KARADENİZ TEKNİK ÜNİVERSİTESİ

Bilgisayar Mühendisliği

Proje Raporu

Ders: Software Engineering
Ders Sorumlusu: Dr.Öğr.Üyesi Sedat GÖRMÜŞ
Dili: English

Grup Üyeleri:
Amal AHMED 341762
Jamiu Oluwaseun OJELEYE 359704
Selen KUTANOĞLU 330061

Project Link : https://bitbucket.org/53un/sms

SMS

smart movie suggester

MOVIE RECOMMENDATION APP

Development Report

October, 2019 - December, 2019

Table of Contents

1. Problem Description

Watching movies with friends or by ourself is indisputably one of the best ways of entertainment in our generation. Watching movies is a great way to relax, veer off to the imaginary world and even spend time with our friends and family.

In order to have a fabulous experience, certain questions that need good decision making has to be answered. Let’s imagine a scenario where we are having a movie night with some friends in a movie room, snacks such as popcorn have been prepared for the night, the room’s ambient lights are adjusted, everyone is seated and the question “**which movie should we watch?**” starts to run through everyone’s mind. The search for an answer to this question is usually done by endless browsing of the internet which takes most of our time and the vast majority of the time, we end up not being satisfied with the movie we watched.

Today, there are numerous websites and applications that offer movie recommendation services, but they require users to register before they can use their service. Even to make it worse, some applications charge fees for this service. Nowadays, most people don’t want to commit their time to sign up or register on a website when they are in the mood to watch a movie. People want to have access to services like this within the blink of an eye. In addition, it’s not everyone that can afford to pay a fee for this service and considering the fact that most movie watchers are teenagers, making services like this free for everyone is nice.

In order to solve today's movie selection problem, we are developing a web-based application that is completely free and can recommend movies in a maximum of 2 seconds without requiring any personal data from users. With the help of the application we would be developing, users can spend their time with their friends instead of spending time choosing the right movie.

1.1 Problem Statement

Selecting the right and perfect movie to watch based on a mixture of genre and plot movie watchers have in mind is not easy and this process takes a lot of time. A movie recommendation web-based application that recommends movies (using movies features and some algorithms) to a user without requiring users details can be developed to give users a better experience whenever they want to watch a movie.

1.2 Stakeholders

The general audience of the application we are developing is made up of everyone who likes to watch movies. The potential key users of the app are movie watchers who are hesitant or reluctant to choose a movie due to the fact that they have a particular mixture of genre or plot they want in a movie. Also, anyone who doesn't want to spend a lot of time choosing movies can save time by using our app.

1.3 Actors and Goals

The users can use the application by visiting the website which is easy to use and has great and user-friendly user interface. Services on the application are available with just a click and no sign up or personal data is required to access any service. The main service provided by the application is the recommendation of movies within a short period of time using fast and efficient algorithms. Potential key users' information are given below:

- **User Category:** Teenagers, Adults, Students, Movie night clubs and Cinema goers (who like to watch movies but have no much time for searching for the right movie).

- **User role:** Users should have an idea of the type of movie they want to watch.
  - ➔ In case they chose the recommendation service based on movie name, they need to know at least one movie name to give us as an input.
  - ➔ In case they chose the recommendation service based on some movie parameters(genres, actors etc), they need to have some knowledge about movies to give us as an input.
  - ➔ In case they chose the recommendation service based on movie subject, they need to write down some information about movie to give us as a text input.

- **Subject matter experience:** The user just needs basic/general movie knowledge.

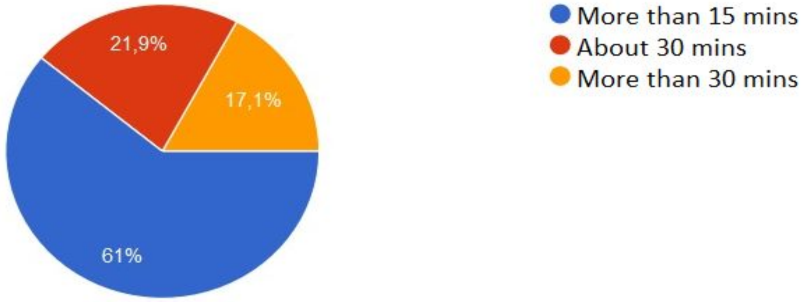
- **Technological experience:** Ability of using basic web site. A novice to web application can use the application.

1.4 Relevant Facts and Assumptions

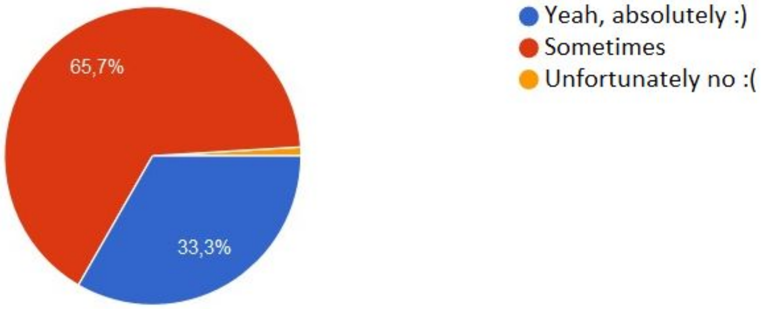
Survey:

Prior to working on the project, we made a survey to get more information on how people find the movie to watch. According to our survey, the process of finding the right movie causes most people to waste more than 15 minutes of their time. In addition, most people are not satisfied with the movie they watch after this 15 minutes endless search even though they limit their search to the type of genre they want. The images below contains the statistical data of the survey we carried out on 105 people:

How much time you spend searching a movie to watch ?
105 answers



Does the movie you choose from any recommendation make you satisfied at the end of it ?
105 answers



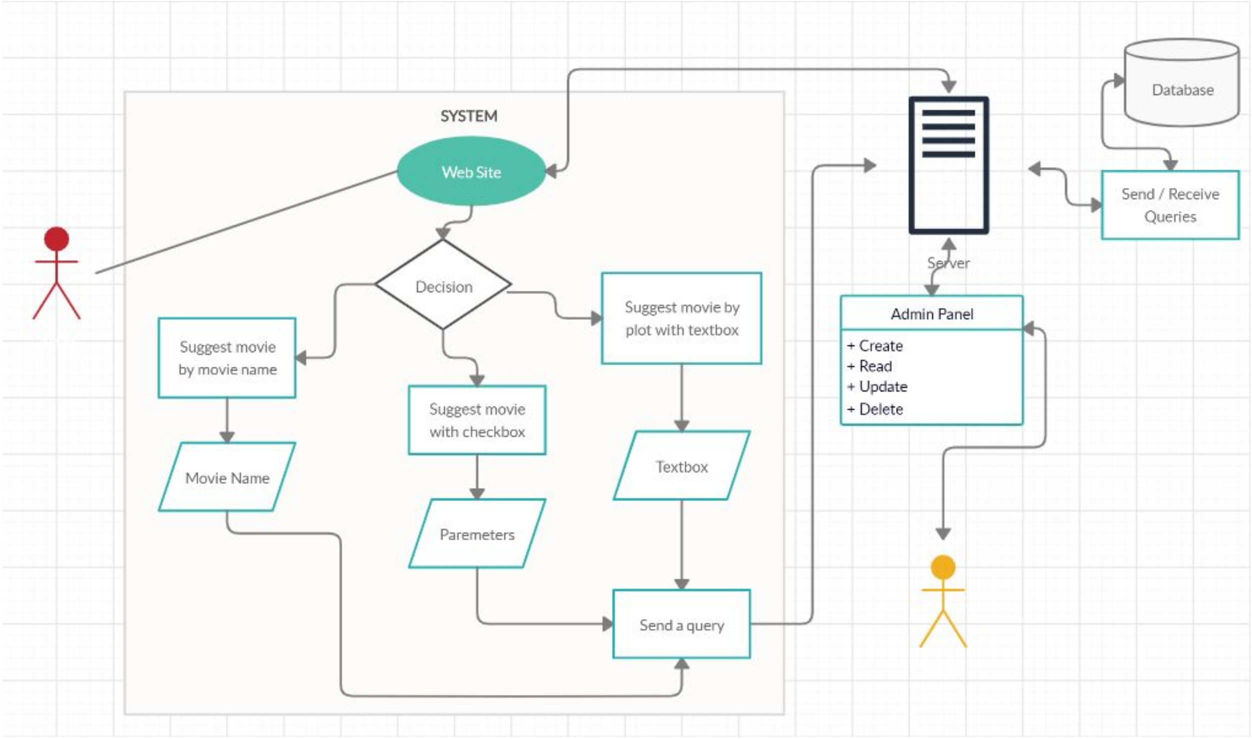
2. Requirements

2.1 Application Use Cases

2.1.1 Use Case List:

1. Just like the first scenario in the problem description, making a movie night is a nice idea but sometimes deciding which movie to watch might be a problem. Using SMS you can solve this problem easily using the “Find movie based on your imagination” section of the application.
2. Another scenario is a situation whereby you came back from work tired and you need a form of entertainment to simmer down your brain. If you decide to watch a movie, the problem of what to watch is enough to make you not to watch any movie. By using our app, you can make this decision as quickly as possible by inputting the information relating to your taste of movie in the app.
3. Students love taking a break from school related activities and one of the activities done by most students during this break is to watch movies. In order to enjoy this break, our application can save them a lot of time by suggesting the best movie for them to watch.
4. Most families have members with different movie taste. During family movie gatherings, selecting a movie that would suit kids, teenagers and adults at the same time is not easy. The next version of our application would be able to find a common movie that can be watched by social groups like this.

2.1.2 Use Case Diagram:



2.2 Specific Functional Requirements

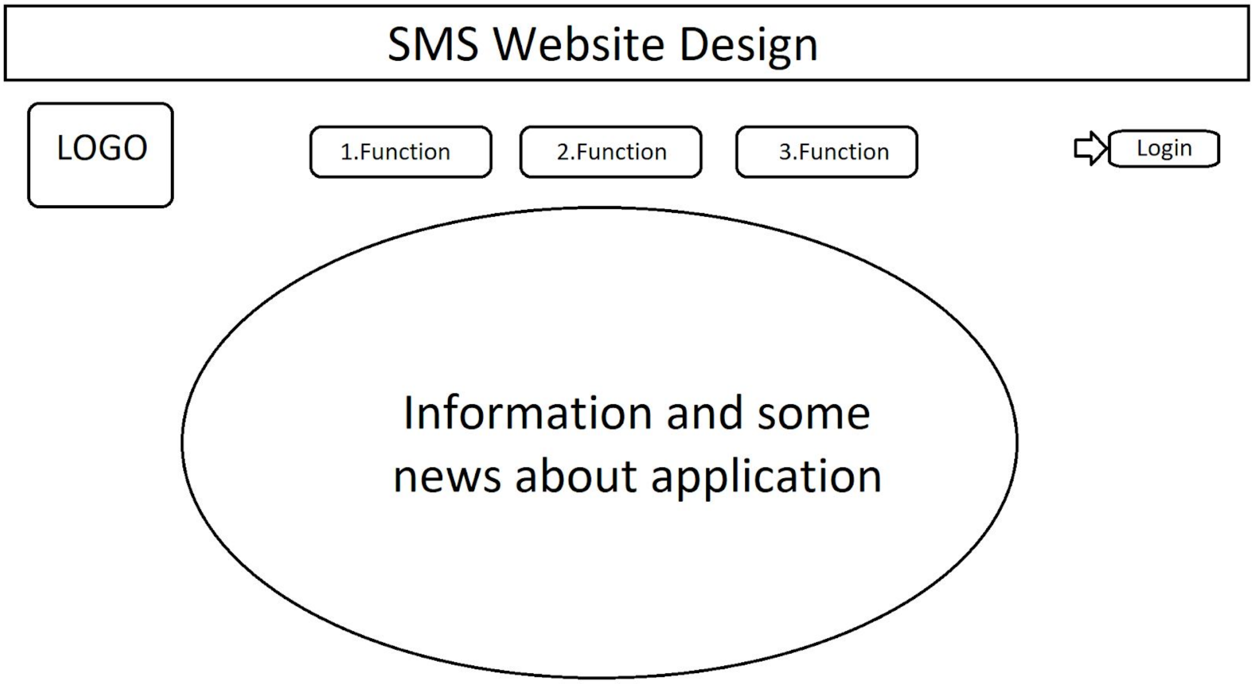
2.2.1 Functional Requirements:

1. Recommendation of movies to users based on similarity to a previously watched movie.
2. Recommendation of movies to users based on the type of genre, actors' names, director's name, etc.
3. Recommendation of movies to user based on the plot the user has imagined in his/her mind.
4. A search of a non-existing movie within the database by the user should not be accepted by the application.

2.2.2 Data Requirements:

1. Users can use the application without entering any private information.
2. Users can input data to find a movie in the application by:
   - ➔ inputting a movie name which is in the form of a string.
   - ➔ checkboxing type of genre, actors' name, director's name, etc.
   - ➔ Inputting the plot the user has imagined in his/her mind in the form of a string.
3. The users are also required to input the number of movies they want the application to recommend.
4. The output displayed by the application to the user is a list of recommended movies by the recommendation system in decreasing order of similarity along side the percentage of similarity.

2.2.3 User Interface Requirements:



1. The logo will represent the project name **‘SMS’** which stands for **“Smart Movie Suggester”**.
2. The main page contains three links for directing users to the web pages containing each recommendation function stated in the functional requirements.
3. Users can choose only one function at a time (i.e the user can only click one of the links at a time).
4. The first link directs the user to a webpage where he/she can find a movie by inputting a previously watched movie name which is in the form of a string.
5. The second link directs the user to a webpage where he/she can find a movie by checkboxing the type of genre, actors’ name, director’s name, etc he/she prefers.
6. The third link directs the user to a webpage where he/she can input the plot he/she imagines in his/her mind in the form of a string.
7. After recommendation, users will take a movie result which includes the name and an IMDB website link of the movie to check out more info about the movie.
8. All pages have similar design so users can use web site easily and efficiently.
9. The Login page is created for just admins of the website so they can update, delete or add a movie.
10. The admins can only login with just their business email address and password.

2.3 Specific Non-Functional(Quality) Requirements

2.3.1 Performance Requirements:

1. **Response Time :**
   Users expect an application to serve them as fast as possible and that’s what the movie recommendation system is aimed to do- by using efficient and optimized techniques and algorithms to suggest movies within a short period of time.
2. **Software Failure:**
   The application might most likely return unsatisfactory result to user due to the fact that the number of movie datasets present in the database is not enough or is outdated. In a situation like this, more datasets has to be added to the database. The performance of the recommendation system is dependent on the number of data present in the database.
3. **Software Success:**
   The recommendation system is successful if it recommends movies that are satisfactory to the user.
4. **Error Detection :**
   The recommendation system should not give any computational or logical error. In case an error is going to occur, the application should detect it.
5. **Scalability:**
   Since the application does not depend heavily on users database and it is more dependent on computational operations, the application is CPU and memory bounded so optimized techniques must be used for computational operations. The system should

be able to handle an increased workload, process and analyse a lot of data, and also adding new features to the system should not be stressful and time consuming.

2.3.2 Security Requirements:

1. **Error Management:** Movies that do not exist in the database must not be searched by the user. If in any situation it was searched, the application should return an error message to the user.
2. **Admin Authentication:** Only the admins must have access to the database from the admin login page.

3. Technologies

3.1 Data Science Python Libraries:

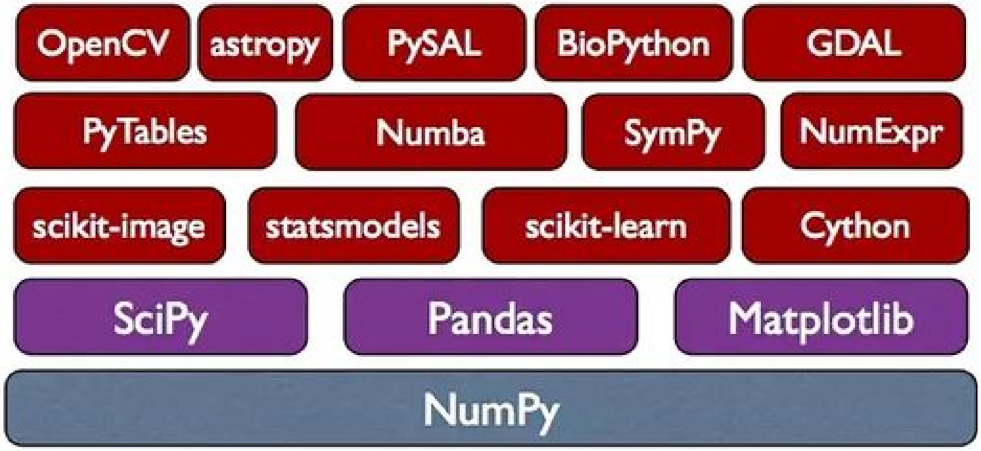
1. **NumPy:** NumPy is a python library that provides fast and space-efficient multidimensional array for efficient mathematical computational operations. Because NumPy stores array elements in memory, it has the ability to perform quick access of data. NumPy also supports quick access of operations such as subindexing, sorting, vectorized mathematical functions, etc[1]. Using the arrays and resources provided by this library would make the recommendation system compute result very fast.

2. **Pandas:** Panda is a python library used for manipulating and analysing data within an array(i.e it's a library for data manipulation and data analysis)[2]. This library is built on NumPy Library. This library can be used to manipulate movie data retrieved from the database.

3. **Scikit-learn:** Is a machine learning python module built on **SciPy** which is a collection of mathematical algorithms and convenience functions(that can be used in building the application) built on the Numpy extension of Python[3].

4. **NLTK:**
   NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language[4]. NLTK package contains a function called **RAKE** which stands for **Rapid Automatic Keyword Extraction algorithm**. It is a domain independent keyword extraction algorithm which tries to determine key phrases in a body of text by analyzing the frequency of word appearance and its co-occurrence with other words in the text. RAKE can be used to extract relevant keywords from users' inputs in the application[5].

The image below shows the hierarchy of the python libraries mentioned above. It shows the libraries that are built on and dependent on others:

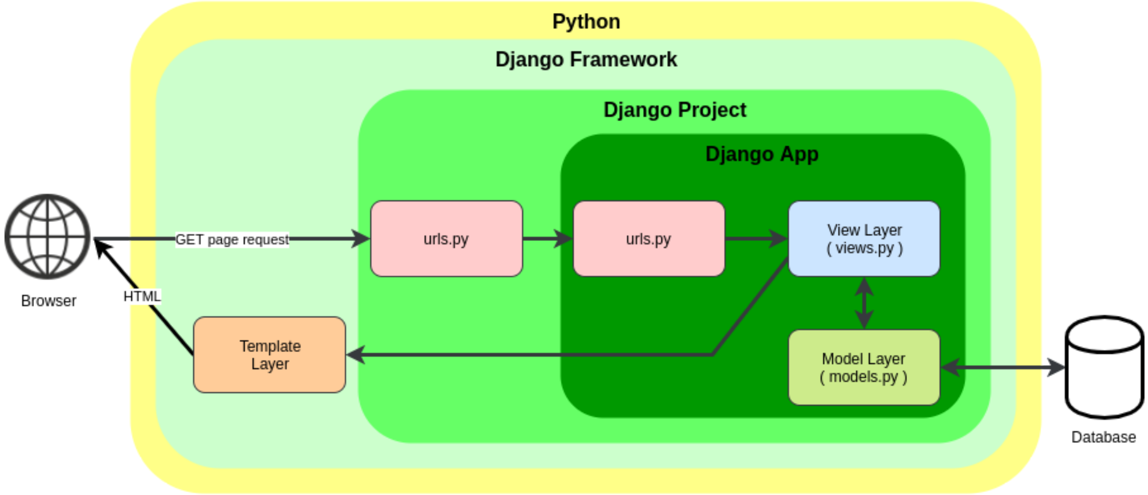


3.2 Django Web Framework:

Django is a free and open-source Python web framework that follows the model view controller (MVC) software architectural pattern which allows for code reusability and high decoupling of the different layers. It includes a default object-relational mapping layer (ORM) which provides a powerful way to interact with the database. Django ORM can be used to interact with application data from various relational databases such as SQLite, PostgreSQL and MySQL[6].

3.3 Django REST Framework:

Django REST framework is a powerful and flexible toolkit for building REST APIs in Django[7]. The diagram below shows a simplified view of the components and layers of a Django Application with a Database storage:

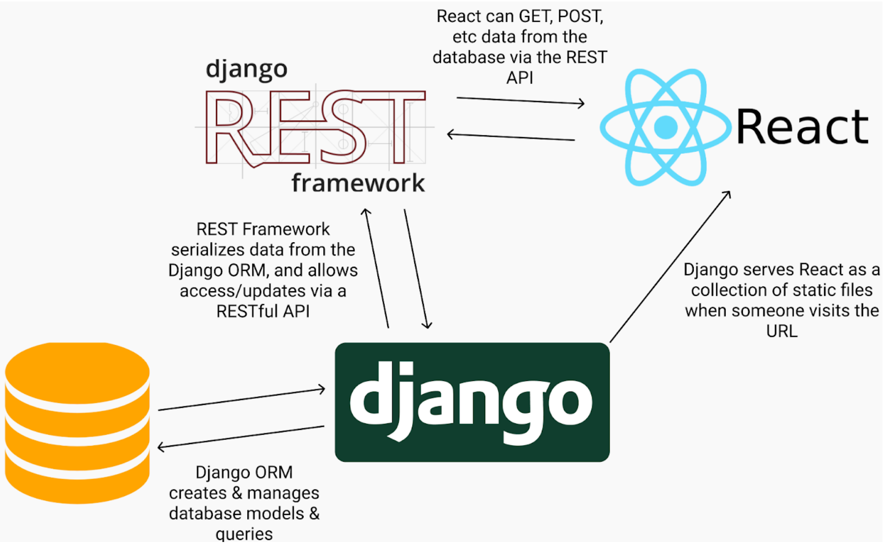


3.4 Axios:

Axios is a javascript library that is used to make HTTP requests/ API calls [9]. It consumes JSON data from Django Restful API which are then displayed on the frontend.

3.5 React Web Framework:

React is a JavaScript framework that allows developers to build web and native frontends for their REST API backends[10].


React can GET, POST, etc data from the database via the REST API
django REST framework
React
REST Framework serializes data from the Django ORM, and allows access/updates via a RESTful API
Django serves React as a collection of static files when someone visits the URL
django
Django ORM creates & manages database models & queries

4. Construction Practices

4.1 Coding Practices

4.1.1 Choice of Programming Language

Two major programming languages would be used for building this application. These PLs were chosen because they have some frameworks and libraries which give developers a major advantage of writing less code and deploying the MVP as soon as possible. Below is an explanation of the PLs based on the subsystem where they are used:

- Front-end:
  There are three main components when it comes to front-end development: **HTML, CSS and JavaScript**. HTML is the structure and content of the site, CSS (Cascading Style Sheets) makes it look pretty, and, lastly, JavaScript is what powers the interactivity of the web application.

- Back-end:
  The back-end development (both the database and the system algorithm) is done using **Python**. Python is chosen because it can be used to build server-side web applications and also It can be used to implement our movie searching algorithm on the backend since it has massive libraries for manipulation of data.
  Most importantly just like it has been explained before, it is chosen because it contains frameworks which can speed-up our progress while building the application.

4.1.2 Programming Conventions

Have you defined coding conventions for names, comments, and layout?

- **Javascript:**
  Naming Conventions: Local variable and function names should be written as **camelCase**.
  Global variables written in **UPPERCASE**
  Constants (like PI) written in **UPPERCASE**

- **HTML and CSS:**
  Hyphens should be used. For example, HTML5 attributes can start with data- (data-quantity, data-price). CSS uses hyphens in property-names (font-size).

- **Python:**
  - ➔ **Classes:**
    Class names should follow the **UpperCaseCamelCase convention** Python's built-in classes, however are typically lowercase words Exception classes should end in "Error"
  - ➔ **Global (module-level) Variables:**

Global variables **should be all lowercase** Words in a global variable name **should be separated by an underscore**.

- ➔ **Instance Variables**:
  Instance variable names **should be all lowercase** Words in an instance variable name should be **separated by an underscore** Non-public instance variables should begin with a single underscore If an instance name needs to be mangled, two underscores may begin its name.
- ➔ **Methods:**
  Method names **should be all lowercase** words in a method name **should be separated by an underscore** Non-public method should begin with a single underscore If a method name needs to be mangled, two underscores may begin its name.
- ➔ Method Arguments:
  Instance methods should have their first argument named ‘self’. Class methods should have their first argument named ‘cls’.
- ➔ **Functions:**
  Function names **should be all lowercase** Words in a function name **should be separated by an underscore**.
- ➔ **Constants:**
  Constant names **must be fully capitalized** Words in a constant name **should be separated by an underscore**.

4.1.3 Error Handling Techniques

Expected errors that can be caused by user inputs are handled from the user interface by preventing users from inputting abnormal inputs into the system.

Non-expected errors that can occur from the backend system are prevented with the use of exceptions or/and conditional statements which are used alongside error codes or messages.

4.1.4 Debug Procedure

The debugging technique that would be used throughout in both the frontend and backend development of the application is the “Print-driven” debugging technique. This technique involves us printing the coding dependencies of expected error location, based on different pieces inside the code(variables, loop iterations, control structures, etc.) so that we can compare the expected vs current values of variables/ control structure paths etc.

This technique is easy to use for smaller sized programs and since our application is decomposed into subsystems which are then divided into smaller programs, this technique can and would be used by every member of the project team to debug all the programs we would be working on.

4.1.5 Test Procedure

The system would be tested by key-users of the application and feedbacks gotten from the key users would be used to improve the application.

4.2 Teamwork and Quality Assurance

- **Integration procedure:**
  Each member of the team are required to follow the coding standards agreed on when writing the code for their part of the project. After the completion of any stage of their code, every member of the group are required to pull request their code using a revision control tool. Every code sent to the main repository are reviewed and merged with the already existing codes in a develop repository by a reviewer.

- **Pair Programming or Individual Programming?**
  Programmers(team members) might be required to work in pairs/ group while writing the test cases for their code. Thereafter, they can work individually while working on their part of the project. The combination of working in group and individual is used in order to make every group member to keep track and have an idea of other member's work.

- **Test case coding:**
  Programmers(team members) must write a test case for their codes before writing the code itself. This method helps programmers to fathom what they are working on before writing a code for the project.

- **Unit testing:**
  It is not mandatory for programmers to write a unit test for their code but if they can, it would ease the work of every member of the during integration testing.

- **Debugging:**
  Programmers are required to debug their code, ensure that it is error/ bug free, and make sure it functions accurately before sending their code for review.

- **Integration-testing:**
  Programmers are not required to perform integration test on their code before they check it into the repository. The integration test of the code would be done as a group after the integration of every code at the end of the project.

- **Code Reviewing:**
  Both the Instant code review(also known as Pair Programming) and the Asynchronous code review (also known as tool-assisted code review) method would be used to review and inspect the codes submitted by a member of the team before checking them into the repository.

The asynchronous code review involves inspecting codes on a revision control tool(such as Git). This method involves a reviewer(usually the best programmer in the group) reviewing each member's code.

The Pair programming method would be used while writing test cases as a group. In addition to this, it would be used to review the codes of the reviewer that inspects other members code on the revision control tool.

4.3 Tools Used

- **Revision control tool:**
  The revision control tool that would be used for maintaining and merging the codes written by every team member is **Git**. The version control repository hosting service that would be used is **Bitbucket**[11]. Bitbucket is web-based and it is chosen over other hosting service due to the fact that it offers an unlimited free private repository service and it is easy to set up.

- **Programming Language Version:**
  - Backend: Python version: 3.6
  - Frontend: Javascript: Its version is dependent on the browser used.

- **Frameworks:**
  Frontend:
  1. **React UI framework:** It is an open-source JavaScript library which is **used** for building user interfaces specifically for single page applications.
     - Version: 0.1.0
  2. **Antd framework:** It is a React UI library that has a plethora of easy-to-use components that are useful for building elegant user interfaces.
     - Version:

  Backend:
  1. **Django Framework:**
     - Version: 2.7

- **Technologies/ Tools Limitations:**
  A programmer is not bound to using a specific technology or tool such as (editor, refactoring tools, debugger, test framework, syntax checker, etc). Every programmer is free to choose any tool they want to use to carry out the tasks they have to complete.

5. Software Architecture and Design

5.1 Design Practices

A Heuristic approach which is understanding a process, devising a plan, carrying out the plan and looking back at the former steps is used to build the application. Divide and Conquer method is also used to divide the system into subsystems which are also divided into sub-programs/ sub-components. These sub-programs are then built and integrated with one another after their completion till the final system is built.

5.2 Overview and Subsystem Decomposition

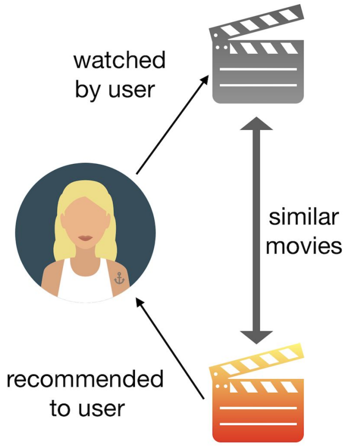
As explained earlier, the system consists of three major subsystems which are:

1. Frontend: which is the user interface.
2. Backend: which consists mainly of the movie database and
3. Movie Recommendation System:which consists of the implementation of the algorithm used for recommending movies to the users.

5.2.1 Movie Recommendation System

The recommendation system is a content based filtering system that recommends similar movie based on the movies a user likes. The content based filtering algorithm involves using a series of discrete characteristics(such as genre, director and actors names, plot, etc) of a movie to find movies with **similar** properties.

In order to find the similarities between two vectors/ samples, a technique known as the **Cosine Similarity** can be used. This technique measures the similarities between two samples by projecting the samples as vectors in an n-dimensional space and finding the cosine of the angle between these two vectors. The resulting cosine of the angle between these vectors is the measure of similarity between the vectors. Also, the range of this result is always between [0,1]. If the resulting value is 0, then the two samples have no features or characteristics in common and if it is a 1, then the two samples are completely similar to one another.



Mathematical Model:

Assuming we have two movies item 1 and item 2 which has a genre, director and actors names, and a plot, in order to find the similarity between these two movies, the following steps below has to be followed:

1. **Data cleaning:**

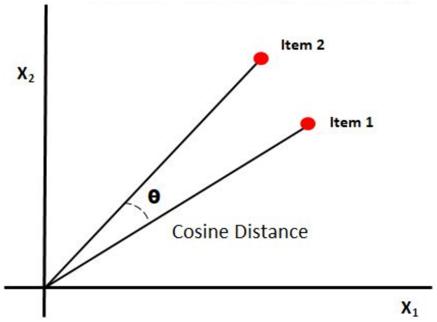
   The plot of a movie contains a lot of irrelevant words so it has to be cleaned by extracting the keywords from it. In order to extract the relevant words from the plot, a **NLP**(Natural Language Processing) algorithm has to be used. Thereafter, genre, actor and director names and the already processed plot words can be merged together to form a single feature.

2. **Modelling:**
   In this stage, the dataset is encoded as integers for use as input to our algorithm. This process is known as feature extraction (or vectorization).
   During feature extraction, a **bag of words**(which is the vocabulary of all the unique words occurring in the dataset) is created then a vectorization technique called the **Count Vectorizer** can be used for this encoding. Countvectorizer counts the number of times a token present in the bag of words shows up in a movie/item data and uses this value to form a vector.
   Illustration of feature extraction: Using the two texts shown below,
   - Text 1: The quick brown fox
   - Text 2: Jumps over the lazy dog

BOW	Jumps	The	brown	dog	fox	lazy	over	quick	the
Text 1	0	1	1	0	1	0	0	1	0
Text 2	1	0	0	1	0	1	1	0	1

Text 1 vector is <0,1,1,0,1,0,0,1,0> and Text2 vector is <1,0,0,1,0,1,1,0,1>

After feature extraction, each vector derived is projected in an n-dimensional space and the cosine of the angle between them is computed using the concept of **dot product**. The dot product between two vectors is equal to the projection of one of them on the other.



$$\boldsymbol{u} \cdot \boldsymbol{v} = [u_1\ u_2\ \dots\ u_n] \cdot \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = u_1 v_1 + u_2 v_2 + \dots + u_n v_n = \sum_{i=1}^{n} u_i v_i$$

Dot Product

Projection of items(movies) in 2-D

$$similarity = cos(\theta) = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{u}\|\,\|\boldsymbol{v}\|} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2}\sqrt{\sum_{i=1}^{n} v_i^2}}$$

$$Where\ similarity\ \mathcal{E}[0, 1]$$

5.2.2 Database

The Database Management System that would be used is SQLite. SQLite works great as the database system for a low traffic websites (a website that gets requests fewer than 100,000) like our web application. We chose SQL over other DBMSs due to the fact that it is fast, reliable, simple and it requires no configuration or maintenance. We find SQLite as an optimum DBMS for now.

The database backend is designed with **Django Framework** which supports connectivity to most relational database. This framework helps web applications to access data from the database using python's object known as **models**. The data queried from the database using this framework are in querysets format which can be converted to a JSON format using a **serializer**. These JSON format data are then consumed in the frontend.

5.2.3 User Interface(Front End)

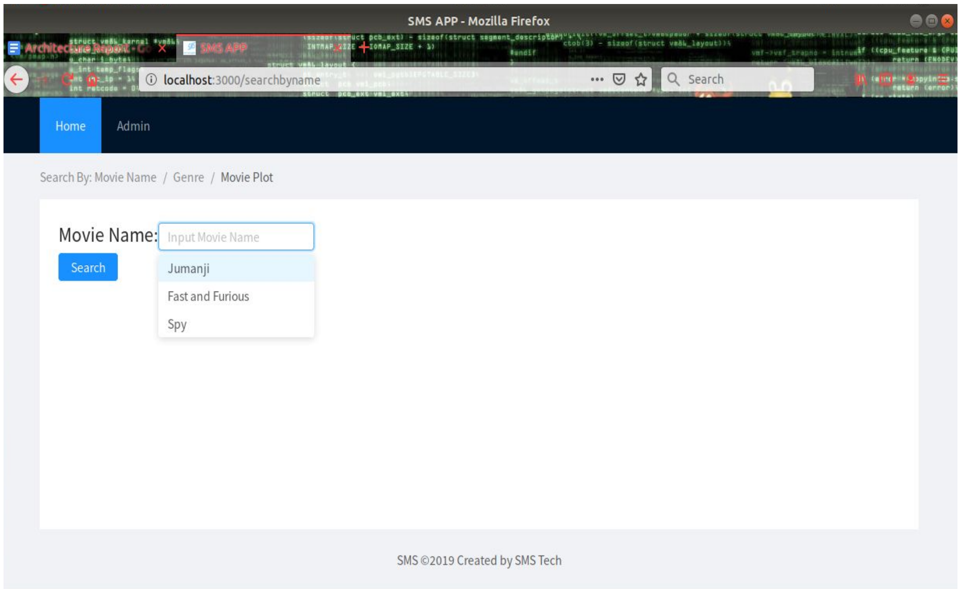
As explained previously, the user interface is designed using **React framework**[9] alongside **Antd framework**[12]. The UI is designed using a OOD(Object Oriented Design) technique which involves creating necessary layouts, components and containers first which are then used to create web pages. This method of design helps in code reuse since most pages contain the same/ similar layouts and containers.

The web application contains of mainly of the **User section** and **Admin section**. which the following pages shown below:

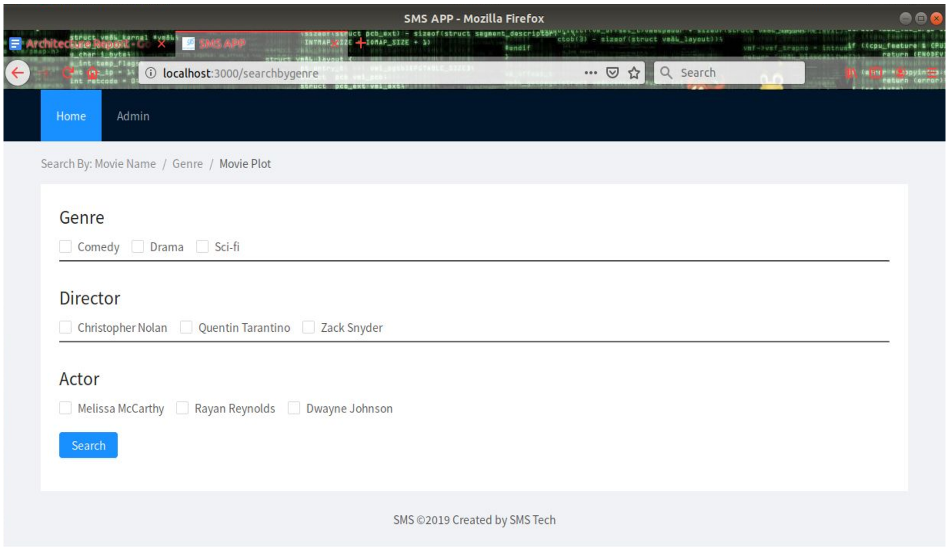
- **User Section:**

  This section of the application provides service to the user. The pages in this section includes:

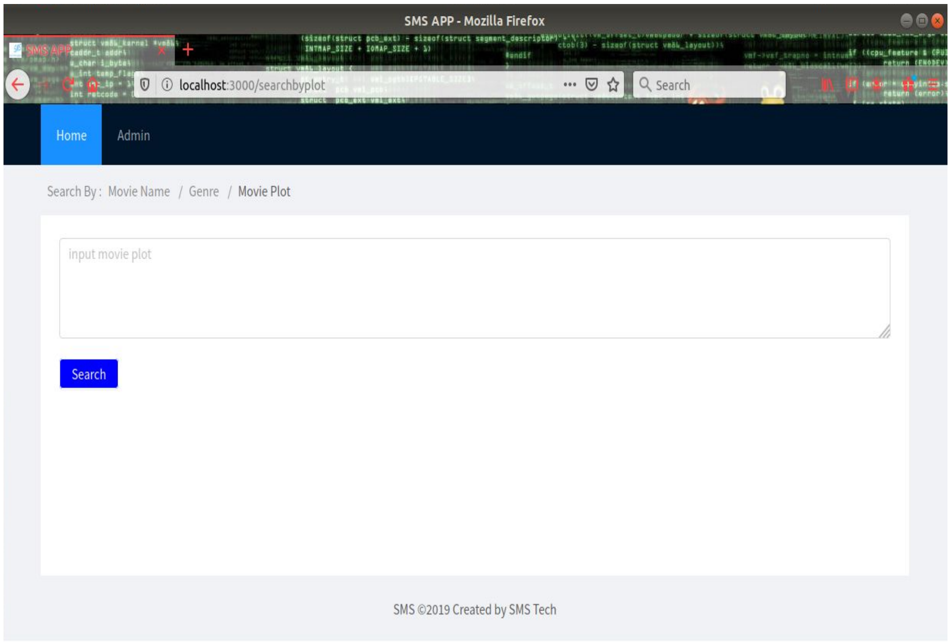
  1. Search by movie name:


SMS APP - Mozilla Firefox
Architecture Report - Go
SMS APP
localhost:3000/searchbyname
Search
Home
Admin
Search By: Movie Name / Genre / Movie Plot
Movie Name:
Input Movie Name
Search
Jumanji
Fast and Furious
Spy
SMS ©2019 Created by SMS Tech

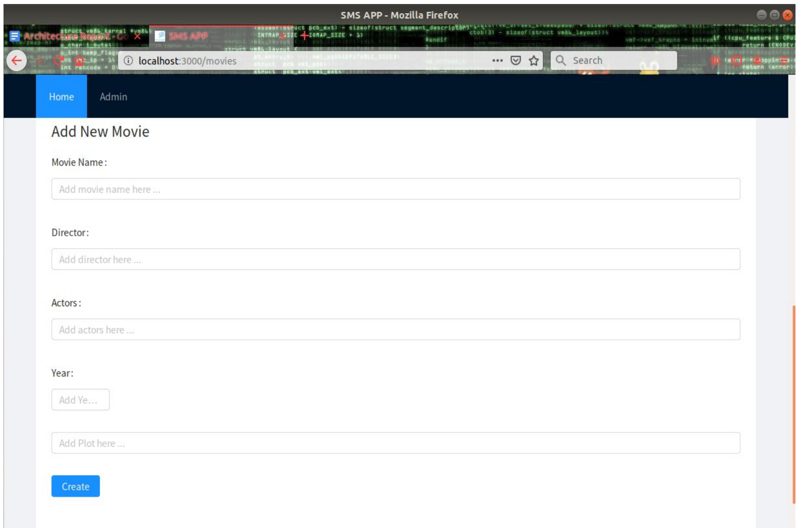
2. Search by genre:

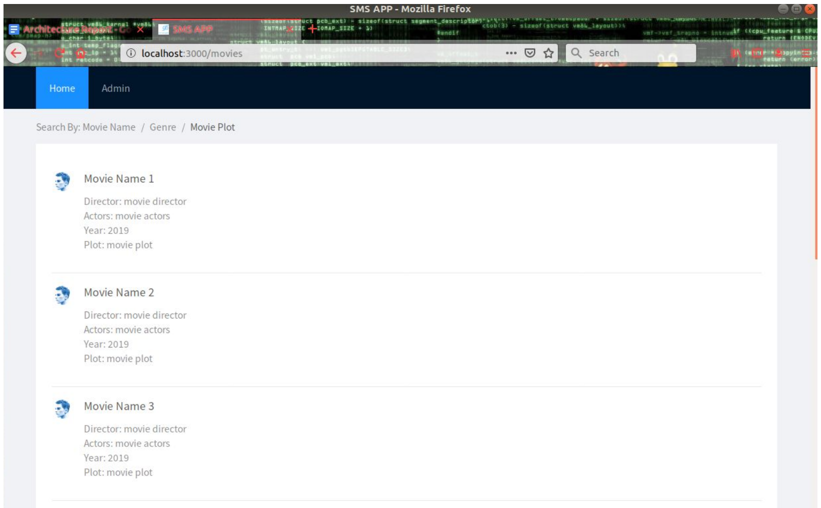

SMS APP - Mozilla Firefox
Architecture Report - Go
SMS APP
localhost:3000/searchbygenre
Search
Home
Admin
Search By: Movie Name / Genre / Movie Plot
Genre
□ Comedy □ Drama □ Sci-fi
Director
□ Christopher Nolan □ Quentin Tarantino □ Zack Snyder
Actor
□ Melissa McCarthy □ Rayan Reynolds □ Dwayne Johnson
Search
SMS ©2019 Created by SMS Tech

3. Search by plot pages:


SMS APP - Mozilla Firefox
SMS APP
localhost:3000/searchbyplot
Search
Home
Admin
Search By : Movie Name / Genre / Movie Plot
input movie plot
Search
SMS ©2019 Created by SMS Tech

- **Admin Section**:This section of the application provides services (such as adding and deleting movie operations) to the admin. The pages in this section includes:
    1. Movies List and Creation Page:



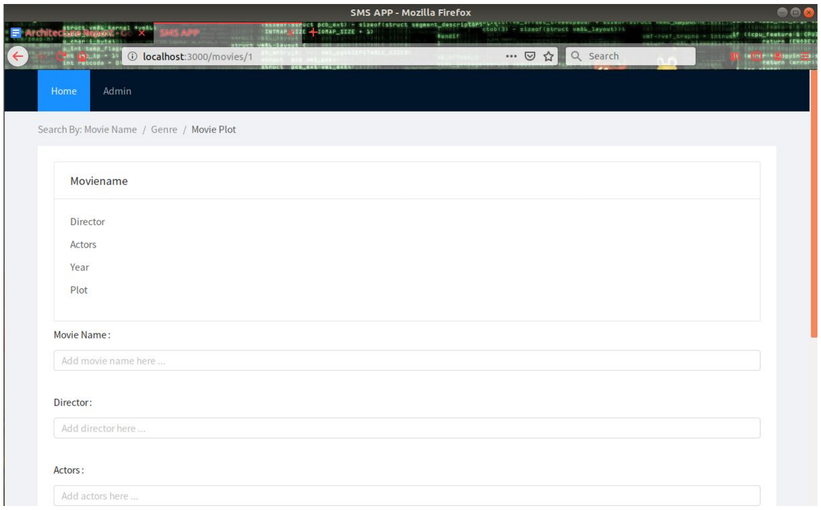

SMS APP - Mozilla Firefox
localhost:3000/movies
Search
Home
Admin
Search By: Movie Name / Genre / Movie Plot

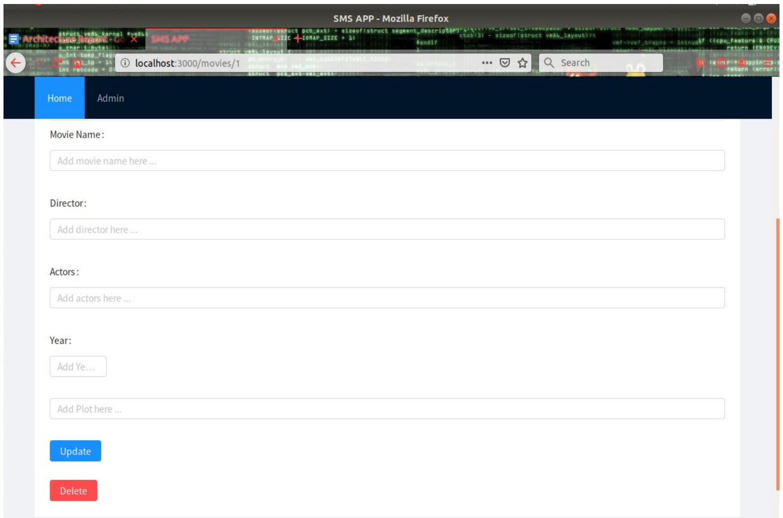
Movie Name 1
Director: movie director
Actors: movie actors
Year: 2019
Plot: movie plot

Movie Name 2
Director: movie director
Actors: movie actors
Year: 2019
Plot: movie plot

Movie Name 3
Director: movie director
Actors: movie actors
Year: 2019
Plot: movie plot

2. Movie Detail, Update and Deletion Page:



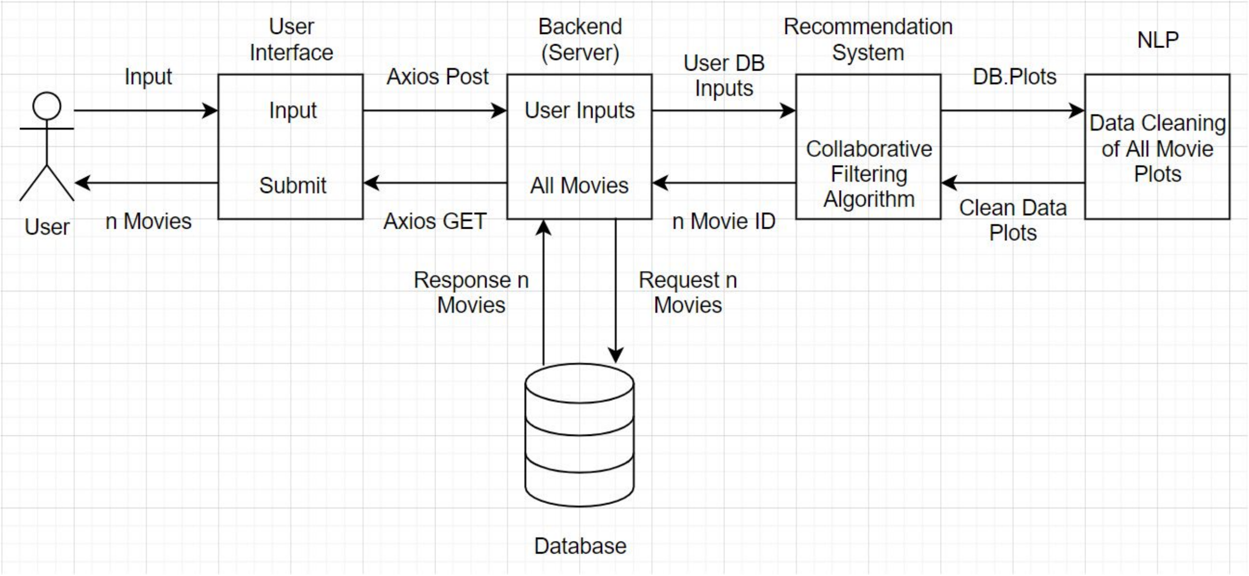


5.2.4 Frontend & Backend Communication:

As explained in the requirement section of this report, Axios would be used for handling http request. It would be used to get data from React forms (frontend) and send it to the backend through an API POST request to the Django REST API. The flow of data from the frontend to the backend(and vice-versa) can be fathomed better from the source code and the diagram below.

Firstly, the movie name or plot (and the number of movie of movies to be suggested) inputted by the user are sent to the backend with the help of axios post request and these data alongside all the movies in the database (which are also queried using Django ORM querying methods) are sent to the recommendation system which uses a collaborative filtering algorithm to suggest similar movies to the user input. Prior to the use of this algorithm, keywords from the movies' plots which were queried from the database are extracted using a Natural Language Processing(NLP) technique. The recommendation system returns the ID of movies to recommend and the movies whose ID was suggested are queried from the DB and sent to the frontend using Axios and Django REST API.

Diagram:



6. Project Inferences

6.1 Achievements

We’ve been able to come up with a minimum viable product(MVP) which solves the problem described in the problem statement. This simple web app contains a:

- Simple user interface.
- Working recommendation system + keyword extraction system (NLP).
- Database for storing movies details.
- Movie name auto-suggester.
- Web page that suggest movies using the movie name provided by the user.
- Web page that suggest movie using the user plot.
- Admin movie creation, read, update and deletion (CRUD) web pages.

6.2 Unachieved Functional Requirements

Due to some reasons, we have not been able to complete the coding of all the functional requirements specified in the requirement analysis. Features such as “recommendation of movies using genre, director and actors” and the integration of the login page with the admin CRUD operation pages (admin authentication) has not been added to the application because we’ve not been chanced to code them due to the workload from other projects.

6.3 Testing Revision

A lot of unit testing were carried out on the recommendation system. At the outset of the testing, the recommendation system (which was programmed with the help of no library) could recommend movies within 27 seconds from a list of about 5000 movies and 17seconds from a list of about 2000 movies but thanks to well optimized libraries available, we could optimize the recommendation system and make it recommend movies within few seconds which will make the user's experience much more better. More tests are still being carried out on the system and other methods for optimizing the codes are been researched by every member of the group.

6.4 Improvement Possibilities

The MVP software at hand can be improved by adding the functional requirements features absent in the application and by adding other features such as:

- User account feature that can save already watched movies by users in order to compute a better result using an improved recommendation system algorithm.
- A social media platform for users to follow and communicate to fellow movie watchers.
- Users searching for movies by media (which can be short video clips or photos of scenes from a movie) which can ease their search experience in a situation when they don’t know the name of a movie.

7. Project Calendar and Test Plan

7.1 Work Breakdown Structure

1. Frontend:
    - Setting up frontend
    - Coding necessary layout for web app
    - Coding necessary components for web app
    - Building web pages using components and layout
    - Routing web pages.

2. Backend:
    - Setting up backend
    - Creating a movie model
    - Creating serializers for the model to convert model instances to JSON
    - Creating API views and endpoints for the model
    - Consuming REST APIs with axios from frontend.

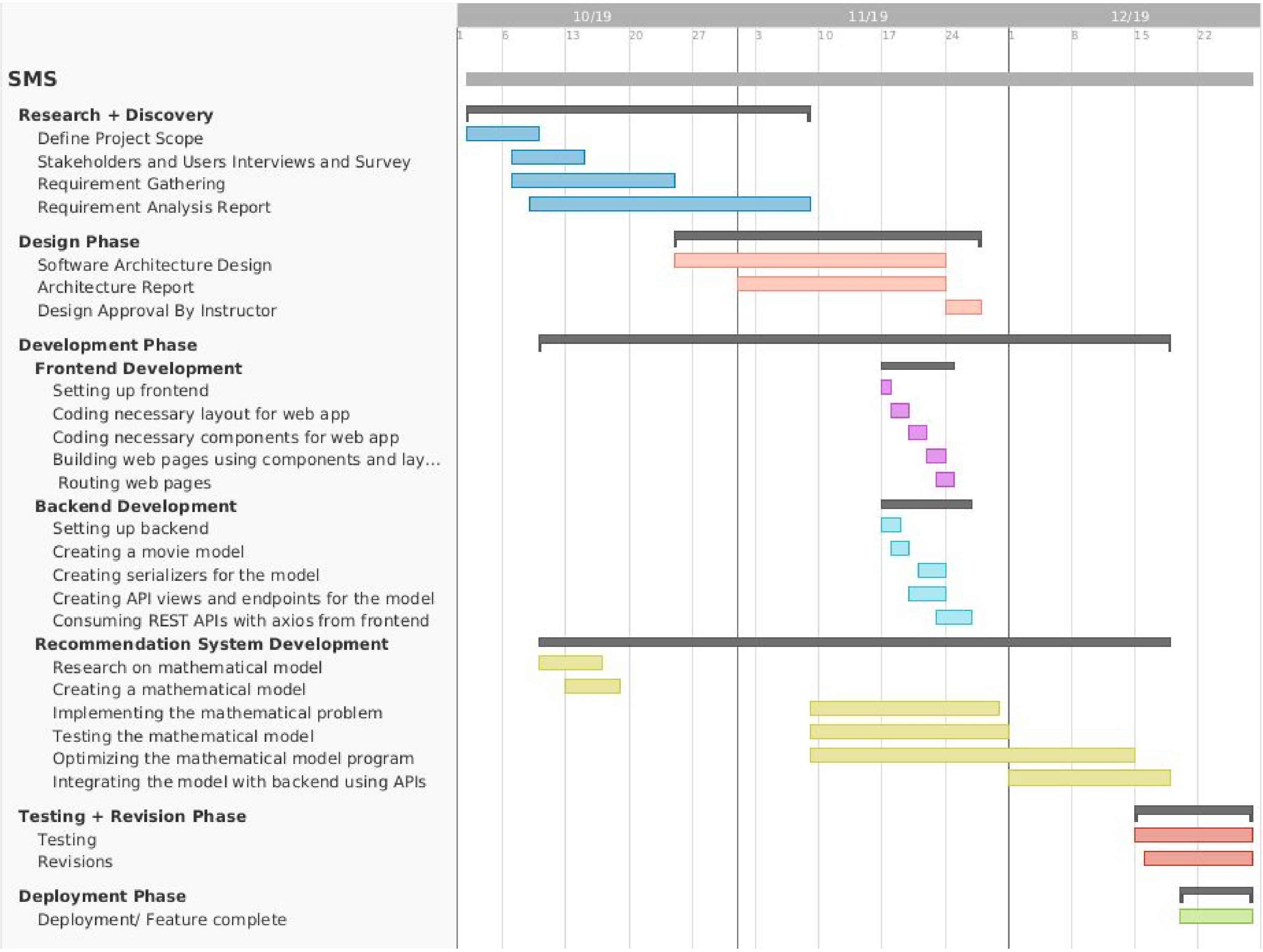
3. Recommendation System:
    - Creating a mathematical model that can recommend suitable movies
    - Implementing the mathematical problem
    - Optimizing the mathematical model program
    - Integrating the model with backend using APIs.

7.2 Communication and Reporting

All teammates have to meet at least once every week to talk about the progress of the project. Every team member’s sub module codes would be shared and merged on a version control system such as Bitbucket. These codes would be reviewed and approved by every team member before it is merged with already existing codes.

In addition, free Web-based applications such as Google Docs and TeamGantt would be used for report writing and scheduling/sharing task among teammates. These tasks must be completed before the deadline suggested by the teammate on the scheduling application.

7.3 Schedule and Milestones



7.4 Project Retrospective

The knowledge we've gained while working on this project is highly valuable to our success in the future projects we would be working on.

Firstly, working as a group increased our productivity and creativity while working on the project **even though all members of the team had almost no idea about the technologies we would be working on**. Together, we've gathered enough experience that can help us while working on similar projects alongside experiences with using Git-based Version Control System(VCS) such as Bitbucket. On the outset of writing the codes for the application, both in group and individual, determining a coding style/ convention has helped us to write clear, understandable and easy to understand codes. As an awareness of working together in a team, writing clean code has many benefits in terms of project readability and enhanceability.

We've learnt how to handle most types of errors and obstacle we come across while writing codes and most importantly, we learnt that documentation is a salient part of software development. Writing down technical details of a project using words and schematic representations (such as UML diagrams, graphs, etc.) helps keep track of all aspects of a software development process and it improves the quality of the software product. In addition to this, It also helps in development, maintenance and knowledge transfer to other developers that would be working on the same project.

Working on this project specifically has added a lot to our knowledge and is an advantage that would have positive impact on future projects.

8. Reference /Bibliography

1. https://numpy.org
2. https://pandas.pydata.org
3. https://scikit-learn.org/stable
4. https://en.wikipedia.org/wiki/Natural_Language_Toolkit
5. https://pypi.org/project/rake-nltk
6. https://www.fullstackpython.com/django-orm.html
7. https://www.django-rest-framework.org
8. https://github.com/axios/axios
9. https://reactjs.org
10. https://creately.com
11. https://bitbucket.org/
12. https://ant.design/