The final implementation of our ‘MovieMad’ website design follows a certain set of patterns which are stated as below:

1. n- tier (3- Tier Architecture)

2. MVC: Model, view and controller

3. DAO and DTO

We are going to explain each of the architectures in terms of how they follow the pattern of our codes in a clear, descriptive and more informative manner.

**1. N-tier (3-Tier Architecture) :** The 3 layers i.e. data, application and the presentation is what makes our code a perfect example of 3- Tier Architecture. Under the resources are the ‘.vm’ files which are basically the functionality of the display of our website which are done by using javalin and apache velocity and this is the presentation layer since it is an overview of all the front end works which are being implemented. The database layer which contains the connection to our MySQL database is connected in such a manner that it can be accessed throughout our entire code. In the model package, there is a utils package where the connection is stored. And finally comes the application layer which is basically the backend work which is basically pretty much everything inside the app package of our code.

Graphical user interface, text

Description automatically generated

This is the application layer.

A picture containing text, device, meter, gauge

Description automatically generated

This is the link to the database layer, the actual database is stored in MySQL.

Text

Description automatically generated with medium confidence

This is the presentation layer.

**2. MVC: Model, view and controller:** Our project implements a Model, View & Controller which are all interconnected as they are supposed to be. Any user actions are conveyed to the controller. The Controller is then in charge of updating the Model, & any requests are handled both ways between the Model & Controller. Next, the Model makes requests in a two-way connection with the View. Notifications are shown in the View with a two-way connection with the Controller. Finally, the View displays any updates to the user.

public static Handler *handleLoginPost* = ctx -> {  
 Map<String, Object> model = ViewUtil.*baseModel*(ctx);  
 //checks login details are correct  
 if (DBDist.*db*.logIn(*getQueryEmail*(ctx),  
 Main.*userDao*.getUserByEmail(*getQueryEmail*(ctx)).getSalt(), *getQueryPassword*(ctx))) {  
 //successful login  
 ctx.sessionAttribute("currentUser", Main.*userDao*.getUserByEmail(*getQueryEmail*(ctx)));  
 model.put("currentUser", ctx.sessionAttribute("currentUser"));  
 model.put("authenticationSucceeded", true);  
 }

From the above image it can be seen that UserController which is in controller package, is called upon by a user interaction, performs actions in the model, and then updates the view. Certain queries are run like get username and salted password which takes in the strings parameter and gets updated in the user class that is in the model and then finally gets displayed using the login.vm which is the view package. This is just one case and there are a lot of similar cases like the admin can delete or movies/shows which works similarly.

**3. DAO and DTO:** Our project follows the DAO pattern, as the only object that can make changes to the database in regards to the state of the program are our DAO classes, contained in our dao package. These are used to allow communication between the database and everything else.

Our project also follows the DTO pattern. The objects used to transfer data around are contained in the DTO package. These are directly related to the tables stored in the database, and are populated with that information upon starting the program.

Graphical user interface

Description automatically generated with low confidence

This is our DAO.

Graphical user interface, application

Description automatically generated

This is our DTO.

The reason for not choosing 2-tiers architecture because the performance tends to have a downfall with the increment of users and is more costly to operate. But the main problem is that multiple requests cannot be responded by the server at a same time which can be done with 3 trier architecture. The main advantage of using 3–Tier architecture is that this architecture helps with much greater flexibility since we, as a developer, have the freedom to update any part of an application without affecting the other since they are independent of each other.