**Project Team 4 - MovieXperience**

**Team Members: Mai Moftah, Triveni Shetty, Deepika Maniyil, Vivek Ravi**

## **PROJECT PROPOSAL**

**Overview and Motivation**

The project we intend to complete for this class is an application that brings together the ability to browse and book our tickets from theatres nearby, and the ability to stream DVD released movies from the comfort of your own home. This application brings together the functions offered by a multitude of different applications; it allows you to book tickets of your favorite movie and theater, it also allows you to watch movies online free anytime and anywhere. We allow users the option to create an account which make them eligible for weekly perks and rewards. Our application is different and is needed because it brings together the 2 different types of movie experiences (at home and at theaters) at an almost free rate.

**Scope and Objectives**

This is a web application project with graphical user interface that is intended to provide usability to teenagers and adults who have basic to advanced proficiency with computer usage. For ‘*Home*’ movie experience, the user can create an account and enjoy unlimited Ordinary/HD streaming movies. For *‘Theater’* movie experience, the application will compute the net pay amount for each transaction based on the number of movie tickets entered, user’s seat preference, movie timings, and the type of theater the user selects.

## **PROJECT ENVIRONMENT AND ARCHITECTURE**

After considering the tools and languages previously used in VMs in this class (Java, Apache, JSP etc.), as well as research done on best languages and tools for web applications, we came to the following final decisions:

Since all developers in the team use different OS and inherently different environments, we decided to simulate identical development environments as much as possible.

**Short Summary:**

We decided to create the Spring 4 MVC application within Eclipse IDE. Continuous and agreed-upon code commit timelines will mean that all team members will be able to pull other members’ code to then test and view code integration.

Briefly, each machine has a local setup of the following:

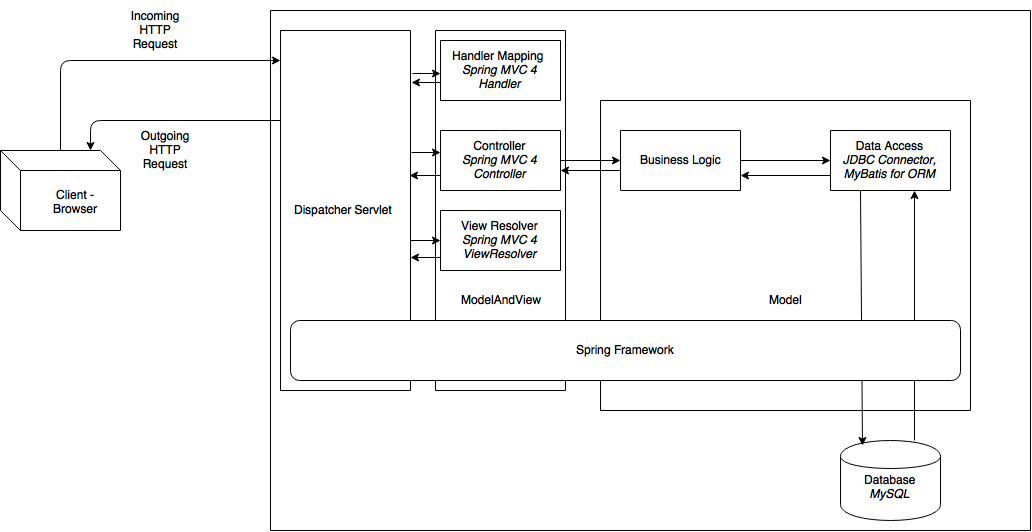
* Backend: Java Version 7
* Framework: Spring 4 MVC
* Web Server: Apache Tomcat 8
* IDE: Eclipse Mars 4.5
* Database: MySQL
* Version Control: Git

**Languages Used:**

* **Backend**: The language to be used for the backend will be Java
* **Frontend**: Our website application will rely on the usage of Bootstrap as well as JQuery for the frontend. To ensure that all designed pages are within the same flow, we have decided to use stylesheets to enforce the uniformity across all pages in the web application.

**Tools and Frameworks:**

* **Framework**: Our development will rely on the usage of the commonly used java web application framework, Spring MVC (Version 4)
* **Web Server + Application server**: Since we are building a website application, we needed a web server that would be able to serve our content in a smooth and efficient manner, and an application server that would be a good middleman for our Spring application. We decided to go with Apache Tomcat 8.
* **Object Relational Mapping:** To allow the retrieval of data from the MySQL database, computations, and displaying of any data, we utilized an ORM tool. For this project we went ahead and used MyBatis ORM tool. After debating between Hibernate ORM and MyBatis ORM, we ended up choosing MyBatis due to time constraints for learning about the tools and overall simplicity.
* **Database**: For our database setup, we went ahead and setup the MySQL database server, which was found to be suitable and compatible with Spring applications. In order to connect the database with the Spring MVC application, we included the information about the database (name of database, database adapter, username and password) within the MyBatis SQL config file. This allows the application to connect correctly with the database and pull desired information.
* **Unit Testing Framework**: We decided to go ahead and use the commonly used java testing framework, JUnit . After generating the desired code, all elements (controllers, views, and models) can be tested within Eclipse IDE with the aid of the JUnit library. Minor changes are made to the database connection, to allow for the access of the test environment instead of production environment, when testing the code. We also used Mockito to test our controllers in our application.
* **Version Control**: We decided to use Git along with Github and create 1 project with all collaborators as members of our team. Constant committing of the code means that all members will have access to the most up-to-date code from the team members.
* **Architecture**: Since we are using Spring MVC as our framework, we will be relying on the Model-View-Controller architecture.



*Spring 4 Application MVC High Level Architecture Diagram*

## **PROJECT REQUIREMENTS**

## **SPRINT 2**

**Story 3:**

**Description**:

As a customer, I want to search for movies, so that I can view a list of movies available in nearby theaters. (genre, name, time playing)

**Acceptance criteria**:



**Tasks**:

Task 56:

3 hrs/ Create a 'Search for movies by genre' view - refer to the acceptance criteria for a rough idea.

Task 57:

2 hrs/ Within the view, create a few select buttons that will link to the corresponding genre page

Task 58:

1 hr/ Create a dropdown list that contain all remaining genres that we will offer in terms of movies - once a genre is selected, the page must automatically populate movies of that genre.

Task 59:

2 hrs/ Design and code the cover flow within the view appropriately so that only movies opening this week are shown in each 'cover'.

Task 60:

3 hrs/ Design the data access layer so that the model can access the db for a specific genre of movies and then respond with the correct list of movies.

Task 61:

3 hrs/ Design a reusable template for each genre 'homepage' - this must contain movies of that genre playing now and opening soon (refer to acceptance criteria doc).

Task 62:

2 hrs/ Create a Controller for 'Search for movies by genre'

Task 63:

3 hrs/ Create a 'Search by theater movie title' view - this must contain a search field and search button. Refer to acceptance criteria doc for a better idea.

Task 64:

2 hrs/ Create 'Search by theater movie title' controller

Task 65:

3 hrs/ Create 'Search by theater movie title' data access layer - that can access the db table to retrieve correct + corresponding movie data.

Task 66:

3 hrs/ Code the 'autocomplete function' for searching by title- refer to acceptance criteria pdf for 'view example' and acceptance criteria. Ensure correct code is added to model, view and controller to enable this function

Task 67:

2 hrs/ Create a 'failed search' view - which will display what happens when the search is unsuccessful. This page must show 'opening shows' as well. Refer to acceptance criteria pdf.

Task 68:

2 hrs/ Create a 'failed search' controller as well

Task 69:

5 hrs/ Create a 'search by showtime' view - ensure dropdown of calendar is functional, ensure search button retrieves correct data for show time selected. Ensure that today's view contain all of today's showtimes and movies (similar to acceptance criteria)

Task 70:

3 hrs/ Create a 'search by showtime' model and the data access layer to retrieve movies playing at a specific time and date based on use selection.

Task 71:

2 hrs/ Create hyperlinks for each that will link to each movie's 'homepage' - this page could provide more information on each movie and allow other functions such as booking of tickets, selection of seats etc.

Task 72:

3 hrs/ Create a function that will only allow the user to see the movie showings for now and after the current clocktime, i.e. do not show movies from today that have already started playing.

Task 73:

3 hrs/ Create a genre table (including: genre and correct movie title)

Task 74:

3 hrs/ Create a movie table (include movie name, movie plot summary, actors in movie, length of movie) with a relation to the genre table

Task 75:

2 hrs/ Create a showtime table with a relation to the genre and to the movie table

Task 76:

3 hrs/ Test all hyperlinks for each movie - ensure that the movie hyperlink links to its corresponding homepage and includes details about the movie (e.g. time, plot summary, poster picture, and even a trailer)

Task 77:

3 hrs/ Test 'search by genre' function to see that correct genres are retrieved when a genre is selected.

Task 78:

3 hrs/ Test 'search by showtime' and see if correct day and time for movies is returned for movies within the db

Task 164:

Update the corresponding controller mapping requests to allow for proper UI display / 3hrs

## **SPRINT 2**

**Story 4:**

**Description**:

As a customer, I want to search for theaters nearby

**Acceptance criteria**:



**Tasks**:

Task 79:

3 hrs/ Test different theater chains from the dropdown list and 'search' - display should show ONLY theaters from that chain

Task 80:

3 hrs/ Test hyperlinks for each theater - this must direct to the correct and specific theater homepage within our app (page should include movies and times for that single theater ONLY)

Task 109:

Create a JPAConfig for Hibernate ORM

Task 110:

Create a RootConfig

Task 111:

Create a model for movies

Task 112:

Create a Constants file to be used by Hibernate setup

Task 113:

Create a DAO interface for movies

Task 114:

Create an implementable for the DAO

Task 115:

Create a GenericService Interface and its implementable

Task 116:

Create a searchservice interface and its implementable

## **SPRINT 2**

**Story 5:**

**Description**:

As a customer, I want to enter my zipcode, so that I can view available movies and theaters.

**Acceptance criteria**:



**Tasks**:

Task 90:

Estimate - 2hours/ Create a database table â€œZip codeâ€: 2hrs

Task 91:

Estimate:2 hours/ Create a database table theaters

Task 92:

Estimate:2 hours/ Create a database table theaters

Task 93:

Estimate: 3hours/ Design and test UI view for the home page

Task 94:

Estimate: 4 hours/ Design and test UI view for the Theater xperience page using Html and JavaScript Where the list of movies and theaters is displayed for the selected zip code.

Task 95:

Estimation: 4 hours/ create and test controller class for validation of zip code

Task 99:

Estimation: 5 hours/ Design a model class for querying the zip code database to check if the user has entered the valid zip code which is maintained in the database.

Task 100:

Estimation: 4 hours/Create an interface which follows the factory design pattern to return user class object for different zip code.

Task 101:

Estimation: 3 hours/ Develop a zip code class which contains the zip code based details and implements the interface mentioned above and will be used by the controller class.

Task 102:

Estimation: one hour/ Test the zip code class

Task 103:

Estimation: 2 hours/ Add code to link the homepage to the theater page

Task 104:

Estimation: 2 hours/ Code acceptance test for theater Xperience page

Task 165:

3hrs/ Update the UI to remove white space in bottom of all pages and allow for more content visibility as well as proper responsiveness.

## **SPRINT 2**

**Story 6:**

**Description**:

As a customer, I want to be able to select the movie I want or the theater of my choice

**Acceptance criteria**:



**Tasks**:

Task 151:

Create a table â€œMovieâ€ using MySQL which will hold the movie details

The table includes the movie id as primary key

Task 152:

Create a table â€œMovieâ€ using MySQL which will hold the movie details

The table includes the movie id as primary key

Task 153:

Task Estimation: 3hrs / Design and test the UI view for the Theater Experience page (movie selection) which has Movie listed in the dropdown based on the zip code selected in the previous page. Also design and test a search button using bootstrap, html and JavaScript.

Task 154:

Task Estimation: 8hrs / Create and test controller class for validation of the input movie using spring framework.

Task 155:

Task Estimation: 5hrs / Design a Data access object for querying the Theater\_movie\_map database to check if the user has entered the valid data in the form using Mybatis Technology. This will open the session in the database and fetches the data based on the query and compares it with the bean class input variables.

Task 156:

Task Estimation: 5hrs / Develop a Bean class (Model class) which stores the data given in the input using the model attribute to the bean class object.

Task 157:

Task Estimation: 3hrs / creating a mapper class in the SQL config file to refer the Bean class object view map to the data access object Movie and Theater database.

## **SPRINT 2**

**Story 7:**

**Description**:

As a customer, I want to be able to enter the number of seats to be booked

**Acceptance criteria**:



**Tasks**:

Task 117:

1 hr/ Research how reservation systems implement the flow for reservation using Spring

Task 118:

2 hrs/ Using Spring and Java, code a service interface that will allow the user to book their ticket

Task 120:

2 hrs/ Using Spring and Java, code a seat Dao and its implementable

Task 121:

1 hr/ Using Javax persistence, code a seat entity model to contain all needed getters and setters for each seat booking

Task 125:

2 hrs/ In the jsp file for the theater booking part, code a input box that will allow input of number of seats

Task 126:

2 hrs/ In the jsp file, code a submit button that will input the data into the database and compare to seats available using mybatis ORM

Task 127:

3 hrs/ create a table that will update the seats available/ unavailable based on booking performed and allow it to be connected to code appropriately

## **SPRINT 3**

**Story 8:**

**Description**:

As a customer, want to be able to select my seat from a seat map.

**Acceptance criteria**:

****

**Tasks**:

Task 159:

Task Estimation: 3hrs / Design and test the UI view for the seat map which will show the list of seats available for a particular movie. The select button is designed to specify the action that the user has selected the movie.

Task 160:

Task Estimation: 8hrs /Create and test a controller class for validation of seat selection using spring framework.

Task 161:

Task Estimation: 2hrs / Develop a Bean class (Model class) which stores the data given in the input using the model attribute to the bean class object.

Task 162:

Task Estimation: 2hrs / Develop a Bean class (Model class) which stores the data given in the input using the model attribute to the bean class object.

Task 163:

Task Estimation: 3hrs / creating a mapper class in the SQL config file to refer the Bean class object view map to the data access object Theater\_movie\_map.

## **SPRINT 2**

**Story 9:**

**Description**:

As a customer, I want to be able to view the seat map for the selected theater/movie.

**Acceptance criteria**:

****

**Tasks**:

Task 122:

2hrs/ Research how jquery can be used to display an interactive seat map

Task 123:

1 hr/ Find helpful plugins to customize a theater seat chart in jquery, javascript

Task 124:

3 hrs/ Add the plugin files and map them in the xml mapping file

Task 133:

5 hrs/ Edit the code of the jquery file to match seat selection in theater (available, unavailable, selected)

Task 134:

6 hrs/ Code the front-end part of the seat map (html) and ensure that all CSS and JS files work appropriately

## **SPRINT 3**

**Story 11:**

**Description**:

As a customer, I want to select my card type during payment and make the payment.

**Acceptance criteria**:



**Tasks**:

Task 166:

2 hours/Create a database table for Payment Summary page.

Task 167:

3 hours/Design and test the UI(view) for the Payment Summary page. Code a page for the Payment Summary using HTML and Java scripts. The UI should contain the total amount to be paid by the customer, a dropdown called â€œCard Typeâ€ which contains values Credit card and Debit card, a text box to enter card number and a textbox to enter password. (as shown in the screen mockup). The UI should contain a SUBMIT button (as shown in the screenmockup) for navigating the entered details successfully to a success message page. This page will be hosted by tomcat web server.

Task 168:

3 hours/Add code to link the Payment Summary page to the success message page. The code should be added to the controller class business logic so that the payment can be done successfully after entering the card number and the password. Here the validation of the input fields are done.

Task 169:

3 hours/Create a model class for the Payment Summary page. This class must be able to check if the card number and the password entered in the input field is valid for the selected card type. If not valid, it must not allow the user to make the payment successfully.

Task 170:

2 hours/Code acceptance tests for PaymentSummary page. Develop and run test cases for the complete payment process.

## **SPRINT 1**

**Story 14:**

**Description**:

As an existing member, I want to be able to log into my account.

**Acceptance criteria**:



**Tasks**:

Task 86:

5 hours/Design and test the UI for the LOGIN page--> Code a page for the LOGIN page using HTML and Java scripts. The UI should contain text input fields for entering Username and Password (as shown in the screen mockup). The UI should contain the GO button (as shown in the screen mockup) for navigating the entered details to the successfully logged in Homepage. This page will be hosted using passenger web server. Develop and run test cases for the view.

Task 87:

5 hours/Add code to link the LOGIN page to the logged in HOME page--> The code should be added to the controller class business logic so that the user can successfully log in if the entered credentials are valid. Here, the validation of the text input field is done.

Task 88:

5 hours/Create a model class for the LOGIN page--> This class must be able to check if the username and password entered in the input field is same as present in the database. If not, the user must not be able to login successfully.

Task 89:

3 hours/Code the acceptance tests for LOGIN page--> Develop and run test cases for the complete login process.

## **SPRINT 1**

**Story 18:**

**Description**:

As a non-member, I want to create an account

**Acceptance criteria**:



**Tasks**:

Task 81:

3 hours/Create a database table for SignUp--> Create a table â€œSignUpâ€ using MYSQL which will hold the Sign Up details for the new customers. Table should include email id as the primary key.

Task 82:

4 hours/Design and test the UI(view) for the SignUp page--> Code a page for the Signing Up using HTML and Java scripts. The UI should contain text input fields for entering the First name, Last name, email id and password (as shown in the screen mockup). The UI should contain a SUBMIT button (as shown in the screenmockup) for navigating the entered details successfully to a success message page. The UI should also contain a CANCEL button (as shown in the screenmockup) for navigating back to the HOME page. This page will be hosted by passenger web server. Develop and run test cases for the view.

Task 83:

5 hours/Add code to link the SignUp page to the success message page--> The code should be added to the controller class business logic so that an account is created based on the details entered by the customer. Here the validation of the input fields (specially email id) is done.

Task 84:

5 hours/Create a model class for the SignUp page--> This class must be able to check if the email id entered in the input field is already present in the database. If already present, it must not allow the user to successfully sign in.

Task 85:

3 hours/Code acceptance tests for SignUp page--> Develop and run test cases for the complete sign up process.

## **SPRINT 2**

**Story 19:**

**Description**:

As an existing member, I want to be able to filter the movie selection by genre.

**Acceptance criteria**:



**Tasks**:

Task 36:

Add a movie by genre drop down box to the already existing HomeXperience- search page(please refer screen mock up in user acceptance criteria) Estimate: 2 hours

Task 37:

Add a radio button Search by movie genre(please refer screen mock up in user acceptance criteria). Estimate : 3 hours

Task 38:

Validate the Page(view) for correct user selection of inputs. Note : To hit the Go button ,user should have selected this radio button and should have selected a valid genre from the drop down list. Estimate: 4 hours

Task 39:

To the already existing class-Movie\_Search (controller) add a new method searchByMovieGenre() for looping through the dataset and retrieving the result..Estimate: 6 hours

Task 40:

Test the connection to the database giving correct userid and password. Estimate : 2 hours

Task 41:

Write and test the database query for retrieving information for a given movie genre. Estimate: 2 hours

Task 42:

Code and test Movie\_Search Class which takes the movie genre from movie genre drop down box and queries the database for results.Estimate: 6 hours

Task 43:

Test the HomeXperience- search\_results for correct query results and proper GUI. Estimate: 6 hours

Task 44:

Group the common UI elements together. Estimate: 3 hours

Task 45:

Upload to github repository and document the search feature implemented. Estimate: 3 hours

Task 48:

In the persistence layer, create or update table "Movie\_details" which has mandatory columnsMovie\_Id,Movie\_Name,Movie\_genre,Last\_updated,Movie\_synopsis,Movie\_release\_date. Estimate: 4 hours

Task 97:

Code and test the code acceptance criteria .Estimate: 4 hours

## **SPRINT 2**

**Story 20:**

**Description**:

As an existing member, I want to be able to filter the movie selection by movie name.

**Acceptance criteria**:



**Tasks**:

Task 27:

Add a "Search by movie name" text box to the created HomeXperience- search page. (please refer screen mock up in user acceptance criteria) Estimate:2 hours

Task 28:

Add a radio button Search by movie name(please refer screen mock up in user acceptance criteria). Estimate: 2 hours

Task 29:

Validate the Page(view) for correct user selection of inputs. ïƒ° Note : To hit the Go button ,user should have selected this radio button and entered a string in the movie text box. Estimate : 5 hours

Task 30:

Create a class Movie\_Search (controller) with method searchByMovieName() for looping through the dataset and retrieves the result. Estimate:9 hours

Task 31:

Test the connection to the database giving correct userid and password. Estimate : 3 hours

Task 32:

Write and test the database query for retrieving information for a given movie name. Estimate 3 hours

Task 33:

Code and test Movie\_Search Class which takes the movie name from movie name search text box and queries the database for results. Estimate : 5 hours

Task 34:

Test the HomeXperience- search\_results for correct query results and proper GUI. Estimate: 4 hours

Task 35:

Group the common UI elements together. Estimate: 4 hours

Task 46:

Create a new page in view called "HomeXperience-search". Estimate: 3 hours

Task 47:

Upload to github repository and document the search feature implemented. Estimate 3 hours

Task 49:

In the persistence layer, create or update table "Movie\_details" which has mandatory columnsMovie\_Id,Movie\_Name,Movie\_genre,Last\_updated,Movie\_synopsis,Movie\_release\_date. Estimate: 4 hours

Task 98:

Code and test the code acceptance criteria. Estimate: 5 hours

## **SPRINT 1**

**Story 21:**

**Description**:

As an existing member, I want to be able to play the videos of my choice.

**Acceptance criteria**:



**Tasks**:

Task 105:

Estimation: 2 hours / Test UI view for the Home Xperience page where the embedded video player for each movie is displayed. The video must be clicked to watch the movie.

Task 106:

Estimation: 4 hours/ Code and test a controller class for the user to select the video from the view.

Task 107:

Estimation: 3 hours / Add code to link the home xperience page to the Running video page. The code should also be written to play the video selected from the Home.

Task 108:

Estimation: 2 hours / Code acceptance test for the running video page. Develop and run test cases for the complete flow of running video played.

## **SPRINT 2**

**Story 22:**

**Description**:

As a database Admin, I want to be able to add/remove movies from the movie database.

**Acceptance criteria**:



**Tasks**:

Task 128:

1 hour/Create a database table for MOVIES. Create a table â€œMOVIESâ€ using MYSQL that will contain all the movie details. Movie\_id is the primary key.

Task 129:

1 hour/Write an INSERT query to add movies to the MOVIES table.

Task 130:

1 hour/Write a DELETE query to delete movies from the MOVIES table.

Task 131:

1 hour/Write a SELECT query to view all the movies present in the MOVIES table. This is to test if the movies are added/deleted from the MOVIES table.

Task 132:

2 hours/Code acceptance tests.

## **SPRINT 2**

**Story 23:**

**Description**:

As a database, I want to be able to automatically add/remove users when prompted.

**Acceptance criteria**:



**Tasks**:

Task 139:

2 hrs: Create a new table â€œuser\_detailsâ€ to the existing ssdi database.

Task 140:

2 hrs: Add user\_id as primary key and auto increment it.

Task 141:

1 hr: Also add email\_id, password columns to the user\_details table.

Task 142:

4 hrs: Update the controller to contain the request mapping for URL(create /delete account) and return the view name with the ModelAndView object.

Task 143:

5 hrs: Create a new CreateUserDAO to instantiate a MyBatis session factory object and code for the CreateUser() and deleteUser() methods.

Task 144:

2 hrs: Code and test SQL query for inserting and deleting user credentials to the database.

Task 145:

4 hrs: Create â€œCreateUser.xmlâ€ to get all values from the created Login.jsp page (front end) and map it to the insert statement SQLquery.

Task 146:

4 hrs: Create â€œDeleteUser.xml to get all values from the created DeleteAccount.jsp page (front end) and map it to the delete statement SQLquery.

Task 147:

4 hrs: Code and test the CreateUser.java and "DeleteUser.java" model classes to get UI form field elements by ID and pass it on to the corressponding DAO class.(user

Task 148:

5 hrs: Code and test for user acceptance tests.(Refer user acceptance criteria).

Task 149:

3hrs: Create metadata for table and its columns.

Task 150:

1 hr: Update the project documents

## **SPRINT 2**

**Story 24:**

**Description**:

As a developer, I want to be able to update movies displayed on the application so that it matches our database

**Acceptance criteria**:



**Tasks**:

Task 35:

1 hour/Create a database table for MOVIES. Create a table MOVIES using MYSQL that will contain all the movie details. Movie\_id is the primary key.

Task 36:

2 hours/Design and test the UI(view) for the movie details page.

Task 37:

3 hours/Create a model class for the MOVIES page. This class must be able to check if all the movies present in the database are displayed on the application.

Task 38:

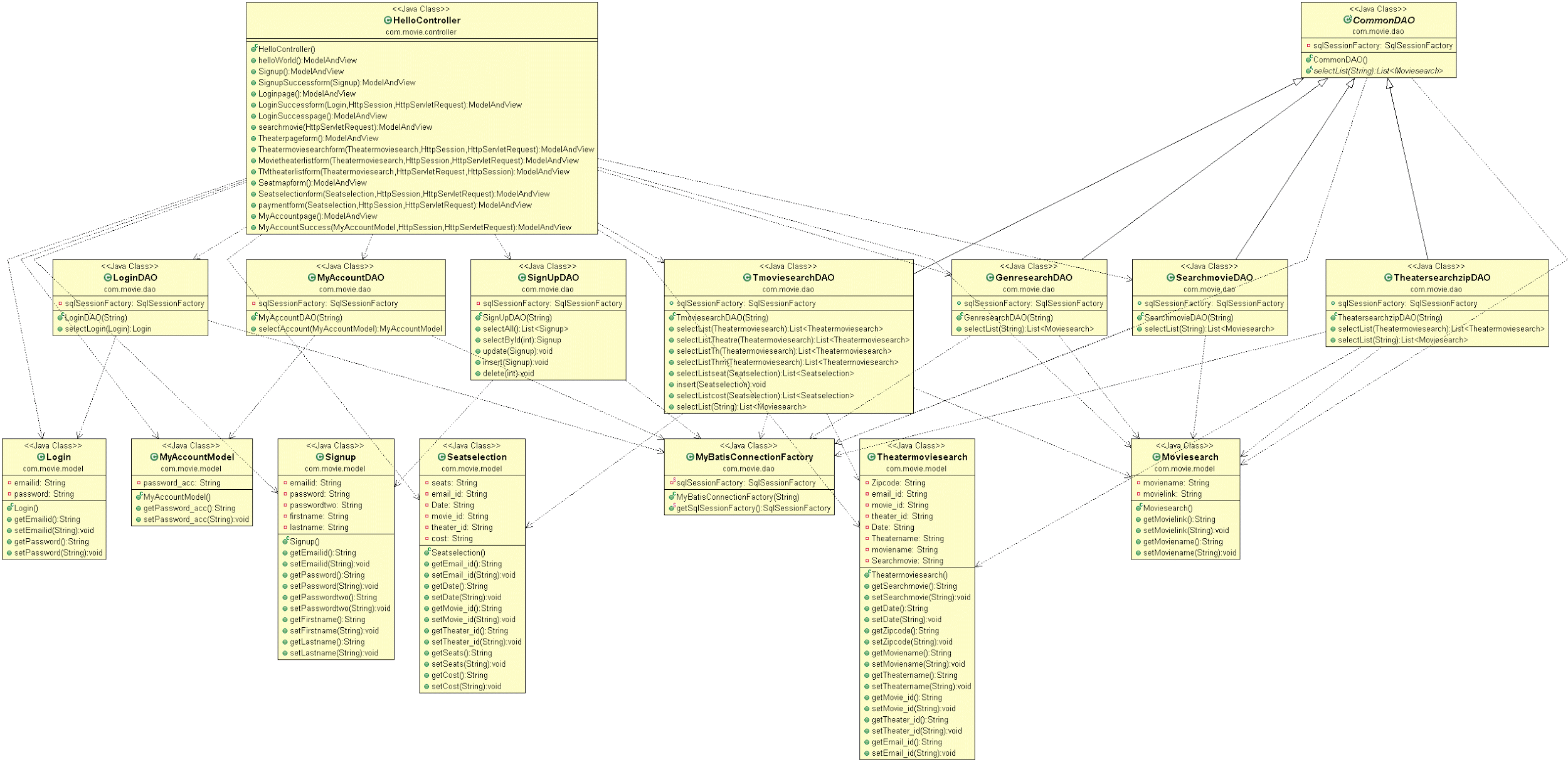
2 hours/Code acceptance tests for the MOVIES page.

## 

## 

## **DETAILED DESIGN**

UML CLASS DIAGRAM:



Generalization: 

Dependency: 

## **TESTING**

To increase the quality of the code, we have used a Regression Testing Framework called Junit to implement unit testing. We have integrated this framework with Eclipse.

**Test 1:**

Requirement(s) under test: #14/As an existing member, I want to be able to log into my account.

Function(s)/Module(s) under test: LoginDAO.java

Initial conditions: Initializing the LoginDAO class with test environment.

Assumptions: The user must already have an account.

Test case input: Enter the email id and password.

Expected result: Succesful Login.

Result: Passed.

**Test 2:**

Requirement(s) under test: #18/As a non-member, I want to create an account.

Function(s)/Module(s) under test: SignUpDAO.java

Initial conditions: Initializing the SignUpDAO class with test environment and the Signup Bean class.

Assumptions: The user doesn’t have an account.

Test case input: Enter the Name,desired email id and set up a new password.

Expected result: Successful SignUp.

Result: Passed.

**Test 3:**

Requirement(s) under test: #20/As an existing member, I want to be able to filter the movie selection by movie name.

Function(s)/Module(s) under test: SearchmovieDAO.java

Initial conditions: Initializing the SearchmovieDAO class with test environment.

Assumptions: The user has an account, is signed in and is in the Home Experience page.

Test case input: Enter the movie name.

Expected result: Either successful search or unsuccessful search.

Result: Passed.

**Test 4:**

Requirement(s) under test: #19/As an existing member, I want to be able to filter the movie selection by genre.

Function(s)/Module(s) under test: GenresearchDAO.java

Initial conditions: Initializing the GenresearchDAO class with test environment.

Assumptions: The user has an account, is signed in and is in the Home Experience page.

Test case input: Enter the genre name.

Expected result: Either successful search or unsuccessful search.

Result: Passed.