**OpenSAP**

**A**

Project Work

Submitted as Minor Project in Bachelor of Engineering

Submitted to

## RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA BHOPAL (M.P)

****

Submitted By

**Pankaj Tiwari(0105CS111068)**

Under the Guidance of

Prof. Sanjay Sharma

## (Department of Computer Science & Engineering)



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

**JAN-JUN 2014**



***CERTIFICATE***

This is to certify that the project entitled **“OpenSAP”** being submitted by **Pankaj Tiwari** student of **6th**  Semester, in Computer Science & Engineering have done their work as MINOR PROJECT for Partial fulfillment of the degree from RGPV, Bhopal (M.P.) is a record of bonafide work carried out by them under our supervision.

###### Prof. Sanjay Sharma Prof. Sreeja Nair

###### **Guide Head**

###### **Department of Computer Department of computer**

**Science & Engineering Science & Engineering**

###### Prof. Amit Shrivastava/Prof Monika Agrawal

###### **Project Coordinators**

**Acknowledgement**

**I would like to place on record my deep sense of gratitude to Prof. Shreeja Nair HOD-Dept. of Computer Science Engineering OIST Bhopal , for his generous guidance, help and useful suggestions.**

**I express my sincere gratitude to Prof. Sanjay Sharma Dept. of Computer Science Engineering, OIST Bhopal, for his stimulating guidance, continuous encouragement and supervision throughout the course of present work.**

**I also wish to extend my thanks to Prof. Amit shrivastava and other colleagues for attending my seminars and for their insightful comments and constructive suggestions to improve the quality of this project work.**

**I am extremely thankful to Dr. V.K.Sahu, Director OIST Bhopal for providing me infrastructural facilities to work in, without which this work would not have been possible.**

**Table of Contents**

* **LIST OF FIGURES……………………………………………………………5**
* **INTRODUCTION……………………………………………………………..7**
* **BACKGROUND AND LITERATURE SURVEY…………………………..8**
* **PROCESS MODEL…………………………………………………………..14**
* **DESIGN……………………………………………………………………….17**
* **TECHNICAL DETAILS…………………………………………………….25**
* **CODING………………………………………………………………………27**
* **TESTING……………………………………………………………………...61**
* **SCREEN LAYOUTS…………………………………………………………67**
* **FUTURE ENHANCEMENTS……………………………………………….77**
* **CONCLUSION……………………………………………………………….79**
* **REFERENCES………………………………………………………………..81**
* **BIBLIOGRAPHY…………………………………………………………….81**

**List Of figures**

**Figure Title Page no.**

**3.1 Waterfall model 15**

**4.1 Use case diagram 18,19**

**4.2 Class Diagram 20**

**4.3 Sequence Diagram 21**

**4.4 Component diagram 22,23**

**4.5 Deployment Diagram 24**

### ABSTRACT

### An application development foundation or framework built for business world context in order to make the business application development for developers easy.

### Inbuilt data structures for business world that can be suitable for any business application development need.

### The developers who create applications for business context do not need to worry about creating everything from scratch,with OpenSAP they can use existing data structures,connectivity facilities and a full stack of different helper files that make the b-app making an ease.

### In broad scenario when there will be no work from scratch,the application development will be more productive and more focused towards the business logic and it will save a lot of work for developers and a good amount in the cost of software production.

### Chapter 1

### Introduction

### Overview

OpenSAP is an effort to make the open source implementation of business framework from scratch that will be available as Open Source and Free Software on Internet.This will lead to minimize the cost gap for students and organisations who want to develop application in business context with minimum effort.

Information for OpenSAP System has been modularized into following modules.

**ADMINISTRATOR MODULE**

In this module when the administrator will enter in to the administrator page and this page consists of two following sub modules.

* **Generic Login System:** Different levels of users will login to a common login system and will be redirected accordingly.
* **Module for different purpose:** In OpenSAP different reusable modules are created with atomic functionality.
* **Business Data structures:** Business Data structures like Cheque,Ledger,Invoice etc to make business context programs ease to code with less work.
* **MVC Seperation:** In order to put models,views and controller logic separate.

**1.2 Project Objective :-.**

* **Can be used anywhere any time as it is a web based application(user location doesn’t matter).**

**1.3 PROJECT SCOPE:**

Objective of this project is to build a business context application foundation which can help developers to build the business world applications easily and avoiding the effort to create everything from scratch. This will help developers and organizations to effectively build the B-apps and reduce the project cost to a large extent. Further as an added advantage of this project of it being open source developers from various part of the world will be able to collaborate into the effective development of the product without any ownership related issue.

**CHAPTER 2**

**BACKGROUND AND LITERATURE SURVEY**

**2.1 Software Requirement Specification:-**

**2.1.1 Introduction:-**

The purpose of designing this software is to provide teacher with all the data and facts related to students sitting at one place.It enables the faculty a better understanding of different students of different semester and branch .

* + 1. **functional requirement:-**

Functional Requirements:

**1. Installable desktop client application:**

Desktop application that can communicate with the remote server and act as desktop

middleware between data entry point and server.This application should contain most of the view related code that is needed to render the response sent from the server in

appropriate way.

**2. Web Client:**

In case of unavailability of desktop application web client will act as rescue.It should be

accesible through any common web browser following w3c standards.

**3. Authentication:**

Different type of users should be assigned different authentication levels according to the task and rights assigned to them.A proper rights management should be maintained.

**4. Communication system:**

Communication system should be there which can help different users communicate in

text based media i.e. message between their working hierarchy.

5. Business logics should be implemented in **the core business APIs.**

* + 1. **Non functional requirement:-**

1.OpenSAP should act as a business application development framework which provided different set of features to make business class applications.

2. It should be leak proof and secure.

3. The Code and Components should be open source and free and there should be no any

licensing cost involved in any direct or indirect sense.

4. Should be scalable on high load or demands.

5. System should be modular with respect to the business needs.

6. Performance

**2.1.4 interface:-**

**2.1.4.1 User Interfaces:-**

-home page which provides all function.

-user interface should be easy to interact

|  |  |
| --- | --- |
| **2.1.4.2 Hardware Interfaces** |  |
| **Monitor screen:-**The software shall display information to the user via monitor.  **Mouse:-**The software shall interact with the movement of the mouse.  **Keyboard:-**The software shall interact with the keystrokes of the keyboard. |

**2.1.4.3 Software interface**

Software interface is developed in JAVA and SQL language.interface of this SOFTWARE developed according to the latest tags of JAVA and SQL.

**2.2 Feasibility Report :-**

**2.2.1 Innovativeness and Usefulness :-**

Technical feasibility centers on the existing manual system of the test

management process and to what extent it can support the system.

According to feasibility analysis procedure the technical feasibility of the

system is analyzed and the technical requirements such as software facilities ,input

are identified.

**2.2.2Market Potential and Competitive advantages:-**

Economic analysis is most frequently used for evaluation of the effectiveness of the

system. This part of feasibility study gives the top management the economic

justification for the new system. This system has a great market potential because,

if the organization implements this system, it need not require any additional

hardware resources as well as it will be saving lot of time.

# DEVELOPMENT METHOD:

The following methods and approaches are used to develop this project.

## MySQL Server

A database management, or DBMS, gives the user access to their data and helps them transform the data into information. These systems allow users to create, update and extract information from their database.

A database is a structured collection of data. Data refers to the characteristics of people, things and events. SQL Server stores each data item in its own fields. In SQL Server, the fields relating to a particular person, thing or event are bundled together to form a single complete unit of data, called a record (it can also be referred to as raw or an occurrence). Each record is made up of a number of fields. No two fields in a record can have the same field name.

## SQL Server Tables

SQL Server stores records relating to each other in a table. Different tables are created for the various groups of information. Related tables are grouped together to form a database.

## Primary Key

Every table in SQL Server has a field or a combination of fields that uniquely identifies each record in the table. The Unique identifier is called the Primary Key, or simply the Key. The primary key provides the means to distinguish one record from all other in a table. It allows the user and the database system to identify, locate and refer to one particular record in the database.

* **Foreign Key**

When a field is one table matches the primary key of another field is referred to as a foreign key. A foreign key is a field or a group of fields in one table whose values match those of the primary key of another table.

* **Referential Integrity**

Not only does SQL Server allow you to link multiple tables, it also maintains consistency between them. Ensuring that the data among related tables is correctly matched is referred to as maintaining referential integrity.

* **Relational Database**

Sometimes all the information of interest to a business operation can be stored in one table. SQL Server makes it very easy to link the data in multiple tables. Matching an employee to the department in which they work is one example. This is what makes SQL Server a relational database management system, or RDBMS. It stores data in two or more tables and enables you to define relationships between the table and enables you to define relationships between the tables.

* **Data Abstraction**

A major purpose of a database system is to provide users with an abstract view of the data. This system hides certain details of how the data is stored and maintained. Data abstraction is divided into three levels.

* + **Physical level:** This is the lowest level of abstraction at which one describes how the data are actually stored.
  + **Conceptual Level:**  At this level of database abstraction all the attributed and what data are actually stored is described and entries and relationship among them.
  + **View level:**  This is the highest level of abstraction at which one describes only part of the database.

**Advantages of RDBMS**

1. Redundancy can be avoided
2. Inconsistency can be eliminated
3. Data can be Shared
4. Standards can be enforced
5. Security restrictions ca be applied
6. Integrity can be maintained
7. Conflicting requirements can be balanced
8. Data independence can be achieved.

**Disadvantages of RDBMS**

A significant disadvantage of the RDBMS system is cost. In addition to the cost of purchasing of developing the software, the hardware has to be upgraded to allow for the extensive programs and the workspace required for their execution and storage. While centralization reduces duplication, the lack of duplication requires that the database be adequately backed up so that in case of failure the data can be recovered.

## JAVA 1.6

The most important characteristic of Java is that it was designed from the outset to be machine independent. We can run Java programs unchanged on any machine and operating system combination that supports Java. Java programs are intrinsically more portable than programs written in other languages. An application written in Java will only require a single set of source code statements, regardless of the number of different computer platforms on which

it is run, so it is very useful for internet application.

* Platform independence - Java programs can be run on many platforms without modification. This portability is assured by using a Virtual machine1. When a Java program is compiled byte-code is created rather than a standard executable file. Effectively, this is machine code for a virtual machine, which is then interpreted by the Java interpreter. The byte-code can be run on any platform which has a suitable interpreter.
* Security - since Java has always been designed with distributed applications in mind, security has been incorporated right from the start, and if anything this has been seen to be too restrictive.

**Java and the JDK**

The Java Development Kit contains all the necessary tools for the development of Java applications and applets, including a compiler, interpreter, class libraries, applet viewer and debugger. The current version of the JDK, 1.4.2, is freely available from Sun's web she- for machines running Windows 95, Windows NT, Solaris SPARC and Solaris x86, and a version for the Macintosh is expected later this year (version 1,0 is currently available). Ports to other machines may also be available from other sources. It should be noted that browsers may not yet support applets created using features in the latest JDK.

**Characteristics of Java**

* Simple
* Secure
* Portable
* Object-oriented
* Robust
* Multithreaded
* Architecture-neutral
* Interpreted
* High performance
* Distributed
* Dynamic

**J2EE**

The Java-2 Enterprise Edition (J2EE™) provides a component-based approach to the design, development, assembly, and deployment of enterprise applications. The J2EE platform offers a multitiered distributed application model, reusable components, a unified security model, flexible transaction control, and web services support through integrated data interchange on Extensible Markup Language (XML)-based open standards and protocols.

**Components of J2EE**

* **SERVLET**

Java servlets are small, platform-independent Java programs that can be used to extend the functionality of a Web server in a variety of ways. Servlets are to the server what applets are to the client—small Java programs compiled to bytecode that can be loaded dynamically and that extend the capabilities of the host. It is a server side programming language.

* **JSP**

Java Server Pages is the extension of servlet to simplify the programming and coding of servlet. It includes HTML tags to make coding easy. Ultimately it convert into the servlet at the time of calling.

**CHAPTER 3**

**PROCESS MODEL**

**3.1 Proposed Process Model:-**

**In this project Waterfall Model is followed.**



**Fig 3.1**

**This model contains 6 phases:-**

* **Feasibility and requirement analysis :-**

Requirement gathering activity involves the analysis of the problem and

collection of the relevant information relating to the product. The main aim of

this activity is to determine whether it would be financially and technically

feasible to develop the product.

* **Requirement gathering:-**

The goal of this phase is to understand the exact requirements of the customer

and to document them properly.

* **System Design :-**

The goal of this phase is to transform the requirement specification into a

structure that is suitable for implementation in some programming language.

* **Testing:-**

In this phase all the modules of the software are tested.

* **Deployment of System:-**

In this all the modules are integrated and then deployed.

* **Operation and Maintenance:-**

Release of software inaugurates the operation and life cycle phase of the

operation.

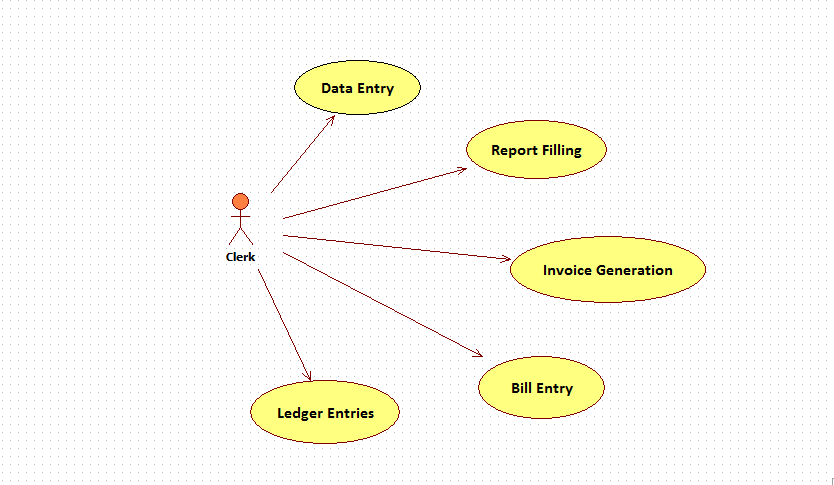
**The phases always occur in this order and do not overlap.**

**CHAPTER 4**

**DESIGN**

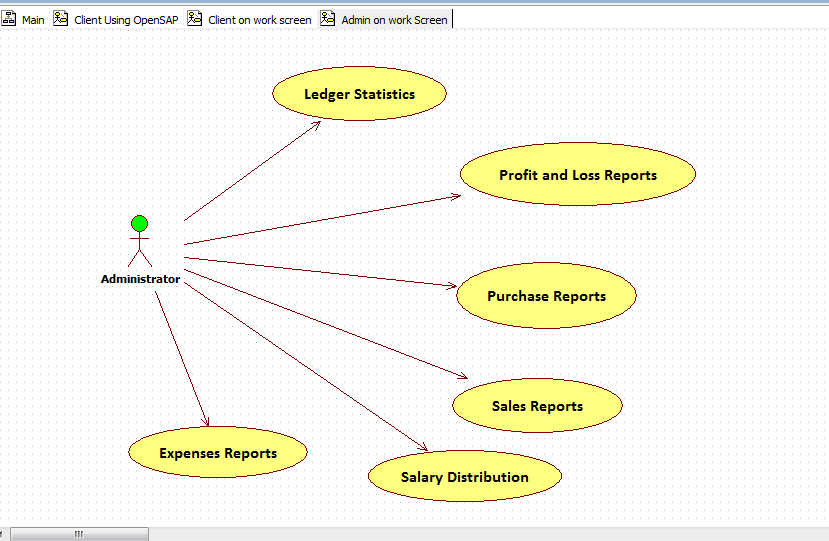
**4.1 Use case diagram**:- A use case diagram can portray the different types of users of a system and the various ways that they interact with the system.

**Fig 4.1**

****

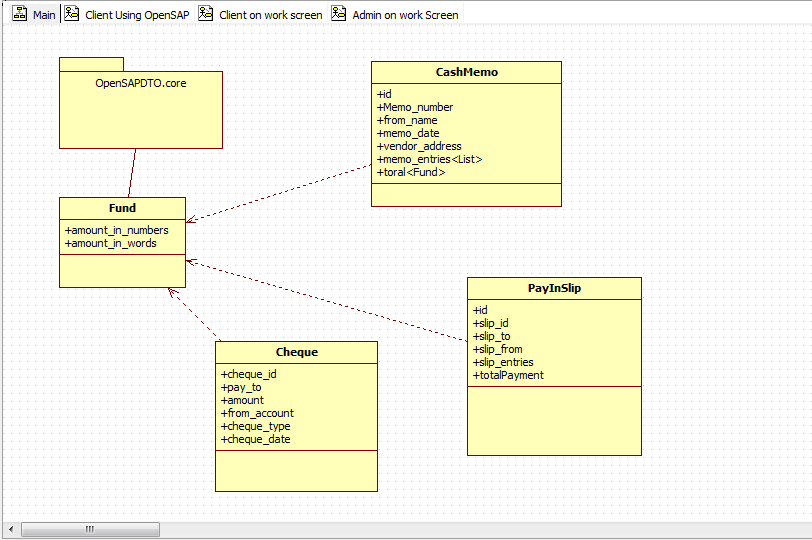
Use case diagram for a cleark level person who interact with different modules.

**Fig 4.2**

****

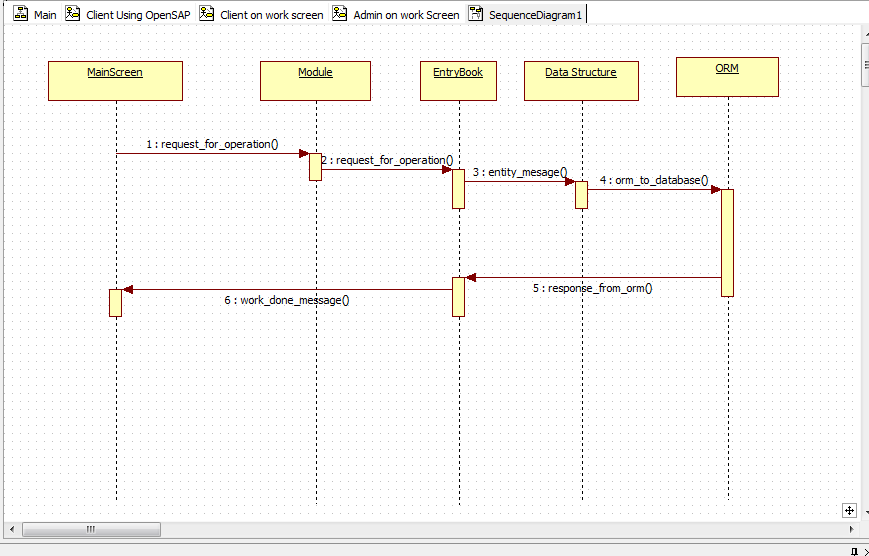
Use case diagram for the Administrator person who used different modules according to his admin panel.

**4.2Class diagram**

****

**4.3 Sequence Diagram:-** A sequence diagram interaction diagram that shows how processes operate with one another and in what order.

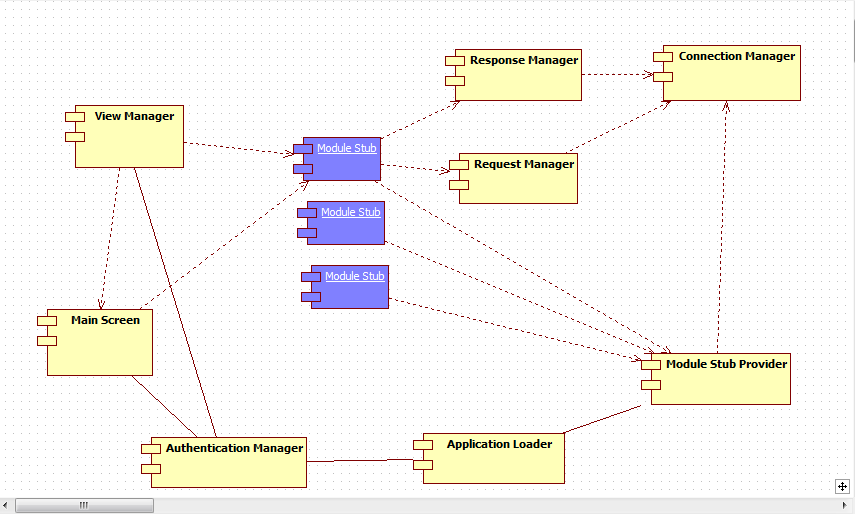
SEQUENCE DIAGRAM OF HOME PAGE:-



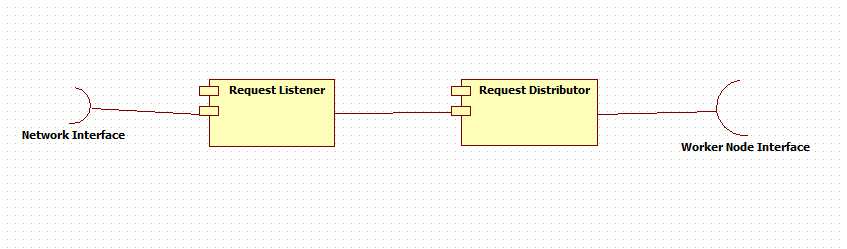
Following diagram shows how module is loaded and its working operationwise in the system.

**4.4 Component Diagram**:- A component diagram depicts how components are wired together to form larger components and or software system. They are used to illustrate the structure of arbitrarily complex systems.

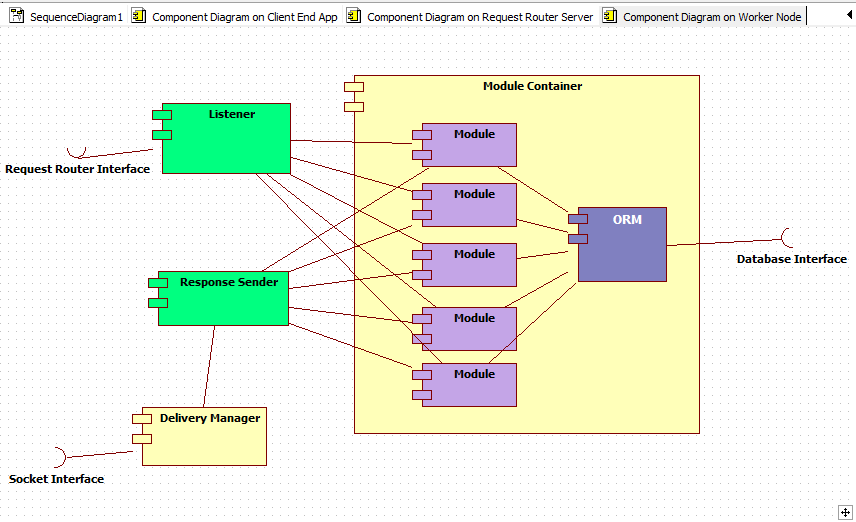
**Fig 4.4.1 Desktop App**



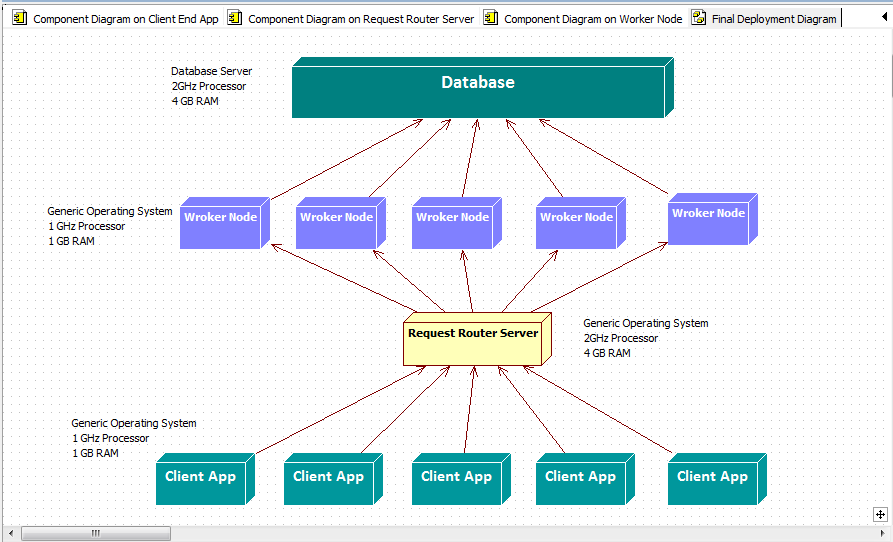
**Fig 4.4.2 Request Routing**

****

**Fig 4.4.3 In Server**

****

**4.5 Deployment Diagram:- A deployment diagram in the**[**Unified Modeling Language**](http://en.wikipedia.org/wiki/Unified_Modeling_Language)**models the physical deployment of**[**artifacts**](http://en.wikipedia.org/wiki/Artifact_(UML))**on**[**nodes**](http://en.wikipedia.org/wiki/Node_(UML))**.**

****

**CHAPTER 5**

**TECHNICAL DETAILS**

**5.1 Software Specification:-**

**Server:**

* Front end Web: HTML,CSS,JavaScript Desktop App : Java,JavaFX
* Back end JSP,Servlet,mysql
* Database My SQL.
* Server MySQL Server,Apache Tomcat 7.0.1
* Connectivity jdbc-hibernate connectivity over jdbc

**Client:**

* Java Virtual Machine Should be installed
* A web browser should be installed to access web-application.
  1. **Hardware Specification:-**

**Client Side**

* 512 MB RAM (minimum).
* 1 GB Hard disk Space(minimum).
* Network Connection

**Server Side**

* 4 GB RAM (minimum).
* 1 GB Hard disk Space(minimum).
* Network Connection

**CHAPTER 6**

**CODING**

**FUND CLASS:**

package openSAPdto.core;

public class Fund {

private float amount\_in\_numbers;

private String amount\_in\_words;

public Fund() {

this.amount\_in\_numbers=0;

this.amount\_in\_words="Zero rupees only";

}

public Fund(float amount\_in\_numbers, String amount\_in\_words) {

this.amount\_in\_numbers = amount\_in\_numbers;

this.amount\_in\_words = amount\_in\_words;

}

public float getAmount\_in\_numbers() {

return amount\_in\_numbers;

}

public void setAmount\_in\_numbers(float amount\_in\_numbers) {

this.amount\_in\_numbers = amount\_in\_numbers;

}

public String getAmount\_in\_words() {

return amount\_in\_words;

}

public void setAmount\_in\_words(String amount\_in\_words) {

this.amount\_in\_words = amount\_in\_words;

}

}

**OpenSAPDto.cheque Package :**

**CLASS CHEQUE:**

package openSAPdto.cheque;

import java.util.Date;

import javax.persistence.Embedded;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import openSAPdto.core.Fund;

@Entity

public class Cheque {

@Id

@GeneratedValue

private int check\_id;

private String PAY\_TO;

@Embedded

private Fund amount;

private String from\_account;

private int check\_type;

// value to be filled frmo checkType.ACC\_DETAILS

private Date checkq\_date;

//

// public String getCheck\_id() {

// return check\_id;

// }

//

// public void setCheck\_id(String check\_id) {

// this.check\_id = check\_id;

// }

public Cheque() {

}

public int getCheck\_id() {

return check\_id;

}

public void setCheck\_id(int check\_id) {

this.check\_id = check\_id;

}

public String getPAY\_TO() {

return PAY\_TO;

}

public void setPAY\_TO(String PAY\_TO) {

this.PAY\_TO = PAY\_TO;

}

public Fund getAmount() {

return amount;

}

public void setAmount(Fund amount) {

this.amount = amount;

}

public String getFrom\_account() {

return from\_account;

}

public void setFrom\_account(String from\_account) {

this.from\_account = from\_account;

}

public int getCheck\_type() {

return check\_type;

}

public void setCheck\_type(int check\_type) {

this.check\_type = check\_type;

}

public Date getCheckq\_date() {

return checkq\_date;

}

public void setCheckq\_date(Date checkq\_date) {

this.checkq\_date = checkq\_date;

}

@Override

public String toString() {

return "[" + this.getCheck\_id() + "]"

+ "{"

+ "PAY\_TO:" + this.PAY\_TO + ""

+ " ,PAY FROM:" + this.from\_account + ""

+ " ==> " + this.getAmount() + "}";

} }

**CLASS ChequeType:**

package openSAPdto.cheque;

public class ChequeType {

public static int ACC\_PAYEE = 0;

public static int HOLDER\_PAYEE = 1;

public static int ANY\_PAYEE = 2;

}

**ChequeBookEntry Class :**

package openSAPdto.cheque;

import openSAPdto.cheque.Cheque;

import java.io.Serializable;

import java.util.List;

import org.hibernate.Query;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

public class CheqeEntryBook {

private SessionFactory factory;

public CheqeEntryBook() {

try {

factory = new Configuration().configure().buildSessionFactory();

} catch (Exception ex) {

System.out.println("Data Manager >> Error in DataManager Constructor:" + ex);

}

}

public void init() {

try {

factory = new Configuration().configure().buildSessionFactory();

} catch (Exception ex) {

System.out.println("ChequeEntryBook >> Error in ChequeEntryBook init:" + ex);

}

}

public void add\_a\_cheque(Cheque s) {

Session session = factory.openSession();

Transaction tx = null;

try {

tx = session.beginTransaction();

int roll\_no = (int) session.save(s);

System.out.println("ChequeEntryBook >> Cheque Saved index = " + roll\_no);

tx.commit();

} catch (Exception ex) {

System.out.println("ChequeEntryBook >> Could not save Cheque index = " + s);

ex.printStackTrace();

} finally {

session.close();

}

}

public void update\_a\_cheque(Cheque s) {

Session session = factory.openSession();

Transaction tx = null;

try {

tx = session.beginTransaction();

// Student s1 = (Student) session.get(Student.class, s.getRoll\_number());

session.update(s);

System.out.println("ChequeEntryBook >> Student updated roll\_no = " + s.getCheck\_id());

tx.commit();

} catch (Exception ex) {

System.out.println("ChequeEntryBook >> Could not updated student roll\_no = " + s.getCheck\_id());

} finally {

session.close();

}

}

public void delete\_a\_cheque(String cheque\_id) {

Session session = factory.openSession();

Transaction tx = null;

try {

tx = session.beginTransaction();

Cheque s = (Cheque) session.get(Cheque.class, cheque\_id);

session.delete(s);

System.out.println("ChequeEntryBook >> Student deleted roll\_no = " + cheque\_id);

tx.commit();

} catch (Exception ex) {

System.out.println("ChequeEntryBook >> Could not delete student roll\_no = " + cheque\_id);

} finally {

session.close();

}

}

public Cheque get\_a\_cheque(String cheque\_id) {

Session session = factory.openSession();

Cheque s = null;

Transaction tx = null;

try {

tx = session.beginTransaction();

s = (Cheque) session.get(Cheque.class, cheque\_id);

System.out.println("ChequeEntryBook >> Student fetched roll\_no = " + cheque\_id);

tx.commit();

return s;

} catch (Exception ex) {

System.out.println("ChequeEntryBook >> Could not fetch student roll\_no = " + cheque\_id);

} finally {

session.close();

}

return s;

}

public List get\_selected\_user\_cheques(String from\_account) {

Session session = factory.openSession();

List results = null;

Transaction tx = null;

try {

tx = session.beginTransaction();

String hql = "FROM Cheque S WHERE S.from\_account =" + from\_account;

Query query = session.createQuery(hql);

results = query.list();

System.out.println("ChequeEntryBook >> List fetched = " + from\_account);

tx.commit();

return results;

} catch (Exception ex) {

System.out.println("ChequeEntryBook >> Could not fetch list from\_account = " + from\_account);

} finally {

session.close();

}

return results;

}

}

public class Subject {

private String subjectCode;

private int semester;

private String subjectName;

public Subject(String subjectCode, int semester, String subjectName) {

this.subjectCode = subjectCode;

this.semester = semester;

this.subjectName = subjectName;

}

public String getSubjectCode() {

return subjectCode;

}

public void setSubjectCode(String subjectCode) {

this.subjectCode = subjectCode;

}

public int getSemester() {

return semester;

}

public void setSemester(int semester) {

this.semester = semester;

}

public String getSubjectName() {

return subjectName;

}

public void setSubjectName(String subjectName) {

this.subjectName = subjectName;

}

public void load(){

}

}

**Hibernate Configuration :**

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE hibernate-configuration SYSTEM

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.dialect">

org.hibernate.dialect.MySQLDialect

</property>

<property name="hibernate.connection.driver\_class">

com.mysql.jdbc.Driver

</property>

<!-- Assume test is the database name -->

<property name="hibernate.connection.url">jdbc:mysql://localhost/d2</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password"> </property>

<property name="show\_sql">true</property>

<property name="hibernate.hbm2ddl.auto">update</property>

<mapping class="openSAPdto.cheque.Cheque"/>

<mapping class="openSAPdto.PayInSLip.PayInSlip"/>

<mapping class="openSAPdto.CashMemo.CashMemo"/>

<mapping class="openSAPdto.core.Fund"/>

</session-factory>

</hibernate-configuration>

**Cheque Module in Servlet:** String operation = request.getParameter("operation");

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

if (module.equals("cheque")) {

if (operation.equals("add")) {

RequestDispatcher rs = request.getRequestDispatcher("cheque/add.jsp");

rs.forward(request, response);

} else if (operation.equals("list")) {

RequestDispatcher rs = request.getRequestDispatcher("cheque/list.jsp");

request.setAttribute("list", new ChequeEntryBook().get\_all\_cheques());

rs.forward(request, response);

}

}

**Module Registry**

package home;

import java.util.HashMap;

public class ModuleRegistry {

HashMap modules;

public ModuleRegistry() {

modules = new HashMap();

}

public HashMap getModules() {

init();

return modules;

}

public void setModules(HashMap modules) {

this.modules = modules;

}

public void init() {

this.modules.put("Cheque", "cheque");

this.modules.put("PaySlip", "payslip");

this.modules.put("Invoice Registry", "invoice");

this.modules.put("Product Entry", "product\_entry");

this.modules.put("Purchase", "purchase");

}

}

**Module Provider Class:**package home;

import java.util.HashMap;

public class ModuleProvider {

ModuleEntry e1,e2;

public ModuleProvider() {

init();

}

public void init() {

e1 = new ModuleEntry();

e1.setModule\_name("cheque");

e1.setModule\_url("cheque");

HashMap h1 = new HashMap();

h1.put("Add a cheque", "add");

h1.put("List all cheques", "list");

e1.setOperations(h1);

e2 = new ModuleEntry();

e2.setModule\_name("payslip");

e2.setModule\_url("payslip");

HashMap h2 = new HashMap();

h2.put("Add a payslip", "add");

h2.put("List all payslip", "list");

h2.put("Payslip Audit", "audit");

e2.setOperations(h2);

}

public ModuleEntry get\_module(String module\_name){

if(module\_name.trim().equals("cheque")){ return e1;}

if(module\_name.trim().equals("payslip")){ return e2;}

return null;

}

}

**Home Servlet:**package home;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Date;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import openSAPdto.cheque.Cheque;

import openSAPdto.cheque.ChequeEntryBook;

import openSAPdto.core.Fund;

public class home extends HttpServlet {

public void login\_filter(HttpServletRequest request, HttpServletResponse response) throws IOException {

if (request.getSession().getAttribute("login") == null) {

response.sendRedirect("login");

}

}

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

if (request.getSession().getAttribute("login") == null) {

response.sendRedirect("login"); // IF LOGIN IS NOT DONE

} else {

//IF LOGIN IS DONE

////////////////////////////////////LOGIN IS OK/////////////////////////////

System.out.println("In home servlet");

if (request.getParameter("operation") != null) {

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

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

if (module.equals("cheque")) {

if (operation.equals("add")) {

RequestDispatcher rs = request.getRequestDispatcher("cheque/add.jsp");

rs.forward(request, response);

} else if (operation.equals("list")) {

RequestDispatcher rs = request.getRequestDispatcher("cheque/list.jsp");

request.setAttribute("list", new ChequeEntryBook().get\_all\_cheques());

rs.forward(request, response);

}

}

if (module.equals("myaccount")) {

if (("operation").equals("logout")) {

request.getSession().invalidate();

response.sendRedirect("/login");

}

}

}

RequestDispatcher rs = request.getRequestDispatcher("index.jsp");

rs.forward(request, response);

////////////////////////////////////////////////////////////////////////////////////////

}// IF LOGIN IS OK END

}

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

if (request.getSession().getAttribute("login") == null) {

response.sendRedirect("login"); // IF LOGIN IS NOT DONE

} else {

//IF LOGIN IS DONE

////////////////////////////////////LOGIN IS OK/////////////////////////////

if (request.getParameter("post\_operation") != null) {

//////////////////////ONE POST CASE////////////////

if (request.getParameter("post\_operation").equals("cheque\_submit")) {

String pay\_to = request.getParameter("pay\_to");

int amount\_numbers = Integer.parseInt(request.getParameter("amount\_numbers").trim());

String amount\_words = request.getParameter("amount\_words");

String pay\_from = request.getParameter("pay\_from");

int cheque\_type = Integer.parseInt(request.getParameter("cheque\_type"));

String cheque\_date = request.getParameter("cheque\_date");

Cheque c = new Cheque();

c.setAmount(new Fund(amount\_numbers, amount\_words));

c.setPAY\_TO(pay\_to);

c.setFrom\_account(pay\_from);

c.setCheckq\_date(new Date());

c.setCheck\_type(cheque\_type);

ChequeEntryBook eb = new ChequeEntryBook();

eb.add\_a\_cheque(c);

}

/////////////////////////////////////////////////////

if (request.getParameter("post\_operation").equals("login")) {

}

////////////////////////////////////////////////////////

doGet(request, response);

}

//////////////////////////////////////////////////////////////////////////////////

}// IF LOGIN IS OK END

}

@Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

}

**View Files:**

**Generic Theming Files:  
(1)header.jsp**<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<link rel="stylesheet" href="http://localhost/bootstrap2/css/bootstrap.css"/>

<link rel="stylesheet" href="http://localhost/bootstrap2/css/bootstrap-theme.css"/>

<title>JSP Page</title>

<style>

.container{

text-align: center;

padding-right: 15px;

padding-left: 15px;

margin-right: auto;

margin-left: auto;

}

</style>

</head>

**(2)navbar.jsp**

<%@page import="home.ModuleRegistry"%>

<%@page import="java.util.Map"%>

<%@page import="java.util.HashMap"%>

<%@page import="java.util.Iterator"%>

<%@page import="java.util.Set"%>

<!-- Navaigation bar -->

<div class="navbar">

<div class="navbar-inner">

<a class="brand" href="#">Company Name</a>

<ul class="nav">

<li class="active"><a href="#">File</a></li>

<li><a href="#">Settings</a></li>

<li class="dropdown">

<a href="#" class="dropdown-toggle" data-toggle="dropdown">

Load Module

<b class="caret"></b>

</a>

<ul class="dropdown-menu">

<%

ModuleRegistry registry = new ModuleRegistry();

HashMap modules = registry.getModules();

Set s = modules.entrySet();

Iterator it = s.iterator();

%>

<%

while (it.hasNext()) {

Map.Entry me = (Map.Entry) it.next();

// out.print(me.getKey() + ": ");

// out.println(me.getValue());

%>

<li><a href="?module=<%=me.getValue()%>"><%= me.getKey()%></a></li>

<%

}

%>

</ul>

</li>

<li class="dropdown">

<a href="#" class="dropdown-toggle" data-toggle="dropdown">

My Account

<b class="caret"></b>

</a>

<ul class="dropdown-menu">

<li><a href="#">View my profile</a></li>

<li><a href="#">View my rights</a></li>

<li><a href="login?operation=logout">Logout</a></li>

</ul>

</li>

</ul>

</div>

</div>

**(3) sidebar.jsp**

<%@page import="java.util.Map"%>

<%@page import="java.util.Iterator"%>

<%@page import="java.util.Set"%>

<%@page import="java.util.HashMap"%>

<%@page import="home.ModuleProvider"%>

<%@page import="home.ModuleEntry"%>

<div class="span3">

<ul class="nav nav-tabs nav-stacked">

<%

if (request.getParameter("module") != null) {

String module\_name = request.getParameter("module");

ModuleEntry e = new ModuleProvider().get\_module(module\_name);

HashMap operations = (HashMap) e.getOperations();

String module\_url = e.getModule\_url();

module\_url = "?module=" + module\_url + "&operation=";

%>

<%

Set set = operations.entrySet();

// Get an iterator

Iterator i = set.iterator();

while (i.hasNext()) {

Map.Entry me = (Map.Entry) i.next();

%>

<li><a href="<%= module\_url%><%=me.getValue()%>"><%= me.getKey()%></a></li>

<%

}

}

%>

</ul>

</div>

**(4) scipts\_plugins.jspf**<script src="http://localhost/bootstrap2/bootstrap\_plugins/jquery.js"></script>

<script src="http://localhost/bootstrap2/bootstrap\_plugins/bootstrap-transition.js"></script>

<script src="http://localhost/bootstrap2/bootstrap\_plugins/bootstrap-alert.js"></script>

<script src="http://localhost/bootstrap2/bootstrap\_plugins/bootstrap-modal.js"></script>

<script src="http://localhost/bootstrap2/bootstrap\_plugins/bootstrap-dropdown.js"></script>

<script src="http://localhost/bootstrap2/bootstrap\_plugins/bootstrap-scrollspy.js"></script>

<script src="http://localhost/bootstrap2/bootstrap\_plugins/bootstrap-tab.js"></script>

<script src="http://localhost/bootstrap2/bootstrap\_plugins/bootstrap-tooltip.js"></script>

<script src="http://localhost/bootstrap2/bootstrap\_plugins/bootstrap-popover.js"></script>

<script src="http://localhost/bootstrap2/bootstrap\_plugins/bootstrap-button.js"></script>

<script src="http://localhost/bootstrap2/bootstrap\_plugins/bootstrap-collapse.js"></script>

<script src="http://localhost/bootstrap2/bootstrap\_plugins/bootstrap-carousel.js"></script>

<script src="http://localhost/bootstrap2/bootstrap\_plugins/bootstrap-typeahead.js"></script>

**Generic Login System:**Account Class

package login;

import javax.persistence.Entity;

import javax.persistence.Id;

@Entity

public class Account {

@Id

private String username;

private String password;

private int right\_level;

public Account() {

}

public Account(String username, String password, int right\_level) {

this.username = username;

this.password = password;

this.right\_level = right\_level;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public int getRight\_level() {

return right\_level;

}

public void setRight\_level(int right\_level) {

this.right\_level = right\_level;

}

}

**RightLevel Class**package login;

public class RightLevel {

public static int Administrator = 1;

public static int Clerk = 2;

}

**Login System Class**package login;

import java.util.List;

import org.hibernate.Query;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

class LoginSystem {

private SessionFactory factory;

public LoginSystem() {

try {

factory = new Configuration().configure().buildSessionFactory();

} catch (Exception ex) {

System.out.println("Data Manager >> Error in login Constructor:" + ex);

}

}

public boolean check(String username, String password) {

Session session = factory.openSession();

Transaction tx = null;

try {

tx = session.beginTransaction();

String hql = "FROM Account A WHERE A.username='" + username + "' AND A.password='" + password+"'";

System.out.println("HQL = "+hql);

Query query = session.createQuery(hql);

List results = query.list();

tx.commit();

Account x = (Account)results.get(0);

if(x!=null) return true;

else return false;

} catch (Exception ex) {

System.out.println("LoginSystem >> Could not find Account username = " + username);

ex.printStackTrace();

} finally {

session.close();

}

return false;

}

public void add(Account a) {

Session session = factory.openSession();

Transaction tx = null;

try {

tx = session.beginTransaction();

session.save(a);

System.out.println("LoginSystem >> Account Saved username = " + a.getUsername());

tx.commit();

} catch (Exception ex) {

System.out.println("LoginSystem >> Could not save Account username = " + a.getUsername());

ex.printStackTrace();

} finally {

session.close();

}

}

}

**Login System View : loginpage.jsp**

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<%@include file="../header.jsp" %>

<body>

<div class="container">

<div class="row">

<div class="span8">

<form class="form-horizontal span5 offset2" method="POST" action="login"

style="margin-top: 30%;">

<input type="hidden" name="post\_operation" value="login"/>

<div class="control-group ">

<label class="control-label">Username</label>

<div class="controls">

<input type="text" id="" placeholder="" name="username">

</div>

</div>

<div class="control-group">

<label class="control-label">Password</label>

<div class="controls">

<input type="password" id="" placeholder="" name="password">

</div>

</div>

<div class="control-group">

<div class="controls">

<button type="submit" class="btn btn-primary">Login</button>

</div>

</div>

<div class="control-group">

<label class="control-label">

<%

if(request.getSession().getAttribute("message")!=null){

out.println(request.getSession().getAttribute("message"));

request.getSession().removeAttribute("message");

}

%>

</label>

</div>

</form>

</div>

</div>

</div>

<%@include file="../script\_plugins.jspf" %>

</body>

</html>

**CHAPTER 7**

**TESTING**

* Software testing is the process of executing a program with intension of finding errors in the code.
* It is a process of evolution of system or its parts by manual or automatic means to verify that it is satisfying specified or requirements or not.
* To purpose of system testing is to check and find out the errors or faults as early as possible so losses due to it can be saved.
* Testing is the fundamental process of software success.
* Testing is not a distinct phase in system development life cycle but should be applicable throughout all phases i.e. design development and maintenance phase.
* Testing is used to show incorrectness and considered to success when an error is detected.

Different levels of testing are used in the test process; each level of testing aims to test different aspects of the system:-

The first level is unit testing. In this testing, individual components are tested to ensure that they operate correctly.

The second level is integration testing. It is a systematic technique for constructing the program structure. In this testing, many tested modules are combined into the subsystems which are then tested. The good here is to see if the modules can be integrated properly.

Third level is integrationtesting. System testing is actually a series of different tests whose primary purpose is to fully exercise computer based system. These tests fall outside scope of software process and are not conducted solely by software engineers.

**7.1 Testing Used:-**

* White Box Testing
* Black box Testing

**7.1.1 White box testing:-** White-box testing tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs.

While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

**Techniques used in white-box testing include:**

* API testing (application programming interface):- testing of the application using public and private APIs.
* [Code coverage](http://en.wikipedia.org/wiki/Code_coverage):-creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once).
* [Fault injection](http://en.wikipedia.org/wiki/Fault_injection) method:- intentionally introducing faults to gauge the efficacy of testing strategies.
* [Mutation testing](http://en.wikipedia.org/wiki/Mutation_testing) methods.
* [Static testing](http://en.wikipedia.org/wiki/Static_testing) method.

**7.1.2 Black-Box Testing**:- Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal implementation. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: [equivalence partitioning](http://en.wikipedia.org/wiki/Equivalence_partitioning), [boundary value analysis](http://en.wikipedia.org/wiki/Boundary_value_analysis), [all-pairs testing](http://en.wikipedia.org/wiki/All-pairs_testing), [state transition tables](http://en.wikipedia.org/wiki/State_transition_table), [decision table](http://en.wikipedia.org/wiki/Decision_table) testing, [fuzz testing](http://en.wikipedia.org/wiki/Fuzz_testing), [model-based testing](http://en.wikipedia.org/wiki/Model-based_testing), [use case](http://en.wikipedia.org/wiki/Use_case) testing, [exploratory testing](http://en.wikipedia.org/wiki/Exploratory_testing) and specification-based testing.

### 7.1.3Alpha testing:- Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.

**7.1.4 Beta testing:-** Beta testing comes after alpha testing and can be considered a form of external [user acceptance testing](http://en.wikipedia.org/wiki/User_acceptance_testing). Versions of the software, known as [beta versions](http://en.wikipedia.org/wiki/Beta_version), are released to a limited audience outside of the programming team. The software is released to groups of people so that further testing can ensure the product has few faults or [bugs](http://en.wikipedia.org/wiki/Computer_bug). Sometimes, beta versions are made available to the open public to increase the [feedback](http://en.wikipedia.org/wiki/Feedback#In_organizations) field to a maximal number of future users.

**7.2 Test Cases & Results:-**

**Cheque Entity Class**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S. No** | **Operation** | **Correct Parameters Passed** | **Incorrect Parameters passed** | **Result** | **Expected Result** |
| 1 | Add a cheque | **Pay\_to:** A/11231089  **Amount:** Fund(1000,”One Thousand”)  **From\_account:**  A/12121335  **Cheque\_type:**  ChequeType.PAYEE  (others default) | Pay\_to:  <not specified>  Null string passed **“ ”** | OpenSAPInvalidDataException:  **Message**: Target Account not specified | Error should be thrown. |
| 2 | Update a cheque | **Pay\_to:** A/11231089  **Amount:** Fund(1000,”One Thousand”)  **From\_account:**  A/12121335  **Cheque\_type:**  ChequeType.PAYEE  (others default) | **From\_account:**  <not specified>  Null string passed **“ ”** | OpenSAPInvalidDataException:  **Message**: Target Account not specified | Error should be thrown at incorrect object. |
| 3 | Delete a cheque | **Cheque\_id:**  String cheque\_id: valid A/c112121212 | **Cheque\_id**  **<**invalid input>  Null String | OpenSAPInvalidDataException:  **Message**: Target cheque\_id not specified | Error should be thrown . |
| 4 | Fetch Cheque List | **From\_account:**  A/12121335 | **From\_account:**  **<**invalid input>  Null String | OpenSAPInvalidDataException:  **Message**: Target account not specified | Error should be thrown . |

**Result: Pass**

**PaySlip Entity Class**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.No** | **Operation** | **Correct Parameters Passed** | **InCorrect Parameters Passed** | **Result** | **Expected Result** |
| 1 | Add a PaySlip | **Slip\_id:**  A/C123234  **Slip\_to:**  A/C1234232 | <null string> passed | OpenSAPInvalidDataException  Message: Account not specified | Error should be thrown at incorrect input |
| 2 | Update a PaySlip | **Slip\_id:**  A/C123234  **Slip\_to:**  A/C1234232 | <null string> passed | OpenSAPInvalidDataException  Message: Account not specified | Error should be thrown at incorrect input |
| 3 | Delete a PaySlip | **Slip\_id:**  A/C123234 | <null string> passed | OpenSAPInvalidDataException  Message: Account not specified | Error should be thrown at incorrect input |
| 4 | Fetch PaySlip List | **Slip\_from:**  A/C123234 | <null string> passed | OpenSAPInvalidDataException  Message: Account not specified | Error should be thrown at incorrect input |

**Result: Pass**

**CashMemo Entity Class**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.No** | **Operation** | **Correct Parameters Passed** | **InCorrect Parameters Passed** | **Result** | **Inference** |
| 1 | Add a CashMemo | **Memo\_id:**  A/C123234  **memo\_to:**  A/C1234232 | <null string> passed | OpenSAPInvalidDataException  Message: Account not specified | Error should be thrown at incorrect input |
| 2 | Update a CashMemo | **Memo\_id:**  A/C123234  **memo\_to:**  A/C1234232 | <null string> passed | OpenSAPInvalidDataException  Message: Account not specified | Error should be thrown at incorrect input |
| 3 | Delete a CashMemo | **Memo\_id:**  A/C123234  **memo\_to:**  A/C1234232 | <null string> passed | OpenSAPInvalidDataException  Message: Account not specified | Error should be thrown at incorrect input |
| 4 | Fetch CashMemo List | **Memo\_id:**  A/C123234  **memo\_to:**  A/C1234232 | <null string> passed | OpenSAPInvalidDataException  Message: Account not specified | Error should be thrown at incorrect input |

**Result: Pass**

**CHAPTER 8**

**SCREEN LAYOUTS**

**LOAD BALANCING SYSTEM**

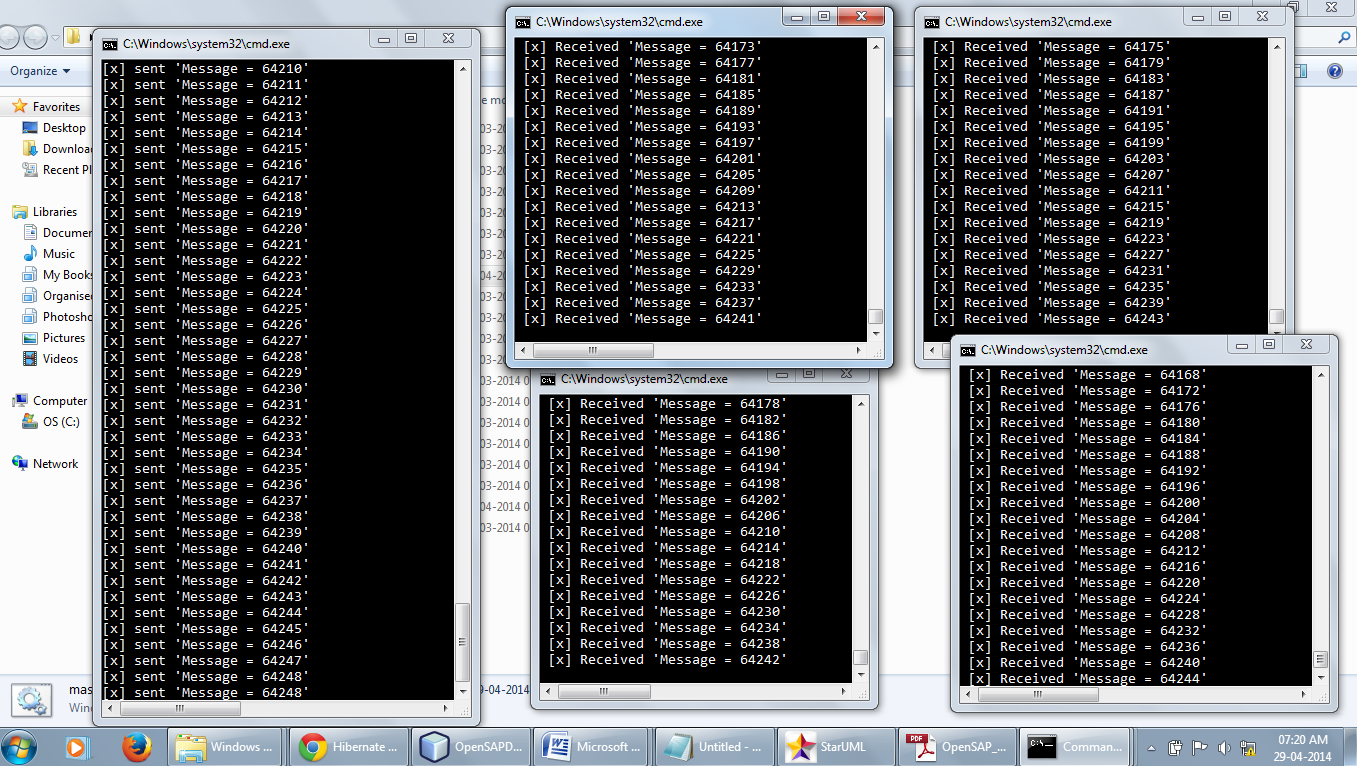
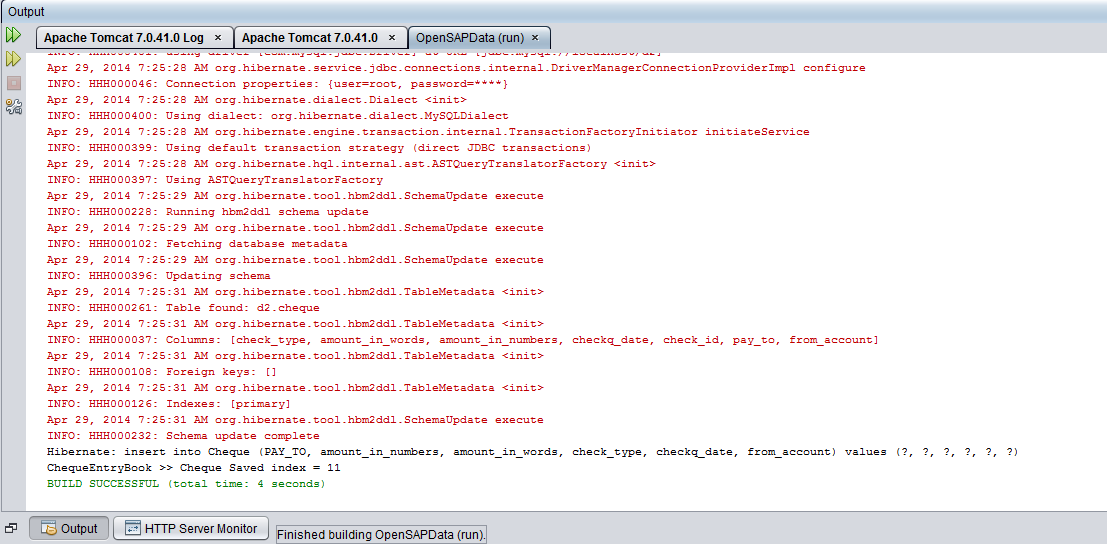
****

Fig: The Load Balacing Reqquest router Server demo where request objects being generated by a master process are evenly distributed to the worker process.

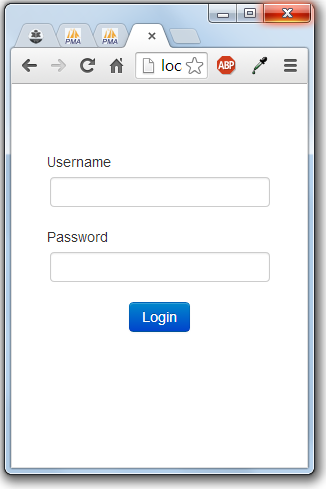
**Working of the DATA Structures:**



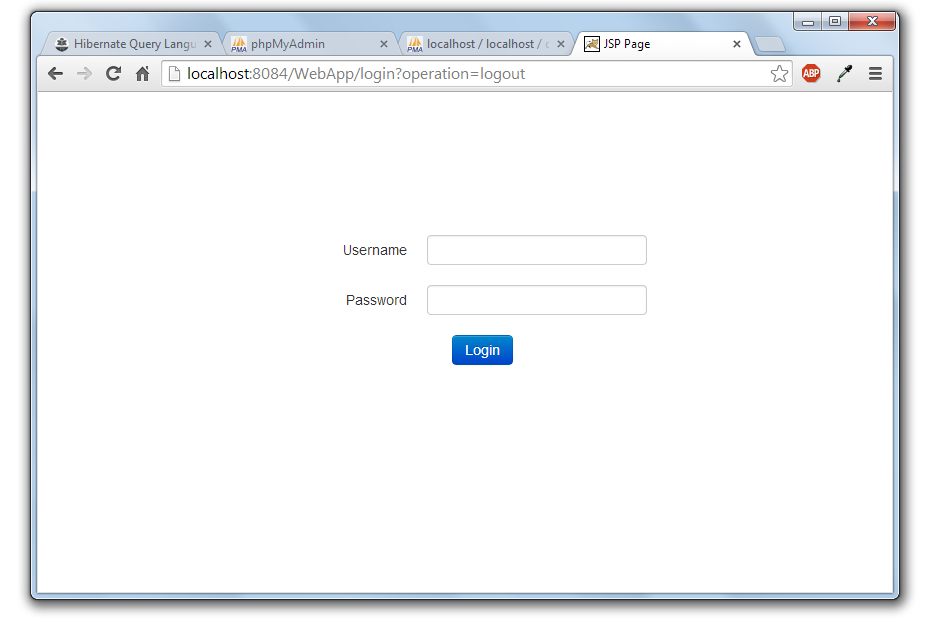
**LOGIN SCREEN:**

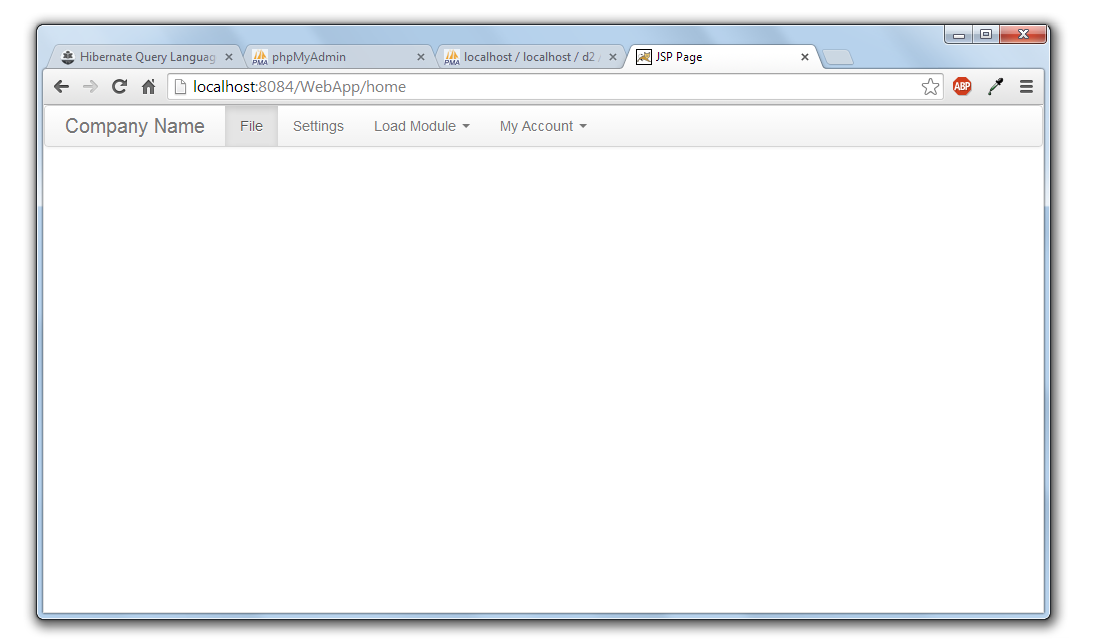
**Screen-1 (Initial)**

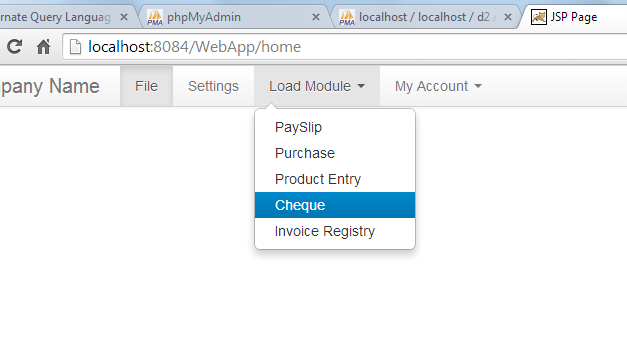
For low screen resolution device web browser



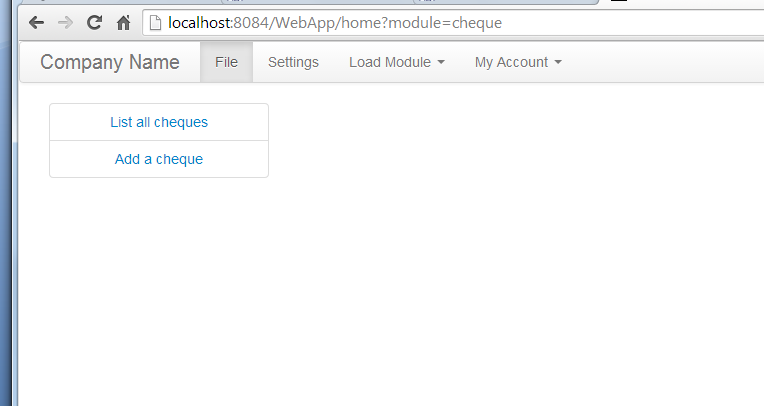
For wide screen browsers:

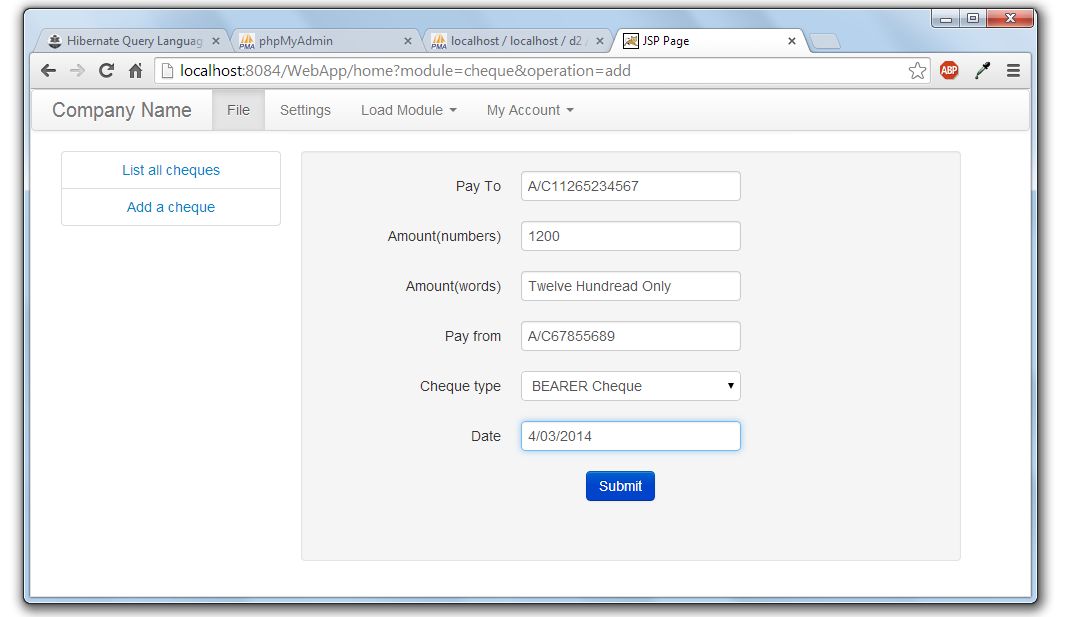


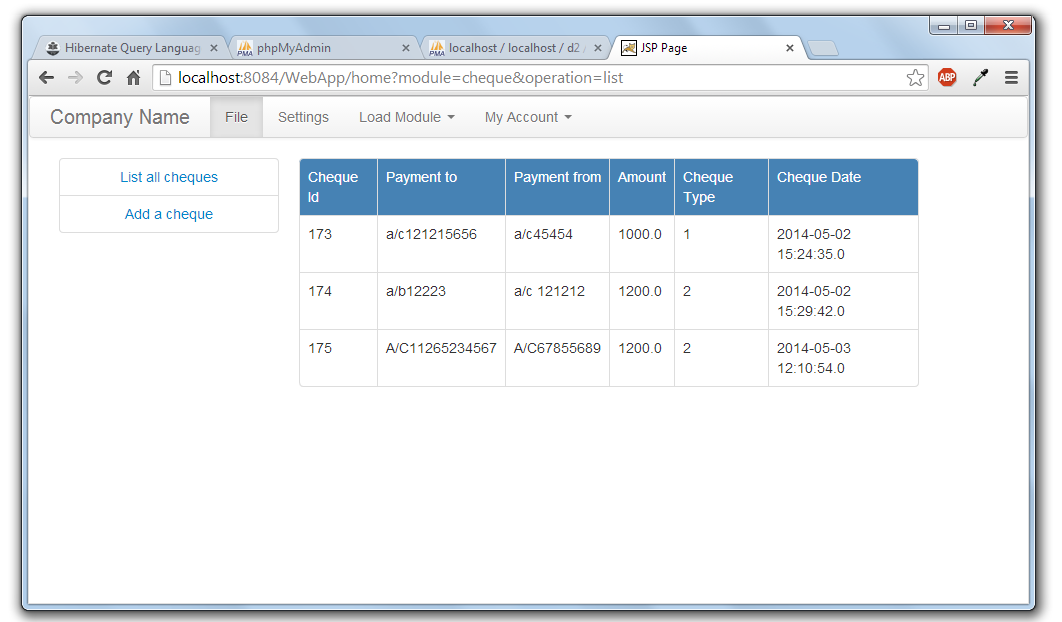
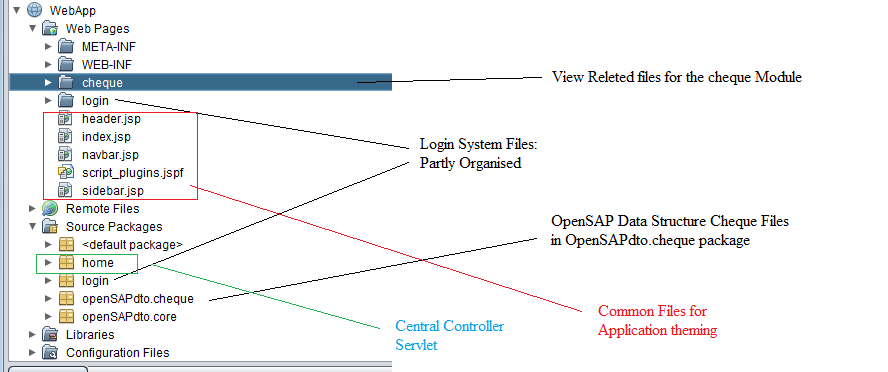
Home Screen After Login   


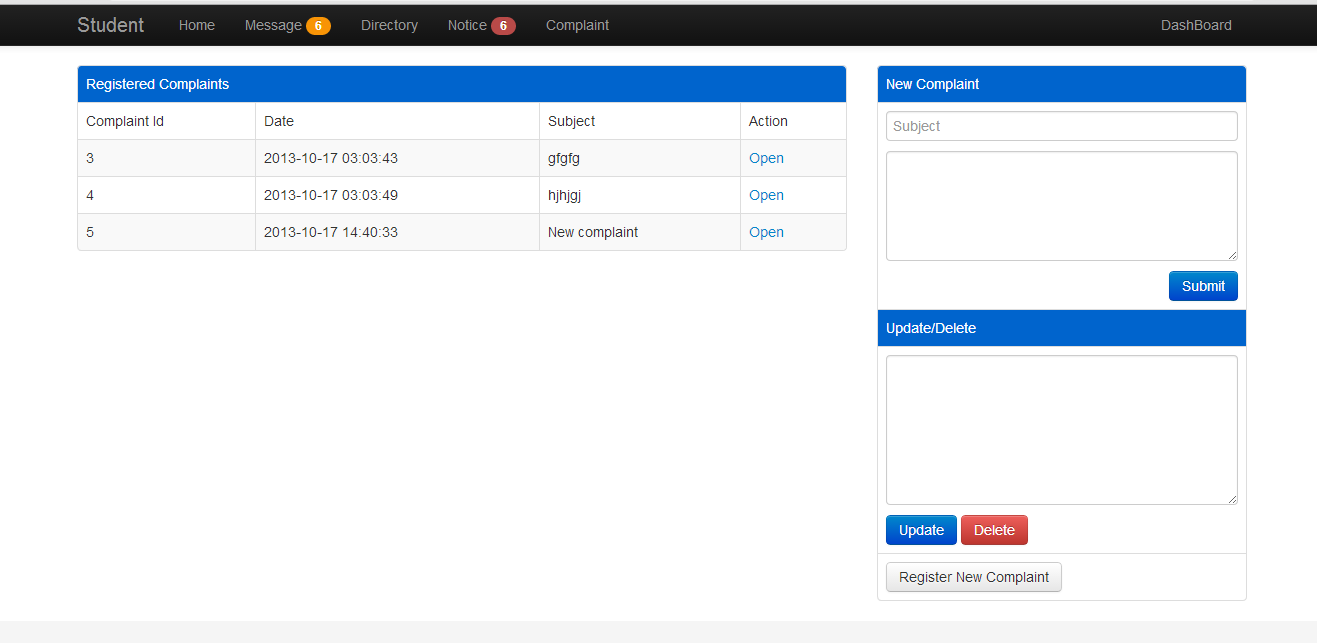
Selecting Cheque Module  


Cheque Module Loaded with list of operations:

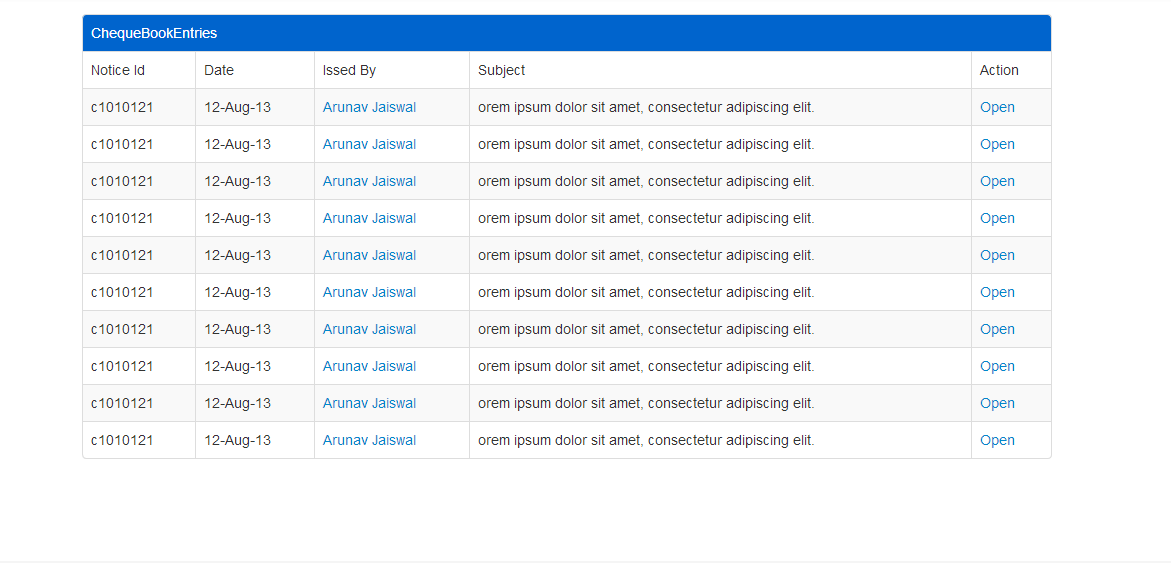


Adding A cheque  


Listing all cheques:  
  
File organisation in Project:  


**A complaint system built with the similar architecture (php based)**

**Fig:** A complaint panel showing the capabilities of complaint module



**Fig:** Table showing chequeBook Entries

**CHAPTER 9**

**FUTURE ENHANCEMENTS**

OpenSAP is a collection of different facilities and helper systems put together for ease of business application development. Motivated from the SAP systems it helps developers to skip working on the infrastructure of the application system and focus on the business goal first.  
The modular architecture of the OpenSAP helps more when it comes on tailoring the application for any specific requirement. Different modules are loosely coupled in the glue of connection components that help abstract the interactivity and most of the time the interaction will be between uniform interaction interfaces. Following are the key areas which will be enhanced in the future and forthcoming releases of OpenSAP:

1. **Module Development**: According to the standard business context different modules can be created which will be used to add up the capability of the system according to the needs.
2. **Enhancements in the Views** : Themes will be developed for the View Manager for view rendering which will help the user to work with more visual interest and will increase the overall productivity of him as well as the organisation.
3. **Web Interface for foreign computers**: Computers which are located away from the business organisation or the personal systems of the Administrators or other people should not be bloated with the OpenSAP desktop client if they want to work outside the working environment. Rather a lightweight web interface will be made which will load as a web application.
4. **OpenSAP SDK and IDE**: Making the modules and manually deploying can sometimes be cumbersome for new developers on the OpenSAP system so in next releases we are planning to prepare our own IDE for the smooth and systematic development of the modules and core enhancement.SDK will provide the API’s for various facilities and IDE will be a way to code and deploy the business code.

**CHAPTER 10**

**CONCLUSION**

OpenSAP starting as the bunch of different application development facilities will soon emerge as a complete application development stack and being Open Source it can prove to be a cheaper alternative for High priced SAP systems which are no doubt useful and successful but still the cost factor associated with them makes it difficult for Low budget organisations, students and researchers to involve in its development and use.

The project is made open-source with the hope that developers from the different parts of the world will take interest and will be able to collaborate in different parts of the OpenSAP without any issue of copyright and ownership.

The Open Source projects are always made keeping in mind the utility and the cost pain associated with the establishment of those utilities.With making this software an open source utility my aim in mind is the development and evolvement of the product without any glitch associated with the licensing -problems.Till now with completion of this report I have made the lengendry web framework which is working quite similar to what the application will be working in near future.Application does have some generic login system that works fine on different levels of rights.It have a module registry that registers different modules and the corresponding operations those modules provide.The developer who will develop the application on this platform will use the ready made Data transfer objects to store the information in the database and if needed he can make his own requirement based DTO by tailoring the existing once.Then He will have to prepare the views corresponding to the different action that follow in the application.The application framework removes the integration headache of developer by providing a tight mechanism that works in flow,the developer have to prepare just the fit-in components for the application workflow.

This type of application framework increases the developer’s productivity to many folds since the developer need not to worry about the integration and code-boilerplate and focus mainly on the business logic of the application.

**References:**(1) Java collection framework studied at tutorialspoint.com  
<http://www.tutorialspoint.com/java/java_list_interface.htm>

(2)HTML,CSS and JavaScript   
<http://www.w3schools.com/>  
(3)JavaBrains Tutorials for Hibernate and JSP:  
<http://javabrains.koushik.org/>

**BIBLIOGRAPHY**

The following books were very helpful during the completion of project:

* **Software Engineering**

By Roger S.Pressman.

* **The Complete Reference Java 7th Edition**

By Herbert Schildt

* **Java How to Program**  
   By Deitel and Deitel
* **SAM Teaches you SAP in 24 Hours 4th Edition**

By George W.Anderson