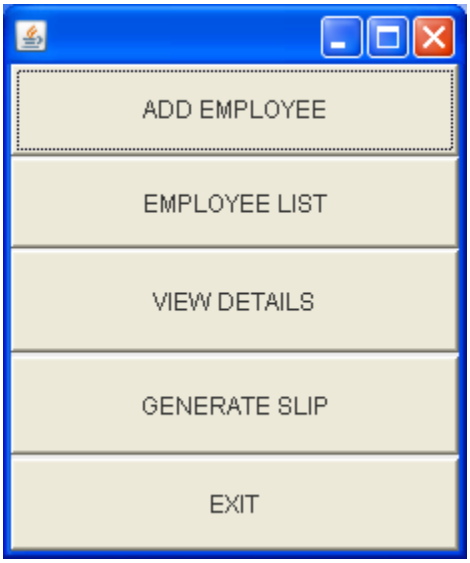
Intro.java



Contains code for username /Password Authentication. When user enters correct login and password It opens Main form otherwise it gives an error message. You can exit by clicking the "X" button on the right top corner of the window.

Press submit button to submit your request and reset to reset the fields.

MenuScrn.java



Using Add employee you can add employee to the firm. Clicking on it opens the add employee window. It calls the AddEmp Class.

Clicking on Employee List shows the list of employees in the firm. It calls the EmpList Class.

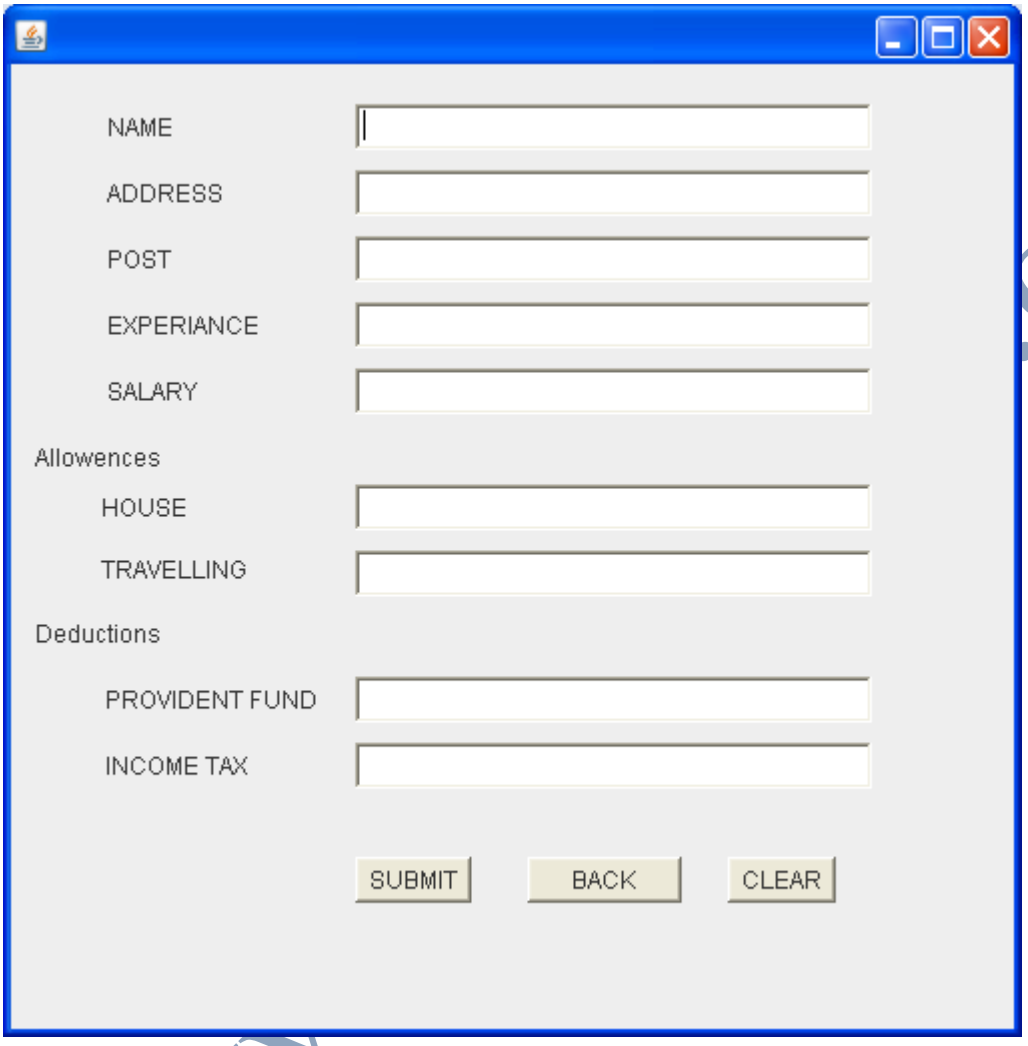
By Clicking on View details you can View/Modify/Delete Records of the employees. It calls EmpDetail.java.

By clicking on generate slip Salary slip is generated. It calls EmpSlip.java

On clicking Exit The Application exits.

Note: You cannot exit using X button in this window.

AddEmp.java

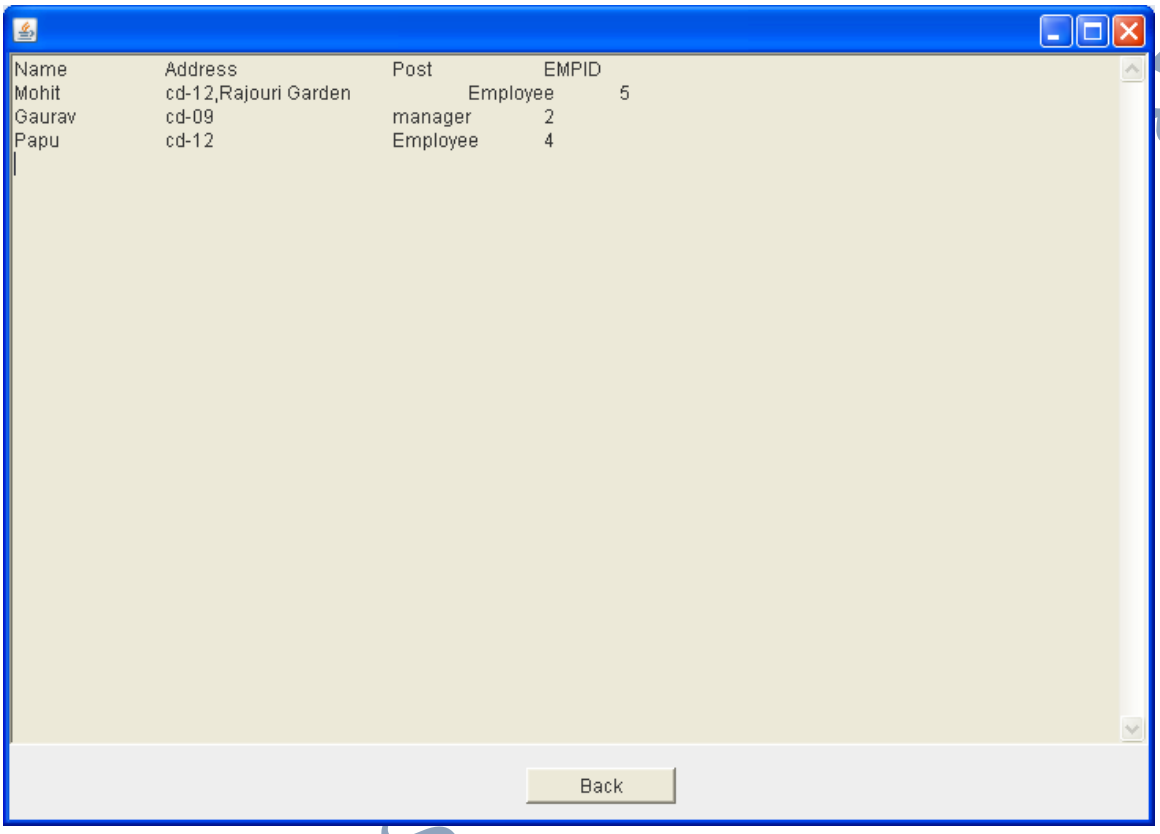


Clicking Back takes you back to main menu screen

Clicking Submit submits the records.

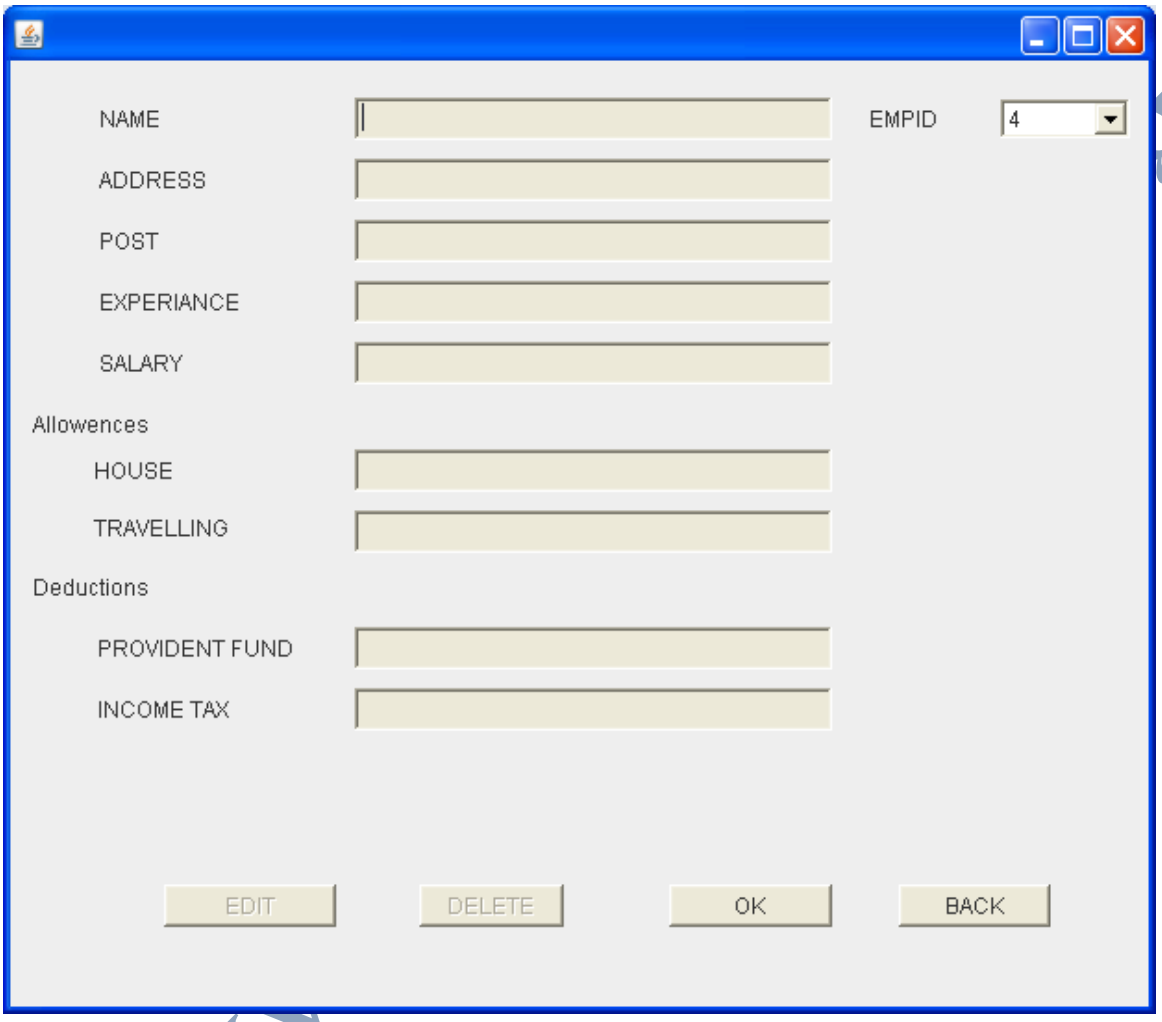
Clicking Clear Clears the Fields

EmpList.java

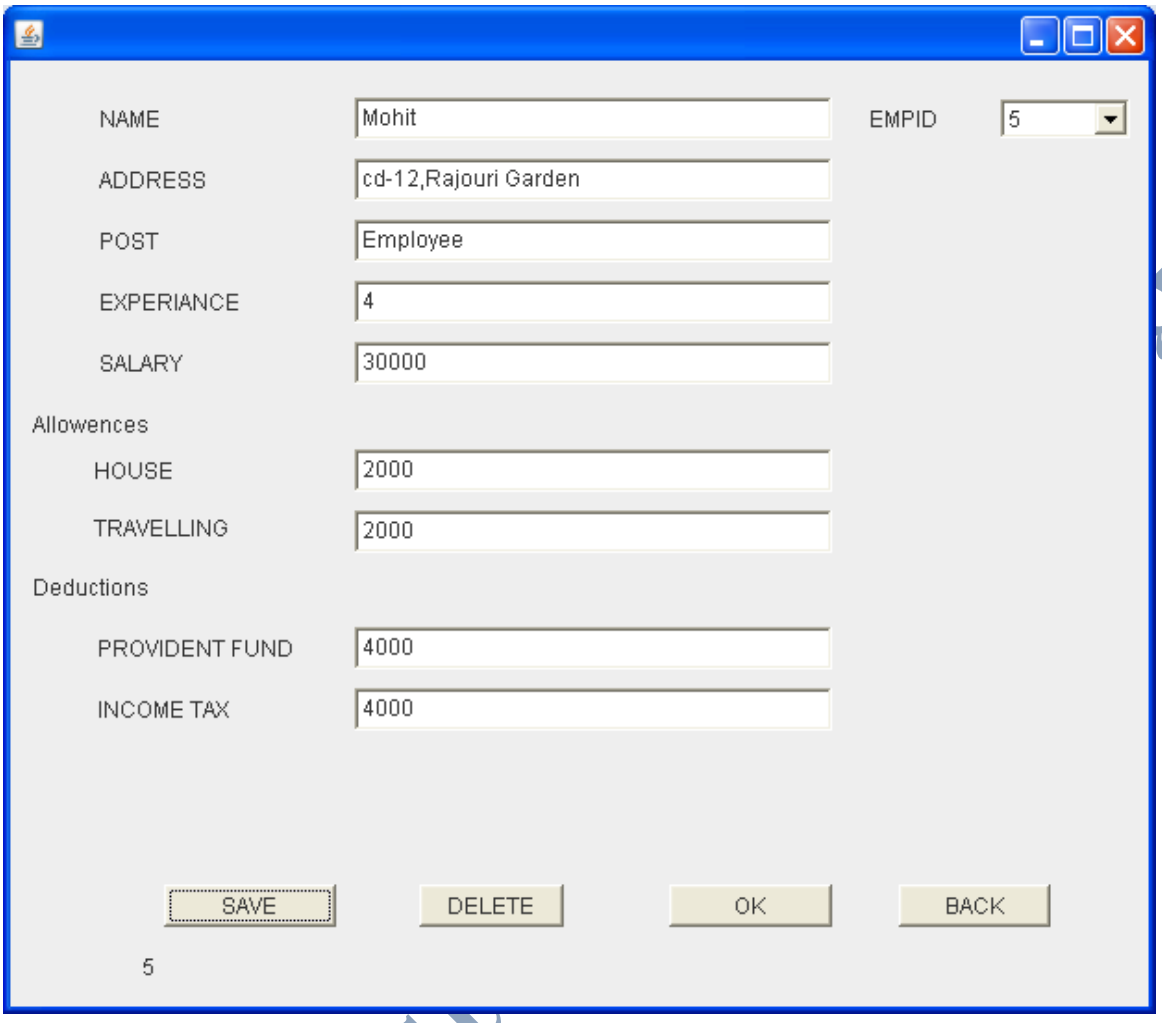


Shows the list of employees. On clicking Back takes you back to the Main Menu screen

EmpDetail.java

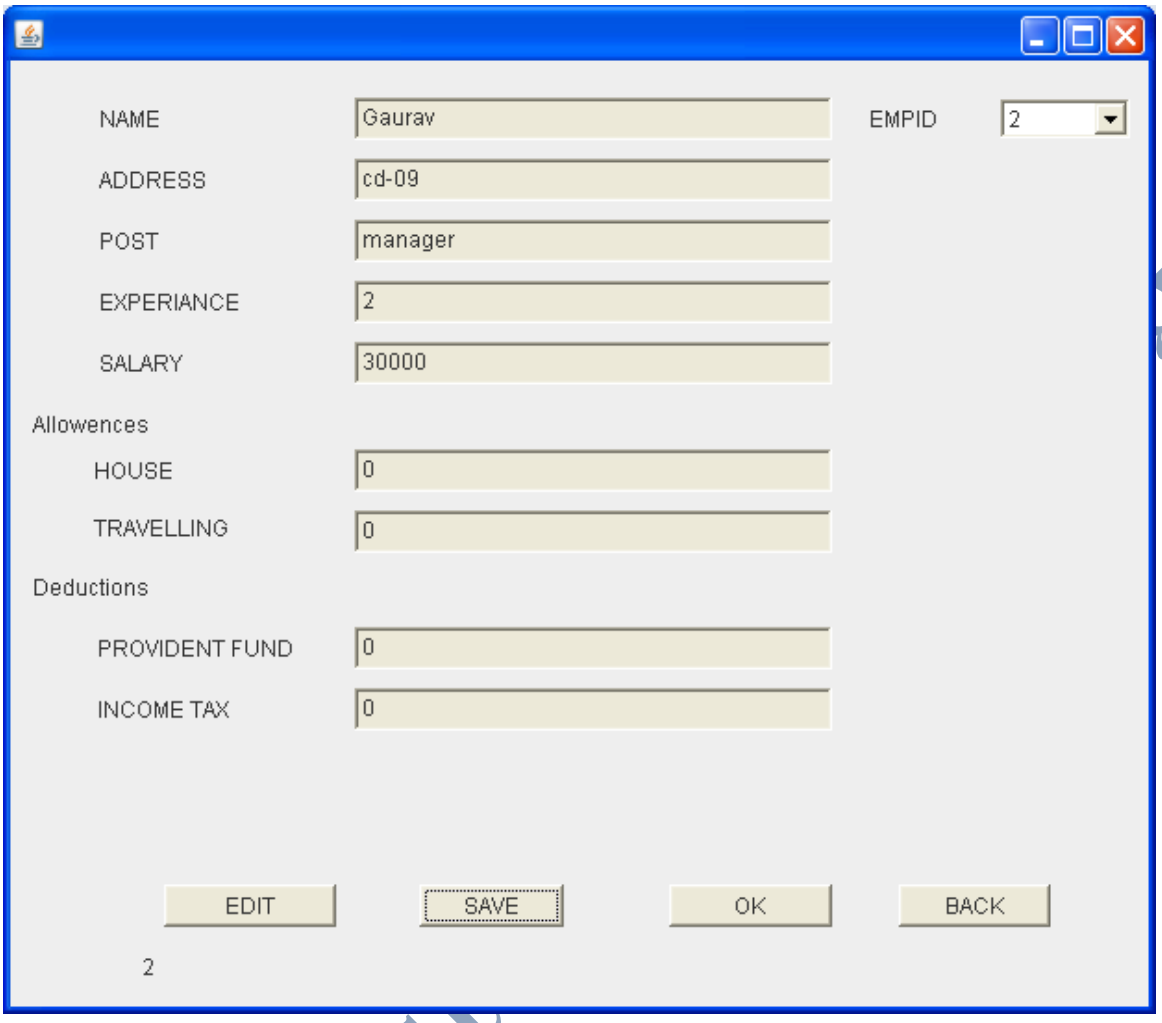


To be able to edit records you must press Edit. Before that please select a record and press OK. This will enable the Edit and Delete Buttons. As Shown below



Clicking on Edit will change it to save after making changes Please click on save to save the changes.

Clicking on Delete will change it Save Please press Save to Confirm the deletion. As shown Below

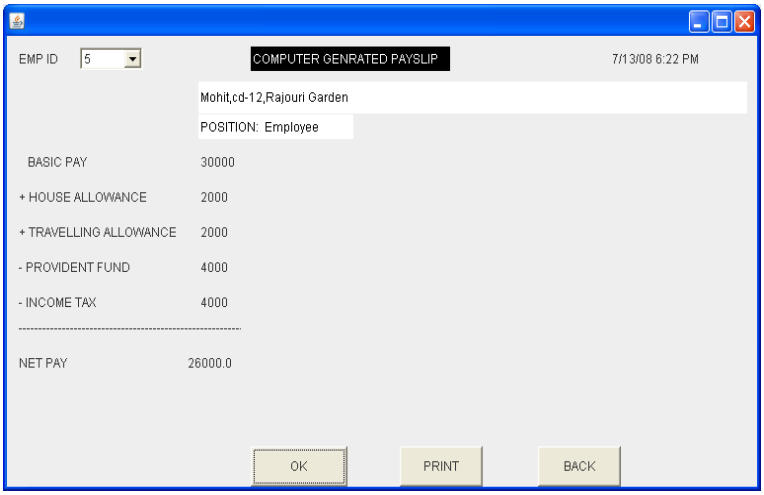


Press save to confirm Deletion

Select another EmpId and press OK to select

Press Back to go back

EmpSlip.Java



Press OK to generate slip for selected ID

Print Function is Currently Disabled and is left for Future scope
Press Back to Go Back.

SYSTEM TESTING

During systems testing, the system is used experimentally to ensure that the software does not fail. In other words, we can say that it will run according to its specifications and in the way users expect. Special test data are input for processing, and the results examined. A limited number of users may be allowed to use the system so that analyst can see whether they try to use it in unforeseen ways. It is desirable to discover any surprises before the organization implements the system and depends on it.

Software modules are tested for their functionality as per the requirements identified during the requirements analysis phase. To test the functionality of the file transfer and chat application, a Quality Assurance (QA) team is formed. The requirements identified during the requirements analysis phase are submitted to the QA team. In this phase the QA team tests the application for these requirements. The development team submits a test case report to the QA team so that the application can be tested in the various possible scenarios.

The software development project, errors can be injected at any stage during development i.e. if there is an error injected in the design phase then it can be detected in the coding phase because there is the product to be executed ultimately on the machine, so we employ a testing process.

During the testing the program to be tested is executed with certain test cases and output of these test cases is evaluated to check the correctness of the program. It is the testing that performs first step in determining the errors in the program.

TEST CASES AND TEST CRITERIA

During Test Cases that are good at revealing the presence of faults is central to successful testing. The reason for this is that if there is a fault in the program, the program can still provide the expected behavior on the certain inputs. Only for the set of inputs the faults that exercise the fault in the program will the output of the program devise from the expected behavior. Hence, it is fair to say that testing is as good as its test case.

The number of test cases used to determine errors in the program should be minimum. There are two fundamental goals of a practical testing activity:-

- maximize the number of errors detected and.
- minimize the number of test cases.

As these two goals are contradictory so the problem of selecting test cases is a complex one. While selecting the test cases the primary objective is to ensure that if there is an error or fault in the program, it is exercised by one of its test cases. An ideal test case is one which succeeds (meaning that there are no errors, revealed in its execution) only it there are no errors in the program one possible set of ideal test cases is one which includes all the possible inputs to the program. This is often called "exhaustive testing", however it is impractical and infeasible as even a small program can have an infinite input domain.

So to avoid this problem we use "test criteria" in selection of the test cases. There are two aspects of the test case selection:-

- specifying a criteria for evaluating the test cases
- generating the set of cases that satisfy a given criteria.

The fully automated process of generating test criteria has not been yet found rather guidelines are only the automated tool available to us previously. The two fundamental properties for a testing criterion are:-

- **Reliability**: a criterion is reliable if all the sets that satisfy the criteria detect the same error. .
- **Validity**: a criterion is valid if for any error in the program there is some set satisfying the criteria that will reveal the error.

The fundamental theorem of testing is that if a testing criterion is valid and reliable, if a set satisfying criteria succeeds then the program contains no errors.

APPROACHES TO TESTING

There are two fundamental approaches to testing they are:

Functional Testing/Black Box Testing:

In functional testing the structure of the program is not considered. Test cases are solely determined on the basis of requirement or specification of the program or module. Internals of the modules and the programs are not considered for selection of test cases. Due to this nature it is also called "black box testing".
The most obvious functional testing procedure is "exhaustive testing", which is impractical.
The other criteria for generating the test case are to generate them "randomly" this strategy has a little chance of resulting in a test case that is close to optimal.
There is no appropriate criterion for developing the test case so we have certain heuristic approaches-:

a) **Equivalence Class Partitioning**: in which the domain of all inputs is divided into a set of equivalence classes, so that if any test in that class succeeds, then every test in that class will succeed i.e. the success of one set element implies the success of the other.

b) **Boundary Value Analysis**: It has been observed that programs work correctly for a set of values in an equivalence class fail on certain values. These values generally lie on the boundary of equivalence class. So in the boundary value analysis we chose an input for a test case from an equivalence class, such that input lies on the edge of equivalence class. Boundary value test cases are also called "extreme cases".

c) **Cause-Effect Graphing**: The problem with the prior approaches is that they consider each input separately i.e. both concentrate on classes and conditions of one input. The combination of inputs is not considered which is used in many cases. One way to exercise the combination of various input conditions is to consider all the valid combinations of the equivalence class of the input conditions. The technique starts with identifying the causes and the effect of system under testing.

STRUCTURAL TESTING:

In the structural approach the test cases are generated an the basis of the actual code of the program or the module to be tested, This structural approach is sometimes called "glass box testing", This testing is concerned with the implementation of the program. The content is not to exercise the various input conditions rather different programming structures and data structures used in the program.

There are three different approaches to structural testing they are-:

a) **Control flow based criteria**:

Most common structure based criteria use control flow based testing in which the control flow graph of the program is considered and coverage of various aspects of graph are specified as criteria. The various control flow based strategies are

- statement coverage,
- branch coverage and
- all path coverage

b) **Data Flow based testing:**

The basic idea behind the data flow based testing is to make sure that during testing, the definitions of variables and their subsequent use is tested. To implement the data flow based testing the data flow graph is first made from the control flow graph.

c) **Mutation Testing:**

In the above two testing techniques the focus is on which path to be executed, but mutation testing is not a path-based approach. The mutation testing requires that the set of test cases must be such that they can distinguish between the original programs and is mutants. In software world there is no fault model as in hardware so most of the techniques do guess work regarding where the fault must lie and then select the test cases to reveal those faults. In Mutation testing faults of some pre-decided types are introduced in the program being tested. Then those faults are found in the mutants.

Implementation, Evaluation And Testing

Implementation is the process of having systems personnel check out and put new equipment into use, train users, install the new application and construct any files of data needed to use it. This phase is less creative than system design. Depending on the size of the organization that will be involved in using the application and the risk involved in its use, systems developers may choose to test the operation in only one area of the firm with only one or two persons. Sometimes, they will run both old and new system in parallel way to compare the results. In still other situations, system developers stop using the old system one day and start using the new one the next.

Evaluation of the system is performed to identify its strengths and weaknesses. The actual evaluation can occur along any of the following dimensions:

1. **Operational Evaluation**: Assessment of the manner in which the system functions, including case of use, response time, overall reliability and level of utilization.
2. **Organizational Impact**: Identification and measurement of benefits to the organization in such areas as financial concerns, operational efficiency and competitive impact.
3. **User Manager Assessment:** Evaluation of the attitudes of senior and user manager within the organization, as well as end-users.
4. **Development Performance**: Evaluation of the development process in accordance with such yardsticks as overall development time and effort, conformance to budgets and standards and other project management criteria.

Maintenance is necessary to eliminate errors in the working system during its working life and to tune the system to any variations in its working environment often small system deficiencies are found as a system is brought into operations and changes are made to remove them. System planners must always plan for resource availability to carry out these maintenance functions. The importance of maintenance is to continue to bring the new system to standards.

CONCLUSION

Since this project has been designed exclusively as a project, certain complexities that do faced by any real life manual problem like total no. of employee, address redundancy etc. are considered in this project. But enhancement to the project can easily be made without changing the current design and programming structure.

BIBLIOGRAPHY

- ***www.google.com***
- ***www.wikipedia.com***
- ***www.studymafia.org***
- ***www.pptplanet.com***