**Online Movies Ticket booking**

**Chapter 1**

**ABSTRACT**

Our project basically manages the ticket booking process of a multiplex, providing an interface to the user to book movie tickets in a more easy way. At the front end we have used Java and at the back end My SQL server. The project proceeds through a sequence of well-designed forms provided with validations to ensure consistency, reliability and most importantly correctness of information fed into the database.

**Overview of the Project**

Cinema-going is one of the most popular out-of-home cultural activities, affecting a serious of social, economic and cultural phenomena in modern societies. Cinemas are considered to be an integral part of cities and they contribute to the definition of a local geography and identity. They also contribute to the preservation of the collective memory, since they constitute a significant social and cultural practice linked to a specific place, which acts as a common reference or landmark for many individuals.

Through this project we present a comprehensive solution for ticket booking in multiplexes. Theater management system, an online ticket selling software that is easy to understand, easy to use and offers the simplicity of fast point-and-click service to the customers.

**This powerful software program is specifically designed for theater owners, to sell tickets online. This intuitive visual interface makes day-to-day aspects of selling, exchanging, refunding, and reporting fast and easy for both the user and administrators. Theater management controls all back-end functionalities like, movie details, ticket rate and show time, customer information and sales history saved in a database, etc. Theater admin manages the report details like counter wise report, daily, weekly, monthly report and movie report etc.**

**Objectives**

The objectives of this study are summarized below:

* The main objective of the project is to design and develop a user friendly efficient computerized Online Movies
* An accurate system without any data
* Secured data storage for Authority
* Secure the user ends data by providing each user’s own personal
* A flexible system which can maneuver the customer-staff relationship in an effective
* To provide better graphical user
* Computerization can be helpful as means of saving time &

**Chapter 2**

**Theoretical Background**

We have done a project on Online Movies and database management and transactions. This system is proposed to be an automate database management & transactions. This stores customer, member, payment, product, receipts, and products information. It also provides the facility of search & advanced search for searching  the records efficiently & immediately. This system provides data storing & report generation with graphical user interface (GUI).

**System Study**

It is always necessary to study and recognize the problems of existing system, which will help in finding out the requirements for the new system. System study helps in finding different alternatives for better solution.

**The project study basically deals with different operations:**

1: Data Gathering

2: Study of Existing System 3: Analyzing Problems

4: Studying various documents

5: Feasibility study for further improvements

**Following are the steps taken during the initial study:**

Initially, we collected all the information, which they wanted to store. Then we studied the working of the current system which is done manually. We noted the limitation of that system which motivated them to have new system. With the help of these documents we got basic ideas about the system as well as input output of the developed system.

The most important thing is to study system thoroughly. Here we are studying both existing system and proposed system so that advantages & disadvantages of both the

systems can be understood. The first task was identifying how system can be computerized. Some analysis and projections was done regarding changes to be made to the existing system. The new developed system for Online Moviesis simple without complexities.

**Existing System**

An Existing system refers to the system that is being followed till now. The Movies shop is working manually. The current system is time consuming and also it is very costly, because it involves a lot of paperwork. To manually handle the system was very difficult task. But now-a-days computerization made easy to work.

The following are the reasons why the current system should be computerized:

* To increase efficiency with reduced
* To reduce the burden of paper
* To save time management for recording details of each and every member and employee.
* To generate required reports

**Proposed System**

The online Movies  management system is user-friendly application. This automated system makes all functionality easier for both owners and customers. It is very simple in design and to implement. The system requirements are very low. System resources and the system will work in almost all configurations.

**It has the following objectives:**

**Enhancement:**

The main objective of Smart Movies  Management System is to enhance and upgrade the existing system by increasing its efficiency and effectiveness. The software improves the working methods by replacing the existing manual system with the computer- based system.

**Automation:**

The Smart Movies  Management System automates each and every activity of the manual system and increases its throughput. Thus the response time of the system is very less and it works very fast.

**Accuracy:**

The Smart Movies  Management System provides the uses a quick response with very accurate information regarding the users etc. Any details or system in an accurate manner, as and when required.

**User-Friendly:**

The software Smart Movies  Management System has a very user-friendly interface. Thus the users will feel very easy to work on it. The software provides accuracy along with a pleasant interface. Make the present manual system more interactive, speedy and user friendly.

**Availability:**

The transaction reports of the system can be retried as and when required. Thus, there is no delay in the availability of any information, whatever needed, can be captured very quickly and easily.

**Maintenance Cost:**

Reduce the cost of maintenance.

**Chapter 3**

**System Analysis & Design**

The way that is followed while carrying on with the development application is as follows:

**Defining a problem**

Defining a problem is one of the important activities of the project. The objective is to define precisely the business problem to be solved & thereby determined the scope of the new system. This phase consist of 2 main tasks. The 1st task within this activity is to review the organization needs that originally initiated the project. The 2nd task is to identify, at an abstract or general level, the expected capabilities of the new system. Thus, it helps us to define the goal to be achieved & the boundary of the system. A clear understanding of the problem will help us in building a better system & reduce the risk of project failure. It also specifies the resources that have to be made available to the project. Three important factors project goal, project bounds & the resource limits are sometimes called the project’s term of reference.

**Feasibility study**

The systems objectives outlined during the feasibility study serve as the basic from which the work of system design is initiated. Much of the activities involved at this stage is of technical nature requiring a certain degree of experience in designing systems, sound knowledge of computer related technology and through understanding of computers available in the market and the various facilities provided by the vendors. Nevertheless, a system cannot be designed in isolation without the active involvement of the user. The user has a vital role to play at this stage too. As we know that data collected during feasibility study wills we utilized systematically during the system design. It should, however be kept in mind that detailed study of the existing system is not necessarily over with the completion of the feasibility study. Depending on the plan of feasibility study, the level of detailed study will vary and the system design stage will also vary in the amount of investigation that still needs to be done.

This investigation is generally an urgent activity during the system. Sometimes, but rarely, this investigation may form a separate stage between feasibility study and computer system design. Designing a new system is a creative process, which calls  for logical as well as lateral thinking. The logical approach involves systematic moves towards the end product keeping in mind the capabilities of the personnel and the equipment at each decision making step. Lateral thought implies encompassing of ideas beyond the usual functions and equipment. This is to ensure that no efforts are being made to fit previous solutions into new situations.

The feasibility study proposes one or more conceptual solutions to the problem set for the project. The objective in assessing feasibility is to determine whether a development project has a reasonable chance of success. It helps us to determine the input & output of the system. The following are the criteria that are considered to confirm the project feasibility.

**The following feasibility study was undertaken for the proposed system:**

**Technical feasibility:**

At first it’s necessary to check that the proposed system is technically feasible or not & to determine the technology and skill necessary to carry out the project. If they are not available then find out the solution to obtain them. Hardware is already available in the University.

**Economic feasibility:**

While considering economic feasibility, it is checked in points like performance, information and outputs from the system. The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project , which will give best , return at the earliest. One of the factors , which affect the development of   a new system , is the cost it would require.

**Social feasibility:**

Although generally there is always resistance, initially to any change in the system is aimed at reliving the work load of the users to extent the system is going to facilitate user to perform

Operations like calculating salary amounts and deductions, generating reports with less possible errors. Thus there is no reason to make system socially unfeasible.

**Chapter 4**

**Modules & Features**

**Module**

There are two basic modules in this system as describe briefly in below

* **Administrative module:**This user is an admin type who has full rights on the system.

**Administrative Module**

This module includes storing and retrieving the details of the data.

* Create , Update, Manage, Delete User
* Creating OfferPlan
* ManageBilling
* Manage User Enquiry throughEmail
* Manage OwnerInfo

**Features**

There are many features in our system. Some salient and new features are:

* Login by FaceRecognition
* Phone number verification through SMS on signup
* Online PaymentGateway
* WebcamIntegration
* Activity Log ofUser’s

**Application Requirements**

**4.4.1 User Interface:**

* HTML has been used for developing the user layout for the
* Javaand JavaScript has been used for creating all the validations and client side scripting
* CSS has been used for designing the web page of the

**4.4.2 Application:**

* Client On Internet : Web Browser , Operating System(Any)
* Web Server :Apache
* Database :MySQL
* Markup Language: HTML, CSS
* Scripting Language : Java, Javascript, JQuery

**Chapter 5**

**System Implementation**

**5.1 Implementation Methodology**

We follow the MVC design pattern for developing our system. Model–view– controller (MVC) is a software design pattern for implementing user interfaces on computers. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user.

* **Model:**The model manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from thecontroller).
* **View:**The view manages the display of
* **Controller:**The controller interprets the mouse and keyboard inputs from the user, informing the model and/or the view to change as

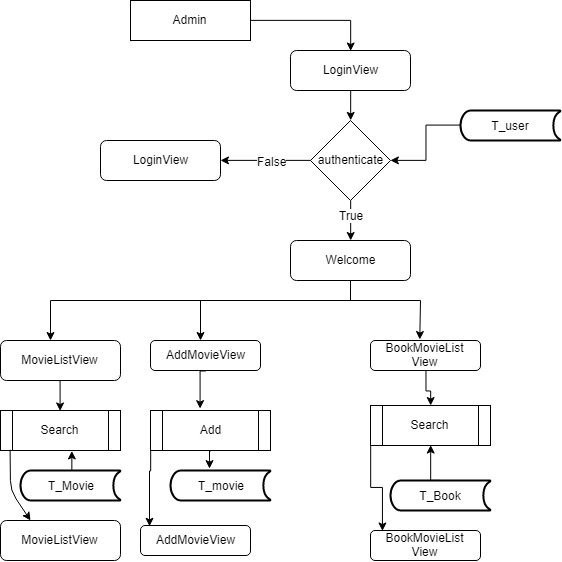
|  |
| --- |
|  |
|  |  |

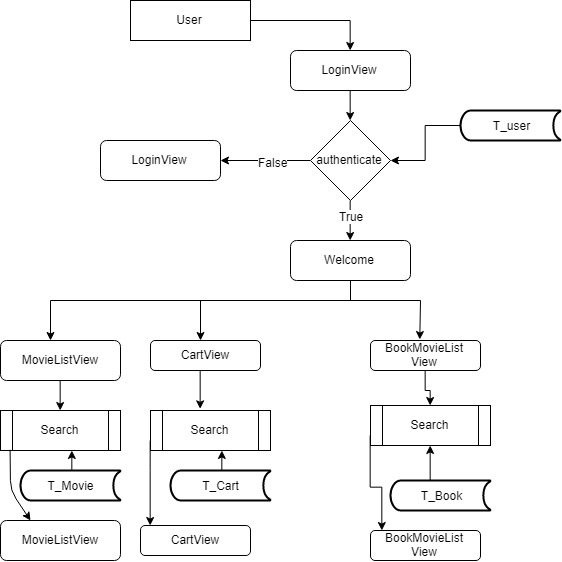
Fig. 5.1: Diagram of A typical collaboration of the MVC components.

**5.2 Data Flow Diagram**

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated.[2] DFDs can also be used for the visualization of data processing.

A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel.



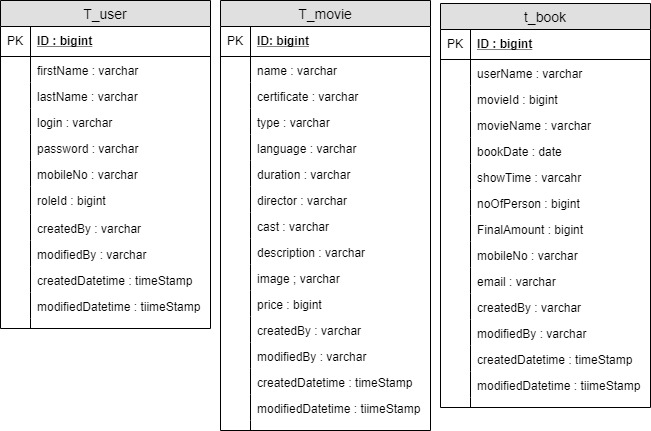
****

**5.3 Entity Relationship Diagram**

An entity-relationship diagram (ERD) is a graphical representation of an information system that shows the relationship between people, objects, places, concepts or events within that system. In software engineering an ER model is commonly formed to represent things that a business needs to remember in order to perform business processes. Consequently, the ER model becomes an abstract data model that defines a data or information structure that can be implemented in a database, typically a relational database.

An ER model is typically implemented as a database. In a simple relational database implementation, each row of a table represents one instance of an entity type, and each field in a table represents an attribute type. In a relational database a relationship between entities is implemented by storing the primary key of one entity as a pointer or "foreign key" in the table of another entity.

There is a tradition for ER/data models to be built at two or three levels of abstraction. Note that the conceptual-logical-physical hierarchy below is used in other kinds of specification, and is different from the three schema approach to software engineering.

****

**5.4 Normalization**

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form by removing duplicated data from the relation tables.

* Normalization is used for mainly twopurpose,
* Eliminating redundant (useless)

Ensuring data dependencies make sense i.e. data is logically stored.

Without Normalization, it becomes difficult to handle and update the database, without facing data loss. Insertion, Update and Deletion Anomalies are very frequent if Database is not normalized.

Normalization rule are divided into following normal form.

* First NormalForm
* Second NormalForm
* Third NormalForm
* BCNF

As per First Normal Form, no two Rows of data must contain repeating group of information i.e each set of column must have a unique value, such that multiple columns cannot be used to fetch the same row. Each table should be organized into rows, and each row should have a primary key that distinguishes it as unique.

As per **First Normal Form**, there are no repeating or duplicate fields in our database and each cell contains only a single value. For example:

As per First Normal Form, there are no repeating or duplicate fields in our database. Our system database does not have any column that has multiple repeating values. So our system database table is in first normal form.

As per Second Normal Form, Second normal form states that it should meet all the rules for 1NF and there must be no partial dependences of any of the columns on the primary key:

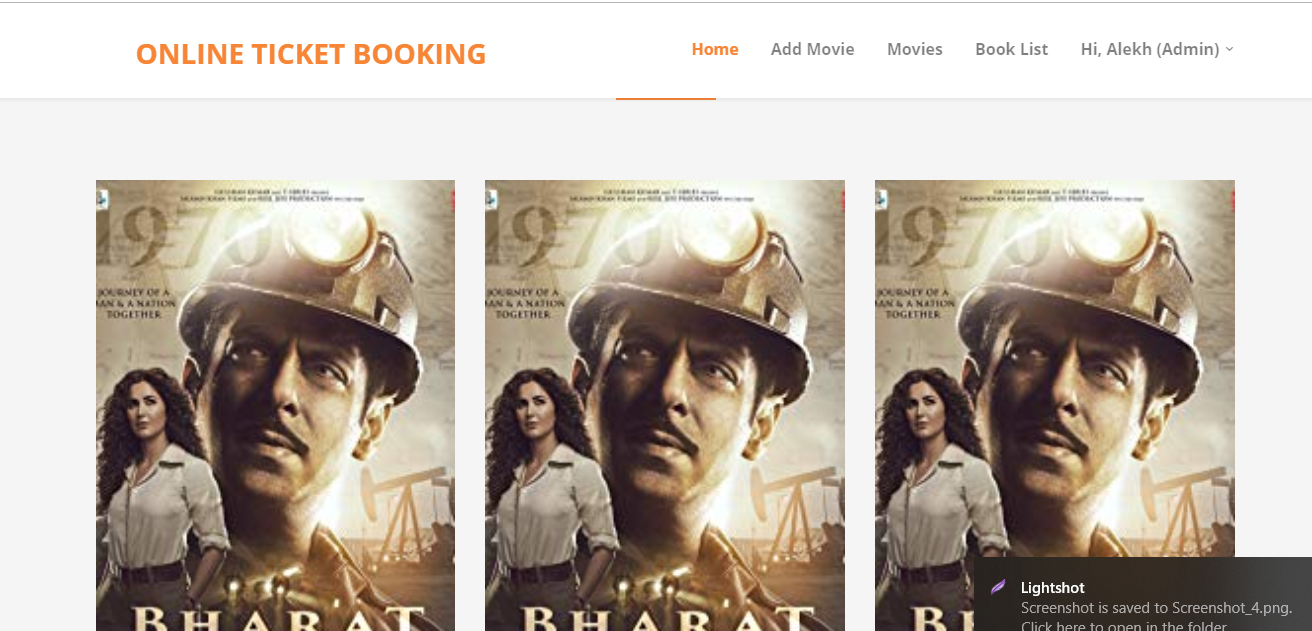
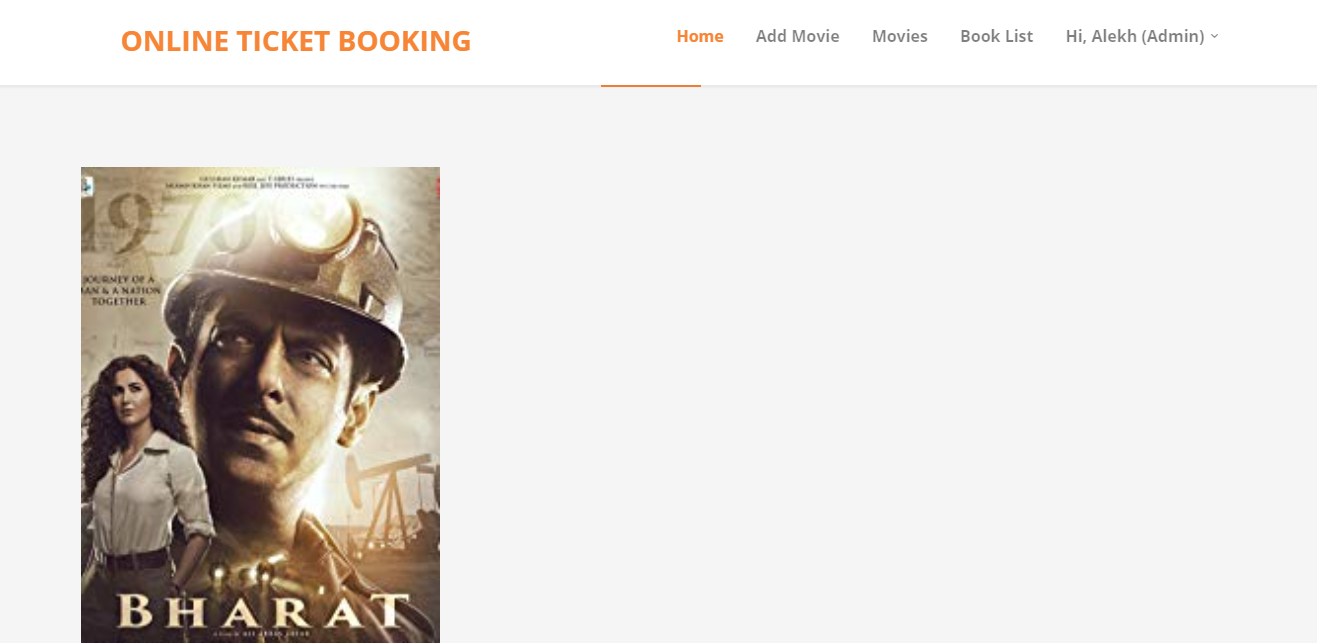
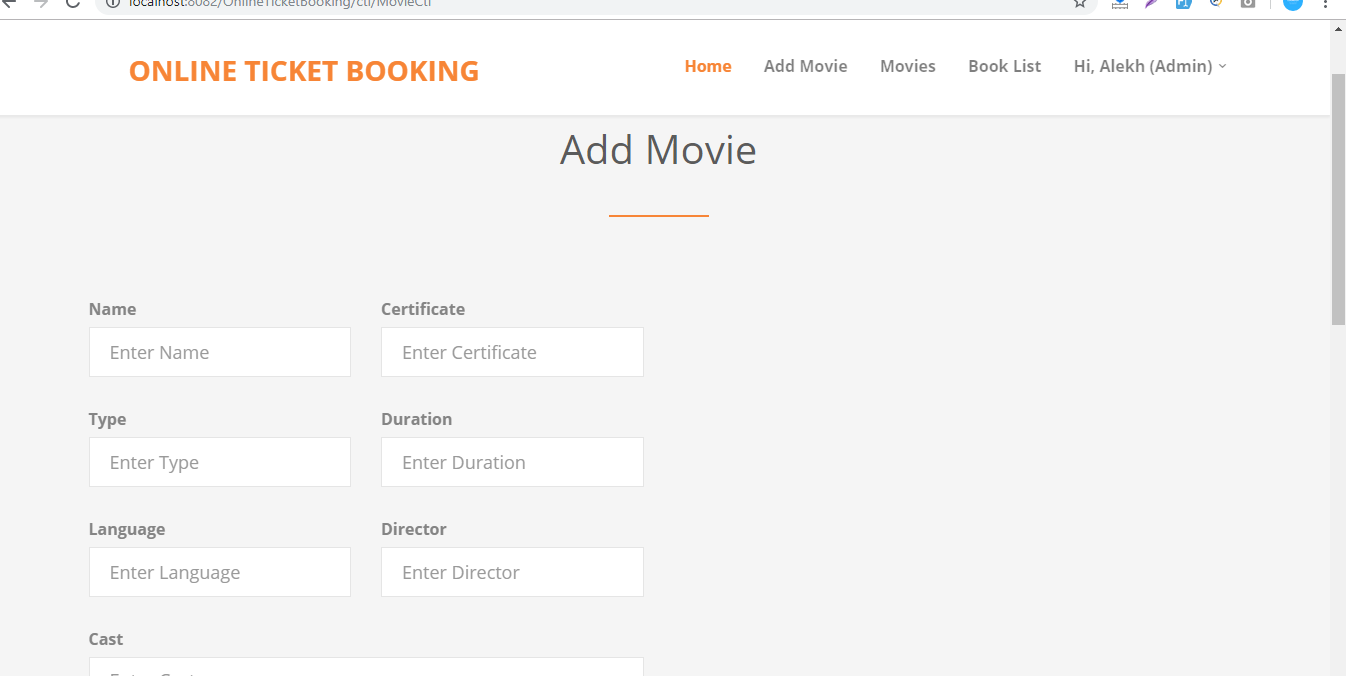
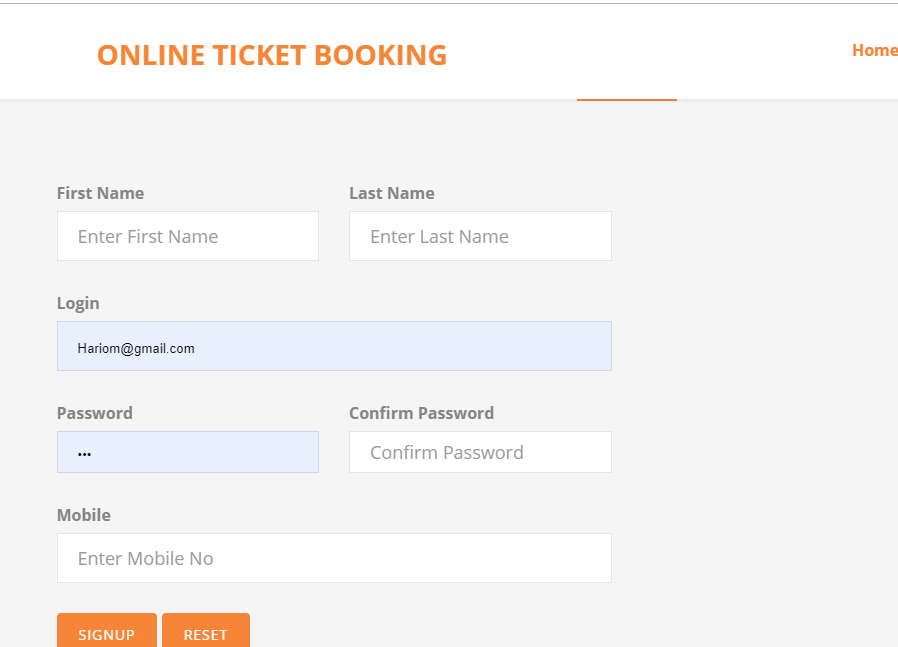
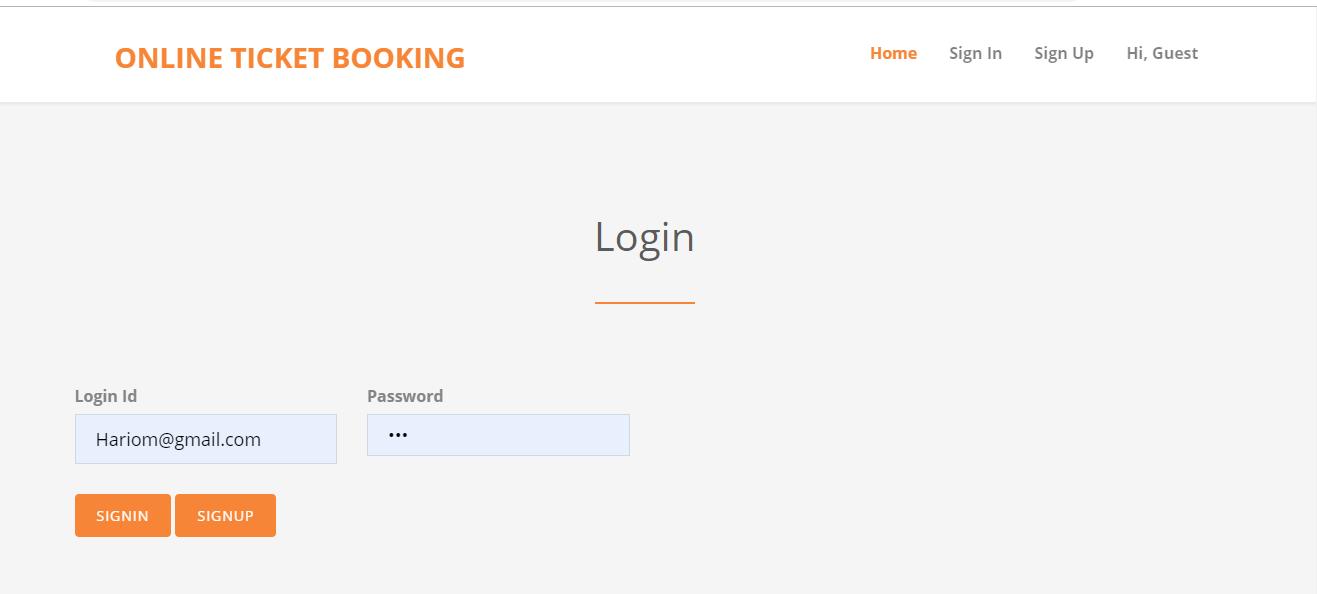
Let’s check with another table of our database:

From the above table we can say it meets all the rules for 1NF and there is no column that depends on the primary key. SO our database table is also in Second Normal Form.

In Third Normal Form, it should meet all the rules for 2NF and no non-key fields can depend upon another.

But from the above table we see, there is dependencies on two non-key value shift\_id and shift\_title . Shift title depends on shift id. So our system database table is not in 3NF.

**Screen shot**



**Source code**

**Beans**

**package** in.co.online.ticket.bean;

**import** java.sql.Timestamp;

/\*\*

\* BaseBean JavaBean encapsulates Generic attributes

\*

\* **@author** Navigable Set

\* **@version** 1.0

\* **@Copyright** (c) Navigable Set

\*

\*/

**public** **abstract** **class** BaseBean **implements** DropdownListBean, Comparable<BaseBean> {

/\*\*

\* Generic Attribute Id For All Child

\*/

**protected** **long** id;

/\*\*

\* Generic Attribute Created BY For All Child

\*/

**protected** String createdBy;

/\*\*

\* Generic Attribute Modified BY For All Child

\*/

**protected** String modifiedBy;

/\*\*

\* Generic Attribute Created Date and TIme For All Child

\*/

**protected** Timestamp createdDatetime;

/\*\*

\* Generic Attribute Modified Date And TIme For All Child

\*/

**protected** Timestamp modifiedDatetime;

/\*\*

\* accessor

\*/

/\*\*

\* **@return** ID of Child

\*/

**public** **long** getId() {

**return** id;

}

/\*\*

\* **@param** Id

\* To set Child ID

\*/

**public** **void** setId(**long** id) {

**this**.id = id;

}

/\*\*

\* **@return** Created By Child

\*/

**public** String getCreatedBy() {

**return** createdBy;

}

/\*\*

\* **@param** CreatedBy

\* To set Child Creatd By

\*/

**public** **void** setCreatedBy(String createdBy) {

**this**.createdBy = createdBy;

}

/\*\*

\* **@return** Modified By Child

\*/

**public** String getModifiedBy() {

**return** modifiedBy;

}

/\*\*

\* **@param** Modified

\* by To set Child ModifiedBY

\*/

**public** **void** setModifiedBy(String modifiedBy) {

**this**.modifiedBy = modifiedBy;

}

/\*\*

\* **@return** Created Date And TIme of Child

\*/

**public** Timestamp getCreatedDatetime() {

**return** createdDatetime;

}

/\*\*

\* **@param** Created

\* Date and Time To set Child Created dATE aND tiME

\*/

**public** **void** setCreatedDatetime(Timestamp createdDatetime) {

**this**.createdDatetime = createdDatetime;

}

/\*\*

\* **@return** Modified Date And Time of Child

\*/

**public** Timestamp getModifiedDatetime() {

**return** modifiedDatetime;

}

/\*\*

\* **@param** Modified

\* By To set Child ModiFied By

\*/

**public** **void** setModifiedDatetime(Timestamp modifiedDatetime) {

**this**.modifiedDatetime = modifiedDatetime;

}

**public** **int** compareTo(BaseBean next) {

// **TODO** Auto-generated method stub

**return** getValue().compareTo(next.getValue());

}

}

Model

**package** in.co.online.ticket.bean;

**import** java.sql.Timestamp;

/\*\*

\* BaseBean JavaBean encapsulates Generic attributes

\*

\* **@author** Navigable Set

\* **@version** 1.0

\* **@Copyright** (c) Navigable Set

\*

\*/

**public** **abstract** **class** BaseBean **implements** DropdownListBean, Comparable<BaseBean> {

/\*\*

\* Generic Attribute Id For All Child

\*/

**protected** **long** id;

/\*\*

\* Generic Attribute Created BY For All Child

\*/

**protected** String createdBy;

/\*\*

\* Generic Attribute Modified BY For All Child

\*/

**protected** String modifiedBy;

/\*\*

\* Generic Attribute Created Date and TIme For All Child

\*/

**protected** Timestamp createdDatetime;

/\*\*

\* Generic Attribute Modified Date And TIme For All Child

\*/

**protected** Timestamp modifiedDatetime;

/\*\*

\* accessor

\*/

/\*\*

\* **@return** ID of Child

\*/

**public** **long** getId() {

**return** id;

}

/\*\*

\* **@param** Id

\* To set Child ID

\*/

**public** **void** setId(**long** id) {

**this**.id = id;

}

/\*\*

\* **@return** Created By Child

\*/

**public** String getCreatedBy() {

**return** createdBy;

}

/\*\*

\* **@param** CreatedBy

\* To set Child Creatd By

\*/

**public** **void** setCreatedBy(String createdBy) {

**this**.createdBy = createdBy;

}

/\*\*

\* **@return** Modified By Child

\*/

**public** String getModifiedBy() {

**return** modifiedBy;

}

/\*\*

\* **@param** Modified

\* by To set Child ModifiedBY

\*/

**public** **void** setModifiedBy(String modifiedBy) {

**this**.modifiedBy = modifiedBy;

}

/\*\*

\* **@return** Created Date And TIme of Child

\*/

**public** Timestamp getCreatedDatetime() {

**return** createdDatetime;

}

/\*\*

\* **@param** Created

\* Date and Time To set Child Created dATE aND tiME

\*/

**public** **void** setCreatedDatetime(Timestamp createdDatetime) {

**this**.createdDatetime = createdDatetime;

}

/\*\*

\* **@return** Modified Date And Time of Child

\*/

**public** Timestamp getModifiedDatetime() {

**return** modifiedDatetime;

}

/\*\*

\* **@param** Modified

\* By To set Child ModiFied By

\*/

**public** **void** setModifiedDatetime(Timestamp modifiedDatetime) {

**this**.modifiedDatetime = modifiedDatetime;

}

**public** **int** compareTo(BaseBean next) {

// **TODO** Auto-generated method stub

**return** getValue().compareTo(next.getValue());

}

}

Controller

package in.co.online.ticket.controller;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.apache.log4j.Logger;

import in.co.online.ticket.bean.BaseBean;

import in.co.online.ticket.bean.RoleBean;

import in.co.online.ticket.bean.UserBean;

import in.co.online.ticket.exception.ApplicationException;

import in.co.online.ticket.exception.DuplicateRecordException;

import in.co.online.ticket.model.UserModel;

import in.co.online.ticket.util.DataUtility;

import in.co.online.ticket.util.DataValidator;

import in.co.online.ticket.util.PropertyReader;

import in.co.online.ticket.util.ServletUtility;

/\*\*

\* Servlet implementation class UserRegistrationCtl

\*/

/\*\*

\* UserRegistration functionality Controller. Performs operation for Validate and add a User

\* As Student Role

\*

\* @author NAvigable set

\* @version 1.0

\* @Copyright (c) Navigable Set

\*

\*/

@WebServlet(name = "UserRegistrationCtl", urlPatterns = { "/UserRegistrationCtl" })

public class UserRegistrationCtl extends BaseCtl {

public static final String OP\_SIGN\_UP = "SignUp";

private static Logger log = Logger.getLogger(UserRegistrationCtl.class);

/\*\*

\* Validate input Data Entered By User

\*

\* @param request

\* @return

\*/

@Override

protected boolean validate(HttpServletRequest request) {

log.debug("UserRegistrationCtl Method validate Started");

boolean pass = true;

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

if (DataValidator.isNull(request.getParameter("firstName"))) {

request.setAttribute("firstName",

PropertyReader.getValue("error.require", "First Name"));

pass = false;

}

if (DataValidator.isNull(request.getParameter("lastName"))) {

request.setAttribute("lastName",

PropertyReader.getValue("error.require", "Last Name"));

pass = false;

}

if (DataValidator.isNull(login)) {

request.setAttribute("login",

PropertyReader.getValue("error.require", "Login Id"));

pass = false;

} else if (!DataValidator.isEmail(request.getParameter("login"))) {

request.setAttribute("login",

PropertyReader.getValue("error.email", "Login"));

pass = false;

}

if (DataValidator.isNull(request.getParameter("confirmPassword"))) {

request.setAttribute("confirmPassword", PropertyReader.getValue(

"error.require", "Confirm Password"));

pass = false;

}

if (DataValidator.isNull(request.getParameter("password"))) {

request.setAttribute("password",

PropertyReader.getValue("error.require", "Password"));

pass = false;

} else if (!DataValidator.isPassword(request.getParameter("password"))) {

request.setAttribute("password",

PropertyReader.getValue("error.password", "Password"));

return false;

}else if (!DataValidator.isPassword(request.getParameter("password"))) {

request.setAttribute("password",

PropertyReader.getValue("error.password", "Password"));

return false;

}

if (!request.getParameter("password").equals(

request.getParameter("confirmPassword"))

&& !"".equals(request.getParameter("confirmPassword"))) {

/\*ServletUtility.setErrorMessage("Confirm Password did not match",

request);\*/

request.setAttribute("confirmPassword", PropertyReader.getValue("error.confirmPassword","Confirm Password"));

pass = false;

}

if (DataValidator.isNull(request.getParameter("mobile"))) {

request.setAttribute("mobile", PropertyReader.getValue("error.require","Mobile No"));

pass = false;

}else if(!DataValidator.isPhoneNo(request.getParameter("mobile"))){

request.setAttribute("mobile", PropertyReader.getValue("error.invalid","Mobile No"));

pass=false;

}

log.debug("UserRegistrationCtl Method validate Ended");

return pass;

}

/\*\*

\* Populates bean object from request parameters

\*

\* @param request

\* @return

\*/

@Override

protected BaseBean populateBean(HttpServletRequest request) {

log.debug("UserRegistrationCtl Method populatebean Started");

UserBean bean = new UserBean();

bean.setId(DataUtility.getLong(request.getParameter("id")));

bean.setRoleId(RoleBean.CUSTOMER);

bean.setFirstName(DataUtility.getString(request

.getParameter("firstName")));

bean.setLastName(DataUtility.getString(request.getParameter("lastName")));

bean.setLogin(DataUtility.getString(request.getParameter("login")));

bean.setPassword(DataUtility.getString(request.getParameter("password")));

bean.setConfirmPassword(DataUtility.getString(request

.getParameter("confirmPassword")));

bean.setMobileNo(DataUtility.getString(request.getParameter("mobile")));

populateDTO(bean, request);

log.debug("UserRegistrationCtl Method populatebean Ended");

return bean;

}

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public UserRegistrationCtl() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* Contains display logic

\*/

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

log.debug("UserRegistrationCtl Method doGet Started");

ServletUtility.forward(getView(), request, response);

}

/\*\*

\* Contains submit logic

\*/

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

System.out.println("in post method");

log.debug("UserRegistrationCtl Method doPost Started");

String op = DataUtility.getString(request.getParameter("operation"));

// get model

UserModel model = new UserModel();

long id = DataUtility.getLong(request.getParameter("id"));

if (OP\_SIGN\_UP.equalsIgnoreCase(op)) {

UserBean bean = (UserBean) populateBean(request);

try {

// System.out.println("in try sign up");

long pk = model.registerUser(bean);

//System.out.println("register");

bean.setId(pk);

request.getSession().setAttribute("UserBean", bean);

ServletUtility.setSuccessMessage("User Successfully Registered", request);

ServletUtility.forward(OTBView.USER\_REGISTRATION\_VIEW, request, response);

return;

} catch (DuplicateRecordException e) {

log.error(e);

ServletUtility.setBean(bean, request);

ServletUtility.setErrorMessage("Login id already exists",

request);

ServletUtility.forward(getView(), request, response);

} catch (ApplicationException e) {

ServletUtility.handleException(e, request, response);

e.printStackTrace();

return;

}

}else if (OP\_RESET.equalsIgnoreCase(op)) {

ServletUtility.redirect(OTBView.USER\_REGISTRATION\_CTL, request, response);

return;

}

log.debug("UserRegistrationCtl Method doPost Ended");

}

/\*\*

\* Returns the VIEW page of this Controller

\*

\* @return

\*/

@Override

protected String getView() {

return OTBView.USER\_REGISTRATION\_VIEW;

}

}

**Testing**

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Also to assess the feature of A software item. Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words software testing is a verification and validation process.

**Verification**

Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to.

**Validation**

Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements.

Testing goes side by side with the implementation that is aimed at ensuring that the system works accurately and efficiently before the live operation is performed .The common view of testing held by the user is process of executing a program with explicit intention of handling errors. The application which has been developed has to be tested to prove its validity. Testing is considered to be the least creative phase of the whole cycle of system design. In the real sense it is the phase, which helps to  bring out the creativity of the other phases, and makes it shine.

The Smart Movies  Management System was tested using the following two techniques of application testing:

**Unit Testing:**

* In the line of strategy the entire individuals function and modules were put to test independently
* By following this strategy all the errors in coding were identified and corrected.
* This method was applied in combination with the White Box and Black Box testing

* Technique to find errors in each
* The effort of specific combination of data on system operation was
* The following were the testes carried out for Graphical User Interface(GUI).
* It was seen that the pages opens properly based on related menu based commands.
* It was tested whether all relevant menus, buttons, icons and other controls are available and properly

**System Testing**

We use this testing method. System testing is the testing to ensure that by putting the software in different environments (e.g., Operating Systems) it still works. System testing is done with full system implementation and environment. It falls under the class of black box testing.

**Performance Testing**

Performance testing is the testing to assess the speed and effectiveness of the system and to make sure it is generating results within a specified time as in performance requirements. It falls under the class of black box testing.

**Multi-user System Testing**

Database Locking Schemes: Whenever more than one person is accessing a record/s some type of process must be used to prevent the outer users from attempting to update the same record at the same time. This process is a locking scheme. In its simplest form, a locking scheme allows only one user at a time to update information in the database.

**Chapter 7**

**7.1 Future Work**

The project has been developed in a very short period of time and all efforts have been taken so that this project is very efficient in its execution there still exists some scope of improvement in our project. The following lists some of the enhancement that can be added incorporate into the project.

Application of the project can be done more attractively. Database management and all maintenance module can be updated which helps the administrator. More security measures can be taken.

There are also few features which can be integrated with this system to make it more flexible. Below list shows the future points to be consider:

* Real-time Chat BOT option for members and trainer, so that members can directly enquiry theirs trainer on any time through the Chat
* Automated Fitness suggestion by enquiring the condition of the
* Real time Claim Processing Bot.
* Video conversation option for trainers and
* Online payment through face
* Barcode generation for membership card and using this, members can take entry toMovies .
* Finger print matching for taking entry toMovies .

.

**Conclusions**

The “**Movies Ticket booking MANAGEMENT SYSTEM**” is successfully designed and developed to fulfilling the necessary requirements, as identified in the requirements analysis phase, such as the system is very much user friendly, form level validation and field level validation are performing very efficiently. The old manual system was suffering from a series of drawbacks. The present project has been developed to meet the aspirations indicated in the modern age.