**Online Movie Review and Rating System**

Introduction

Movie Review and Rating Portal in Java is a web application that running over the browsers and helps to users to find review and rating of the movies at same time user can review and existing movies.

In this project, Admin will be able to add/remove movies to the list whereas, a user would be able to watch and share their review for a particular movie in the form of a comment. Also, it will allow users to rate a movie based on their experience.

****The whole project is developed using  Servlet and JSP. At the front end, we have used HTML, CSS, and Bootstrap. At the data access layer, we have used JDBC API. The Database used here is MYSQL. The whole project is following the MVC(Model View & Controller) design pattern.****

Objective

The main objective is to build a secured, robust Online Movie review and rating Application. It maintains the record of the Movies, users, review and raring results and, etc so that it would be easy to access at any time 24\*7.

Each module is built independent of each other. This application is developed for showcasing the popularity of the movie for competition purposes. Admin is the main user of the website who will Add the new released movie and will manage every activity like adding Users or manage users and movies.

This application will help to review and rating their favorite movie Gathering information related to any topic let it be the best places in the world, the best gadgets available, or the best movie to watch online, one would always look up to get it through social media platforms or any sources available online. Movies indeed dominate the world of entertainment and, to choose which one to watch from the list of millions is confusing. Out of many, one way to tackle this confusion is to go by reviews and ratings available online for a movie.

Methodologies

There are two main users of this application one is the Admin and another one is the user let us see the responsibility.

#### ****Admin****

* ****Manage Admin User****: It will allow Admin to manage the portal.
* ****Manage Register User****: In this module, Admin can manage each and every user registered on the portal.
* ****Manage Movies Category****: This module will allow Admin to categorize movies based on genres, language, or ratings.
* ****Manage Movies****: Admin can add or remove movies in this module.
* ****Manage Movie Ratings****: In this module, Admin can manage movies based on their ratings.
* ****Manage User Reviews****: Here Admin would be able to manage reviews given by users for movies.

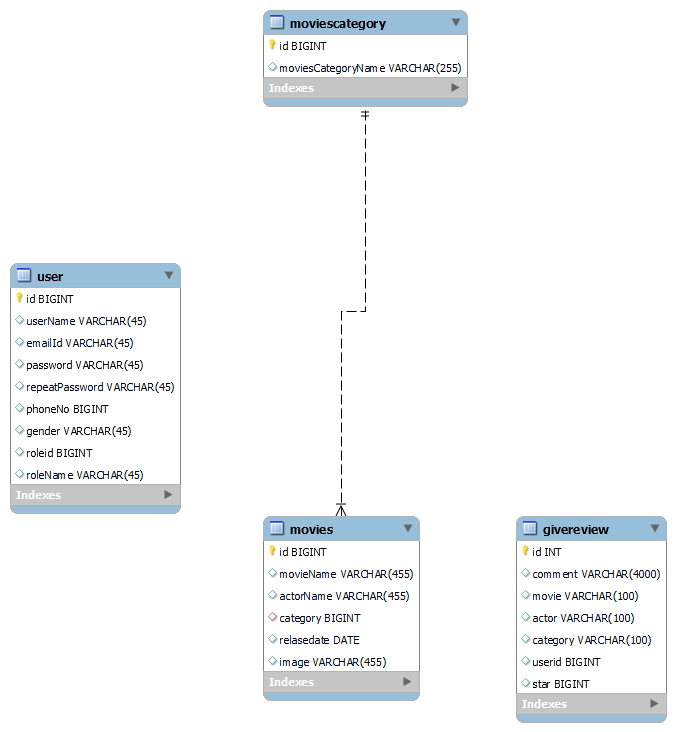
#### ****User****

* ****Request for Registration:****In this module, Users can register themselves on the portal.
* ****Login to the Portal:****Registered Users can log in to the portal with this module.
* ****Manage Profile:****In this module, every registered User can manage their profiles by editing names and other details.
* ****View Reviews:****In this module, a user would be able to view reviews for different movies.
* ****Give Reviews:****This module will allow the user to give their reviews based on their movie experience.
* ****Give Ratings:****Similarly, in this module, a user can rate a movie depending on their experience.

**Note: The login, profile section and registrationare also present in the system.**

**System Analysis and Design**

**E R DIAGRAM**

****

**Feasibility:**

This project will be developed on computer, so first check whether the technology is technically available or not. Now a day’s computer interaction with any job becomes common for any kind of job or work.

And because of increasing usage of Computer, Computer is also available with a variety of hardware. Vendors can fulfill any type of hardware requirement. The whole project is developed by some special tools or by using languages and databases, which are also available in a variety.

Preliminary investigation of a system examines the feasibility of a system that is useful to an organization. It is the first phase of system development.

The main objective of this phase is to identify the current deficiencies in the user’s environment and to determine which existing problem are going to be solve in proposed system and also which new function needs to be added in proposed system.

An important outcome of such preliminary investigation is to determine whether the system that will meet all needed requirements.

Thus, three tests are carried out on the system namely operation, technical and economical.

Any project is beneficial if and only satisfies the organization requirement. For any new system setup, it only meets to be communicated and work the other supporting system.

The new system meets all existing operations since it provides right information at a right time to the right user. A Leigh man can easily operate with the system.

Technical Feasibility examines whether the technology needed is available and if it is available then it feasible to carry out all project activities.

The technical needs of a system include:

* The facility to produce outputs in a given time.
* Ability to process large number of transaction at a particular speed.
* Giving response to users under certain conditions.

The technology needed for our system is mainly:

* Latest version of browsers.
* Any operating system.

These technologies are available which helps to carry out the system efficiently.

Economical feasibility of a system examines whether the finance is available for implementing the new system and whether the money spent is recoverable the satisfaction.

The cost involves is in designing and developing a good investment for the organization.

Thus, hardware requirements used for proposed system are very standard. Moreover, by making use of proposed system to carry out the work speedily will increase and also saves the valuable time of an organization.

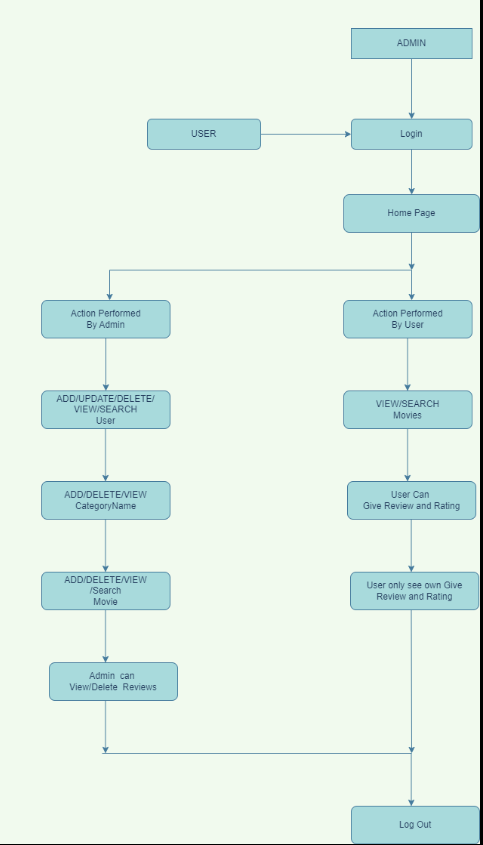
In the proposed system the finance is highly required for the installation of the software’s which can also be recovered by implementing a better system.



**Chapter-3**

**Design**

**System Flow Chart:**

****

**Data dictionary**

**Data validation:**

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been integrated in the system in almost every area where there is a possibility for the user to commit errors. The system will not recognize invalid data.

Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been integrated where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with pop up menus.

**Different Type Of validation:**

* Data type validation;
* Range and constraint validation;
* Code and Cross-reference validation; andStructured validation

**Coding**

**Implementation and Testing:**

**Black-Box Testing**:

Black Box Testing, also known as Behavioural Testing, is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

This can be following way:

* Input interfacing
* Processing
* Output interfacing



This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories:

* Incorrect or missing functions
* Interface errors
* Errors in data structures or external database access
* Behaviour or performance errors
* Initialization and termination errors.

**White-Box Testing:**

White Box Testing ,also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.

The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential.

White box testing is testing beyond the user interface and into the nitty-gritty of a system.

This method is named so because the software program, in the eyes of the tester, is like a white/ transparent box; inside which one clearly sees.

**Limitations and Future Application of the Project**

**Futures Enhancement:**

* In future we can expand this project on the cloud.
* In future, we can allow admin to allow add prize module for the winner.

**Limitation :**

In this, we don’t have a facility to block users from the system.

**Source code**

**Beans**

**package** in.co.movie.review.bean;

**import** java.util.Date;

**public** **class** MoviesBean **extends** BaseBean{

**private** String movieName;

**private** String ActorName;

**private** Date Realeasedate;

**private** String image;

**public** String getImage() {

**return** image;

}

**public** **void** setImage(String image) {

**this**.image = image;

}

**private** String category;

**private** Long categoryId;

**public** Long getCategoryId() {

**return** categoryId;

}

**public** **void** setCategoryId(Long categoryId) {

**this**.categoryId = categoryId;

}

**public** String getMovieName() {

**return** movieName;

}

**public** **void** setMovieName(String movieName) {

**this**.movieName = movieName;

}

**public** String getActorName() {

**return** ActorName;

}

**public** **void** setActorName(String actorName) {

ActorName = actorName;

}

**public** Date getRealeasedate() {

**return** Realeasedate;

}

**public** **void** setRealeasedate(Date realeasedate) {

Realeasedate = realeasedate;

}

**public** String getCategory() {

**return** category;

}

**public** **void** setCategory(String category) {

**this**.category = category;

}

**public** String getKey() {

**return** id+"";

}

**public** String getValue() {

**return** category+" "+categoryId;

}

}

**Controller**

package in.co.movie.review.Controller;

import java.io.IOException;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import in.co.movie.review.Model.MoviesModel;

import in.co.movie.review.Utility.DataUtility;

import in.co.movie.review.Utility.ServletUtility;

import in.co.movie.review.bean.BaseBean;

import in.co.movie.review.bean.MoviesBean;

@WebServlet(name = "MoviesListCtl" ,urlPatterns = "/MoviesList")

public class MoviesListCtl extends BaseCtl {

private static final long serialVersionUID = 1L;

public static final String OP\_SEARCH = "search";

public MoviesListCtl() {

super();

}

protected BaseBean populateBean(HttpServletRequest request) {

MoviesBean bean = new MoviesBean();

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

bean.setCategoryId(DataUtility.getLong(request.getParameter("category")));

bean.setMovieName(DataUtility.getString(request.getParameter("MovieName")));

bean.setActorName(DataUtility.getString(request.getParameter("actorName")));

bean.setRealeasedate(DataUtility.getDate(request.getParameter("Releasedate")));

populateDTO(bean, request);

return bean;

}

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

MoviesModel model = new MoviesModel();

MoviesBean bean = null;

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

if (id > 0) {

model.delete(id);

ServletUtility.setSuccessMessage("Movies Deleted Successfully !!", request);

}

List list = null;

try {

System.out.println("in do get");

list = model.list();

} catch (Exception e) {

e.printStackTrace();

}

if (list == null && list.size() == 0) {

ServletUtility.setErrorMessage("No record found", request);

}

ServletUtility.setList(list, request);

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

}

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

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

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

MoviesModel model = new MoviesModel();

MoviesBean bean = new MoviesBean();

bean = (MoviesBean) populateBean(request);

List list = null;

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

ServletUtility.redirect(MRView.MOVIES\_LIST\_CTL, request, response);

return;

}

if (OP\_SEARCH.equalsIgnoreCase(op)) {

try {

System.out.println("In Post");

list = model.search(bean);

ServletUtility.setList(list, request);

ServletUtility.setbean(bean, request);

} catch (Exception e) {

e.printStackTrace();

}

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

}

}

@Override

protected String getView() {

return MRView.MOVIES\_LIST\_VIEW;

}

}

**Model:**

**package in.co.movie.review.Model;**

**import java.sql.Connection;**

**import java.sql.Date;**

**import java.sql.PreparedStatement;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import java.util.ArrayList;**

**import java.util.List;**

**import in.co.movie.review.Exception.ApplicationException;**

**import in.co.movie.review.Exception.DuplicateRecordException;**

**import in.co.movie.review.Utility.JDBCDataSource;**

**import in.co.movie.review.bean.GiveReview;**

**import in.co.movie.review.bean.MoviesBean;**

**import in.co.movie.review.bean.MoviesCategoryBean;**

**import in.co.movie.review.bean.UserBean;**

**public class MoviesModel {**

**public Integer nextPk() throws Exception {**

**Connection conn = null;**

**int pk = 0;**

**try {**

**conn = JDBCDataSource.getConnection();**

**PreparedStatement ps = conn.prepareStatement("SELECT MAX(ID) FROM movies");**

**ResultSet rs = ps.executeQuery();**

**while (rs.next()) {**

**pk = rs.getInt(1);**

**}**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**return pk + 1;**

**}**

**public MoviesBean findByMovieName(String movieName) throws Exception {**

**MoviesBean bean = null;**

**Connection conn = null;**

**try {**

**conn = JDBCDataSource.getConnection();**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**try {**

**PreparedStatement ps = conn.prepareStatement("SELECT \* FROM movies WHERE movieName=?");**

**ps.setString(1, movieName);**

**ResultSet rs = ps.executeQuery();**

**while (rs.next()) {**

**bean = new MoviesBean();**

**bean.setId(rs.getLong(1));**

**bean.setMovieName(rs.getString(2));**

**bean.setActorName(rs.getString(3));**

**bean.setCategory(rs.getString(4));**

**bean.setRealeasedate(rs.getDate(5));**

**bean.setImage(rs.getString(6));**

**}**

**rs.close();**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**return bean;**

**}**

**public long add(MoviesBean bean) throws Exception {**

**System.out.println("in add method");**

**Connection conn = null;**

**int pk = 0;**

**MoviesModel model = new MoviesModel();**

**MoviesBean moviebean = findByMovieName(bean.getMovieName());**

**if (moviebean != null) {**

**throw new DuplicateRecordException("Movie is already exite");**

**}**

**try {**

**conn = JDBCDataSource.getConnection();**

**pk = nextPk();**

**conn.setAutoCommit(false);**

**PreparedStatement ps = conn.prepareStatement("INSERT INTO movies VALUES(?,?,?,?,?,?)");**

**ps.setLong(1, pk);**

**ps.setString(2, bean.getMovieName());**

**ps.setString(3, bean.getActorName());**

**ps.setLong(4, bean.getCategoryId());**

**ps.setDate(5, new Date(bean.getRealeasedate().getTime()));**

**ps.setString(6, bean.getImage());**

**ps.executeUpdate();**

**conn.commit();**

**ps.close();**

**} catch (Exception e) {**

**e.printStackTrace();**

**try {**

**conn.rollback();**

**} catch (Exception e2) {**

**e.printStackTrace();**

**throw new ApplicationException("Exception : add rollback exception " + e.getMessage());**

**}**

**} finally {**

**JDBCDataSource.closeconnection(conn);**

**}**

**return pk;**

**}**

**public List list() throws Exception {**

**System.out.println("in model list");**

**ArrayList list = new ArrayList();**

**try {**

**Connection conn = null;**

**conn = JDBCDataSource.getConnection();**

**PreparedStatement pstmt = conn.prepareStatement("SELECT movies.id,movieName,actorName,moviescategory.moviesCategoryName,relasedate,image\r\n" +**

**"FROM movies\r\n" +**

**"INNER JOIN moviescategory ON movies.category = moviescategory.id");**

**ResultSet rs = pstmt.executeQuery();**

**while (rs.next()) {**

**MoviesBean bean = new MoviesBean();**

**bean.setId(rs.getLong(1));**

**bean.setMovieName(rs.getString(2));**

**bean.setActorName(rs.getString(3));**

**bean.setCategory(rs.getString(4));**

**bean.setRealeasedate(rs.getDate(5));**

**bean.setImage(rs.getString(6));**

**list.add(bean);**

**}**

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**return list;**

**}**

**public static long delete(long id) {**

**int i = 0;**

**try {**

**Connection conn = JDBCDataSource.getConnection();**

**PreparedStatement stmt = conn.prepareStatement("DELETE from movies where id=?");**

**stmt.setLong(1, id);**

**i = stmt.executeUpdate();**

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**return i;**

**}**

**public List search(MoviesBean bean) throws Exception {**

**StringBuffer sql = new StringBuffer("SELECT movies.id,movieName,actorName,moviescategory.moviesCategoryName,relasedate,image FROM movies INNER JOIN moviescategory ON movies.category = moviescategory.id WHERE 1=1");**

**if (bean != null) {**

**if (bean.getId() > 0) {**

**sql.append(" AND id = " + bean.getId());**

**}**

**if (bean.getMovieName() != null && bean.getMovieName().length() > 0) {**

**sql.append(" AND movieName like '" + bean.getMovieName() + "%'");**

**}**

**}**

**ArrayList list = new ArrayList();**

**Connection conn = null;**

**try {**

**conn = JDBCDataSource.getConnection();**

**PreparedStatement pstmt = conn.prepareStatement(sql.toString());**

**ResultSet rs = pstmt.executeQuery();**

**while (rs.next()) {**

**bean = new MoviesBean();**

**bean.setId(rs.getLong(1));**

**bean.setMovieName(rs.getString(2));**

**bean.setActorName(rs.getString(3));**

**bean.setCategory(rs.getString(4));**

**bean.setRealeasedate(rs.getDate(5));**

**bean.setImage(rs.getString(6));**

**list.add(bean);**

**}**

**rs.close();**

**} catch (Exception e) {**

**e.printStackTrace();**

**} finally {**

**JDBCDataSource.closeconnection(conn);**

**}**

**return list;**

**}**

**}**

**DateUtility**

**package in.co.movie.review.Utility;**

**import java.sql.Time;**

**import java.sql.Timestamp;**

**import java.text.SimpleDateFormat;**

**import java.util.Date;**

**public class DataUtility {**

**public static final String APP\_DATE\_FORMATE = "MM/dd/yyyy";**

**public static final SimpleDateFormat formatter = new SimpleDateFormat(APP\_DATE\_FORMATE);**

**public static final String APP\_TIME\_FORMATE = "HH:mm:ss";**

**public static final SimpleDateFormat timeformatter = new SimpleDateFormat(APP\_TIME\_FORMATE);**

**public static String getString(String val) {**

**if (DataValidator.isNotNull(val)) {**

**return val.trim();**

**} else {**

**return val;**

**}**

**}**

**public static int getInt(String val) {**

**if (DataValidator.isInteger(val)) {**

**return Integer.parseInt(val);**

**} else {**

**return 0;**

**}**

**}**

**public static String getStringData(Object val) {**

**if (val != null) {**

**return val.toString();**

**} else {**

**return "";**

**}**

**}**

**public static long getLong(String val) {**

**if (DataValidator.isLong(val)) {**

**return Long.parseLong(val);**

**} else {**

**return 0;**

**}**

**}**

**public static Date getDate(String val) {**

**Date date = null;**

**try {**

**date = formatter.parse(val);**

**} catch (Exception e) {**

**}**

**return date;**

**}**

**public static Date getDate1(String val) {**

**Date date = null;**

**try {**

**date = formatter.parse(val);**

**} catch (Exception e) {**

**}**

**return date;**

**}**

**/\*\***

**\* Converts Date into String**

**\***

**\* @param date**

**\* @return**

**\*/**

**public static String getDateString(Date date) {**

**try {**

**if (date != null) {**

**return formatter.format(date);**

**} else {**

**return "";**

**}**

**} catch (Exception e) {**

**return "";**

**}**

**}**

**/\*\***

**\* Converts Time into String**

**\***

**\* @param time**

**\* @return**

**\*/**

**public static String getTimeString(Time time){**

**try {**

**if (time != null) {**

**return timeformatter.format(time);**

**} else {**

**return "";**

**}**

**} catch (Exception e) {**

**return "";**

**}**

**}**

**public static Time getTime(String val) {**

**Time time = null;**

**try {**

**time = (Time) timeformatter.parse(val);**

**System.out.println("time is :"+time);**

**} catch (Exception e) {**

**}**

**return time;**

**}**

**public static Date getDate(Date date, int day) {**

**return null;**

**}**

**public static Timestamp getTimestamp(long l) {**

**Timestamp timeStamp = null;**

**try {**

**timeStamp = new Timestamp(l);**

**} catch (Exception e) {**

**return null;**

**}**

**return timeStamp;**

**}**

**public static long getTimestamp(Timestamp tm) {**

**try {**

**return tm.getTime();**

**} catch (Exception e) {**

**return 0;**

**}**

**}**

**public static Timestamp getCurrentTimestamp() {**

**Timestamp timeStamp = null;**

**try {**

**timeStamp = new Timestamp(new Date().getTime());**

**} catch (Exception e) {**

**}**

**return timeStamp;**

**}**

**}**

**Screen Snapshot**

**ADD SS after configuration**