

## **CSE 3001 Software Engineering**

### **Project Report**

**Project Title: Movie Recommendation Chatbot  
using Django**

#### **Team Members :**

Vibhor Sharma	(17BCE2152)
Manish R Reddy	(17BCE0098)
L Dinesh Kumar	(17BCE2142)

# Contents

1. Abstract.....	3
2. Introduction .....	3
2.1 Purpose.....	3
2.2 Product Scope.....	3
3. Software Development Life Cycle Model Used.....	4
4. Overall Description.....	5
4.1 Product Perspective.....	5
4.2 Product Functions.....	5
4.3 User classes and characteristics.....	5
4.4 Operating Environment.....	5
4.5 Design and Implementation Constraints.....	6
4.6 User Documentation.....	6
4.7 Assumptions and Dependencies.....	6
5. External Interface Requirements.....	6
5.1 User Interfaces.....	6
5.2 Hardware Interfaces.....	6
5.3 Software Interfaces.....	7
5.4 Communications Interfaces.....	7
6. System Features.....	7
6.1 Register.....	7
6.2 Login.....	8
6.3 Chatbot conversation.....	8
6.4 Personalized recommendation.....	8
6.5 Manage Users.....	9
7. Use Case Diagram.....	10
8. System Sequence Diagram.....	11
8.1 UC-1: Register User.....	11
8.2 UC-2: Login User.....	12
8.3 UC-3: Chatbot Conversation.....	13
8.4 UC-4: Personalized Recommendation.....	13
8.5 UC-5: Manage Users.....	14
9. Functional Requirements Specification.....	15
10. Other Nonfunctional Requirements.....	15
10.1 Performance Requirements.....	15
10.2 Safety Requirements.....	15
10.3 Security Requirements.....	15
10.4 Software Quality Attributes.....	16
10.5 Business Rules.....	16
11. UML Diagrams.....	17
11.1 Class Diagram.....	17
11.2 Collaboration Diagram.....	17
11.3 State Diagram.....	18
11.4 Activity Diagram.....	19
11.5 Component Diagram.....	19
11.6 Deployment Diagram.....	20
12. Testing.....	21
13. References.....	23
14. Appendix.....	23
Codes.....	23
Screenshots.....	47

# 1. Abstract

In today's world and era, there is a whole plethora of movies to watch and often making this choice is not so easy. Hence, we propose to implement a movie recommendation conversational agent which can collect data from the 'The Movie DB' website and use a variety of filters to suggest movies to the user. We also plan to make the recommendations personalised to the user by allowing them to make an account on our website which will keep track of their preferences in the long term.

In particular, we plan to allow the user to save their preferences and use clustering algorithms to figure out the user's preference and use it while fetching movie recommendations. The chatbot is one which can respond to normal conversations such as greetings and will fetch the requests from the user and use it to generate movie recommendations.

## 2. Introduction

### 2.1 Purpose

We will be making a movie recommendation system using django. We will be making a chatbot based website so that the user can interact with the chatbot and get recommendations about the kind of movies that he is interested in viewing and then he can finally decide the movie the user wants to view based upon the recommendations.

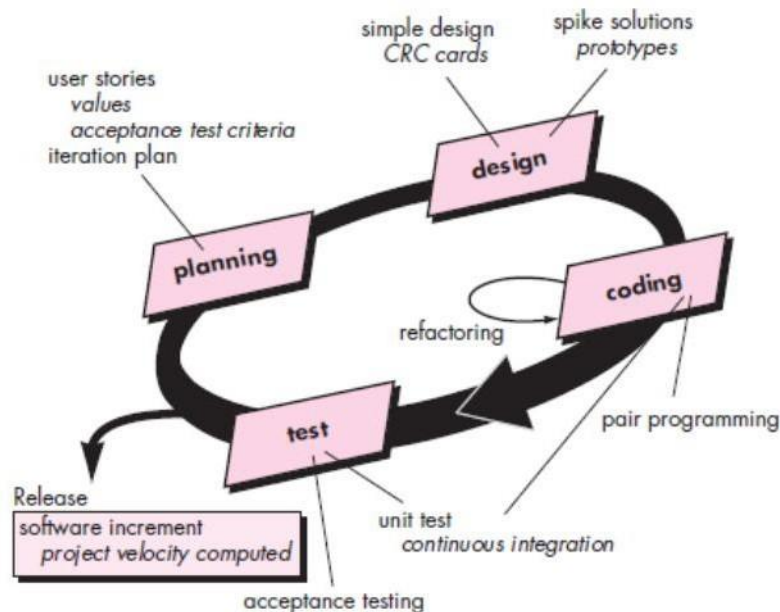
.

### 2.2 Product Scope

We propose to implement a movie recommendation conversational agent which can collect data from the 'The Movie DB' website and use a variety of filters to suggest movies to the user. We also plan to make the recommendations personalized to the user by allowing them to make an account on our website which will keep track of their preferences in the long term. In particular, we plan to allow the user to save their preferences and use clustering algorithms to figure out the user's preference and use it while fetching movie recommendations. The chatbot is one which can respond to normal conversations such as greetings and will fetch the requests from the user and use it to generate movie recommendations

### 3. Software Development Life Cycle Model Used

We are going to using the **Extreme Programming framework in Agile Model** for our project.



**Extreme Programming Workflow**

An agile model has the following benefits which are advantageous to our project-

- Continuous delivery of software
- Continuous update according to changes
- Intended to deliver working software quickly
- Based on an iterative and incremental approach to software development
- Evolve quickly to meet changing requirements

We plan to use Agile Model for our project as-

- Our project consists of several (3-4) phases which have to be made one by one and the requirements of each components can only be decided after the previous phases have been completed.
- For now, we have decided on two of these components. First, we will build a chatbot which will be able to filter and recommend movies to users. In the second phase, we will personalize the suggestions to be tailored to individual users which will be based on the history of choices they have made in the past.

- Hence, we are using Extreme Programming(Agile) Model because it allows us to focus more on the code and less on the planning i.e. we will be able to prototype our project faster.
- Agile will allow us to add more components to the project as we continue working on it and the software planning is need to be extended only after each phase is completed.
- We can solve any roadblock when building our project as the agile model allows our planning to be more flexible than other models

## 4. Overall Description

### 4.1 Product Perspective

This software is a self –contained product purely created from scratch. The software is created for recommending a movie of their choice and their favorite genre

### 4.2 Product Functions

- Authentication – User sign-up/login
- User verification
- Interactive Chatbot
- Request the chatbot for movies of a particular genre
- Based upon the request, the chatbot suggests movies of that genre
- Also , the website will suggest movies based on the history of the movies that the user watches.

### 4.3 User Classes and Characteristics

- Users – People who login to the website to get recommendations of movies
- Analysts – People who analyze what type of movies the users watch ,to get know certain trends in movie watching
- Administrators – People who supervise the working of the website and take care of the backend and database of the website.

### 4.4 Operating Environment

- Operating System(i.e Windows 7 or above, also supports MacOS)
- Web browser (Mozilla firefox ,Google chrome or Internet explorer)
- Memory, a.k.a RAM (i.e 1GB RAM or above)
- Graphics Card (Intel Integrated Graphics)
- Hard Disk Space (i.e ,50 GB available)
- I/O Ports

## **4.5 Design and Implementation Constraints**

When the user does not like the recommended movies by the chatbot of the website, there is a problem. In this case the chatbot asks more questions to the user to get to know specifically what type of movies the user wants to watch and after analyzing the answers to the questions, the chatbot suggests the movies.

## **4.6 User Documentation**

A help page will be provided to the user the first time the user logs in to the website and proper directions will be given to the user as he keeps on using the Software.

## **4.7 Assumptions and Dependencies**

Assuming there is a movie night party in the house, then this website can be used to view the movie of the audience's choice and genre in the house. Based on the type of genre the audience wants to view, the website will give some recommendations of movies. The audience can choose from these movie recommendations.

# **5. External Interface Requirements**

## **5.1 User Interfaces**

- Keyboard
- Mouse
- Front end –Website

## **5.2 Hardware Interfaces**

- Any operating system
- Browser supporting HTML and Javascript
- Processor – Intel processor
- Memory –1GB
- Hard Drive Capacity –50GB

### 5.3 Software Interfaces

- Any operating system
- Database – SQL in the host server is used for storing data related to movies and user requests
- Django – Django in python is used as the back end for the website.

### 5.4 Communications Interfaces

This project supports all types of web browsers.

## 6. System Features

The following are the essential components(functional units) of our project-

### 6.1 Register

#### Description and Priority

This feature allows our project to accept new end users into our system environment. It is of low priority as our first goal is to get our website up and running before we accept new users. Registering on our website will allow the user to get personalized recommendations

#### Stimulus/Response Sequences

- **End User** requests to create an account.
- **The System** displays the account creation page.
- **End User** inputs the required user data.
- **The System** verifies that the information meet basic criteria such as correct amount of characters and validity of email address.
- **The System** verifies the user information by checking if the name exists on the **Database**, and if it does not stores the user data in the **Database**, sends the user an account confirmation email, and displays the “activate account by email” page.
- **End User** activates the account by clicking on a link embedded in the confirmation email, which informs the system the account has been confirmed
- **The System** activates the account by manipulating a field in the **Database** and displays the “registration successful” page to the user.

## 6.2 Login

### Description and Priority

Allows user to login to the website, allowing them access to the full functionality of the website such as personalized recommendations, some statistics etc. It is of low priority right now.

### Stimulus/Response Sequences

- **End User** enters the user login and password.
- **The System** does a basic validation to make sure that the password and username are not empty strings and don't contain an illegal number of characters.
- **The System** verifies the entered data by checking the Database. It then logs the user in and displays the End User's "Student Center" page

## 6.3 Chatbot conversation

### Description and Priority

This is the main frontal interface of the project which will be used by the user to interact with our system and get output in the form of movie suggestions and other related information.

### Stimulus/Response Sequences

- End User issues a query
- The System extracts keywords from query, interprets intent of the user and according to the filters( keywords) gets movie suggestions from the database.
- End User if satisfied with the result finishes conversation else the conversation is continued till user gets satisfactory result.
- End User submits the question

## 6.4 Personalized Recommendation

### Description and Priority

The user's history of liked movies is taken into account and will be used to determine the types of movies preferred by the user and will be presented to them in the form of personalized recommendations in one section of the page.

### Stimulus/Response Sequences

- End User watches movie recommended by chatbot and then comes back to website to rate it.
- The System clusters the movies watched by end user recently and extracts common features.
- The System uses these features to generate a list of movies to be presented to the user as suggestions
- End User decides validity of suggestions.



## 6.5 Manage Users

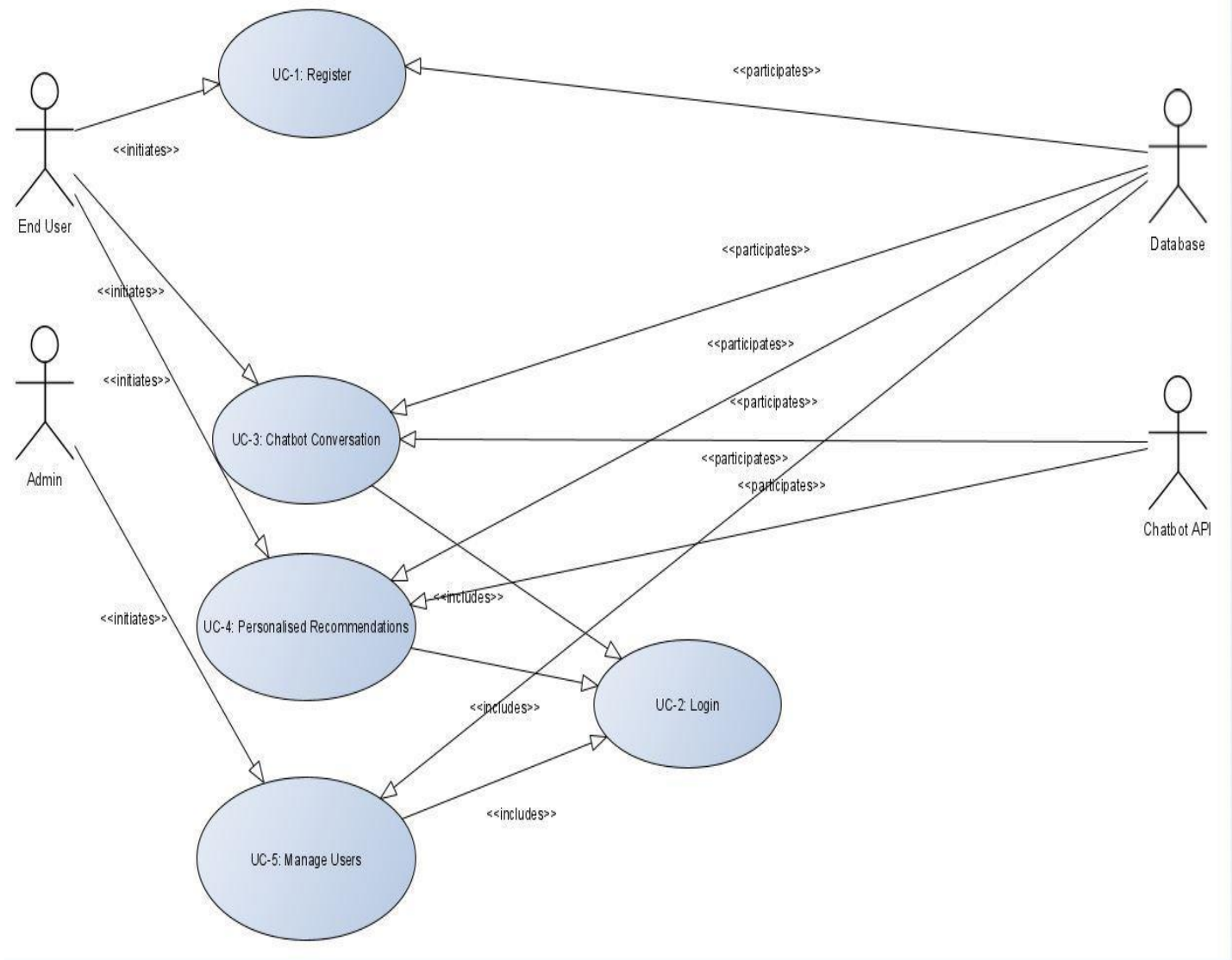
### Description and Priority

The user's history of liked movies is taken into account and will be used to determine the types of movies preferred by the user and will be presented to them in the form of personalized recommendations in one section of the page.

### Stimulus/Response Sequences

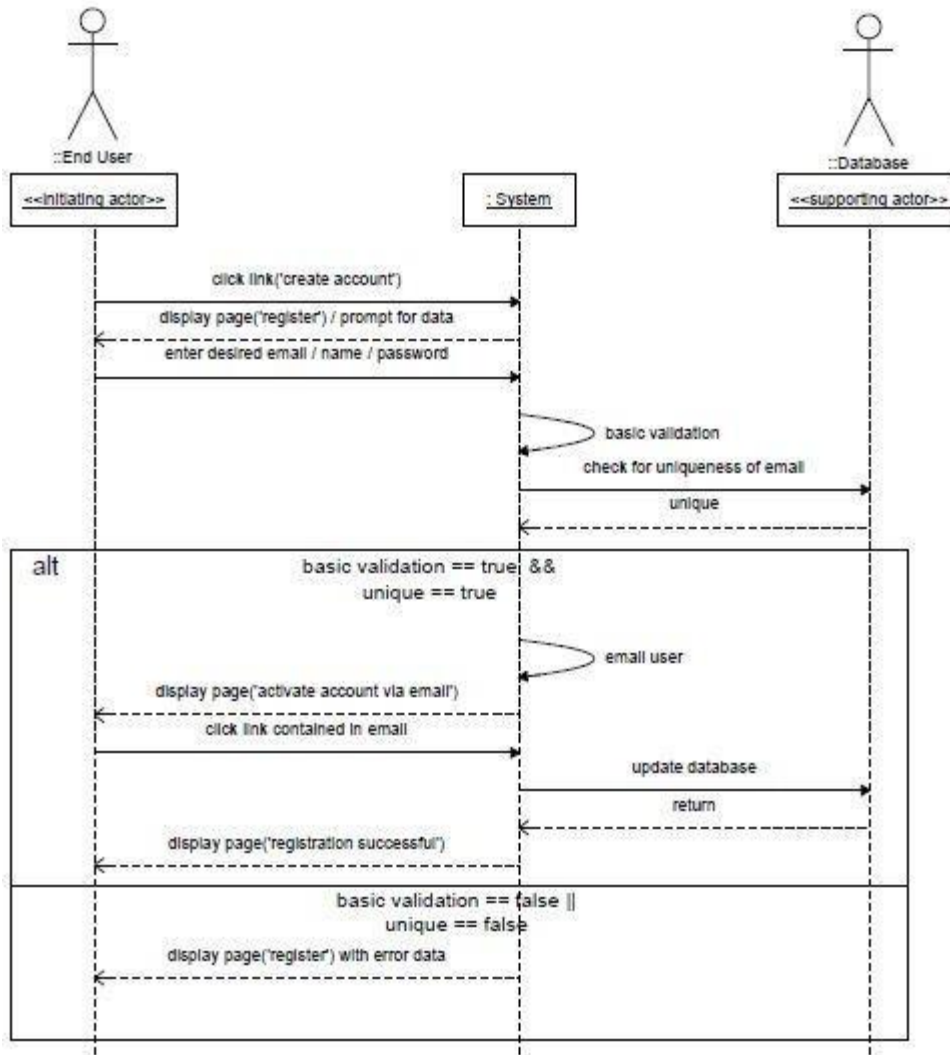
- **End User** selects new course(s) to add to or remove from their enrollment list.
- **The System** modifies the Database entries related to the user's enrollment list.
- **The System** displays an "add/remove successful" page to the user

## 7. Use Case Diagram



# 8.System Sequence Diagram

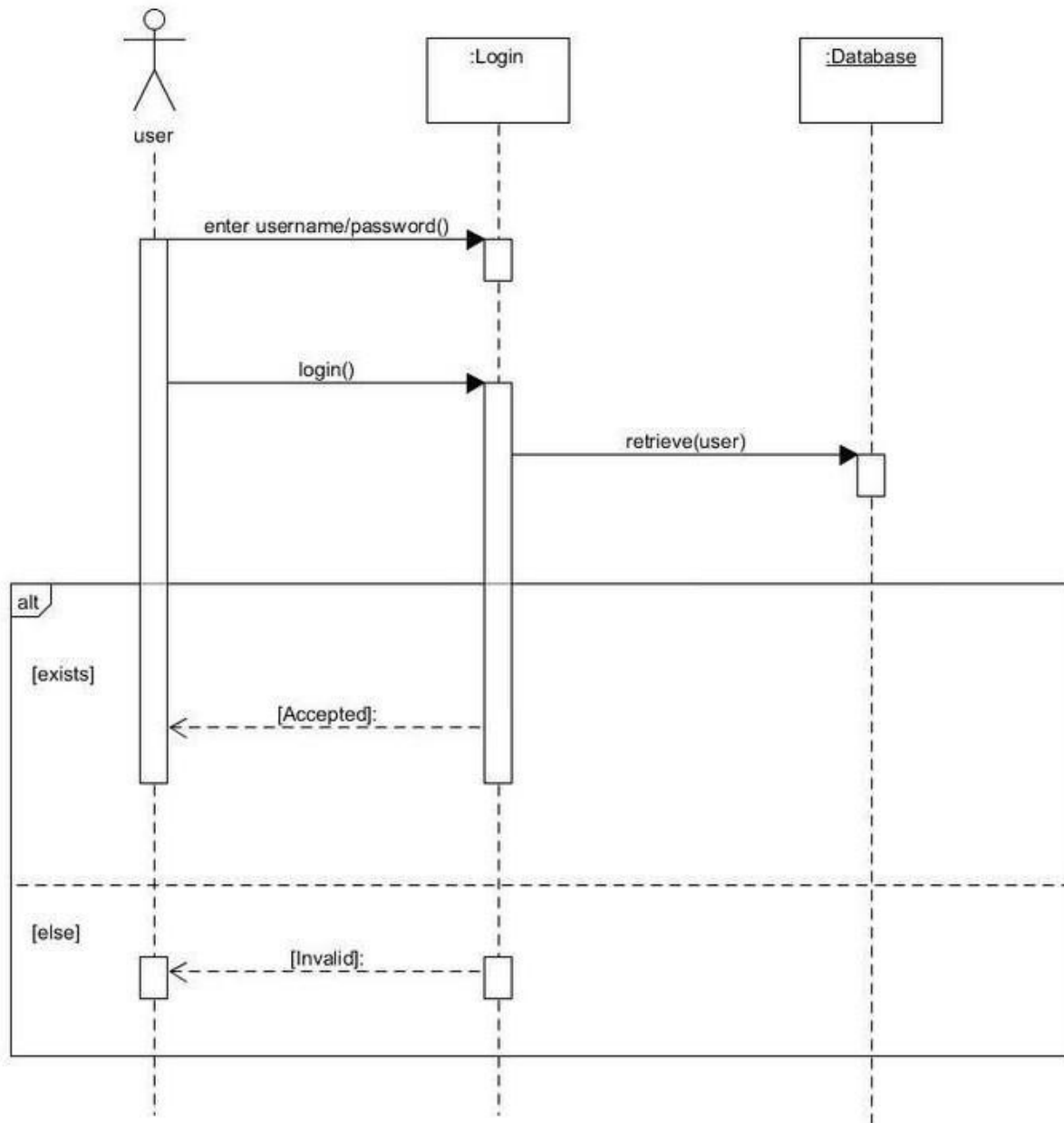
## 8.1 UC-1: Register User



### Create a User Account Explanation:

The user account creation is fairly self-explanatory. The user requests to create an account and then the system prompts the user for information. When the user submits his/her personal information (name, email, and password), the system checks to make sure it is valid and the account is created.

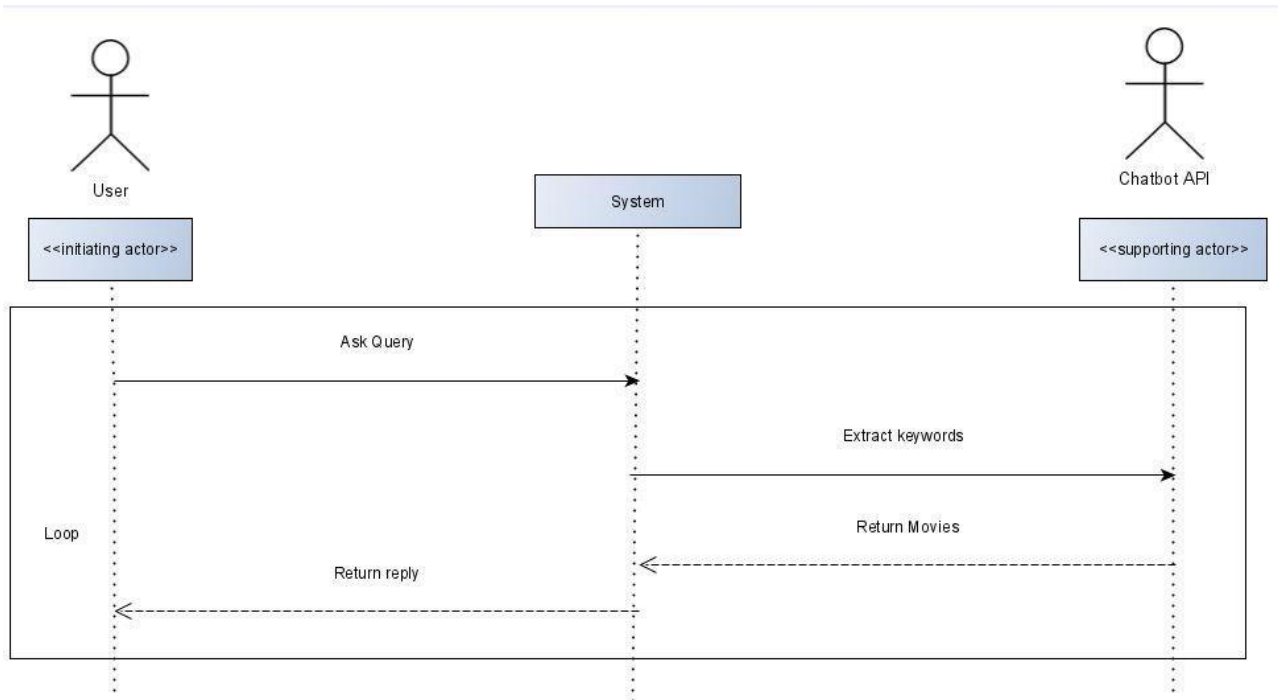
## 8.2 UC-2: Login User



### Log into an Account Explanation:

In order to log in the user first requests to log in to the website, and the system displays the log in screen. From here the user inputs their email and password into the displayed fields and clicks log-in. The system checks if the information meets basic criteria and then if the username and password exist in the database. It then creates a session for the user.

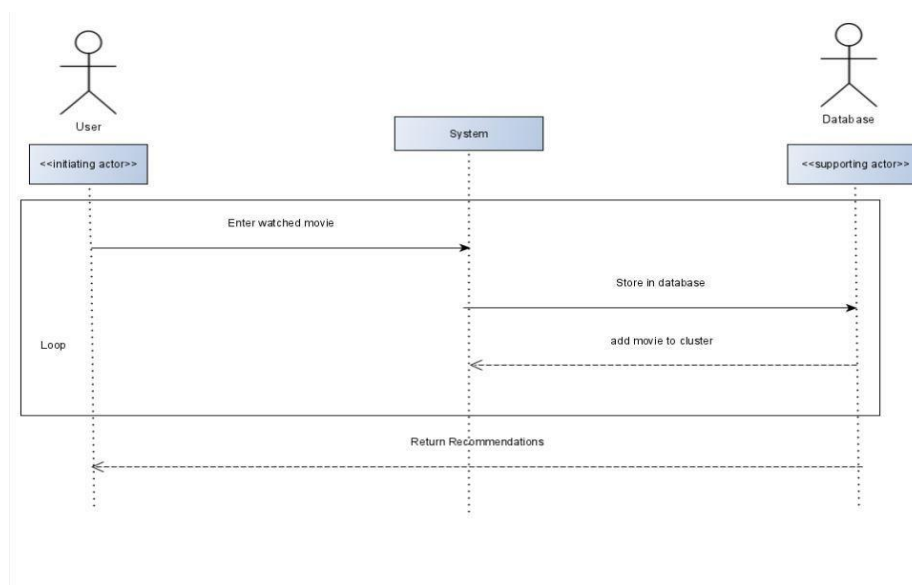
### 8.3 UC-3: Chatbot Conversation



#### Chatbot Conversation Explanation:

The end user can chat with our system and receive recommendations from it till they are satisfied. The user sends a query from which the keywords are extracted through NLP techniques and relevant movies are extracted from the database according to the keywords. It is then presented to the user.

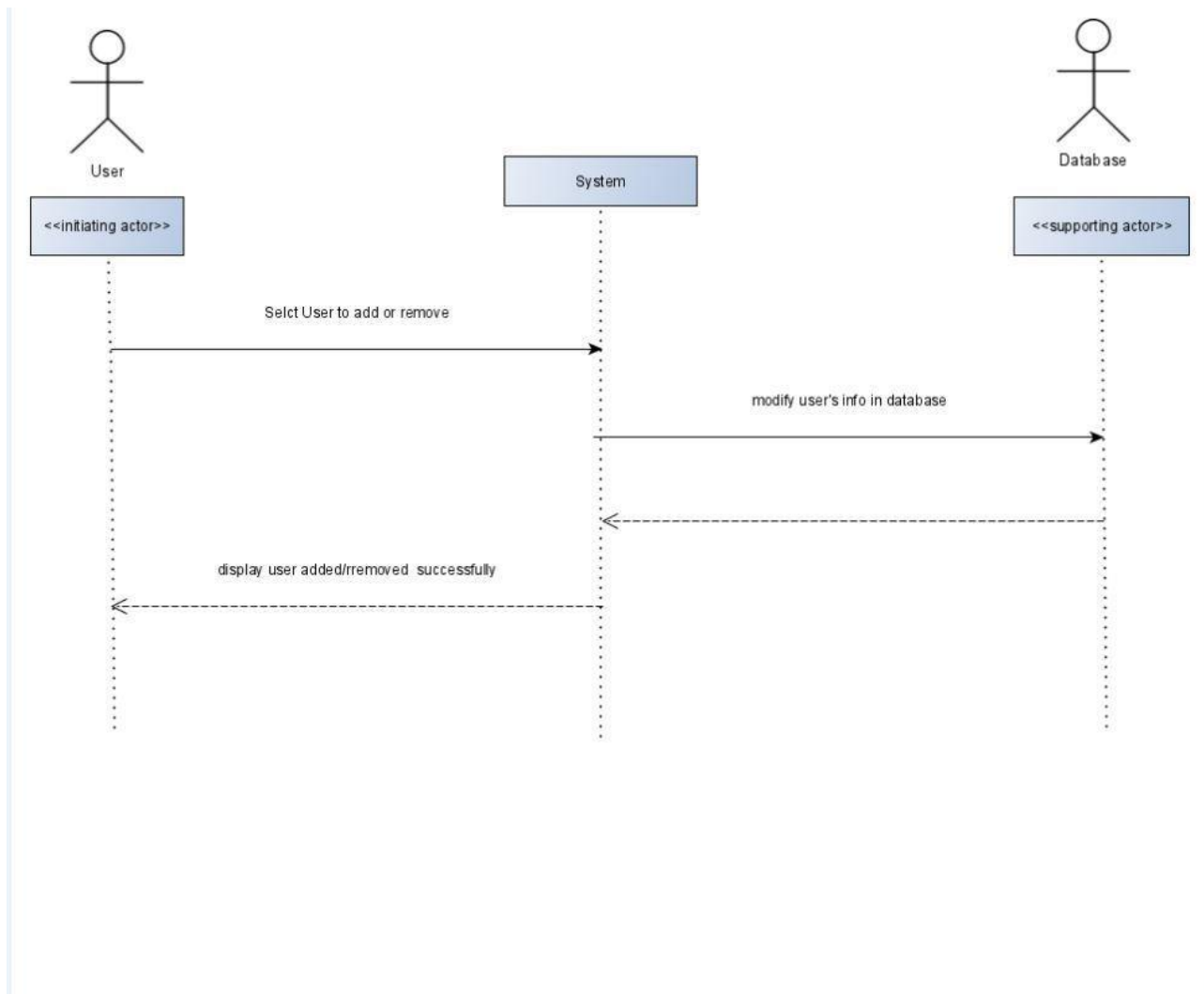
### 8.4 UC-4: Personalized Recommendation



### Personalized Recommendations Explanation:

The end user can enter the movies which he/she has liked in the database whose features are then extracted and it is then sent to a clustering algorithm which will keep track of the preferences of movies and will be used to provide similar movies and hence, provide personalized recommendations.

## 8.5 UC-5: Manage Users



### Manage Users Explanation:

The admin will have access to the user database and can manage the users in this way. The admin can access the database records and add and remove users as they wish from the database if they deem it to be necessary.

## 9. Functional Requirements Specification

The Movie Recommendation System has two active actors and one system(Chatbot API) along with a database.

The general users who use this website/API. They can login to the website, converse with the chatbot about movies, filter movies based on keywords and also get personalized recommendations based on previous history and have discussions with other people on a forum

The admin manage this website. They will be given access to the database to manage users and will also be responsible for moderating the discussion forums.

The Chatbot API will be responsible for the request response cycles between user and system and the database will have user login information along with movie watch history.

## 10. Other Nonfunctional Requirements

### 10.1 Performance Requirements

- Speed: The system should search the results for a query in less than 10seconds.
- Response time: It must have a response time of at least 3seconds.
- Efficiency: The SUD will assign each user a session ID and the system will use a searching algorithm that is efficient to search the database.

### 10.2 Safety Requirements

- The personal data gathered by this website will not be shared with any other party and will remain confidential.
- The user can deny the use of personal information at any point of time.

### 10.3 Security Requirements

i. Usability: The SUD(System Under Development) will have a clean, yet a very stylish, website such that the loading time of different pages can be minimized in order to increase the Browsing speed.

ii. Reliability:

a. Frequency/ Severity of failure: The system must be up 365 days a year with 99% uptime. The system will only be down when the server will be under maintenance.

b. Recoverability: The SUD will have a backup hard drive such that failure of the primary hard drive will not result in the loss of data.

c. Predictability: The server maintenance will generally be carried out during that time when there is minimal traffic, i.e during those hours in which minimum number of users use the website.

iii. Supportability:

a. Compatibility: The SUD will be compatible with internet browsers such as Google Chrome, Mozilla Firefox. It should be compatible with operating systems Windows XP and higher. When talking about mobile browsers, it shall be compatible with Google Chrome, Samsung Internet, Safari

b. Maintainability: The SUD will be maintained by the developers as some updation in the database has to be done by the development team.

## 10.4 Software Quality Attributes

- Availability
- Correctness
- Maintainability
- Usability

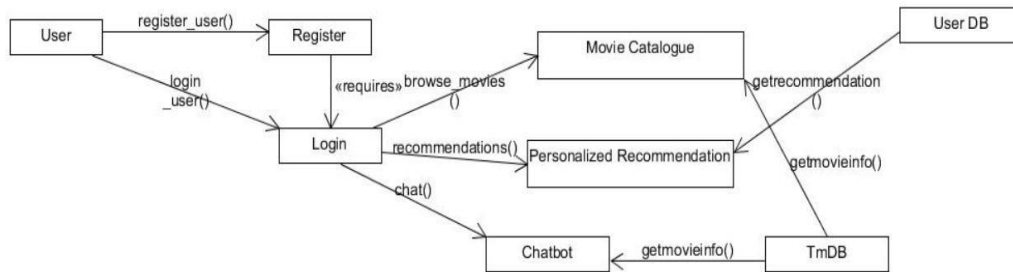
## 10.5 Business Rules

- The user must be logged in to get the personalized recommendations.
- Users can select who can see their data.
- Users can remove their history at any point of time.
- Users can stop sharing their data if they want to at their will.
- The user must be notified about any changes in the terms and conditions.

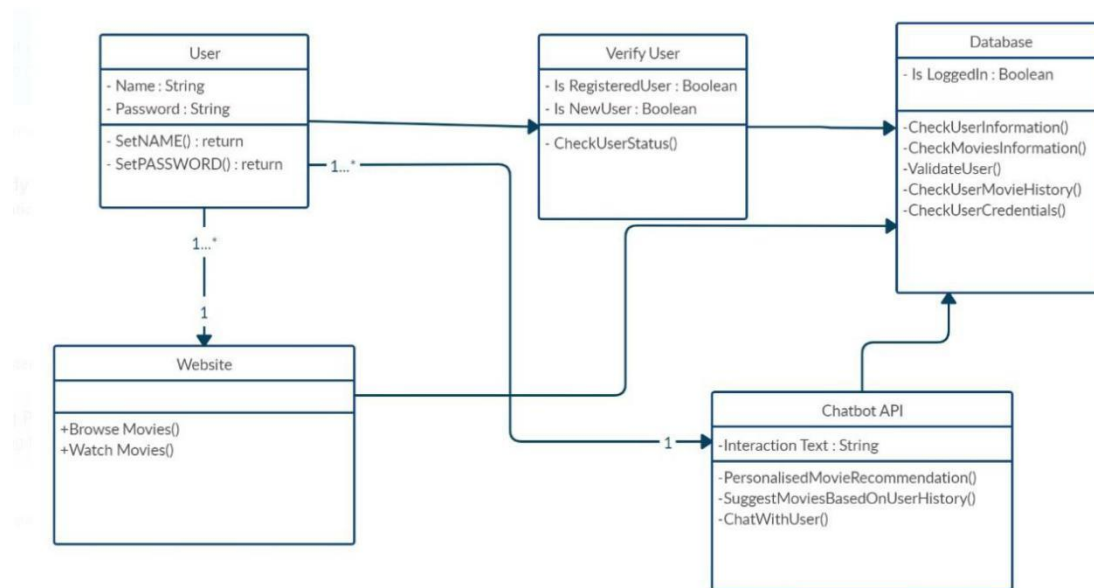


# 11.UML Diagrams

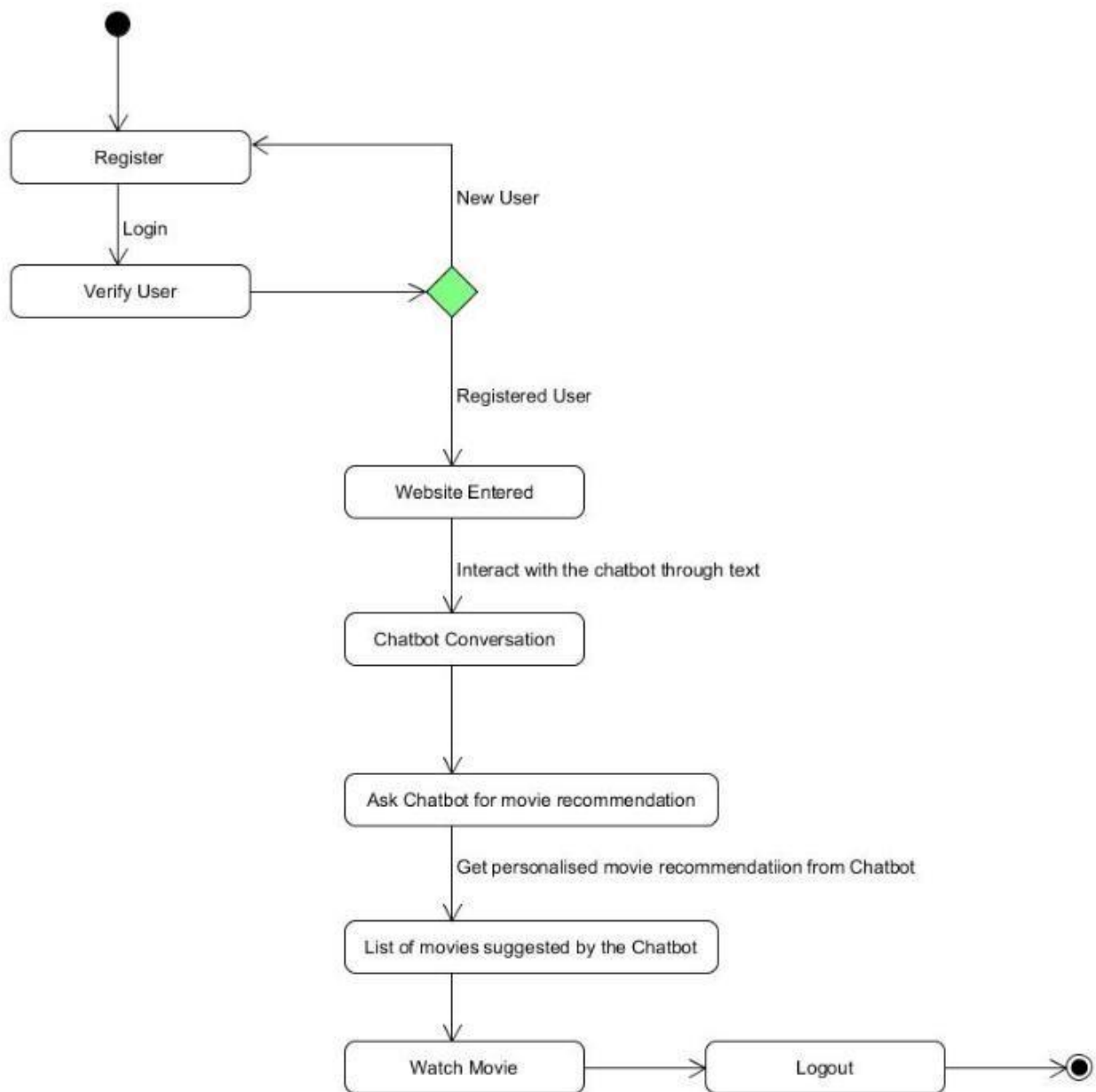
## 11.1 Class Diagram



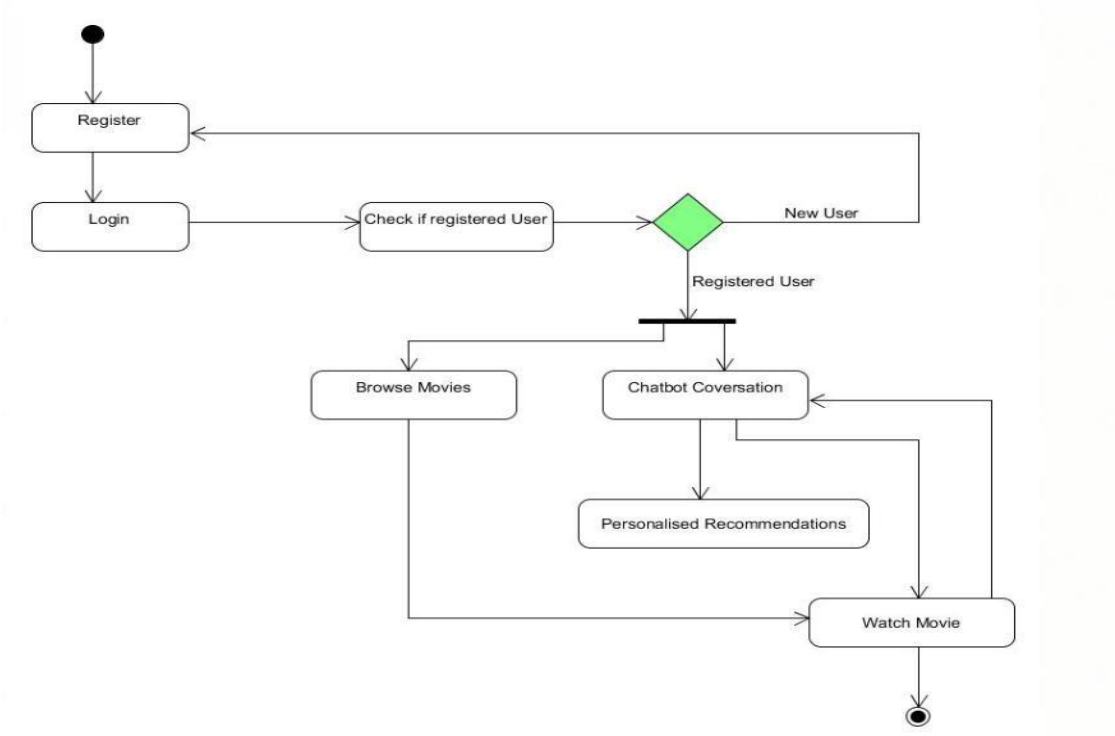
## 11.2 Collaboration Diagram



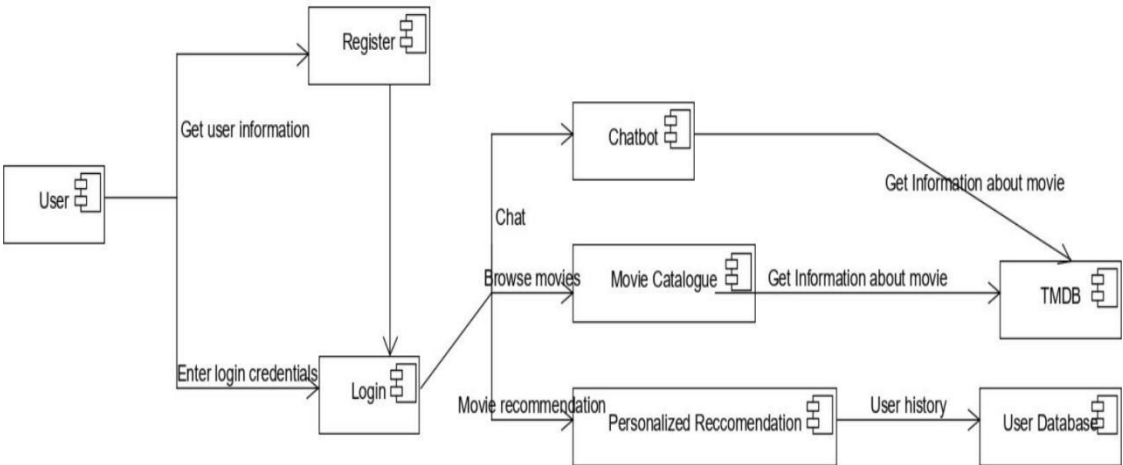
### 11.3 State Diagram



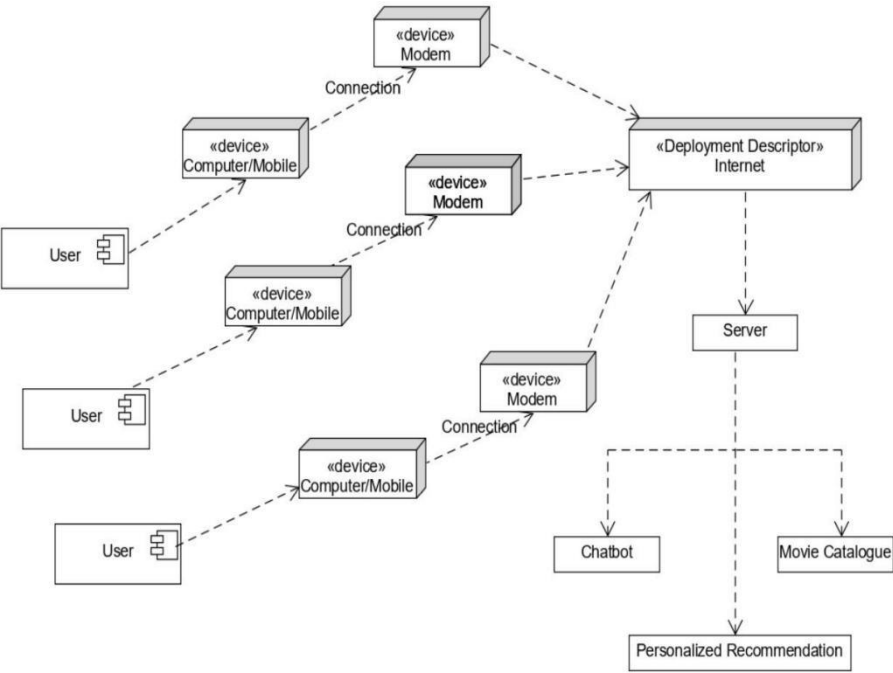
11.4 Activity Diagram



11.5 Component Diagram



# 11.6 Deployment Diagram



## 12.Test Cases

Test Case No	Test scenario	Test Steps	Test Data	Expected Results	Status
1	Register user with valid data	<ul style="list-style-type: none"> <li>Go to site <a href="http://127.0.0.1">http://127.0.0.1</a></li> <li>Click SignUp button</li> <li>Enter valid information</li> <li>Submit Registration form</li> </ul>	Username= Manishr99  Password= manishrreddy	User should be registered into the application	Pass
2	Check Customer Login with valid data	<ul style="list-style-type: none"> <li>Go to site <a href="http://127.0.0.1">http://127.0.0.1</a></li> <li>Enter Username</li> <li>Enter Password</li> <li>Click Submit</li> </ul>	Username= Manishr99  Password= manishrreddy	User should login into the application	Pass
3	Check Customer Login with invalid Data	<ul style="list-style-type: none"> <li>Go to site <a href="http://127.0.0.1">http://127.0.0.1</a></li> <li>Enter Username</li> <li>Enter Password</li> <li>Click Submit</li> </ul>	Username= Manishr99  Password= Reddy14678	User should not login to the application	Fail
4	Chat with chatbot with a query with keywords 'tell' and 'movie'	<ul style="list-style-type: none"> <li>Click on chatbot form and type the query</li> <li>Press Enter</li> </ul>	Query- tell me about the movie Extraction	Fetches information about the movie Extraction	Pass
5	Chat with chatbot with a invalid query with only keyword 'tell'	<ul style="list-style-type: none"> <li>Click on chatbot form and type the invalid query</li> <li>Press Enter</li> </ul>	Query- tell me about Extraction	Does not fetch information about the movie Extraction	Fail
6	Chat with chatbot with a query with keywords 'popular' and 'movie'	<ul style="list-style-type: none"> <li>Click on chatbot form and type the query</li> <li>Press Enter</li> </ul>	Query- show some popular movie	Gives you list of some popular movies which are trending	Pass
7	Chat with chatbot with a query with keywords 'genre' and	<ul style="list-style-type: none"> <li>Click on chatbot form and type the query</li> <li>Press Enter</li> </ul>	Query – show some movies of genre action	Gives you list of some popular movies of action genre	Pass

	‘movies’				
8	Chat with chatbot with a query with keywords ‘kids’ and ‘movie’	<ul style="list-style-type: none"> <li>Click on chatbot form and type the query</li> <li>Press Enter</li> </ul>	Query – movie for kids	Gives you list of some movies which are meant for kids	Pass
9	Chat with chatbot with a query with keywords ‘review’ and ‘movie’	<ul style="list-style-type: none"> <li>Click on chatbot form and type the query</li> <li>Press Enter</li> </ul>	Query – review of movie Ad Astra	Will give you the review of the movie Ad Astra	Pass
10	Chat with chatbot with a query with keywords ‘popularity’ and ‘movie’	<ul style="list-style-type: none"> <li>Click on chatbot form and type the query</li> <li>Press Enter</li> </ul>	Query – popularity of movie Extraction	Will show you a number value of how popular extraction movie is	Pass
11	Personalized Recommendation	<ul style="list-style-type: none"> <li>Login with the username and password</li> <li>After that type names of five movies in the blank spaces provided</li> <li>Then Press Submit</li> </ul>	Query-Extraction Ad Astra Gravity Looper Interstellar	Based on the movies that you have typed, it will recommend you list of movies of similar kind and genre	Pass

## 13. References

1. <https://www.themoviedb.org/>
2. <https://pypi.org/project/tmdbv3api/>
3. <https://docs.djangoproject.com/en/3.0/topics/templates/>
4. <https://docs.djangoproject.com/en/3.0/ref/contrib/admin/>
5. <https://docs.djangoproject.com/en/3.0/ref/class-based-views/base/#>
6. <https://jquery.com/>

## 14. Appendix

### Codes

#### Front-end : (templates)

##### A. Home Page (Index)

```
{% extends 'base.html' % }
{% load static % }
{% block js % }
<head>
<script type="text/javascript" src="/static/js/script.js"></script>
<style type="text/css">
#outer
{
margin-top:3em;
margin-bottom: 4em;
margin-left: 13em;
}
.ip
{
border-top-style: none;
border-left-style: none;
border-right-style: none;
border-bottom-width: 2px;
border-bottom-style: ridge;
border-bottom-color: #999999;
margin: 1em;
background:transparent;
}
.ip:focus
{
outline:none;
```

```

}
body
{
background-image:url('{ % static "img/back1.jpg"% }');
background-repeat:no-repeat;
background-size:cover;
}
#sub
{
margin-left: 43em;
}
.navbar
{
padding-right: 50px;
padding-bottom: 15px;
}
a{
/*background: #c9df8a;*/
text-decoration: none;
}
a:link {
color: white;
text-decoration: none;
font-size:20px;
}
a:visited {
color: white;
}
a:active {
color: yellow;
}
a:hover
{
color:#d3d3d3;
}
#app
{
margin-left:60px;
}

</style>
</head>
{% endblock %}
<body>
{% block content %}
<!-- Image and text -->
<nav class="navbar navbar-light" style="background-color: black;">
<a href="/" style="font-weight: bold; font-family:Brush Script MT, cursive; font-size:

```



```

24px">ZELDA</a>
<a href="/catalogue1/" style="margin-left:-1030px;">Catalogue</a>
<span align="right" style="color:#d3d3d3;font-size:20px;"><a href="/go_to_signup/">Sign
Up/Login</a></span>
</nav>
<br>
<div id="app" class="container top-padding">
<div class="row">
<div class="col-md-12">
<div class="card col-md-6" v-for="message in messages" v-bind:class="{ 'user-message':
message.user, 'chat-message': message.chat_bot, 'offset-md-6': message.chat_bot}">
<div class="card-body">
[[message.text]]
</div>
</div>
</div>
</div>
</div>
<div id="text-box" class="row top-padding">
<div class="col-md-12">
<textarea class="form-control" style="width:110%;margin-top: 10px;" v-
bind:placeholder="placeholder" v-model="input" v-bind:class="{ 'border-danger': send_blank}" v-
on:change="check_content"></textarea>
<i style="margin-right:-90px;margin-top: 12px;" class="fas fa-arrow-circle-right send-btn" v-
on:click="add_message"></i>
</div>
</div>
</div>
{% endblock %}
</body>

```

## B. Sign-up/Login Page

```

{% extends 'base.html' %}
{% load static %}
{% block js %}
<head>
<script type="text/javascript" src="/static/js/script.js"></script>
<link rel="stylesheet" type="text/css" href="{% static 'bootstrap.min.css' %}">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<script src="//code.jquery.com/jquery-1.11.1.min.js"></script>
<style type="text/css">
#outer
{
margin-top:3em;
margin-bottom: 4em;
margin-left: 13em;

```

```

}
.ip
{
border-top-style: none;
border-left-style: none;
border-right-style: none;
border-bottom-width: 2px;
border-bottom-style: ridge;
border-bottom-color: #999999;
margin: 1em;
background: transparent;
}
.ip:focus
{
outline: none;
}
body
{
background-image: url('{ % static "img/back1.jpg" % }');
background-repeat: no-repeat;
background-size: cover;
}
#sub
{
margin-left: 43em;
}
.navbar
{
padding-right: 50px;
padding-bottom: 15px;
}
a{
/*background: #c9df8a;*/
text-decoration: none;
}
a:link {
color: white;
text-decoration: none;
font-size: 20px;
}
a:visited {
color: white;
}
a:active {
color: yellow;
}
a:hover
{

```

```

color:#d3d3d3;
}
#app
{
margin-left:60px;
}
</style>
</head>
{% endblock %}
<body>
{% block content %}
<!-- Image and text -->
<nav class="navbar navbar-light" style="background-color: black;">
<a href="/" style="font-weight: bold; font-family:Brush Script MT, cursive; font-size: 24px;margin-left: -1230px">ZELDA</a>
<a href="/catalogue1/" style="margin-left: -1200px;font-size:22px;">Catalogue</a>
</nav>
<div class="container" style="margin-top:60px;">
<div id="loginbox" style="margin-top:50px;" class="mainbox col-md-6 col-md-offset-3 col-sm-8 col-sm-offset-2">
<div class="panel panel-info">
<div class="panel-heading">
<div class="panel-title">Log In</div>
</div>
<div style="padding-top:30px" class="panel-body">
<div style="display:none" id="login-alert" class="alert alert-danger col-sm-12"></div>
<form action="/login/" method="post" id="loginform" class="form-horizontal" role="form">
{% csrf_token %}
<div style="margin-bottom: 25px" class="input-group">
<span class="input-group-addon"><i class="glyphicon glyphicon-user"></i></span>
<input id="login-username" type="text" class="form-control" name="username"
placeholder="Username" required>
</div>

<div style="margin-bottom: 25px" class="input-group">
<span class="input-group-addon"><i class="glyphicon glyphicon-lock"></i></span>
<input id="login-password" type="password" class="form-control" name="pass"
placeholder="Password" required>
</div>
<div class="input-group">
<div class="checkbox">
<label>
<input id="login-remember" type="checkbox" name="remember" value="1"> Remember me
</label>
</div>
</div>
<div style="margin-top:10px" class="form-group">
<div class="col-sm-12 controls">

```

```

<input type="submit" id="btn-login" class="btn btn-success" style="background-
color:#d9edf7;border:none;color:#31708F" value="Log in">
</div>
</div>
<div class="form-group">
<div class="col-md-12 control">
<div style="border-top: 1px solid#888; padding-top:15px; font-size:85%">Don't have an account!
<a href="#" style="color:#042A59;font-size:15px;" onClick="$('#loginbox').hide();
$('#signupbox').show()">Sign Up Here</a>
</div>
</div>
</div>
</form>
</div>
</div>
</div>
<div id="signupbox" style="display:none; margin-top:50px" class="mainbox col-md-6 col-md-offset-
3 col-sm-8 col-sm-offset-2">
<div class="panel panel-info">
<div class="panel-heading">
<div class="panel-title">Sign Up</div>
<div style="float:right; font-size: 55%; position: relative; top:-23px"><a id="signinlink"
style="color:#042A59" href="#" onclick="$('#signupbox').hide();
$('#loginbox').show()">Login</a></div>
</div>
<div class="panel-body">

<form action="/sign_up/" method="post" id="signupform" class="form-horizontal" role="form">
{ % csrf_token % }

<div class="form-group">
<label for="firstname" class="col-md-3 control-label">First Name</label>
<div class="col-md-9">
<input type="text" class="form-control" name="first_name" placeholder="First Name" required>
</div>
</div>

<div class="form-group">
<label for="firstname" class="col-md-3 control-label">Last Name</label>
<div class="col-md-9">
<input type="text" class="form-control" name="last_name" placeholder="Last Name" required>
</div>
</div>

<div class="form-group">
<label for="email" class="col-md-3 control-label">Email</label>
<div class="col-md-9">
<input type="text" class="form-control" name="email" placeholder="Email Address" required>

```

```

</div>
</div>

<div class="form-group">
<label for="Mobile Number" class="col-md-3 control-label">Username</label>
<div class="col-md-9">
<input type="text" class="form-control" name="username" placeholder="Username" required>
</div>
</div>
<div class="form-group">
<label for="password" class="col-md-3 control-label">Password</label>
<div class="col-md-9">
<input type="password" class="form-control" name="password1" placeholder="Password" required>
</div>
</div>
<div class="form-group">
<label for="Confirm Password" class="col-md-3 control-label">Confirm Password</label>
<div class="col-md-9">
<input type="password" class="form-control" name="password2" placeholder="Confirm Password"
required>
</div>
</div>
<div class="form-group">
<div class="col-md-offset-3 col-md-9">
<input type="submit" id="btn-signup" class="btn btn-info" value="Sign Up">
</div>
</div>
</form>
</div>
</div>
</div>
</div>
</div>
{% endblock %}
</body>

```

### C. Catalogue Page

```

{% extends 'base.html' %}
{% load static %}
{% block js %}
<head>

```

```

<script type="text/javascript" src="/static/js/script.js"></script>
<style type="text/css">
#outer
{
margin-top:3em;
margin-bottom: 4em;
margin-left: 13em;
}
.ip
{
border-top-style: none;
border-left-style: none;
border-right-style: none;
border-bottom-width: 2px;
border-bottom-style: ridge;
border-bottom-color: #999999;
margin:1em;
background:transparent;
}
.ip:focus
{
outline:none;
}
body
{
background-image:url({'% static "img/back1.jpg"% });
background-repeat:no-repeat;
background-size:cover;
}
#sub
{
margin-left: 43em;
}
.navbar
{
padding-right: 50px;
padding-bottom: 15px;
}
a{
/*background: #c9df8a;*/
text-decoration: none;
}
a:link {
color: white;
text-decoration: none;
font-size:20px;
}
a:visited {

```

```

color: white;
}
a:active {
color: yellow;
}
a:hover
{
    color:#d3d3d3;
}
#result
{
padding:15px;
margin-left: 20px;
}
.obj
{
float:left;
padding-left:50px;
}
.text1
{
float:bottom;
font-size:20px;
font-weight: 700;
font-family: Calibri;
}
</style>
</head>
{% endblock %}
<body>
{% block content %}
<!-- Image and text -->
<nav class="navbar navbar-light" style="background-color: black;">
<a href="/" style="font-weight: bold; font-family:Brush Script MT, cursive; font-size:
24px">ZELDA</a>
<a href="/catalogue1/" style="margin-left:-1000px;">Catalogue</a>
<span align="right" style="color:#d3d3d3;font-size:20px;"><a href="/go_to_signup/">Sign
Up/Login</a></span>
</nav>
<br>
<h5 align="center">MOST POPULAR MOVIES OF THIS WEEK</h5>
<div id="result">
<div class="obj">
<img src= "https://image.tmbd.org/t/p/w500{{ poster.0 }}" width="230" height="270" alt="not
found">
<p class="text1">{{ ans.0 }}</p>
</div>
<div class="obj">

```

```

<img src= "https://image.tmbd.org/t/p/w500{ {poster.1} }" width="230" height="270" alt="not
found">
<p class="text1">{ {ans.1} }</p>
</div>
<div class="obj">
<img src= "https://image.tmbd.org/t/p/w500{ {poster.2} }" width="230" height="270" alt="not
found">
<p class="text1">{ {ans.2} }</p>
</div>
<div class="obj">
<img src= "https://image.tmbd.org/t/p/w500{ {poster.3} }" width="230" height="270" alt="not
found">
<p class="text1">{ {ans.3} }</p>
</div>
<div class="obj">
<img src= "https://image.tmbd.org/t/p/w500{ {poster.4} }" width="230" height="270" alt="not
found">
<p class="text1">{ {ans.4} }</p>
</div>
<br>
<div class="obj">
<img src= "https://image.tmbd.org/t/p/w500{ {poster.5} }" width="230" height="270" alt="not
found">
<p class="text1">{ {ans.5} }</p>
</div>
<div class="obj">
<img src= "https://image.tmbd.org/t/p/w500{ {poster.6} }" width="230" height="270" alt="not
found">
<p class="text1">{ {ans.6} }</p>
</div>
<div class="obj">
<img src= "https://image.tmbd.org/t/p/w500{ {poster.7} }" width="230" height="270" alt="not
found">
<p class="text1">{ {ans.7} }</p>
</div>
<div class="obj">
<img src= "https://image.tmbd.org/t/p/w500{ {poster.8} }" width="230" height="270" alt="not
found">
<p class="text1">{ {ans.8} }</p>
</div>
<div class="obj">
<img src= "https://image.tmbd.org/t/p/w500{ {poster.9} }" width="230" height="270" alt="not
found">
<p class="text1">{ {ans.9} }</p>
</div>
</div>
{ % endblock % }
</body>

```



## D. Personalized Recommendation

```
{% extends 'base.html' %}  
{% load static %}  
{% block js %}  
<head>  
<script type="text/javascript" src="/static/js/script.js"></script>  
<style type="text/css">  
#outer  
{  
margin-top:3em;  
margin-bottom: 4em;  
margin-left: 13em;  
}  
.ip  
{  
border-top-style: none;  
border-left-style: none;  
border-right-style: none;  
border-bottom-width: 2px;  
border-bottom-style: ridge;  
border-bottom-color: #999999;  
margin:1em;  
background:transparent;  
}  
.ip:focus  
{  
outline:none;  
}  
body  
{  
background-image:url('{% static "img/back1.jpg"% }');  
background-repeat:no-repeat;  
background-size:cover;  
}  
#sub  
{  
margin-left: 43em;  
}  
.navbar  
{  
padding-right: 50px;  
padding-bottom: 15px;  
}  
a{
```

```

/*background: #c9df8a;*/
text-decoration: none;
}
a:link {
color: white;
text-decoration: none;
font-size:20px;
}
a:visited {
color: white;
}
a:active {
color: yellow;
}
a:hover
{
color:#d3d3d3;
}
#result
{
padding:15px;
margin-left: 20px;
}
.obj
{
float:left;
padding-left:50px;
}
.text1
{
float:bottom;
font-size:20px;
}
.navbar .navbar-light
{
position:fixed;
}
</style>
<!-- <script type="text/javascript">
function Displaydiv()
{
var T = document.getElementById("result");
T.style.display = "block";
}
</script> -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
</head>
{% endblock %}

```

```

<body>
{ % block content % }
<!-- Image and text -->
<nav class="navbar navbar-light" style="background-color: black;">
<a href="/cust_home" style="font-weight: bold; font-family:Brush Script MT, cursive; font-size:
24px">ZELDA</a>
<a href="/catalogue/" style="margin-left:-820px;">Catalogue</a>
<a href="/personal/" style="margin-left:-820px;">Recommendation</a>
<span align="right" style="color:#d3d3d3;font-size:20px;">Welcome
{ { user.first_name } }&nbsp;&nbsp;&nbsp;<a href="/">Logout</a></span>
</nav>

<form action = '/personal/' autocomplete='off' method="post" id="user_choice">
{ % csrf_token % }
<br><br>
<h4 align="center">Enter a few movies of your choice to get personalized
recommendation:</h4><br>
<div id="outer">
<input type="text" class="ip" name="user_movie1" id = "user_movie1" required>
<input type="text" name="user_movie2" class="ip" id = "user_movie2" required>
<input type="text" name="user_movie3" class="ip" id = "user_movie3" required>
<input type="text" name="user_movie4" class="ip" id = "user_movie4" required>
<input type="text" name="user_movie5" class="ip" id = "user_movie5" required>
</div>
<div id="sub">
<input type="submit" value = "Submit" style="padding: 10px 18px 10px 18px;border-
style:none;border-radius: 10px;background-color: #bfbfbf;"></div>
</form>
<div id="result">
<div class="obj">
<img src= "https://image.tmbd.org/t/p/w500{ { poster.0 } }" width="230" height="300" alt="not
found">
<p class="text1">{ { ans.0 } }</p>
</div>
<div class="obj">
<img src= "https://image.tmbd.org/t/p/w500{ { poster.1 } }" width="230" height="300" alt="not
found">
<p class="text1">{ { ans.1 } }</p>
</div>
<div class="obj">
<img src= "https://image.tmbd.org/t/p/w500{ { poster.2 } }" width="230" height="300" alt="not
found">
<p class="text1">{ { ans.2 } }</p>
</div>
<div class="obj">
<img src= "https://image.tmbd.org/t/p/w500{ { poster.3 } }" width="230" height="300" alt="not
found">
<p class="text1">{ { ans.3 } }</p>

```

```

</div>
<div class="obj">
<img src= "https://image.tmbd.org/t/p/w500{ { poster.4 } } " width="230" height="300" alt="not
found">
<p class="text1">{ { ans.4 } }</p>
</div>
</div>
{ % endblock % }
</body>

```

## E. Home page after login

```

{ % extends 'base.html' % }
{ % load static % }
{ % block js % }
<head>
<script type="text/javascript" src="/static/js/script.js"></script>
<style type="text/css">
#outer
{
margin-top:3em;
margin-bottom: 4em;
margin-left: 13em;
}
.ip
{
border-top-style: none;
border-left-style: none;
border-right-style: none;
border-bottom-width: 2px;
border-bottom-style: ridge;
border-bottom-color: #999999;
margin: 1em;
background:transparent;
}
.ip:focus
{
outline:none;
}
body
{
background-image:url('{ % static "img/back1.jpg"% }');
background-repeat:no-repeat;
background-size:cover;
}
#sub

```

```

{
margin-left: 43em;
}
.navbar
{
padding-right: 50px;
padding-bottom: 15px;
}
a{
/*background: #c9df8a;*/
text-decoration: none;
}
a:link {
color: white;
text-decoration: none;
font-size:20px;
}
a:visited {
color: white;
}
a:active {
color: yellow;
}
a:hover
{
color:#d3d3d3;
}
#app
{
margin-left:60px;
}
</style>
</head>
{% endblock %}
<body>
{% block content %}
<!-- Image and text -->
<nav class="navbar navbar-light" style="background-color: black;">
<a href="/cust_home" style="font-weight: bold; font-family:Brush Script MT, cursive; font-size:
24px">ZELDA</a>
<a href="/catalogue/" style="margin-left:-800px;">Catalogue</a>
<a href="/personal/" style="margin-left:-800px;">Recommendation</a>
<span align="right" style="color:#d3d3d3;font-size:20px;">Welcome
{ {user.first_name} }&nbsp;&nbsp; <a href="/">Logout</a></span>
</nav>
<br>
<div id="app" class="container top-padding">
<div class="row">

```

```

<div class="col-md-12">
<div class="card col-md-6" v-for="message in messages" v-bind:class="{ 'user-message':
message.user, 'chat-message': message.chat_bot, 'offset-md-6': message.chat_bot}">
<div class="card-body">
[[message.text]]
</div>
</div>
</div>

</div>
<div id="text-box" class="row top-padding">
<div class="col-md-12">
<textarea class="form-control" style="width:110%;margin-top: 10px;" v-
bind:placeholder="placeholder" v-model="input" v-bind:class="{ 'border-danger': send_blank}" v-
on:change="check_content"></textarea>
<i style="margin-right:-90px;margin-top: 12px;" class="fas fa-arrow-circle-right send-btn" v-
on:click="add_message"></i>
</div>
</div>
</div>
{% endblock %}
</body>

```

## F. Base Page (which all other pages are extending)

```

<!DOCTYPE html>
<html>
<head>
<title>{% block title %}{{ title }}{% endblock %}</title>

<meta charset="utf-8"/>

<link rel="stylesheet" type="text/css" href="/static/css/bootstrap.min.css">
<link rel="stylesheet" type="text/css" href="/static/css/style.css">
<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.0.13/css/all.css"
integrity="sha384-
DNOHZ68U8hZfKXOrtjWvjxusGo9WQnrNx2sqG0tfsghAvtVIRW3tvkXWZh58N9jp"
crossorigin="anonymous">

<script type="text/javascript" src="/static/js/dev-vue.js"></script>
{% block css %} {% endblock %}

{% block js %} {% endblock %}
</head>
<body>

```

```

        {% block content %} {% endblock %}
</body>
</html>

```

## Back-end (Django)

### A. **views.py**

```

from django.shortcuts import render, render_to_response, redirect
from django.template import RequestContext
from django.contrib.auth.models import User, auth
from django.http import HttpResponse, Http404
from django.contrib import messages
import json
from django.views.decorators.csrf import csrf_exempt
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from chatterbot import ChatBot
import operator
import tmdbv3api as tb
from django.contrib.auth import login as auth_login
tmdb = tb.TMDb()
tmdb.api_key = 'ef9d09282b5ec0b221147c2cff9fe58d'
tmdb.language = 'en'
tmdb.debug = True
movie = tb.Movie()

@csrf_exempt
def get_response(request):
    response = {'status': None}

    if request.method == 'POST':
        data = json.loads(request.body.decode('utf-8'))
        message = data['message']
        stop_words = set(stopwords.words('english'))
        word_tokens = word_tokenize(message)
        filtered_sentence = [w for w in word_tokens if not w in stop_words]
        filtered_sentence = []
        for w in word_tokens:
            if w not in stop_words:
                filtered_sentence.append(w)
        #filtered_sentence is the list of processed words in the query

        chat_response = "No keyword used/Invalid response."
        if 'popular' in filtered_sentence:

```

```

movie =tb.Movie()
popular = movie.popular()
string=""
for i in range(4):
    string+=popular[i].title+", "
string+=popular[4].title #
chat_response='Some popular movies right now are '+string

```

```

genre_to_id={"ScienceFiction":878,"Drama":18,"Action":28,"Comedy":35,"Thriller":53,"Horror":27,"Romance":10749,"Fantasy":14,"History":46,"Adventure":12,"Mystery":9648}

```

```

if 'genre' in filtered_sentence:
    discover=tb.Discover()
    i=filtered_sentence.index('genre')
    string=""
    for i in range(i+1,len(filtered_sentence)):
        string += filtered_sentence[i]
    string2=""
    if string in genre_to_id.keys():
        a=genre_to_id[string]
        popular = discover.discover_movies({
"with_genres":a})
        for i in range(4):
            string2+=popular[i].title+", "
            string2+=popular[4].title
            chat_response="Some popular "+string+ " movies right now are
"+string2
    else:
        chat_response="The query should be in the form of: \"suggest me some
movie of genre g\" and g should be from the following: Science Fiction, Drama, Action, Comedy,
Thriller, Horror, Romance, Fantasy, History, Adventure, Mystery"

```

```

if 'kids' in filtered_sentence:
    discover = tb.Discover()
    popular = discover.discover_movies({
'certification_country': 'US',
'certification.lte': 'G',
'sort_by': 'popularity.desc'})
    string = "
    for i in range(4):
        string+=popular[i].title+", "
    string+=popular[4].title
    chat_response='Some popular kids movies right now are '+string

```

```

if 'review' in filtered_sentence:
    movie = tb.Movie()

```



```

k=filtered_sentence.index('review')
if 'movie' not in filtered_sentence[k:]:
    chat_response="The query should be in the form of: \"give review of
movie x\"
else:
    i=filtered_sentence.index('movie')
    string=""
    for i in range(i+1,len(filtered_sentence)):
        string += filtered_sentence[i]
    string=string[:len(string)-1]
    search = movie.search(string)
    chat_response = movie.reviews(search[0].id)
    chat = chat_response[0].entries
    count=0
    chat_response=""
    for i in chat["content"]:
        if count!=300:
            chat_response+=i
            count+=1
        else:
            chat_response+=".... \"+\n\n For more check out:
"+chat["url"]

            break

if 'cast' in filtered_sentence:
    movie = tb.Movie()
    k=filtered_sentence.index('cast')
    if 'movie' not in filtered_sentence[k:]:
        chat_response="The query should be in the form of: \"cast of movie x\"
    else:
        i=filtered_sentence.index('movie')
        string=""
        for i in range(i+1,len(filtered_sentence)):
            string += filtered_sentence[i]
        string=string[:len(string)-1]
        search = movie.search(string)
        chat_response = movie.credits(search[0].id)
        chat = chat_response.cast
        chat_response = "Starring: "
        for i in chat:
            chat_response+=i["name"]+" as "+i["character"]+" || "

if 'popularity' in filtered_sentence:
    movie = tb.Movie()
    k=filtered_sentence.index('popularity')
    if 'movie' not in filtered_sentence[k:]:
        chat_response="The query should be in the form of: \"popularity of
movie x\"

```

```

        else:
            i=filtered_sentence.index('movie')
            string=""
            for i in range(i+1,len(filtered_sentence)):
                string += filtered_sentence[i]
            string=string[:len(string)-1]
            search = movie.search(string)
            chat_response=search[0].popularity

    if 'tell' in filtered_sentence:
        movie = tb.Movie()
        k=filtered_sentence.index('tell')
        if 'movie' not in filtered_sentence[k:]:
            chat_response="The query should be in the form of: \"tell about movie
x\"

        else:
            i=filtered_sentence.index('movie')
            string=""
            for i in range(i+1,len(filtered_sentence)):
                string += filtered_sentence[i]
            string=string[:len(string)-1]
            search = movie.search(string)
            chat_response=search[0].overview

    response['message'] = {'text': chat_response, 'user': False, 'chat_bot': True}
    response['status'] = 'ok'

else:
    response['error'] = 'no post data found'
    return HttpResponse(json.dumps(response),content_type="application/json")

def home(request, template_name="home.html"):
    context = {'title': 'Movie Recommendation ChatBot'}
    return render_to_response(template_name, context)

def go_to_signup(request):
    return render(request,'sign_up.html',{ })

def sign(request):
    if request.method == 'POST':
        fname = request.POST['first_name'];
        lname = request.POST['last_name'];
        uname = request.POST['username'];
        email = request.POST['email'];
        pass1 = request.POST['password1'];
        pass2 = request.POST['password2'];

```

```

        if pass1==pass2:
            if User.objects.filter(username=uname).exists():
                print('Username taken')
            elif User.objects.filter(email=email).exists():
                print("Email taken")
            else:
                user =
User.objects.create_user(username=uname,password=pass1,email=email,first_name=fname,last_name
=lname)
                user.save();
                print('User created')
                context = {
                    'user':user
                }
                return render(request,'logged_in_home.html',context)
        else:
            print('Password not matching')
            return redirect('/go_to_signup')
    else:
        return render(request,'sign_up.html',{ })

def cust_home(request):
    return render(request,'logged_in_home.html',{ })

def login(request):
    if request.method == 'POST':
        uname = request.POST['username'];

        password = request.POST['pass'];

        user = auth.authenticate(username=uname,password = password)
        if user is not None:
            auth_login(request,user)
            context={
                'user':user
            }
            return render(request,'logged_in_home.html',context)
        else:
            return redirect('/login/')
    else:
        return render(request,'sign_up.html',{ })

def catalogue(request):
    movie = tb.Movie()
    popular = movie.popular()
    pop_mov = []
    pop_mov_pos = []

```

```

for i in range(10):
    pop_mov.append(popular[i].title[:27])
    pop_mov_pos.append(popular[i].poster_path)
context={
    'ans':pop_mov,
    'poster':pop_mov_pos
}
return render(request,'catalogue.html',context)

```

```

def catalogue1(request):
    movie = tb.Movie()
    popular = movie.popular()
    pop_mov = []
    pop_mov_pos = []
    for i in range(10):
        pop_mov.append(popular[i].title[:27])
        pop_mov_pos.append(popular[i].poster_path)
    context={
        'ans':pop_mov,
        'poster':pop_mov_pos
    }
    return render(request,'catalogue1.html',context)

```

```

def personalized(request):
    discover = tb.Discover()
    movie = tb.Movie()
    if request.method == 'POST':
        var1 = request.POST['user_movie1']
        var2 = request.POST['user_movie2']
        var3 = request.POST['user_movie3']
        var4 = request.POST['user_movie4']
        var5 = request.POST['user_movie5']
        movie_list=[]
        search1 = movie.search(var1)
        search2 = movie.search(var1)
        search3 = movie.search(var1)
        search4 = movie.search(var1)
        search5 = movie.search(var1)
        var1 = search1[0].id
        var2 = search2[0].id
        var3 = search3[0].id
        var4 = search4[0].id
        var5 = search5[0].id
        movie_list.append(var1)
        if var2 not in movie_list:
            movie_list.append(var2)
        if var3 not in movie_list:
            movie_list.append(var3)

```

```

if var4 not in movie_list:
    movie_list.append(var4)
if var5 not in movie_list:
    movie_list.append(var5)
genre_count={878:0,18:0,28:0,35:0,53:0,27:0,10749:0,14:0,46:0,12:0,9648:0}
for i in movie_list:
    m = movie.details(i)
    genre_id=[]
    for k in m.genres:
        genre_id.append(k["id"])
    for j in genre_id:
        if j in genre_count.keys():
            genre_count[j]+=1
genre_count = dict(sorted(genre_count.items(),
key=operator.itemgetter(1),reverse=True))
all_counts=list(genre_count.values())
top3_genres=[]
for i in range(3):
    for genre, count in genre_count.items():
        if count == all_counts[i]:
            top3_genres.append(genre)
popular = discover.discover_movies({
'with_genres':top3_genres,
'sort_by': 'popularity.desc'})
string = []
poster = []
for i in range(4):
    string.append(popular[i].title)
    poster.append(popular[i].poster_path)
string.append(popular[4].title)
poster.append(popular[4].poster_path)

context = {
'ans':string,
'poster':poster
}
return render(request,'personalized.html',context)
else:
    return render(request,'personalized.html',{ })

```

## B. urls.py

```
from django.contrib import admin
from django.urls import path

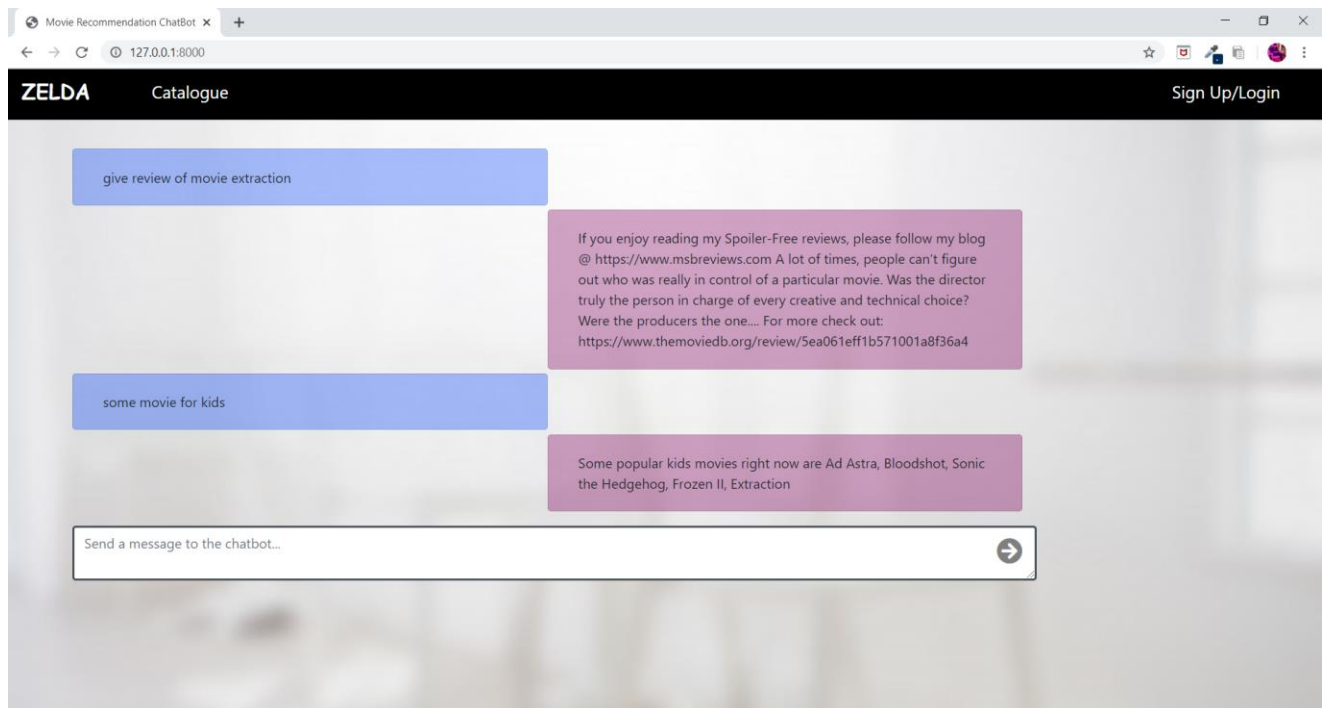
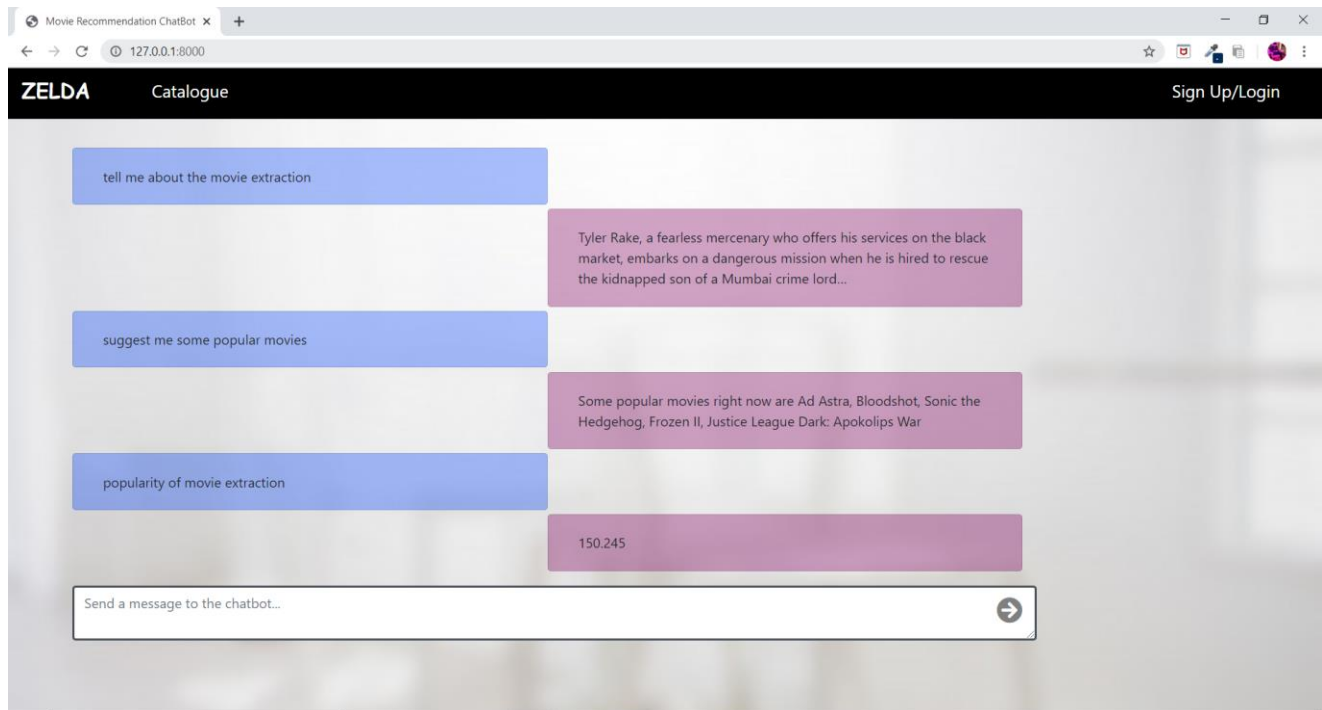
from comedy.views import home, get_response, go_to_signup,
login, sign, catalogue, cust_home, personalized, catalogue1

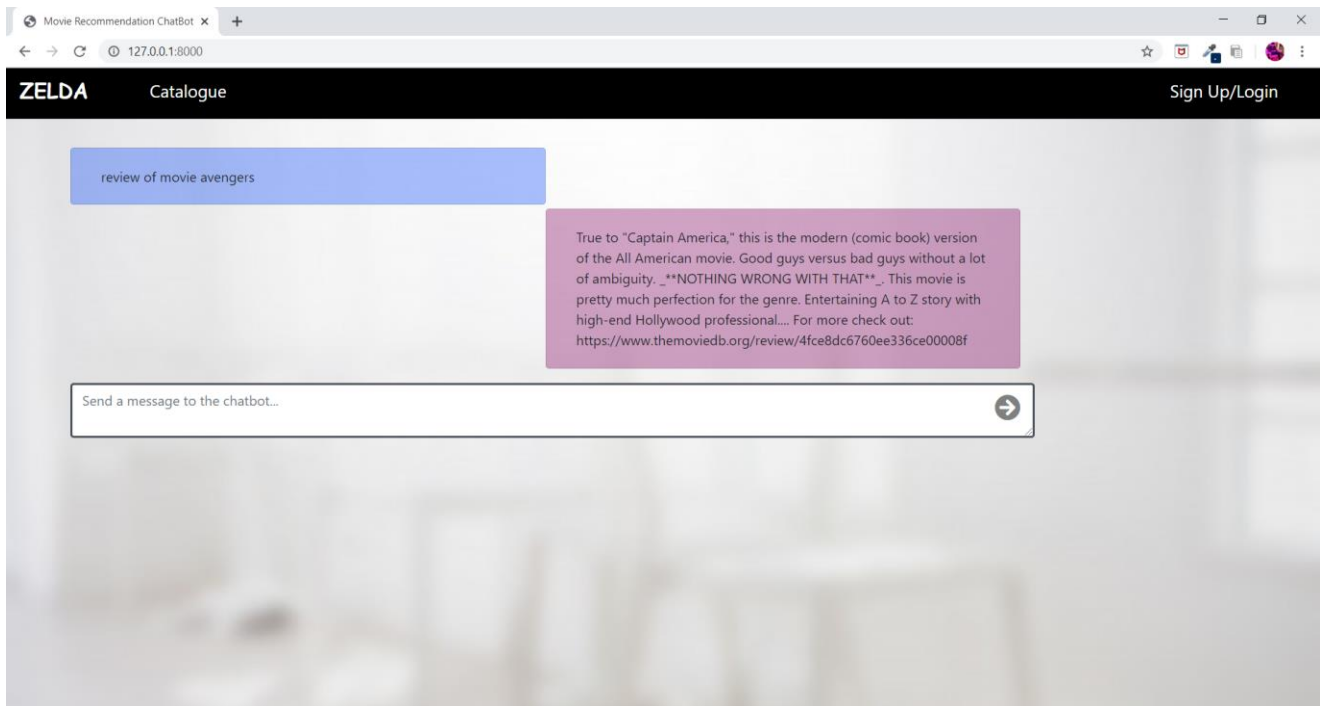
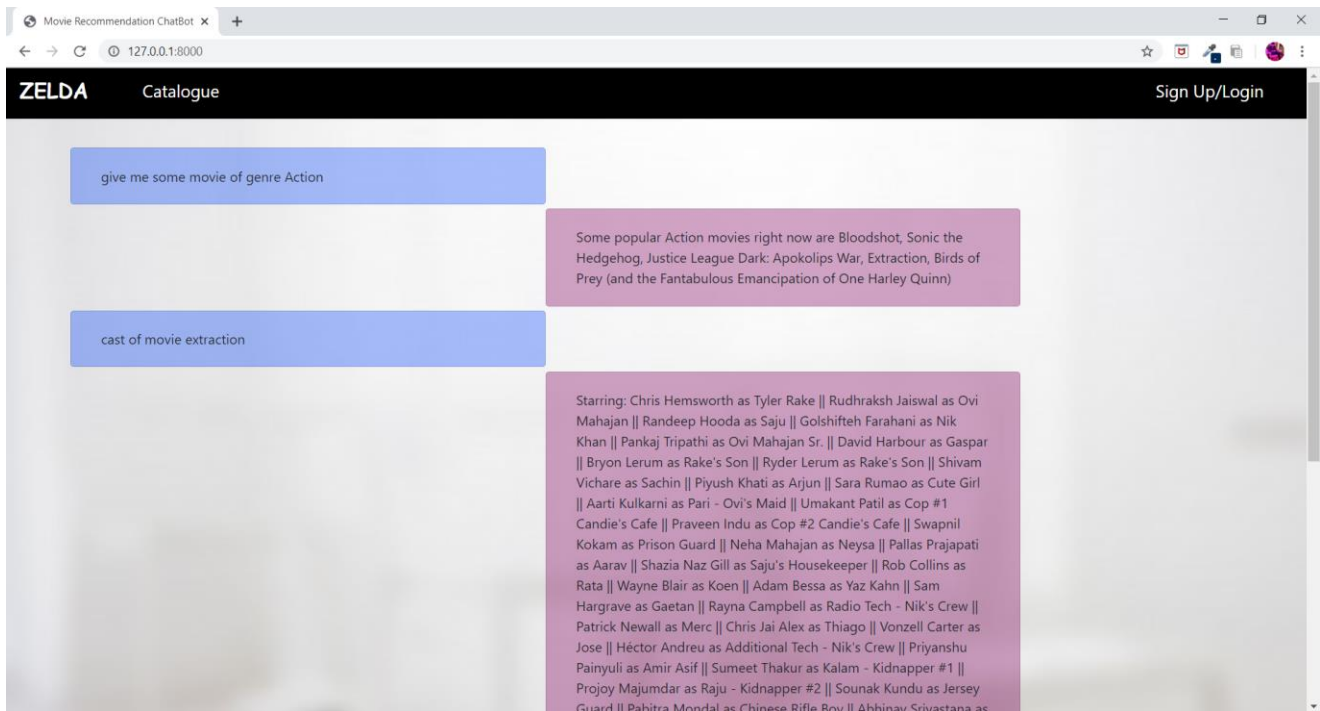
from django.conf.urls.static import static
from django.conf import settings

urlpatterns = [
    path('admin/', admin.site.urls),
    path("", home),
    path('get-response/', get_response),
    path('go_to_signup/', go_to_signup),
    path('sign_up/', sign),
    path('login/', login),
    path('catalogue/', catalogue),
    path('catalogue1/', catalogue1),
    path('cust_home/', cust_home),
    path('personal/', personalized)
]

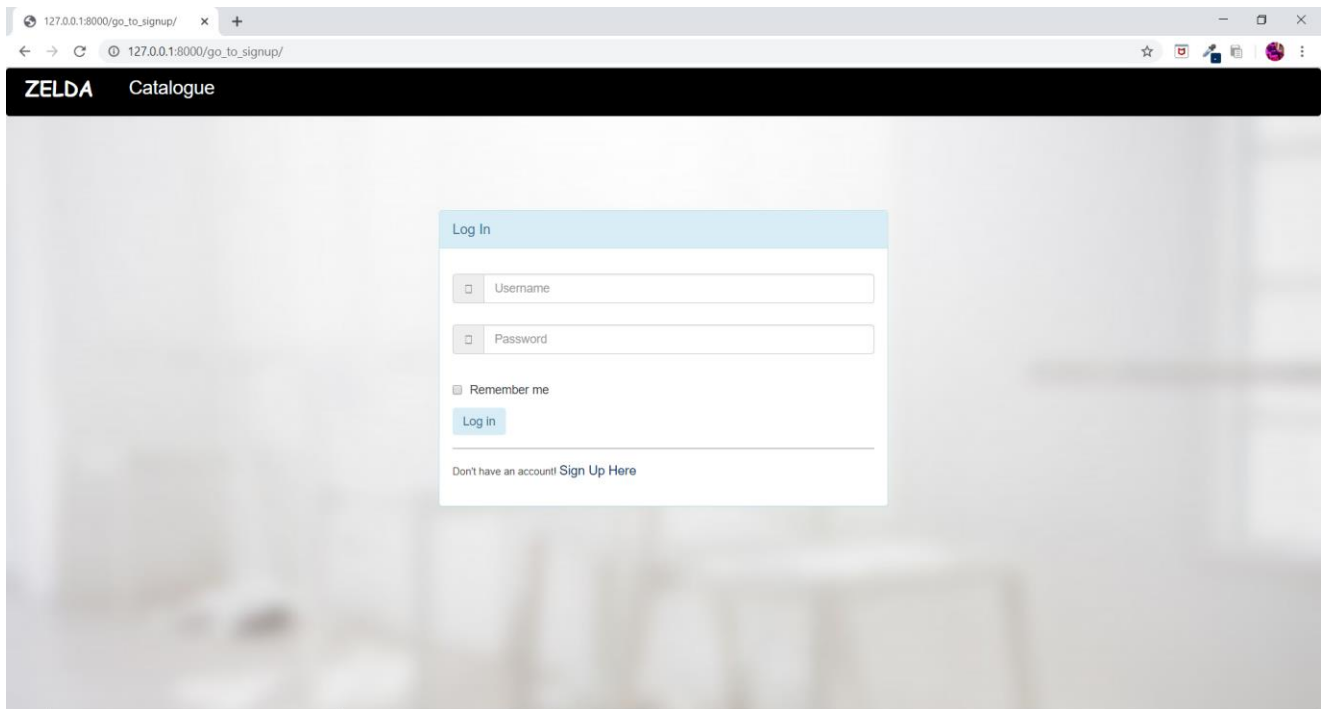
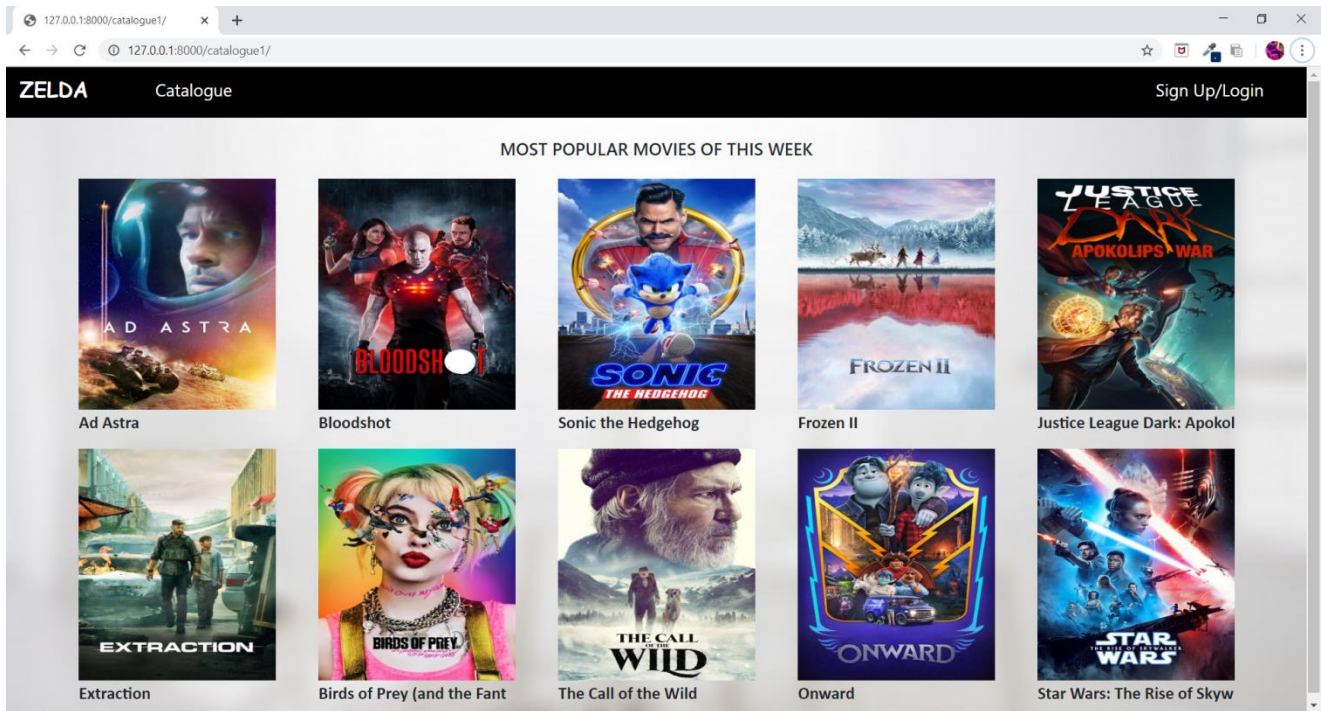
if settings.DEBUG == True:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

## Screenshots









127.0.0.1:8000/go\_to\_signup/#

ZELDA Catalogue

Sign Up

Login

**First Name**

First Name

**Last Name**

Last Name

**Email**

Email Address

**Username**

Username

**Password**

Password

**Confirm Password**

Confirm Password

Sign Up

127.0.0.1:8000/personal/

ZELDA Catalogue Recommendation Welcome Dhinesh Logout

Enter a few movies of your choice to get personalized recommendation:

extraction

avengers

cars

hulchul

hulchul

Submit

127.0.0.1:8000/personal/

127.0.0.1:8000/personal/

ZELDA


Catalogue

Recommendation

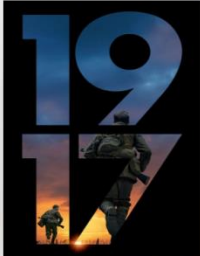
Welcome Dhinesh Logout

Enter a few movies of your choice to get personalized recommendation:


Submit




Extraction



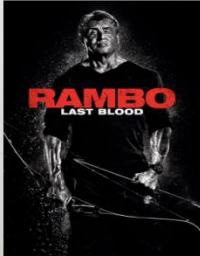
1917



The Dark Knight



Enemy of the State



Rambo: Last Blood

