TABLE OF CONTENTS

ARDENT COMPUTECH PVT. LTD.

Ardent Computech Private Limited is an ISO 9001-2008 certified Software Development Company in India. It has been operating independently since 2003. It was recently merged with ARDENT TECHNOLOGIES.

Ardent Technologies

ARDENT TECHNOLOGIES is a Company successfully providing its services currently in UK, USA, Canada and India. The core line of activity at ARDENT TECHNOLOGIES is to develop customized application software covering the entire responsibility of performing the initial system study, design, development, implementation and training. It also deals with consultancy services and Electronic Security systems. Its primary clientele includes educational institutes, entertainment industries, resorts, theme parks, service industry, telecom operators, media and other business houses working in various capacities.

Ardent Collaborations

ARDENT COLLABORATIONS, the Research Training and Development Department of ARDENT COMPUTECH PVT LTD is a professional training Company offering IT enabled services & industrial trainings for B-Tech, MCA, BCA, MSc and MBA fresher's and experienced developers/programmers in various platforms. Summer Training / Winter Training / Industrial training will be provided for the students of B.TECH, M.TECH, MBA and MCA only. Deserving candidates may be awarded stipends, scholarships and other benefits, depending on their performance and recommendations of the mentors.

Associations

Ardent is an ISO 9001:2008 company.It is affiliated to National Council of Vocational Training (NCVT), Directorate General of Employment & Training (DGET), Ministry of Labor & Employment, and Government of India.

1. Abstract

This project is aimed at developing an Expense Manager that is of importance to an individual person. The system is a standalone / desktop application that can be accessed by any user. User's logging in should be able to note their daily expenses. User's logging in may also access by date, category and date & category report of their expenses. EM should have super users for approval of users.

2. Introduction and Objectives of the Project

A person should be able to

- Sign in to the EM
- Recover your password
- Login in EM
- Note their expense
- Edit previously noted expense
- View date type report
- View date and category type report
- View category type report

3. Project Category

- Desktop Application (using DBMS (Data Base Management System) and OOPS (Object Oriented Programs)

4. Tools/Platform, Hardware and Software Requirement specifications.

1. Netbeans IDE 8.0.2
2. Oracle 11g xe
3. JDK 1.7K
4. Microsoft Windows 7/8

Hardware Requirement Specification

Client Machine	
HDD	500 MB
Processor	Pentium 4 or newer processor that supports SSE2
Memory	1024 MB

Software Requirement Specification

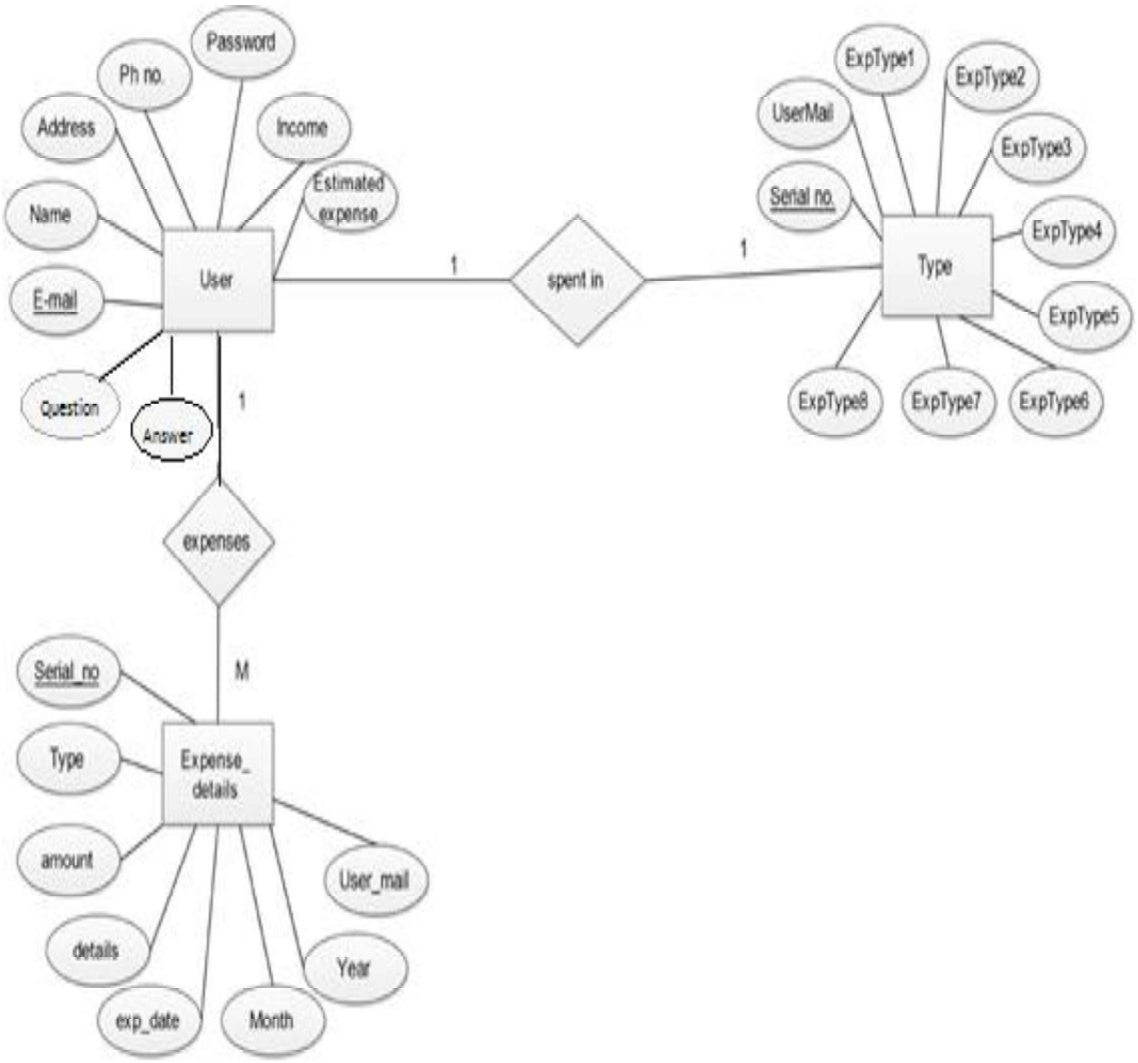
Client Machine	
SOFTWARE	NetBeans IDE 8.0.2 , Oracle 11g xe JDK 1.7.0
Language	Java SQL (**Structured Query Language**)

5. Goals of Implementation

The implementation aims at tracking and reporting expenses of individual.

6. Data Model

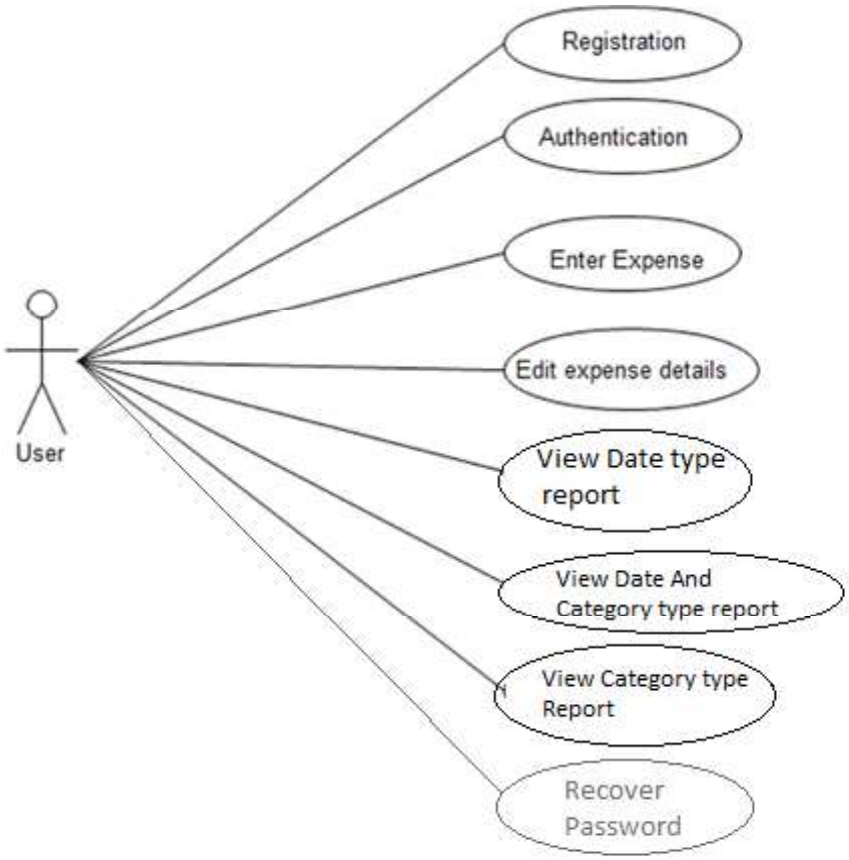
ER Diagram



7. Functional Requirements

Functional Requirements are those that refer to the functionality of the system, i.e., what services it will provide to the user. Non-functional (supplementary) requirements pertain to other information needed to produce the correct system and are detailed separately.

8. Use Case Diagram



9. Use Case Descriptions

Use Case Name:	**Enter Expense**
Priority	Essential
Trigger	Menu selection and form filling
Precondition	User must login first.
Basic Path	1. User enters an amount. 2. User chooses a date. 3. Type details of the expense. 4. Chooses a Type. 5. User clicks on the submit button. 6. Data saves to the database and give a confirmation dialog box
Alternate Path	NA
Post Condition	The user remains in that frame.
Exception Path	Message appeared “something went wrong in the dialog box”

Use Case Name:	**Edit expense**
Priority	Essential
Trigger	Menu selection
Precondition	User must login first.
Basic Path	1. Selects a date and a type. 2. If there exists a record that matches the chosen date and type EM shows a record with that date and type else return error message.
	3. User modifies the details. 4. User hits the Update button. 5. The EM program returns error or success message.
Alternate Path	NA
Post Condition	The user is in the edit expense frame
Exception Path	Message appeared “something went wrong in the dialog box”

Use Case Name:	**View date type report**
Priority	Essential
Trigger	Menu selection
Precondition	User must login first.
Basic Path	1. User selects a date. 2. EM returns the report if there exist entries on that date else shows error message.
Alternate Path	NA
Post Condition	The user is on View expense frame
Exception Path	Message appeared “something went wrong in the dialog box”

Use Case Name:	**View category type report**
Priority	Essential
Trigger	Menu selection
Precondition	User must login first.
Basic Path	1. User selects a category. 2. EM returns the report if there exist entries on that month else shows error message.
Alternate Path	NA
Post Condition	The user is on View expense frame
Exception Path	Message appeared “something went wrong in the dialog box”

Use Case Name:	**View category & date type report**
Priority	Essential
Trigger	Menu selection
Precondition	User must login first.
Basic Path	1. User selects a month. 2. EM returns the report if there exist entries on that date else shows error message.
Alternate Path	NA
Post Condition	The user is on View expense frame
Exception Path	Message appeared “something went wrong in the dialog box”

10. Non Functional Requirements

In addition to the obvious features and functions that you will provide in your system, there are other requirements that don't actually DO anything, but are important characteristics nevertheless. These are called "non-functional requirements" or sometimes "Quality Attributes." For example, attributes such as performance, security, usability, compatibility aren't a "feature" of the system, but are a required characteristic. You can't write a specific line of code to implement them; rather they are "emergent" properties that arise from the entire solution. The specification needs to describe any such attributes the customer requires. You must decide the kind of requirements that apply to your project and include those that are appropriate.

Each requirement is simply stated in English. Each requirement must be objective and quantifiable; there must be some measurable way to assess whether the requirement has been met.

Often deciding on quality attributes requires making tradeoffs, e.g., between performance and maintainability. In the APPENDIX you must include an engineering analysis of any significant decisions regarding tradeoffs between competing attributes.

Here are some examples of non-functional requirements:

Performance requirements

Requirements about resources required, response time, transaction rates, throughput, benchmark specifications or anything else having to do with performance.

Operating constraints

List any run-time constraints. This could include system resources, people, needed software, The application must run without any manual intervention.

Platform constraints

Discuss the target platform. Be as specific or general as the user requires. If the user doesn't care, there are still platform constraints.

Since the application will be developed in JEE it is platform independent.

Accuracy and Precision

Requirements about the accuracy and precision of the data. (Do you know the difference?) Beware of 100% requirements; they often cost too much.

Modifiability

Requirements about the effort required to make changes in the software. Often, the measurement is personnel effort (person- months).

Minimal

Portability

The effort required to move the software to a different target platform. The measurement is most commonly person-months or % of modules that need changing.

Minimal

Reliability

Requirements about how often the software fails. The measurement is often expressed in MTBF (mean time between failures). The definition of a failure must be clear. Also, don't confuse reliability with availability which is quite a different kind of requirement. Be sure to specify the consequences of software failure, how to protect from failure, a strategy for error detection, and a strategy for correction.

Security

One or more requirements about protection of your system and its data. The measurement can be expressed in a variety of ways (effort, skill level, and time,) to break into the system.

Do not discuss solutions (e.g. passwords) in a requirements document.

Only secured users can access the application.

No one can go to any independent page without logging in.

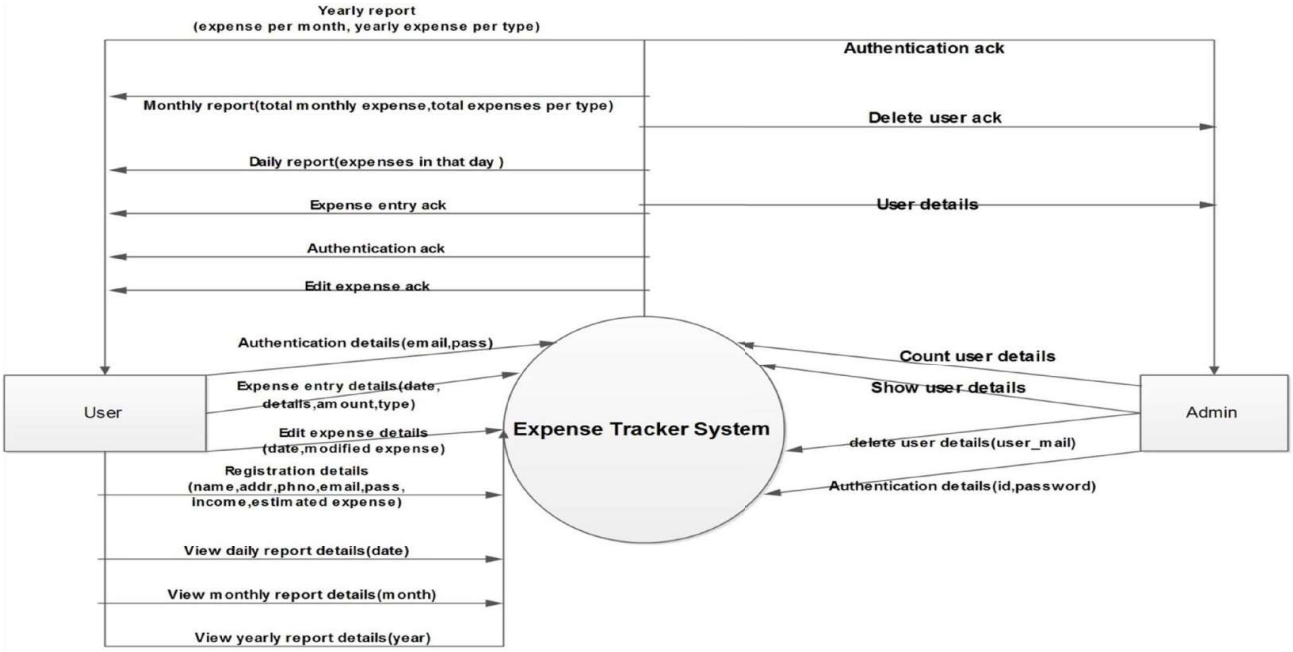
Usability

Requirements about how difficult it will be to learn and operate the system. The requirements are often expressed in learning time or similar metrics.

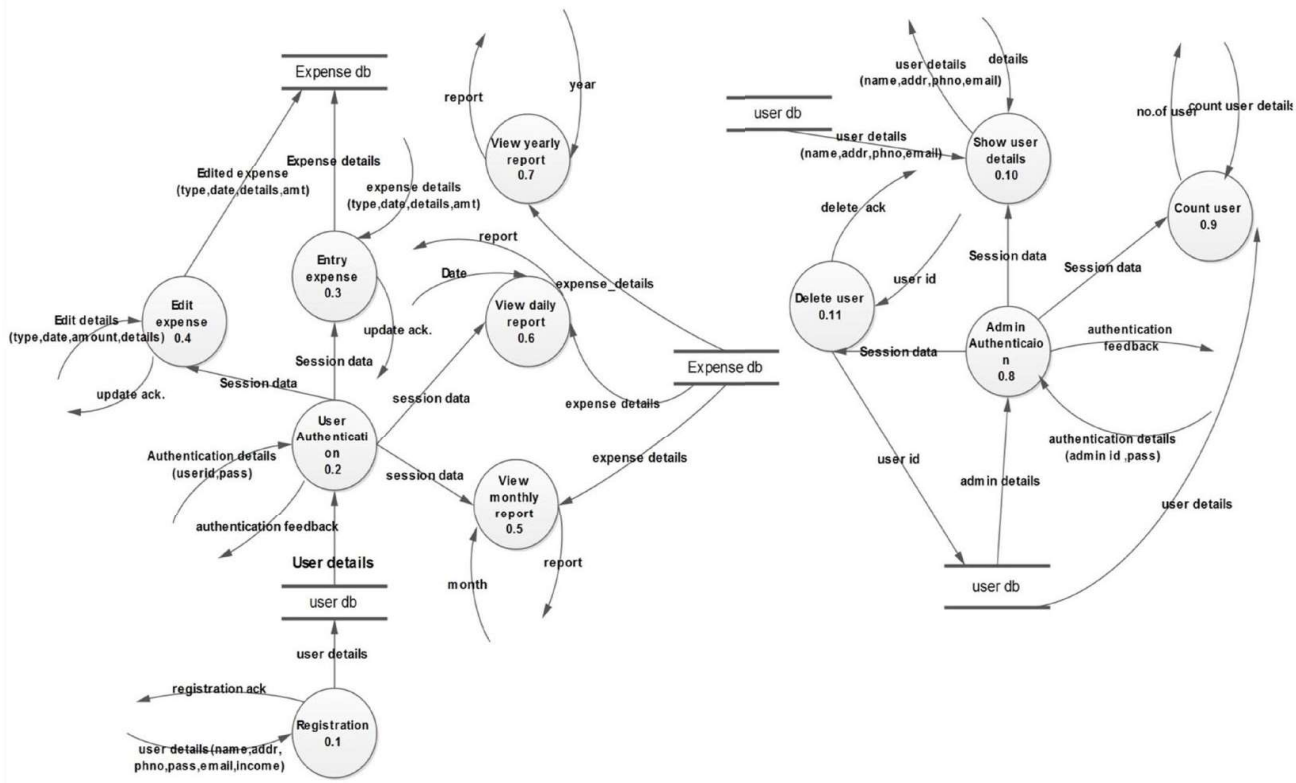
11. Software Engineering Paradigm Applied

Data Flow Diagrams

Level 0



Level 1



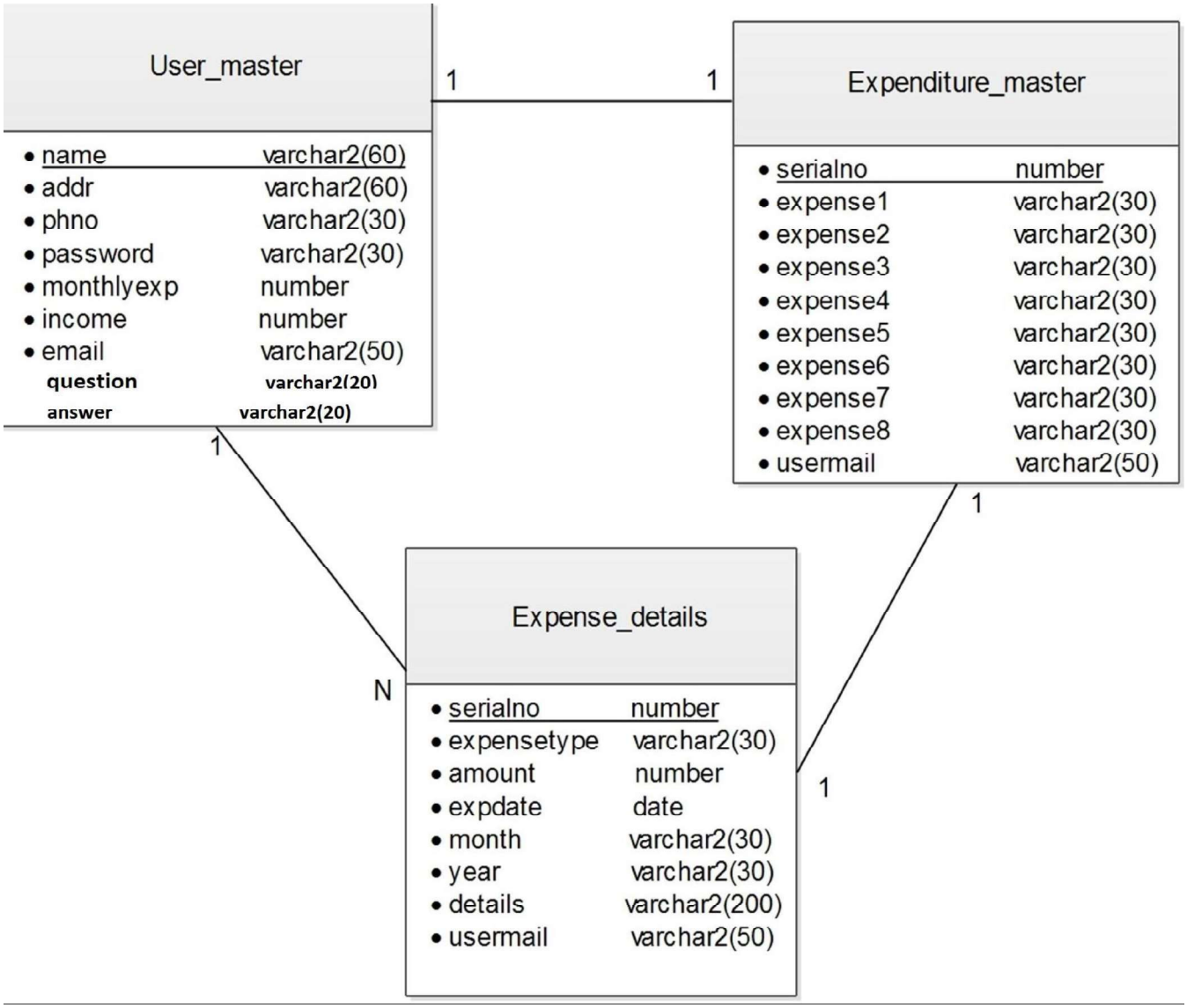
12. Project Planning

Project planning is concerned with identifying the following for every project:

- Activities.
- Milestones.
- Deliverables.

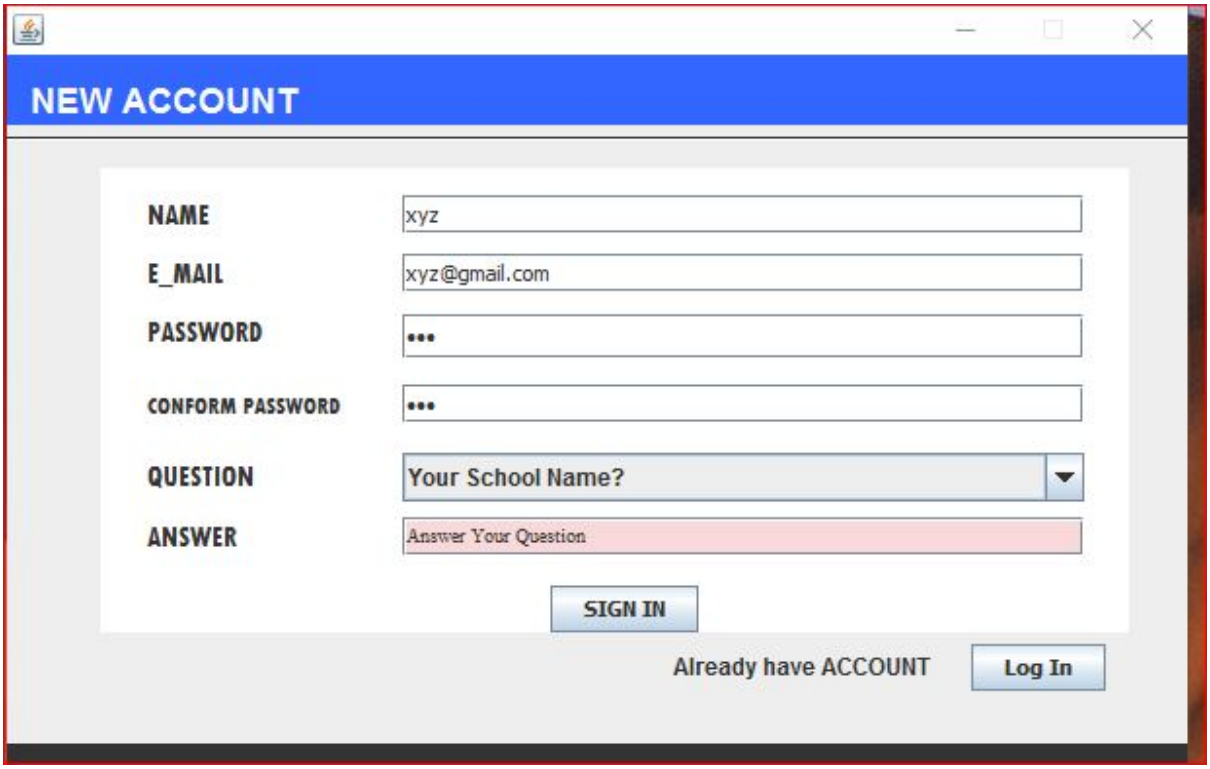
A plan must be drawn up to guide the development towards the project goal.
A plan is drawn up at the start of a project.
This plan should be used as the driver for the project.
The initial plan is not static, and must be modified as the project progresses.

13. DEVELOPMENT OF EM.

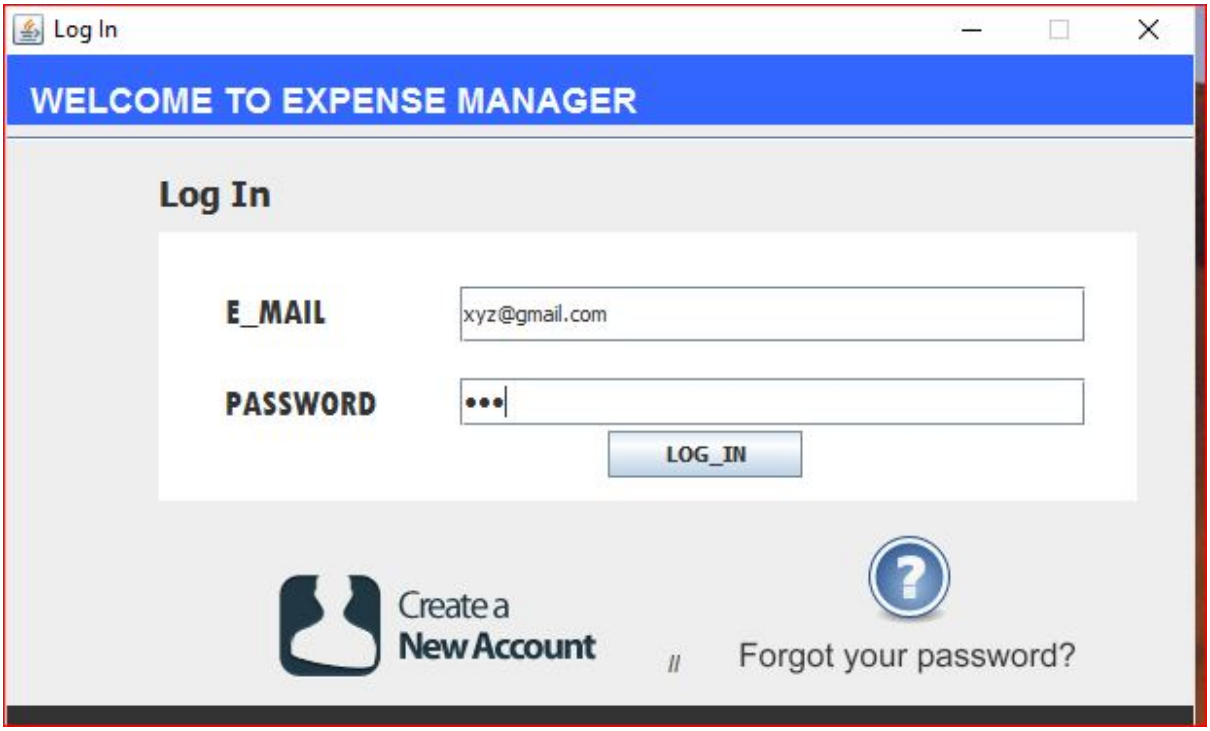


14. User Interface Design

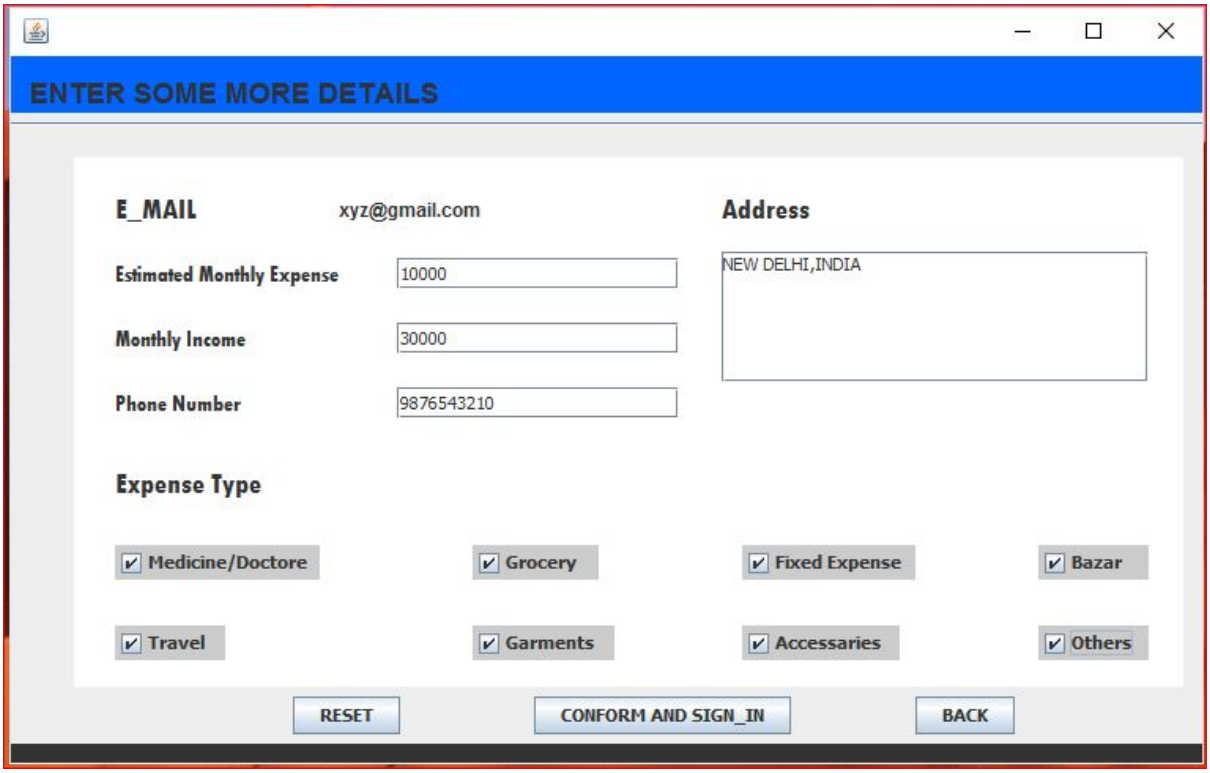
NEW ACCOUNT FRAME



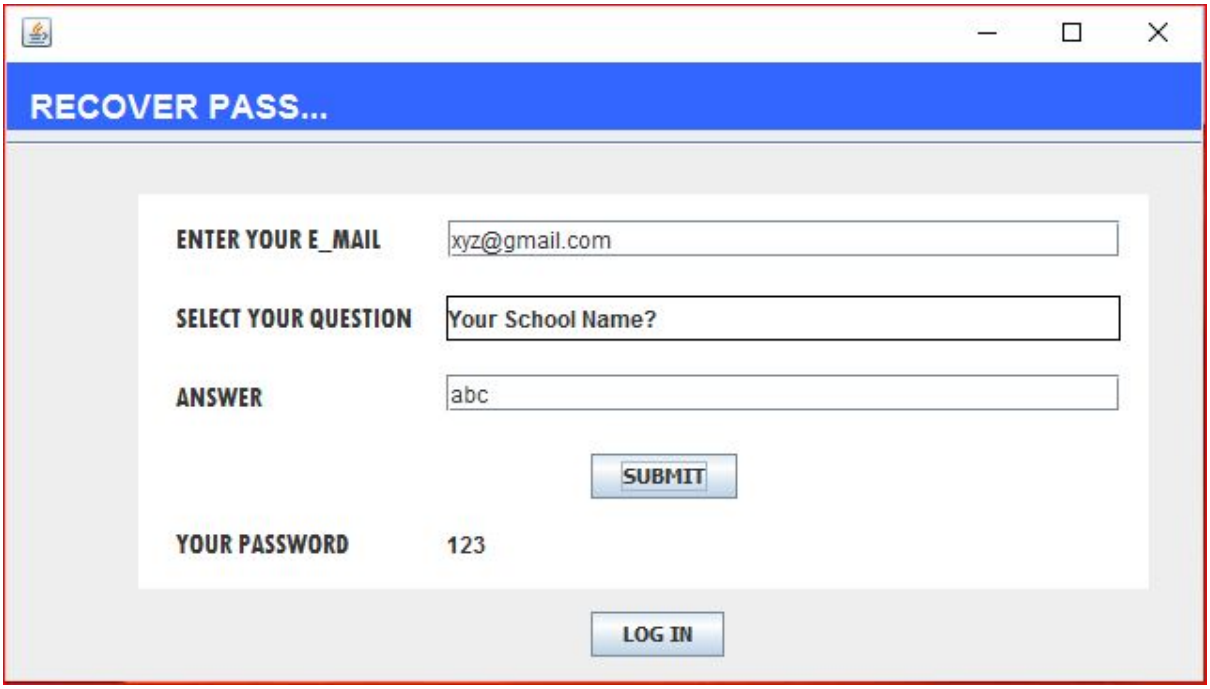
LOGIN FRAME



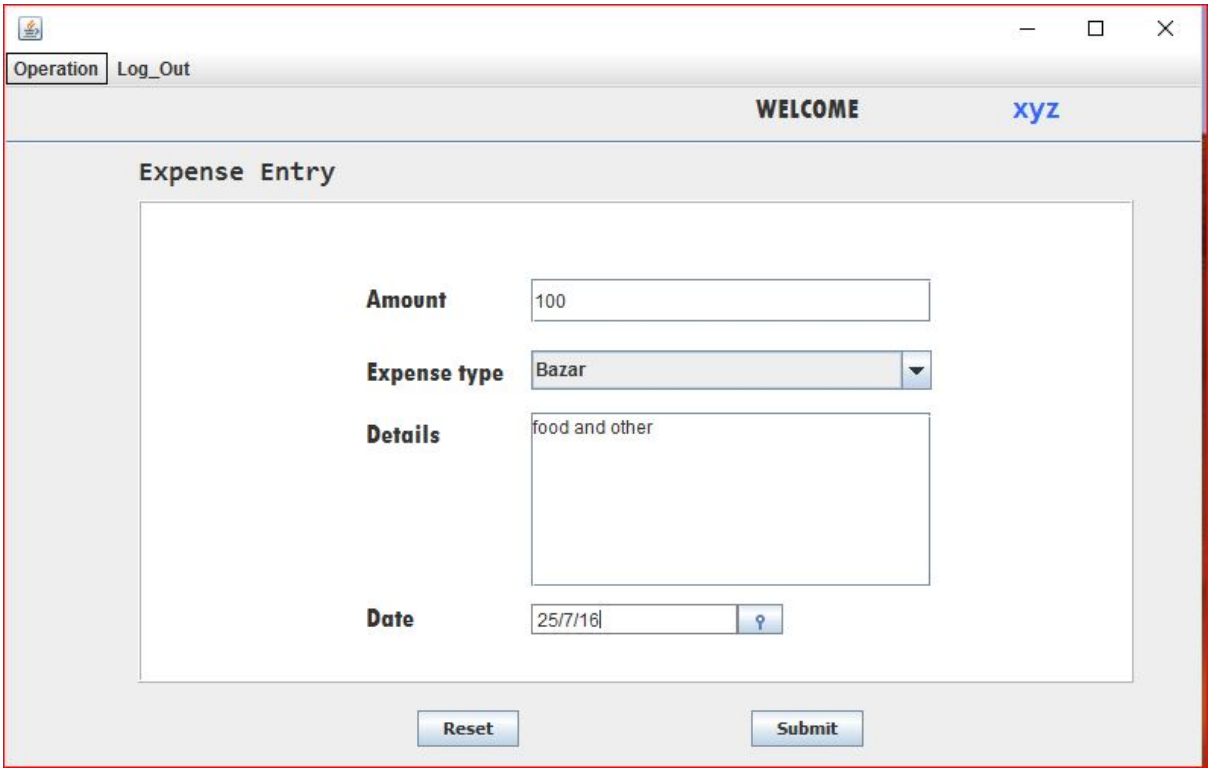
DETAILS FRAME



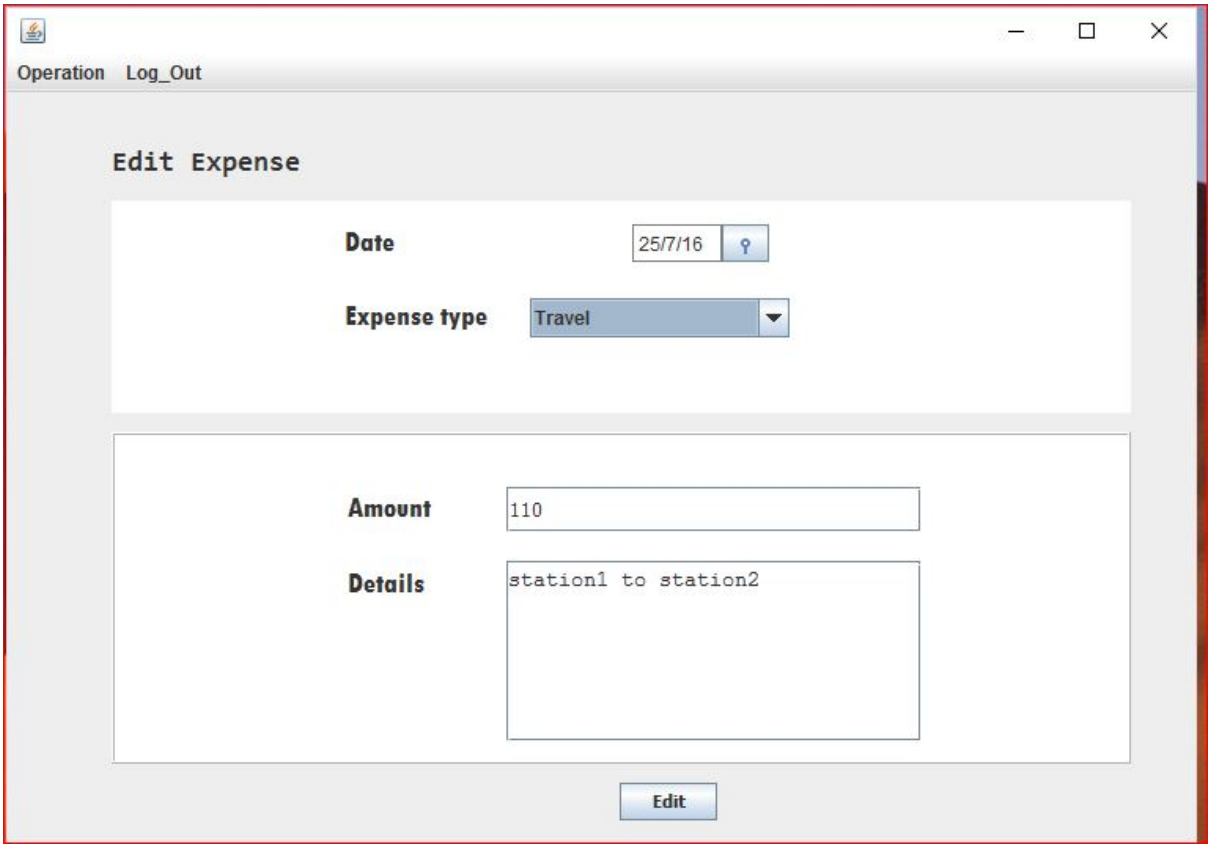
RECOVER PASSWORD FRAME



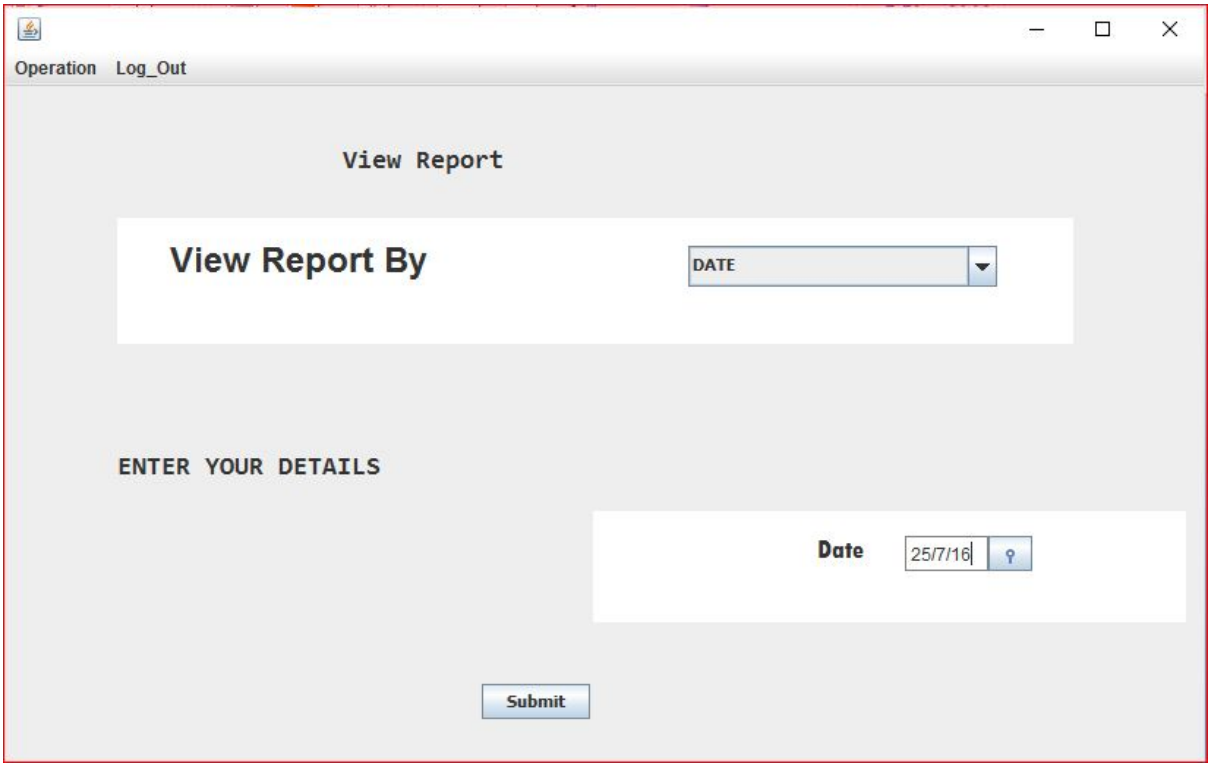
EXPENSE ENTRY



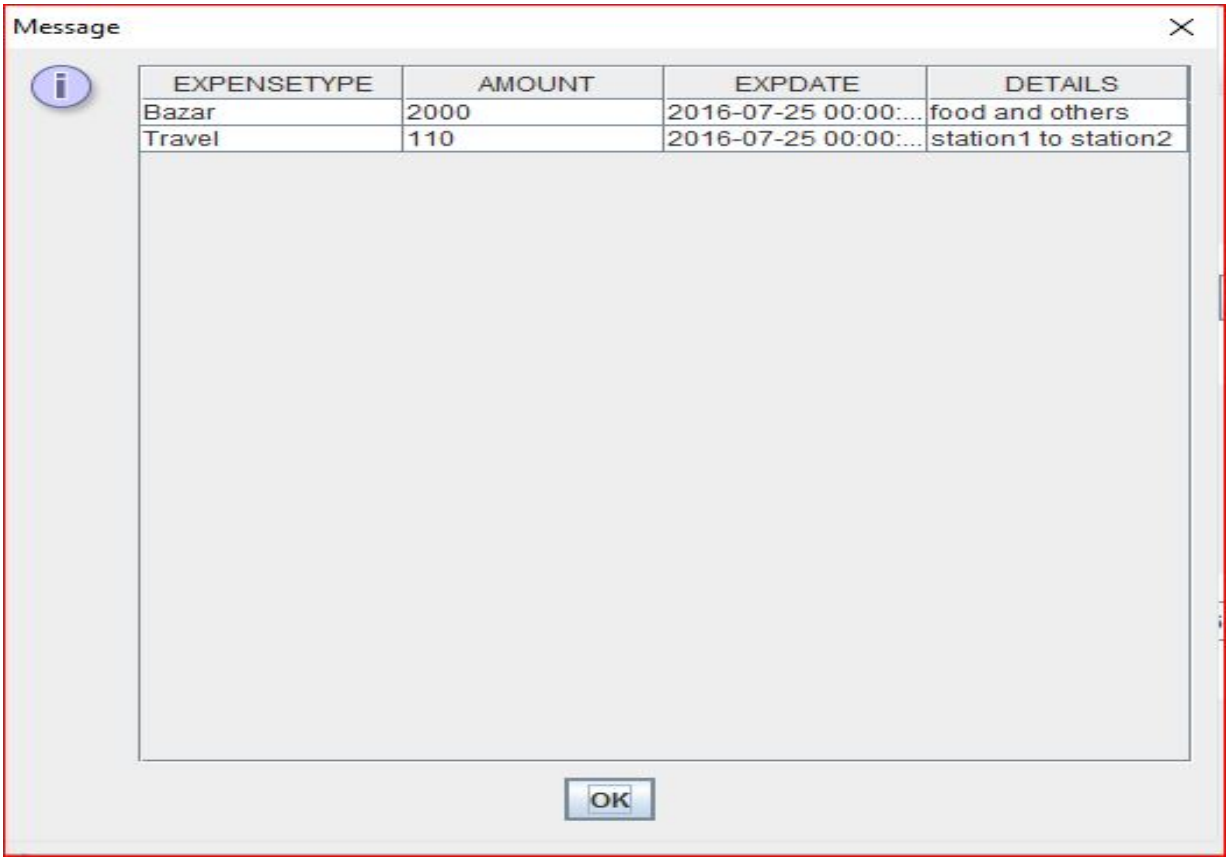
EDIT EXPENSE FRAME



VIEW EXPENSE FRAME



DETAILS OF EXPANSE



15. Code

MAIN CLASS :

```
package ClassPath;

public class Main {

   public static void main(String...arg){

      new frames.log_in().setVisible(true);

   }

}
```

ExpenseDetails :

```
package beans;

import dao.Dao;

import dao.DaoImpl;


public class ExpenseDetails {

private  int serialno;

private  String expense1;

private  String expense2;

private  String expense3;

private  String expense4;

private  String expense5;

private  String expense6;

private  String expense7;

private  String expense8;

private  String email;

public int getSerialno() {

            return serialno;
```

```
}
public void setSerialno(int serialno) {
        this.serialno = serialno;
}
public String getExpense1() {
        return expense1;
}
public void setExpense1(String expense1) {
        this.expense1 = expense1;
}
public String getExpense2() {
        return expense2;
}
public void setExpense2(String expense2) {
        this.expense2 = expense2;
}
public String getExpense3() {
        return expense3;
}
public void setExpense3(String expense3) {
        this.expense3 = expense3;
}
public String getExpense4() {
        return expense4;
}
public void setExpense4(String expense4) {
        this.expense4 = expense4;
```

```
}
public String getExpense5() {
        return expense5;
}
public void setExpense5(String expense5) {
        this.expense5 = expense5;
}
public String getExpense6() {
        return expense6;
}
public void setExpense6(String expense6) {
        this.expense6 = expense6;
}
public String getExpense7() {
        return expense7;
}
public void setExpense7(String expense7) {
        this.expense7 = expense7;
}
public String getExpense8() {
        return expense8;
}
public void setExpense8(String expense8) {
        this.expense8 = expense8;
}
public String getEmail() {
        return email;
```

```
}
public void setEmail(String email) {
        this.email = email;
}



}
```

ExpenseValues :

```
package beans;
import dao.Dao;
import dao.DaoImpl;
import java.sql.Date;

public class ExpenseValues {
        private String expensetype;
        private int amount;
        private java.sql.Date  expdate;
        private int month;
        private int year;
        private String details;
        private String usermail;

   @Override
   public String toString() {
      return "ExpenseValues{" + "expensetype=" + expensetype + ", amount=" + amount + ",
expdate=" + expdate + ", month=" + month + ", year=" + year + ", details=" + details + ", usermail=" +
usermail + '}';
   }
```

```
public String getExpensetype() {
        return expensetype;
}
public void setExpensetype(String expensetype) {
        this.expensetype = expensetype;
}
public int getAmount() {
        return amount;
}
public void setAmount(int amount) {
        this.amount = amount;
}

public String getDetails() {
        return details;
}
public void setDetails(String details) {
        this.details = details;
}
public String getUsermail() {
        return usermail;
}
public void setUsermail(String usermail) {
        this.usermail = usermail;
}
```

```
        /*public int insertExpenseValues()
        {
                Dao d=new DaoImpl();
                int serialno=d.getSerial("expense_details");
                int i=d.insertExpenseValues(serialno,expensetype, amount, expdate, details,
usermail);
                return i;
        }*/

    public Date getExpdate() {
        return expdate;
    }

    public void setExpdate(Date expdate) {
        this.expdate = expdate;
    }

    public int getMonth() {
        return month;
    }

    public void setMonth(int month) {
        this.month = month;
    }

    public int getYear() {
        return year;
```

```
    }


    public void setYear(int year) {
        this.year = year;
    }
}
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

UserDetails :

```
package beans;
import dao.Dao;
import dao.DaoImpl;
public class UserDetails {
private String name;
private String addr;
private String phno;
private String password;
private String email;
private int monthlyexp;
private int income;
private String ques;
private String ans;


    @Override
```

```
    public String toString() {

        return "UserDetails{" + "name=" + name + ", addr=" + addr + ", phno=" + phno + ", password=" +
password + ", email=" + email + ", monthlyexp=" + monthlyexp + ", income=" + income + ", ques=" +
ques + ", ans=" + ans + '}';

    }



public String getName() {

        return name;

}

public void setName(String name) {

        this.name = name;

}

public String getAddr() {

        return addr;

}

public void setAddr(String addr) {

        this.addr = addr;

}

public String getPhno() {

        return phno;

}

public void setPhno(String phno) {

        this.phno = phno;

}

public String getPassword() {

        return password;

}
```

```
public void setPassword(String password) {
        this.password = password;
}
public String getEmail() {
        return email;
}
public void setEmail(String email) {
        this.email = email;
}
public int getMonthlyexp() {
        return monthlyexp;
}
public void setMonthlyexp(int monthlyexp) {
        this.monthlyexp = monthlyexp;
}
public int getIncome() {
        return income;
}
public void setIncome(int income) {
        this.income = income;
}
public String getQues(){
   return ques;
}
public void setQues(String ques){
   this.ques=ques;
}
```

```
public String getAns(){
   return ans;
}
public void setAns(String ans){
   this.ans=ans;
}

public boolean login(String email,String password)
{
        Dao d=new DaoImpl();
        boolean b=d.loginUser(email, password);
        return b;
}

public int updateUser()
{
        Dao d=new DaoImpl();
        int i=d.updateUser(name, addr, phno, password, monthlyexp, income, email);
        return i;
}
public int deleteUser()
{
        Dao d=new DaoImpl();
        int i=d.deleteUser(email);
        return i;
}
}
```

EmailValidator :

```
package bo;

import java.util.regex.Matcher;

  import java.util.regex.Pattern;


public class EmailValidator {


private Pattern pattern;

private Matcher matcher;


private static final String EMAIL_PATTERN =

   "^[_A-Za-z0-9-\\+]+(\\.[_A-Za-z0-9-]+)*@"

   + "[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$";


public EmailValidator() {

   pattern = Pattern.compile(EMAIL_PATTERN);

}

public boolean validate(final String hex) {


   matcher = pattern.matcher(hex);

   return matcher.matches();


}

}


/*

 * To change this template, choose Tools | Templates
```

```
 * and open the template in the editor.
 */
```

InsertBo :

```
package bo;
import beans.ExpenseValues;
import dao.Dao;
import dao.DaoImpl;
import javax.swing.JOptionPane;

public class InsertBo {

   ExpenseValues ex;

   public InsertBo(ExpenseValues ex) {
      this.ex = ex;
   }



   public boolean ExpenseNote(){
      boolean state=false;
      Dao d=new DaoImpl();

      if(d.insertExpenseValues(ex)==1)
      {
         state=true;
         JOptionPane.showMessageDialog(null,"Entry Successfull");
      }
```

```
        else
        JOptionPane.showMessageDialog(null,"Same type of expense can not be noted ,please edit");


      return state;
   }
}


/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
```

LoginBo :

```
package bo;
import dao.Dao;
import dao.DaoImpl;
public class LoginBo {

   public boolean login(String uname,String pass){


      boolean state=false;
      Dao d=new DaoImpl();
      if(d.loginUser(uname, pass))
         state=true;
      return state;


   }
}
```

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
```

PhonenoValidator :

```
package bo;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
public class PhonenoValidator {
   public  boolean validate(String phno) {

    // String sPhoneNumber = "605-8889999";
     //String sPhoneNumber = "605-88899991";
     //String sPhoneNumber = "605-888999A";

     Pattern pattern = Pattern.compile("^((\\+|00)(\\d{1,3})[\\s-]?)?(\\d{10})$");
     Matcher matcher = pattern.matcher(phno);

     if (matcher.matches()) {
        return true;
     } else {
        return false;
     }
}
}
/*
```

```
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
```

RegistrationBo :

```
package bo;
import beans.ExpenseDetails;
import beans.UserDetails;
import dao.*;
public class RegistrationBo {
   UserDetails u;
   ExpenseDetails exd;
  static Dao d;
   public RegistrationBo(UserDetails u,ExpenseDetails exd) {
       this.u = u;
       this.exd=exd;
   }
   static
   {
   d=new DaoImpl();
   }


  public int Register(){
      int state=0;
      if(d.isExists(u)){

          if(d.registerUser(u)==1){
```

```
            d.isertExpenseDetails(exd);
                state=1;
        }

    }
    return state;
  }
}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
```

ReportBo :

```
package bo;
import dao.Dao;
import dao.DaoImpl;
import java.sql.ResultSet;
public class ReportBo {

    public ResultSet getReportBydate(String uid,java.sql.Date date){

        ResultSet res=null;
       try{

      Dao r=new DaoImpl();
```

```
    res=r.getExpenseValuesByDate(uid, date);
  return res;
    }
    catch(Exception e){
        e.printStackTrace();
    }
    return res;
}

  public ResultSet getReportBycategory(String uid,String type){

     ResultSet res=null;
    try{

   Dao r=new DaoImpl();

    res=r.getExpenseValuesByCategory(uid,type);
  return res;
    }
    catch(Exception e){
        e.printStackTrace();
    }
    return res;
}

   public ResultSet getReportByDateAndCategory(String uid,String type,java.sql.Date Date){
```

```
        ResultSet res=null;
       try{

      Dao r=new DaoImpl();

       res=r.getExpenseValuesByDateAndCategory(uid, type,Date);
    return res;
       }
       catch(Exception e){
           e.printStackTrace();
       }
       return res;
   }
 }

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
```

Controller :

```
package controller;
import beans.ExpenseDetails;
import beans.ExpenseValues;
import beans.UserDetails;
import bo.InsertBo;
import bo.LoginBo;
import bo.RegistrationBo;
```

```
import javax.swing.JOptionPane;
public class Controller {

    public void registerUser(UserDetails user,ExpenseDetails ed){
        RegistrationBo rb=new RegistrationBo(user,ed);
        if(rb.Register()==1)
            new frames.load( null, true).setVisible(true);

        else
            JOptionPane.showMessageDialog(null,"user ALready Exists");



    }
    public boolean  noteExpense(ExpenseValues ev){
        boolean state=false;
            state=new InsertBo(ev).ExpenseNote();





        return state;
    }

    public boolean userlogin(String uname,String pass){
        System.out.print("in controller");
        LoginBo l=new LoginBo();
        if(l.login(uname, pass))
```

```
        return true;
      else
        return false;
  }

}
```

INTERFACE dao :

```
package dao;
import beans.ExpenseDetails;
import java.util.ArrayList;
import beans.ExpenseValues;
import beans.UserDetails;
import java.sql.ResultSet;

public interface Dao {
int registerUser(UserDetails u);
boolean loginUser(String mail,String password);
public int updateUser(String name, String addr, String phno, String password,int monthlyexp, int
income, String email);

 public boolean isExists(UserDetails u);
public int deleteUser(String email);
//public ResultSet getList(String uname);
public int isertExpenseDetails(ExpenseDetails ed);
public int insertExpenseValues(ExpenseValues ex);
public int editExpenseValues(String expensetype,int amount,String expdate,String details,String
usermail);
public int getSerial(String table);
```

```
public  ResultSet getExpenseValuesByDate(String uid,java.sql.Date date);

public ResultSet getExpenseValuesByCategory(String uid,String type);

public ResultSet getExpenseValuesByDateAndCategory(String uid,String type,java.sql.Date dSate);

public ArrayList<String> getExpenseType(String email);

public boolean isExpensePresent(ExpenseValues ev);

public boolean updateDetails(ExpenseValues ev);

public ResultSet getDetailsByDateAndType(java.sql.Date date,String type);


//    public ResultSet getforgotpwd(String email);

}
```

DaoImpl :

```
package dao;

import beans.ExpenseDetails;

import java.sql.*;

import java.util.ArrayList;

import dbcon.*;

import beans.ExpenseValues;

import beans.UserDetails;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.JOptionPane;

public class DaoImpl implements Dao{

            Connection con;

        ResultSet res;

        boolean status;

        public static ArrayList<String> al;
```

```
    {
  con=DbCon2.getConnected();
  al=new ArrayList<String>();
}
          public  String getuname(String mail){
         try{

        PreparedStatement ps=DbCon2.getConnected().prepareStatement("select name from
user_master where email=?");
        ps.setString(1, mail);
         ResultSet r=ps.executeQuery();
        if(!r.next())
           return r.getString("name");





         }
         catch(Exception e){
            e.printStackTrace();

         }
         return null;

    }
//
    @Override
          public boolean isExists(UserDetails u){
```

```
        boolean status=false;
        try{

         PreparedStatement ps=DbCon2.getConnected().prepareStatement("select * from
user_master where email=?");
         ps.setString(1, u.getEmail());
         ResultSet r=ps.executeQuery();
         if(!r.next())
            status=true;



      }
      catch(Exception e){
         JOptionPane.showMessageDialog(null, e);
      }
      return status;


   }
//
//       @Override
//       public ResultSet getList(String uname){
//      ResultSet list=null;
//           try {
//
//          PreparedStatement ps=DbCon2.getConnected().prepareStatement("select
expense1,expense2,expense3,expense4,expense5,expense6,expense7,expense8 from
expenditure_master where usermail=?");
//          ps.setString(1, uname);
//          list=ps.executeQuery();
```

```
//          System.out.print(list);
//      } catch (SQLException ex) {
//          Logger.getLogger(DaoImpl.class.getName()).log(Level.SEVERE, null, ex);
//      }
//          return list;
//
//      }
        @Override
        public int registerUser(UserDetails u) {
                // TODO Auto-generated method stub
                int i=0;
         System.out.println(u);
                String sql="insert into user_master values(?,?,?,?,?,?,?,?,?)";
                try{
                        con=DbCon2.getConnected();
                        System.out.println("Con "+con);
                        PreparedStatement ps=con.prepareStatement(sql);

            ps.setString(1, u.getName());
                        ps.setString(2, u.getAddr());
                        ps.setString(3, u.getPhno());
                        ps.setString(4, u.getPassword());
                        ps.setInt(5, u.getMonthlyexp());
                        ps.setInt(6, u.getIncome());
            ps.setString(7, u.getEmail());
                        ps.setString(8,u.getQues());
            ps.setString(9,u.getAns());
```

```
                i=ps.executeUpdate();
        }catch(Exception e){e.printStackTrace();}
        return i;
}

@Override
public boolean loginUser(String mail, String password) {
        // TODO Auto-generated method stub
        boolean b=false;
        String sql="select * from user_master where email=? and password=?";
        try{
                Connection con=DbCon2.getConnected();
                PreparedStatement ps=con.prepareStatement(sql);
                System.out.println(mail+"\n.....\n "+password);
                ps.setString(1, mail);
                ps.setString(2, password);
                ResultSet rs=ps.executeQuery();
      System.out.print("in Dao imp"+rs);
                if(rs.next())
                {
                        b=true;
          return b;
                }
        }catch(Exception e){e.printStackTrace();}
        return b;
}
```

```
public int updateUser(String name, String addr, String phno, String password,
                        int monthlyexp, int income, String email)
{
        return 0;
}

public int deleteUser(String email)
{
        return 0;
}

@Override
public int isertExpenseDetails(ExpenseDetails ed) {
        // TODO Auto-generated method stub
        int i=0;
        String sql="insert into expenditure_master values(?,?,?,?,?,?,?,?,?,?)";
        try{
                con=DbCon2.getConnected();
                PreparedStatement ps=con.prepareStatement(sql);
                ps.setInt(1,this.getSerial("expenditure_master"));
                ps.setString(2, ed.getExpense1());
                ps.setString(3, ed.getExpense2());
                ps.setString(4, ed.getExpense3());
                ps.setString(5, ed.getExpense4());
                ps.setString(6, ed.getExpense5());
                ps.setString(7, ed.getExpense6());
```

```
                    ps.setString(8, ed.getExpense7());
                    ps.setString(9, ed.getExpense8());
                    ps.setString(10,ed.getEmail() );
                    i=ps.executeUpdate();
            }catch(Exception e){e.printStackTrace();}
            return i;
        }

    @Override
    public boolean isExpensePresent(ExpenseValues ev){
      try{
        PreparedStatement ps=con.prepareStatement("select * from expense_details where
expdate=? and expensetype=?");
        ps.setDate(1,ev.getExpdate());
        ps.setString(2,ev.getExpensetype());
        status=ps.executeQuery().next();



      }catch(Exception e){
        e.printStackTrace();
      }
          return status;
    }
      @Override
      public int insertExpenseValues(ExpenseValues ev) {
            // TODO Auto-generated method stub
            int i=0;
            String sql="insert into expense_details values(sl.nextval,?,?,?,?,?,?,?)";
```

```
                try{

                        if(!this.isExpensePresent(ev)){
                        PreparedStatement ps=con.prepareStatement(sql);

                        ps.setString(1,ev.getExpensetype());
                        ps.setInt(2,ev.getAmount() );
                        ps.setDate(3,ev.getExpdate());
            ps.setInt(4,ev.getMonth());
            ps.setInt(5,ev.getYear());
                        ps.setString(6, ev.getDetails());
                        ps.setString(7, ev.getUsermail());
                        i=ps.executeUpdate();
            }
                }catch(Exception e){e.printStackTrace();}
                return i;
        }

    @Override
    public boolean updateDetails(ExpenseValues ev){
      try{

          PreparedStatement ps=con.prepareStatement("update expense_details set details=?
,amount=?  where expdate=? and expensetype=?");
          ps.setString(1,ev.getDetails());
            ps.setInt(2,ev.getAmount());
              ps.setDate(3,ev.getExpdate());
```

```
            ps.setString(4,ev.getExpensetype());
            if(ps.executeUpdate()>0)
               status=true;

    }catch(Exception e){
        e.printStackTrace();
    }



    return status;
}

      @Override
      public int editExpenseValues(String expensetype, int amount,
                              String expdate, String details, String usermail) {
                  // TODO Auto-generated method stub

                  return 0;
      }
@Override
      public int getSerial(String table)
      {
                  int serial=10;
                  try{

                              String sql="select max(serialno) from "+table;
                              PreparedStatement ps=con.prepareStatement(sql);
```

```
                ResultSet rs=ps.executeQuery();
                if(rs.next())
                {
                        serial=rs.getInt("max(serialno)")+10;
                }
            }catch(Exception e){e.printStackTrace();}
            return serial;
        }

    @Override
    public ResultSet getDetailsByDateAndType(java.sql.Date date,String type){

            String sql="select amount,details from expense_details where expdate=? and
EXPENSETYPE=? ";

            try{
                Connection con=DbCon2.getConnected();
                PreparedStatement ps=con.prepareStatement(sql);
                ps.setDate(1, date);
             ps.setString(2,type);
                res=ps.executeQuery();
            return res;

            }catch(Exception e){e.printStackTrace();}
            return res;
    }
//
//
```

```
@Override
public ResultSet getExpenseValuesByDate(String uid,java.sql.Date date) {
        // TODO Auto-generated method stub
System.out.println("in DaoImpl "+date+" "+uid);
        ResultSet rs=null;
        String sql="select expensetype,amount,expdate,details from expense_details where
expdate=? and usermail=?";

        try{

                PreparedStatement ps=con.prepareStatement(sql);
                ps.setDate(1, date);
 ps.setString(2, uid);
                rs=ps.executeQuery();
 return rs;

        }catch(Exception e){e.printStackTrace();}
        return rs;
}

@Override
public ResultSet getExpenseValuesByCategory(String uid,String type) {
        // TODO Auto-generated method stub
        ResultSet rs=null;
        String sql="select EXPENSETYPE,amount,expdate,details from expense_details
where EXPENSETYPE=? and usermail=?";
```

```
            try{

                PreparedStatement ps=con.prepareStatement(sql);
                ps.setString(1, type);
        ps.setString(2, uid);
                rs=ps.executeQuery();
                return rs;

            }catch(Exception e){e.printStackTrace();}
            return rs;
        }

        @Override
        public ResultSet getExpenseValuesByDateAndCategory(String uid,String type,java.sql.Date
date) {
            // TODO Auto-generated method stub
            ResultSet rs=null;
            String sql="select expensetype,amount,expdate,details as total from
expense_details where EXPENSETYPE=? and usermail=? and expdate=?";

            try{
                Connection con=DbCon2.getConnected();
                PreparedStatement ps=con.prepareStatement(sql);
                ps.setString(1, type);
        ps.setString(2, uid);
        ps.setDate(3, date);
                rs=ps.executeQuery();
                return rs;
```

```
            }catch(Exception e){e.printStackTrace();}
            return rs;
    }

    @Override
    public ArrayList<String> getExpenseType(String email) {
System.out.println("email-"+email);
            String sql="select * from expenditure_master where usermail=?";

            try{
        al.clear();
                    Connection con=DbCon2.getConnected();
                    PreparedStatement ps=con.prepareStatement(sql);
                    ps.setString(1, email);
                    ResultSet rs=ps.executeQuery();
                    if(rs.next())
                    {

            for(int i=1;i<9;i++){
              String type=rs.getString("expense"+i);
                            if(type!=null)
                al.add(rs.getString("expense"+i));
            }
                    }
            }catch(Exception e){e.printStackTrace();}
            return al;
```

```
	}
}


package dbcon;


import java.sql.Connection;
import java.sql.DriverManager;



public class DbCon2 {
	static Connection con;


	public static Connection getConnected() {
		try{
			Class.forName("oracle.jdbc.OracleDriver");



	con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","expenseManage
r","1234");


}
		catch(Exception e){e.printStackTrace();}
		return con;
	}
}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

formpopTable :

```
package utility;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.util.Vector;
import javax.swing.table.DefaultTableModel;
public class formpopTable {
    public static DefaultTableModel buildTableModel(ResultSet rs)
        throws SQLException {

    ResultSetMetaData metaData = rs.getMetaData();

    // names of columns
    Vector<String> columnNames = new Vector<String>();
    int columnCount = metaData.getColumnCount();
    for (int column = 1; column <= columnCount; column++) {
        columnNames.add(metaData.getColumnName(column));
    }

    // data of the table
    Vector<Vector<Object>> data = new Vector<Vector<Object>>();
```

```
    while (rs.next()) {
        Vector<Object> vector = new Vector<Object>();
        for (int columnIndex = 1; columnIndex <= columnCount; columnIndex++) {
            vector.add(rs.getObject(columnIndex));
        }
        data.add(vector);
    }

    return new DefaultTableModel(data, columnNames);

}
}
```

build.xml :

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- You may freely edit this file. See commented blocks below for -->
<!-- some examples of how to customize the build. -->
<!-- (If you delete it and reopen the project it will be recreated.) -->
<!-- By default, only the Clean and Build commands use this build script. -->
<!-- Commands such as Run, Debug, and Test only use this build script if -->
<!-- the Compile on Save feature is turned off for the project. -->
<!-- You can turn off the Compile on Save (or Deploy on Save) setting -->
<!-- in the project's Project Properties dialog box.-->
<project name="ExpenseManager" default="default" basedir=".">
    <description>Builds, tests, and runs the project ExpenseManager.</description>
    <import file="nbproject/build-impl.xml"/>
```

```
<!--

There exist several targets which are by default empty and which can be
used for execution of your tasks. These targets are usually executed
before and after some main targets. They are:

  -pre-init:                 called before initialization of project properties
  -post-init:                called after initialization of project properties
  -pre-compile:              called before javac compilation
  -post-compile:             called after javac compilation
  -pre-compile-single:       called before javac compilation of single file
  -post-compile-single:      called after javac compilation of single file
  -pre-compile-test:         called before javac compilation of JUnit tests
  -post-compile-test:        called after javac compilation of JUnit tests
  -pre-compile-test-single:  called before javac compilation of single JUnit test
  -post-compile-test-single: called after javac compilation of single JUunit test
  -pre-jar:                  called before JAR building
  -post-jar:                 called after JAR building
  -post-clean:               called after cleaning build products

(Targets beginning with '-' are not intended to be called on their own.)

Example of inserting an obfuscator after compilation could look like this:

    <target name="-post-compile">
        <obfuscate>
            <fileset dir="${build.classes.dir}"/>
```

```
    </obfuscate>
  </target>
```

For list of available properties check the imported

nbproject/build-impl.xml file.

Another way to customize the build is by overriding existing main targets.

The targets of interest are:

```
-init-macrodef-javac:     defines macro for javac compilation
-init-macrodef-junit:     defines macro for junit execution
-init-macrodef-debug:     defines macro for class debugging
-init-macrodef-java:      defines macro for class execution
-do-jar:                  JAR building
run:                      execution of project
-javadoc-build:           Javadoc generation
test-report:              JUnit report generation
```

An example of overriding the target for project execution could look like this:

```
  <target name="run" depends="ExpenseManager-impl.jar">
    <exec dir="bin" executable="launcher.exe">
      <arg file="${dist.jar}"/>
    </exec>
  </target>
```

```
    Notice that the overridden target depends on the jar target and not only on
    the compile target as the regular run target does. Again, for a list of available
    properties which you can use, check the target you are overriding in the
    nbproject/build-impl.xml file.

    -->
</project>
```

16.DATABASE :

Create Application Express workspace and credentials.

workspace-expenseManager

username-expenseManager

password-1234

```
CREATE TABLE  "EXPENDITURE_MASTER"
  (         "SERIALNO" NUMBER,
            "EXPENSE1" VARCHAR2(30),
            "EXPENSE2" VARCHAR2(30),
            "EXPENSE3" VARCHAR2(30),
            "EXPENSE4" VARCHAR2(30),
            "EXPENSE5" VARCHAR2(30),
            "EXPENSE6" VARCHAR2(30),
            "EXPENSE7" VARCHAR2(30),
            "EXPENSE8" VARCHAR2(30),
```

```
        "USERMAIL" VARCHAR2(50),
         PRIMARY KEY ("SERIALNO") ENABLE
  ) ;ALTER TABLE  "EXPENDITURE_MASTER" ADD CONSTRAINT "FK_PERORDERS" FOREIGN KEY
("USERMAIL")
         REFERENCES  "USER_MASTER" ("EMAIL") ENABLE;
```

```
CREATE TABLE  "EXPENSE_DETAILS"
  (      "SERIALNO" NUMBER,
        "EXPENSETYPE" VARCHAR2(30),
        "AMOUNT" NUMBER,
        "EXPDATE" DATE,
        "MONTH" VARCHAR2(30),
        "YEAR" VARCHAR2(30),
        "DETAILS" VARCHAR2(200),
        "USERMAIL" VARCHAR2(50),
         PRIMARY KEY ("SERIALNO") ENABLE,
         CONSTRAINT "U1" UNIQUE ("EXPENSETYPE", "EXPDATE", "USERMAIL") ENABLE
  ) ;ALTER TABLE  "EXPENSE_DETAILS" ADD CONSTRAINT "UFK2" FOREIGN KEY ("USERMAIL")
         REFERENCES  "USER_MASTER" ("EMAIL") ENABLE;
```

```
CREATE TABLE  "USER_MASTER"
  (         "NAME" VARCHAR2(60),
            "ADDR" VARCHAR2(60),
            "PHNO" VARCHAR2(30),
            "PASSWORD" VARCHAR2(30),
            "MONTHLYEXP" NUMBER,
            "INCOME" NUMBER,
            "EMAIL" VARCHAR2(50),
            "QUES" VARCHAR2(60) NOT NULL ENABLE,
            "ANS" VARCHAR2(60) NOT NULL ENABLE,
             PRIMARY KEY ("EMAIL") ENABLE
  ) ;
```

```
create sequence "SL"
start with 1
increment by 1
nocache
nocycle
noorder
/
```

17. Testing

Team Interaction

The following describes the level of team interaction necessary to have a successful product.

- The Test Team will work closely with the Development Team to achieve a high quality design and user interface specifications based on customer requirements. The Test Team is responsible for visualizing test cases and raising quality issues and concerns during meetings to address issues early enough in the development cycle.
- The Test Team will work closely with Development Team to determine whether or not the application meets standards for completeness. If an area is not acceptable for testing, the code complete date will be pushed out, giving the developers additional time to stabilize the area.
- Since the application interacts with a back-end system component, the Test Team will need to include a plan for integration testing. Integration testing must be executed successfully prior to system testing.

Test Objective

The objective our test plan is to find and report as many bugs as possible to improve the integrity of our program. Although exhaustive testing is not possible, we will exercise a broad range of tests to achieve our goal. We will be testing a Binary Search Tree Application utilizing a pre-order traversal format. There will be eight key functions used to manage our application: load, store, clear, search, insert, delete, list in ascending order, and list in descending order. Our user interface to utilize these functions is designed to be userfriendly and provide easy manipulation of the tree. The application will only be used as a demonstration tool, but we would like to ensure that it could be run from a variety of platforms with little impact on performance or usability.

Process Overview

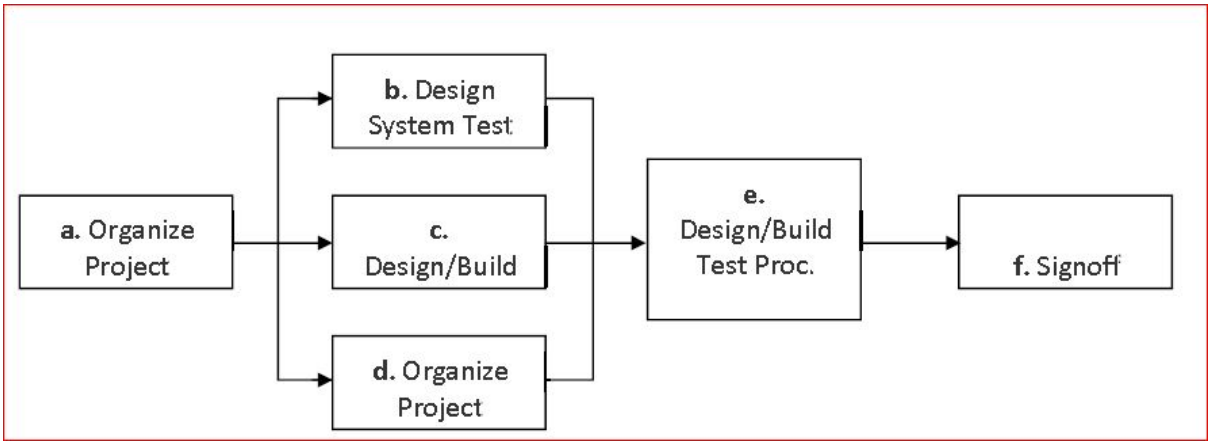
The following represents the overall flow of the testing process:

1. Identify the requirements to be tested. All test cases shall be derived using the current Program Specification.
2. Identify which particular test(s) will be used to test each module.
3. Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.
4. Identify the expected results for each test.
5. Document the test case configuration, test data, and expected results.
6. Perform the test(s).
7. Document the test data, test cases, and test configuration used during the testing process. This information shall be submitted via the Unit/System Test Report (STR).
8. Successful unit testing is required before the unit is eligible for component integration/system testing.
9. Unsuccessful testing requires a Bug Report Form to be generated. This document Shall describe the test case, the problem encountered, its possible cause, and the sequence ofevents that led to the problem. It shall be used as a basis for later

technical analysis.

10. Test documents and reports shall be submitted. Any specifications to be reviewed, revised,or updated shall be handled immediately.

Testing Process



The diagram above outlines the Test Process approach that will be followed

a. **Organize Project** involves creating a System Test Plan, Schedule & Test Approach, and assigning responsibilities.
b. **Design/Build System Test** involves identifying Test Cycles, Test Cases, Entrance & Exit Criteria, Expected Results, etc. In general, test conditions/expected results will be identified by the Test Team in conjunction with the Development Team. The Test Team will then identify Test Cases and the Data required. The Test conditions are derived from the Program Specifications Document.
c. **Design/Build Test Procedures** includes setting up procedures such as Error Management systems and Status reporting.
d. **Build Test Environment** includes requesting/building hardware, software and data setups.

Case1

Tested By:	Subham Ruj
Test Type	Unit Testing
Test Case Number	2
Test Case Name	User Identification
Test Case Description	password so that he/she can able to go for the further options. The test case will check the application for the same since a user can only login with the correct userid, password..
Item(s) to be tested	
2	Verification of the user_id and password with the record in the database.
Specifications	
Input	Expected Output/Result
1) Correct User id and password	1) Successful login
2) Incorrect Id or Password	2) Failure Message

e. Execute System Tests The tests identified in the Design/Build Test Procedures will be executed. All results will be documented and Bug Report Forms filled out and given to the Development Team as necessary.

f. Signoff - Signoff happens when all pre-defined exit criteria have been achieved.

Test Strategy:

The following outlines the types of testing that will be done for unit, integration, and system testing. While it includes what will be tested, the specific use cases that determine how the testing is done will be detailed in the Test Design Document. The test cases that will be used for designing use cases is shown in Figure 2.1 and onwards.

Case2

Tested By:	Subham Ruj
Test Type	Unit Testing
Test Case Number	2
Test Case Name	Expense Entry
Test Case Description	The User enters an expense details i.e. its type, date, amount. The test case checks whether the user is able to note an entry.
Item(s) to be tested	
1	Check whether the user is trying to enter same type of expense in same day.
2	Check if the user can actually note an entry.

Specifications

Input	Expected Output/Result
1) Check if the user can actually note an entry. 2) Check whether user able to enter same type of expense in a date for more than one time.	1) The user is able to note an entry. 2) Return an error message.

Case3

Tested By:	Subham Ruj
Test Type	Unit Testing
Test Case Number	3
Test Case Name	Expense Edit
Test Case Description	The User updates a previously noted expense. The test case checks whether the user can update an entry.
Item(s) to be tested	
1	Whether the user can update an entry in the database.
Specifications	
Input	Expected Output/Result
1) Check if the user can actually edit an expense details.	1) The user is able to update an entry.

Unit Testing

Unit Testing is done at the source or code level for language-specific programming errors such as bad syntax, logic errors, or to test particular functions or code modules. The unit test cases shall be designed to test the validity of the programs correctness.

White Box Testing

In white box testing, the UI is bypassed. Inputs and outputs are tested directly at the code level and the results are compared against specifications. This form of testing ignores the function of the program under test and will focus only on its code and the structure of that code. Test case designers shall generate cases that not only cause each condition to take on all possible values at least once, but that cause each such condition to be executed at least once. To ensure this happens, we will be applying Branch Testing. Because the functionality of the program is relatively simple, this method will be feasible to apply.

Each function of the binary tree repository is executed independently; therefore, a program flow for each function has been derived from the code.

Black Box Testing

Black box testing typically involves running through every possible input to verify that it results in the right outputs using the software as an end-user would. We have decided to perform Equivalence Partitioning and Boundary Value Analysis testing on our application.

System Testing

The goals of system testing are to detect faults that can only be exposed by testing the entire integrated system or some major part of it. Generally, system testing is mainly concerned with areas such as performance, security, validation, load/stress, and configuration sensitivity. But in our case well focus only on function validation and performance. And in both cases we will use the black-box method of testing.

18. System Security measures (Implementation of security for the project developed)

- Only authorized users are allowed.
- Without signing in users are not allowed to enter the expense entry detail frame.
- Valid email must be entered. (i.e XYZ@gmail.com, MYZ@yahoo.com, etc).
- Valid phone number must be entered. (i.e +91-9641365727, +123-69691714).

19. Database/Data security

- Database is present in the user machine.
- Oracle's default security are applied.

20. Creation of User profiles and access rights

- Click on the New account on the login frame.
- Fill all the required details then click on the Sign in button in the sign in frame.
- Now fill all the details of the expense then click on the 'Confirm and Sign in' button in details frame or if the details are needed to be deleted then click on the Reset button.
- If all the details are entered correctly then an account will be created for the user and he can access the database.

21. Cost Estimation of the Project along with Cost Estimation Model

Analogous estimate of effort or cost

Used for Early Estimate or Individual Activity Estimate

Sample example shown below is for two major deliverables of a software project. You use a previous project as a benchmark for analogous estimation. Using your experience you will estimate a multiplier.

Multipliers:

1. Prototyping: 0.75.
2. Testing: 0.5
3. 3. Deployment: 0.5

Finally, if you want to convert to cost, you would use current rates for the resource.

WBS Id	Previous Similar Project Activity	Previous Effort	Current Project Estimate	Multiplier	Effort (Previous Effort * 0.75)	Cost (Rs. 500/hr.)
1	Prototyping	20 Work-Hours	Prototyping	0.75	15 Work hours	Rs.7500 /-
2	Testing	5 Work-Hours	Testing	0.50	2.5 work-hours	Rs. 1250/-
Total					17.5 work-hours	Rs. 8750/-

Note: Effort is also called Size and unitf estimation is called either Work-Hour, person hours.

22. Future scope and further enhancement of the Project

E M has lot of enhancement options. In future EM will store invoices. It may try to analyse the user behaviour and preferences and accordingly suggest spending wisely. AI concepts can be applied to make EM intelligent.

23. Bibliography

1. Roger S. Pressman. Software Engineering :A Practitioner's Approach(Sixth Edition ,International Edition) McGraw-Hill, 2005.

2. Ian Summerville. Software Engineering (Seventh Edition). Addison-Wesley, 2004.

3. Frederick P. Brooks. The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition. Addison-Wesley Pub Co; 1st edition (August 2, 1995).

4. 1st Edition: 1975. Tells the story of the IBM 360 Operating System, and what failed why.

5. UML = Unified Modeling Language

6. Martin Fowler. UML Distilled. Addison-Wesley Pub Co; 3rd edition (September 19, 2003).

7. A classic. Selected quotes.

8. Ken Arnold and James Gosling, The Java Programming Language, second ed., Addison-Wesley, 1998.

9. Gary Cornell and Cay S. Horst Mann, Core Java, second ed., SunSoft Press, 1997.

10. Robert Eckstein and Marc Loy and Dave Wood, Java Swing, O'Reilly, 1998.

11. Robert Englander, Developing Java Beans, O'Reilly, 1997.

12. David Flanagan, Java in a Nutshell, second ed., O'Reilly, 1996.

13. James Gosling and Bill Joy and Guy Steele, the Java Language Specification, AddisonWesley, 1996.

14. George Reese, Database Programming with JDBC and Java, O'Reilly, 1997.