

---

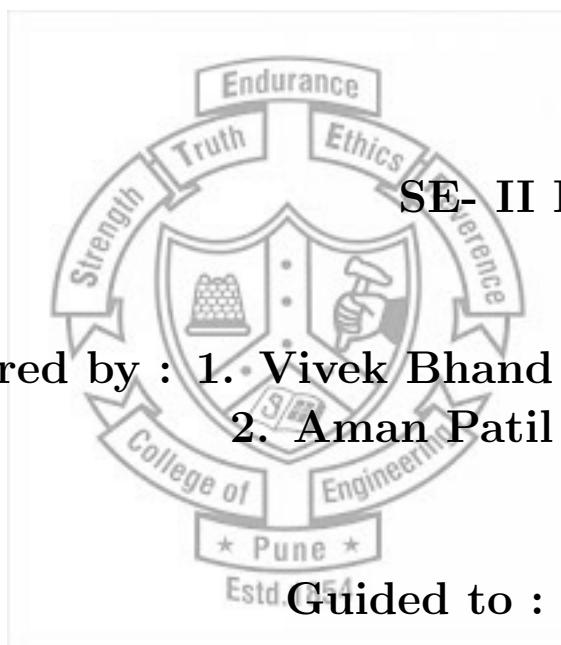
# Project Report

for

Not-Flix

SE- II Mini Project

Prepared by : 1. Vivek Bhand (111903129)  
2. Aman Patil (111903135)



Guided to : Harish D.G.  
Professor  
Samrudhi Kulkarni

April 12, 2022

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	State of the art survey . . . . .	4
1.2	Intended Audience and Reading Suggestions . . . . .	4
1.3	Project Scope . . . . .	4
<b>2</b>	<b>Overall Description</b>	<b>5</b>
2.1	Problem Statement . . . . .	5
2.2	Objectives . . . . .	5
2.3	Operating Environment . . . . .	5
2.4	Design . . . . .	5
<b>3</b>	<b>Constraint</b>	<b>6</b>
3.1	Constraints for the project . . . . .	6
3.2	Advantages over existing Project . . . . .	6
3.3	Disadvantages . . . . .	6
<b>4</b>	<b>System Features</b>	<b>7</b>
4.1	Description and Priority . . . . .	7
4.2	Functional Requirements . . . . .	7
<b>5</b>	<b>Other Nonfunctional Requirements</b>	<b>8</b>
5.1	Performance Requirements . . . . .	8
5.2	Security Requirements . . . . .	8
5.3	Software Quality Attributes . . . . .	8
<b>6</b>	<b>Project Planning using Automated Tools</b>	<b>9</b>
6.1	Gantt Diagram . . . . .	9
6.2	Cost Estimation . . . . .	10
6.3	Work Flow . . . . .	11
<b>7</b>	<b>UML Diagrams</b>	<b>12</b>
7.1	Activity Diagram . . . . .	13
7.2	Class Diagram . . . . .	14
7.3	Sequence Diagram . . . . .	15
7.4	User Case . . . . .	15
7.5	Entity Relationship Diagram . . . . .	16
<b>8</b>	<b>Code ScreenShots</b>	<b>17</b>

<b>9 Future Scope</b>	<b>29</b>
<b>10 Conclusion</b>	<b>30</b>
<b>11 GitHub Repository Link</b>	<b>31</b>



# 1 Introduction

## 1.1 State of the art survey

To assess the present state of streaming sites and their attractiveness to consumers, we ran a survey to gather information on the most popular streaming platforms, the reasons for their appeal over competitors, and their disadvantages over others.

We discovered that Netflix is one of the most popular streaming sites after reviewing all of the poll responses. The appeal of Netflix as a more preferable option was due to its ease of use, appealing user interface, and categorical specification of the streaming content. Even with these advantages, however, the number of Netflix users was very low. The cause could be attributed to the subscription service's high cost. Other streaming services were also unsatisfactory.

So, to address the need for a streaming site with a user-friendly, appealing, and simple-to-use interface, we've chosen to create a streaming web platform with all of Netflix's features at a far lower cost. Our proposed solution, "NOT-FLIX" (website name), offers its users the best streaming video experience available without putting a strain on their wallet.

## 1.2 Intended Audience and Reading Suggestions

This SRS is for developers, project managers, users and testers. Further the discussion will provide all the internal, external, functional and also non-functional informations about "NOT-Flix WEBSITE".

## 1.3 Project Scope

"NOT-Flix WEBSITE" is a online streaming platform which provides all feature and the user experience offered by NetFlix.

After getting signing up to a Not-Flix user is redirected to a subscription page..He/She will be added as a user to the database after completing his/her payment process. After that he/she is redirected to the homepage, where he/she can see all the suggestions of the content he/she would like to watch. User profile will contain all his personal informations, subscription details and notifications

## 2 Overall Description

### 2.1 Problem Statement

”NOT-Flix WEBSITE” is a online streaming website with User friendly Interface and implementations of similar features like NetFlix.

### 2.2 Objectives

The following are the Objectives of our project:

- User friendly Site with less expenses.
- Categorical classification of the content.
- System for real-time search.

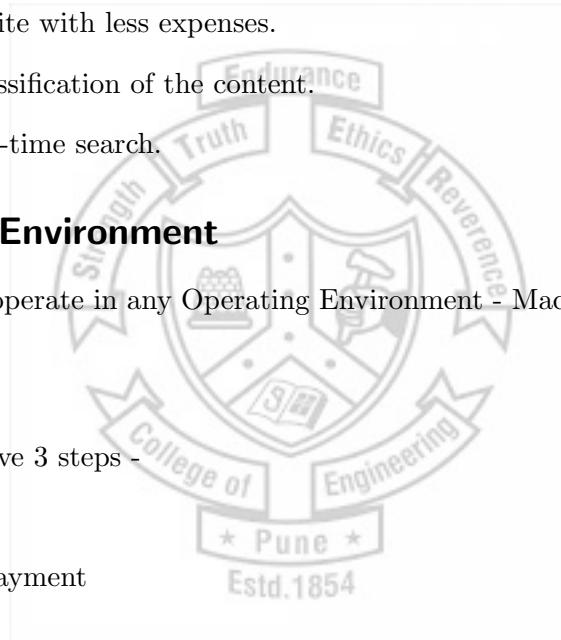
### 2.3 Operating Environment

The website will be operate in any Operating Environment - Mac, Windows, Linux etc.

### 2.4 Design

Student activities have 3 steps -

- Sign Up
- Subscription Payment
- User Profile
- Streaming Content



# **3 Constraint**

## **3.1 Constraints for the project**

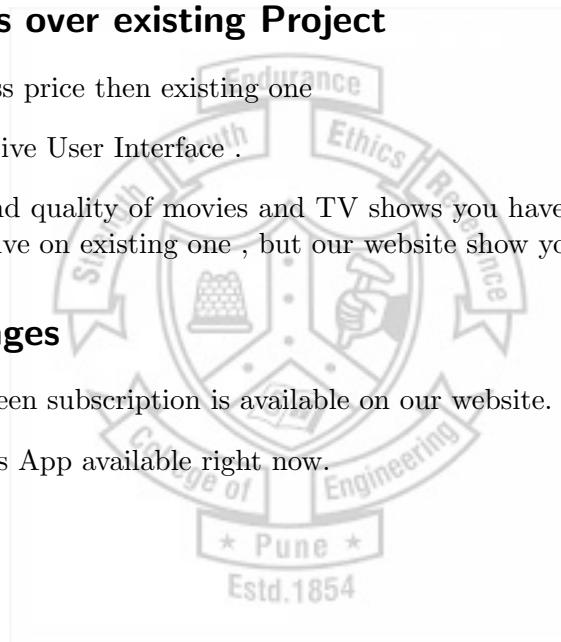
- Due to limited memory availability not able to show all the movies and shows on the website.
- Due to less budget not able to purchase all new movies which are recently released.
- Only single screen is available on a single email id .

## **3.2 Advantages over existing Project**

- Available at less price then existing one
- Provide attractive User Interface .
- The number and quality of movies and TV shows you have access to will depend on where you live on existing one , but our website show you all the content.

## **3.3 Disadvantages**

- Only single screen subscription is available on our website.
- No Android/Ios App available right now.



## 4 System Features

”NOT-Flix WEBSITE” is an online streaming website. So the main art of this product is to watch content and provide a good user experience while using it.

### 4.1 Description and Priority

”NOT-Flix WEBSITE” has features that are main and also some are sub. But all the feature is necessary for this software.

The features with priority up to down -

1. Streaming of Content : This is the goal feature of this software.
2. Ability to search content
3. Categorical classification of content
4. Preview and thumbnail at home page
5. Season wise classification of the shows

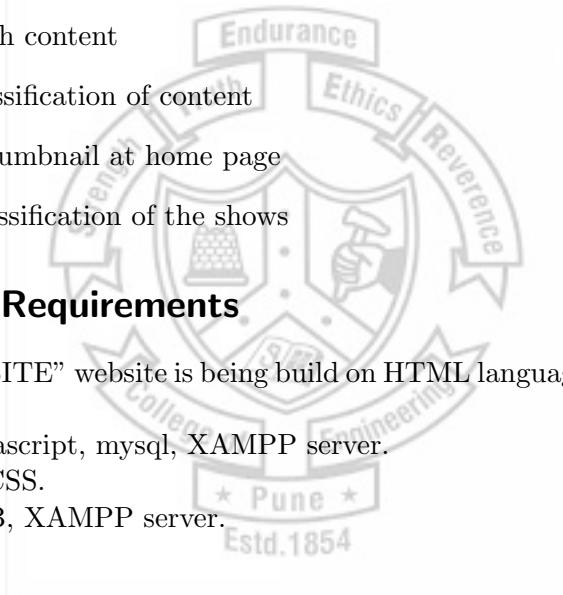
### 4.2 Functional Requirements

The ”Not-Flix WEBSITE” website is being build on HTML language, CSS, php, JavaScript and InnoDB.

Back-End - php, Javascript, mysql, XAMPP server.

Font-End - HTML, CSS.

Database - MongoDB, XAMPP server.



# 5 Other Nonfunctional Requirements

## 5.1 Performance Requirements

”NOT-Flix WEBSITE” is a website which provides a smooth user experience as same as NetFlix. It provides a cheap alternate to using Netflix.

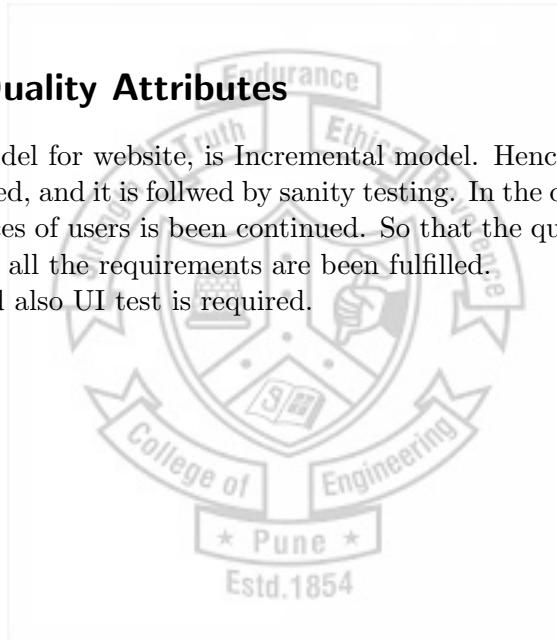
## 5.2 Security Requirements

No one without registered users can enter to the website. Protection against SQL injection attacks.

## 5.3 Software Quality Attributes

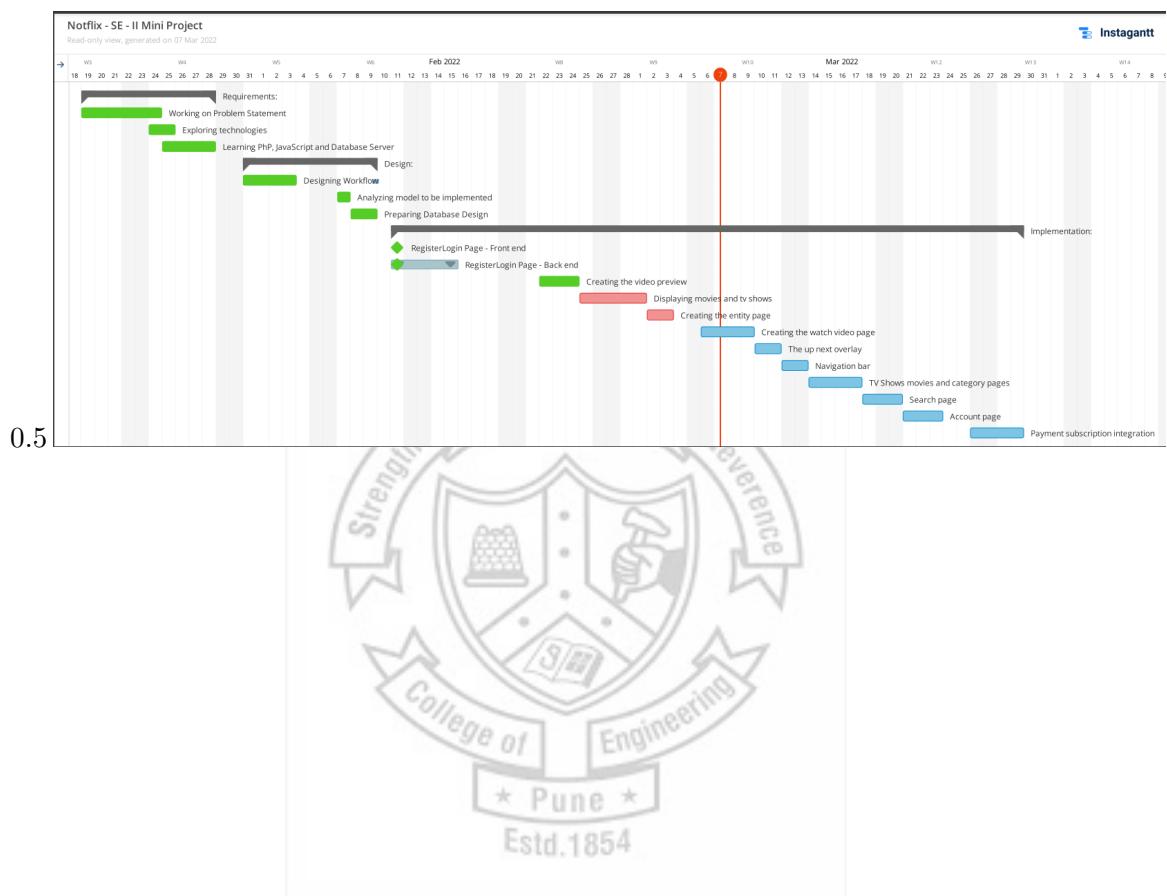
Our development model for website, is Incremental model. Hence, testing is done after each new feature added, and it is followed by sanity testing. In the development phase also testing and conferences of users is been continued. So that the quality of the software is been maintained and all the requirements are been fulfilled.

Database, logical and also UI test is required.



# 6 Project Planning using Automated Tools

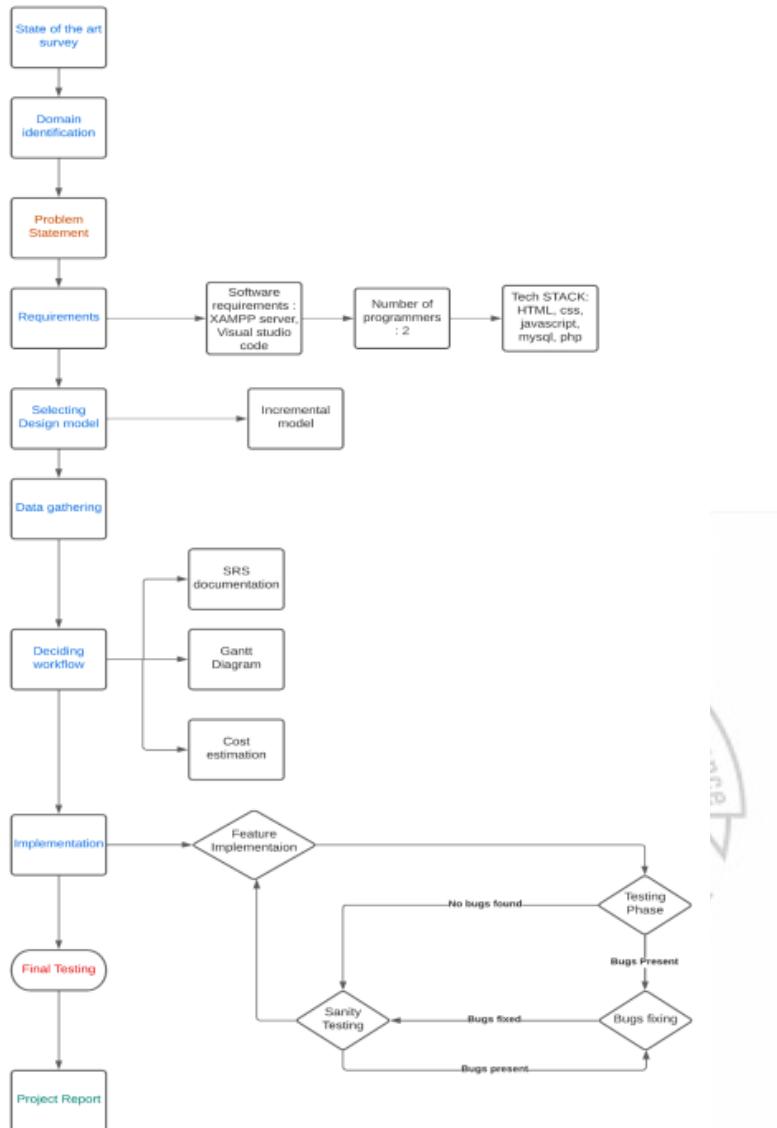
## 6.1 Gantt Diagram



## 6.2 Cost Estimation

<b>Requirements:</b>		32h	19/Jan	28/Jan	<div style="width: 100%;">100%</div>
1	<span style="color: green;">✓</span> Working on Problem State...	AR, Vi	8h	19/Jan	24/Jan <div style="width: 100%;">100%</div> 
2	<span style="color: green;">✓</span> Exploring technologies	AR, Vi	12h	24/Jan	25/Jan <div style="width: 100%;">100%</div> 
3	<span style="color: green;">✓</span> Learning PhP, JavaScript an...	AR, Vi	12h	25/Jan	28/Jan <div style="width: 100%;">100%</div> 
<a href="#">+ Add task</a> <a href="#">+ Add section</a>					
<b>Design:</b>		29h	31/Jan	09/Feb	<div style="width: 100%;">100%</div>
6	<span style="color: green;">✓</span> Designing Workflow	AR, Vi	16h	31/Jan	03/Feb <div style="width: 100%;">100%</div> 
7	<span style="color: green;">✓</span> Analyzing model to be impl...	AR, Vi	3h	07/Feb	07/Feb <div style="width: 100%;">100%</div> 
8	<span style="color: green;">✓</span> Preparing Database Design	AR, Vi	10h	08/Feb	09/Feb <div style="width: 100%;">100%</div> 
<a href="#">+ Add task</a> <a href="#">+ Add section</a>					
<b>Implementation:</b>		134h	11/Feb	29/Mar	<div style="width: 15%;">15%</div>
11	<span style="color: green;">✓</span> RegisterLogin Page - Front ...	Aman Rajput	5h	11/Feb	11/Feb <div style="width: 100%;">100%</div> 
+	<span style="color: green;">✓</span> RegisterLogin Page - Back e...	Vivek	20h	11/Feb	15/Feb <div style="width: 100%;">100%</div> 
16	<span style="color: green;">✓</span> Creating the video preview	AR, Vi	12h	22/Feb	24/Feb <div style="width: 100%;">100%</div> 
17	<span style="color: green;">✓</span> Displaying movies and tv sh...	Aman Rajput	30h	25/Feb	01/Mar <div style="width: 0%;">0%</div> 
18	<span style="color: green;">✓</span> Creating the entity page	Vivek	8h	02/Mar	03/Mar <div style="width: 0%;">0%</div> 
19	<span style="color: green;">✓</span> Creating the watch video pa...	Vi, AR	10h	06/Mar	09/Mar <div style="width: 0%;">0%</div> 
20	<span style="color: green;">✓</span> The up next overlay	AR, Vi	6h	10/Mar	11/Mar <div style="width: 0%;">0%</div> 
21	<span style="color: green;">✓</span> Navigation bar	AR, Vi	4h	12/Mar	13/Mar <div style="width: 0%;">0%</div> 
22	<span style="color: green;">✓</span> TV Shows movies and categ...	Vi, AR	8h	14/Mar	17/Mar <div style="width: 0%;">0%</div> 
23	<span style="color: green;">✓</span> Search page	AR, Vi	8h	18/Mar	20/Mar <div style="width: 0%;">0%</div> 
24	<span style="color: green;">✓</span> Account page	AR, Vi	7h	21/Mar	23/Mar <div style="width: 0%;">0%</div> 
25	<span style="color: green;">✓</span> Payment subscription integ...	Vi, AR	16h	26/Mar	29/Mar <div style="width: 0%;">0%</div> 
<a href="#">+ Add task</a> <a href="#">+ Add section</a>					

### 6.3 Work Flow

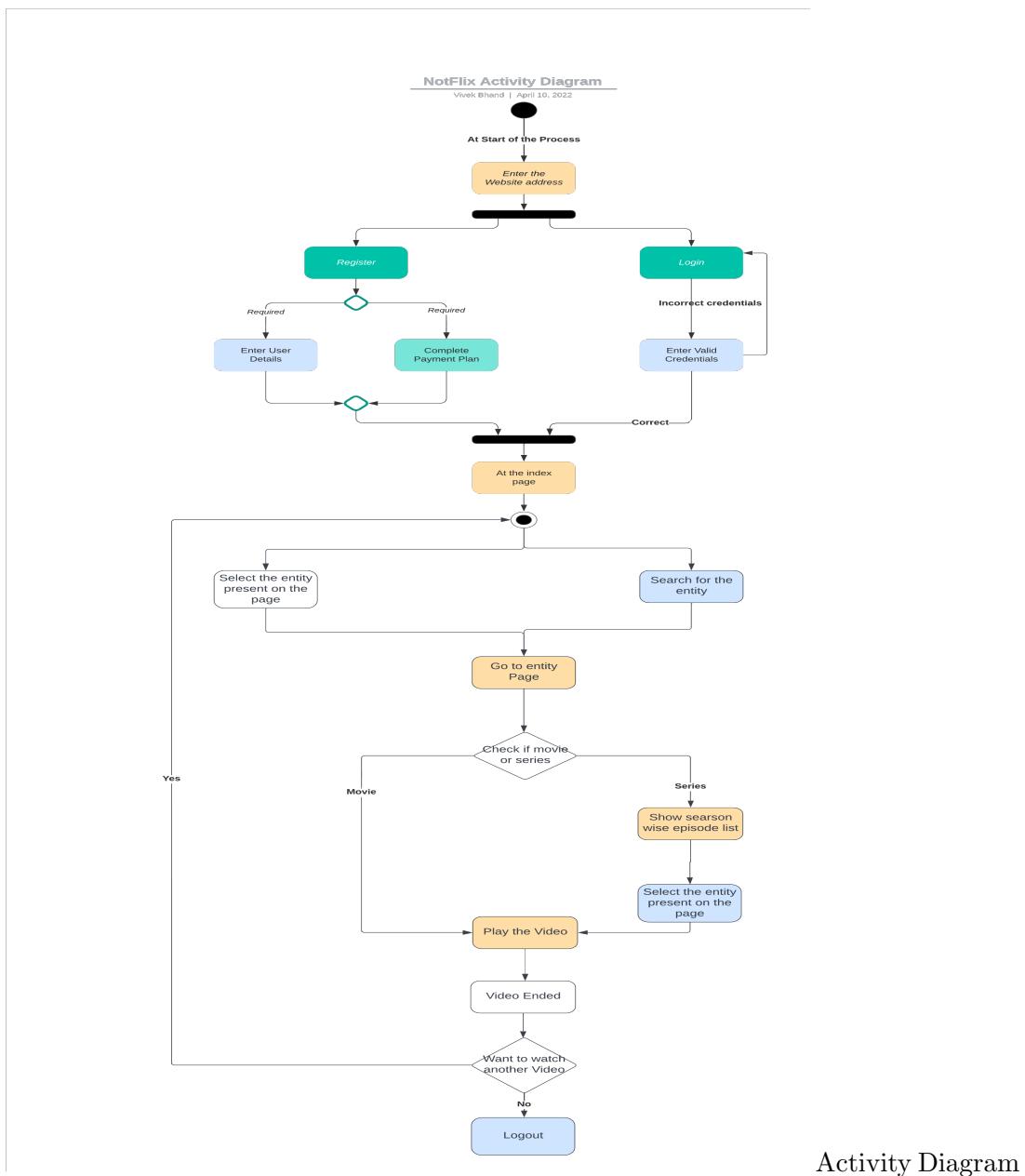


Workflow is the overview of the project. Connection of all the entities are dependable to each others. This gives the simple idea about the functional activities of the project.

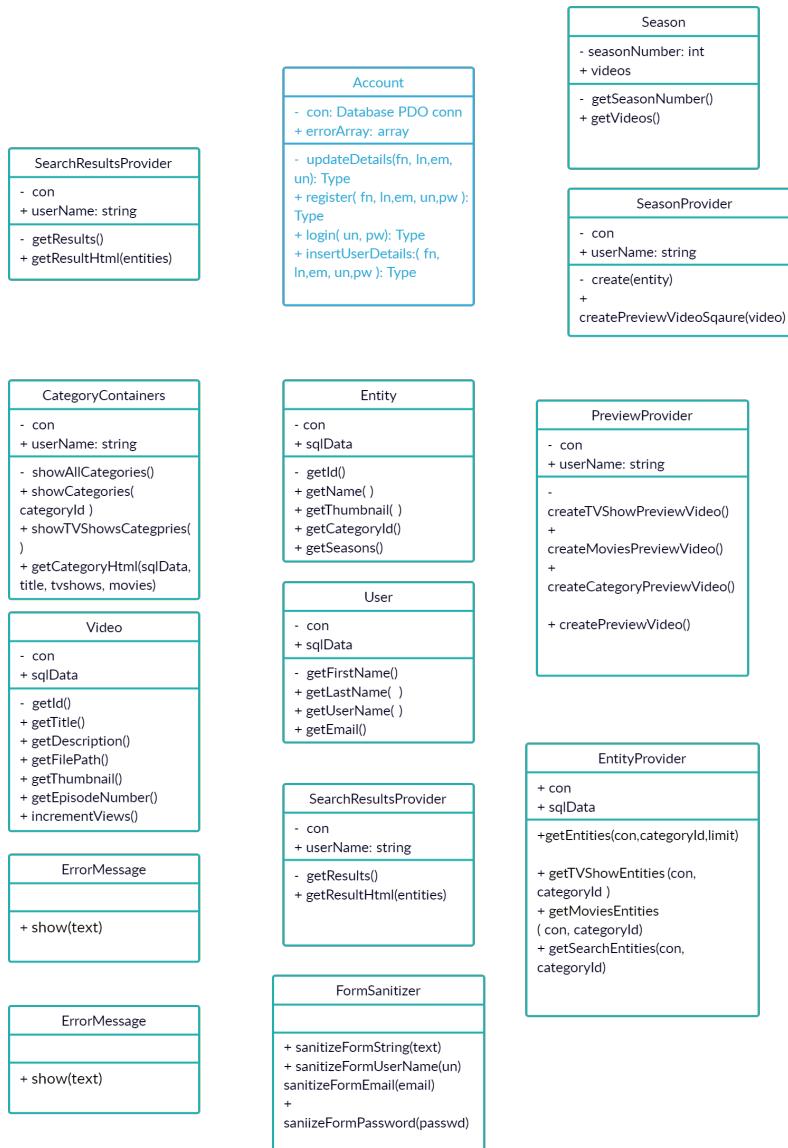


# 7 UML Diagrams

## 7.1 Activity Diagram

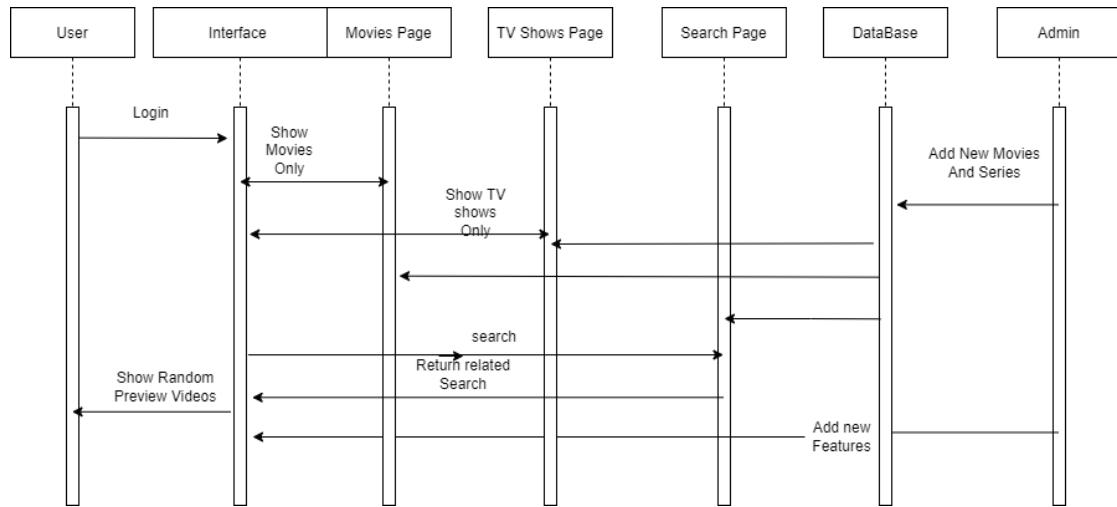


## 7.2 Class Diagram



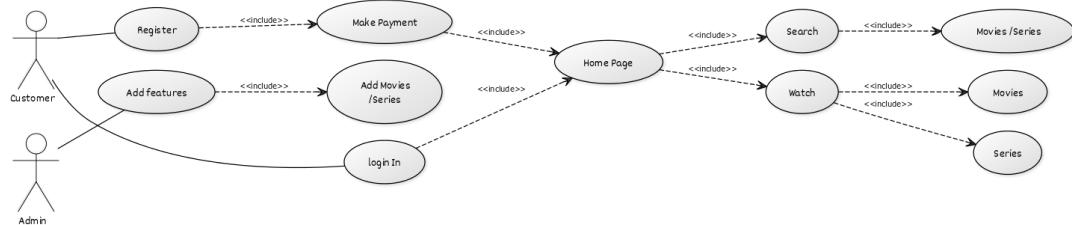
Activity Diagram

### 7.3 Sequence Diagram



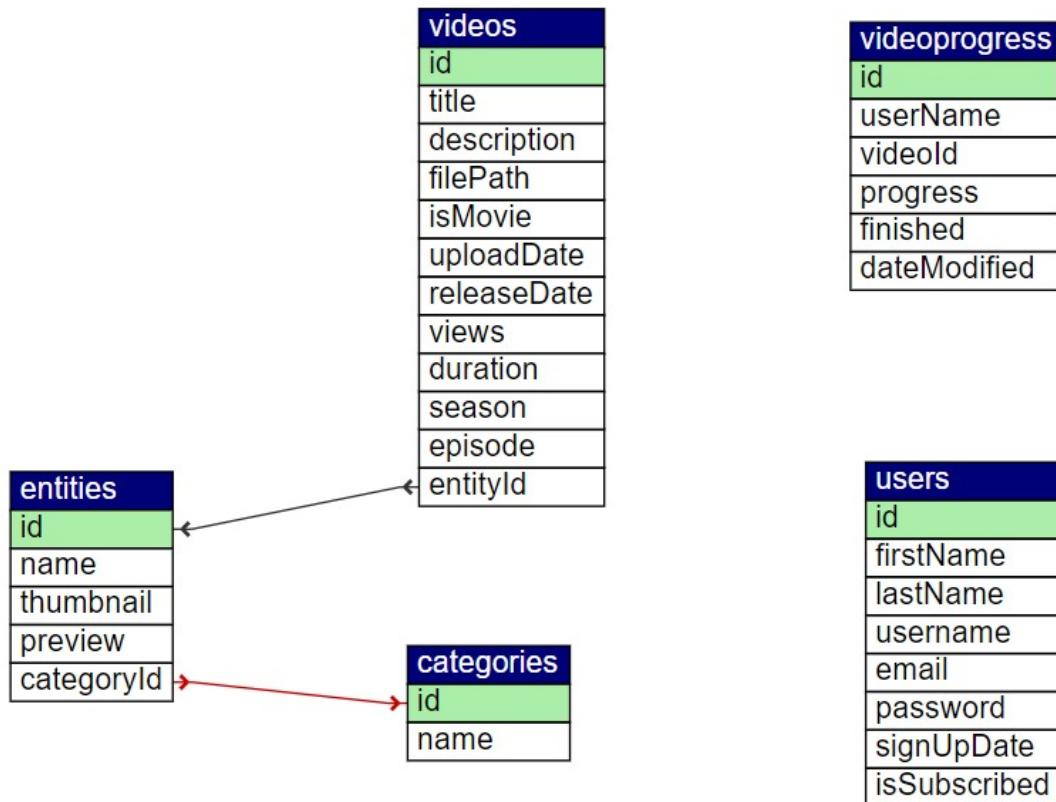
Sequence Diagram

### 7.4 User Case

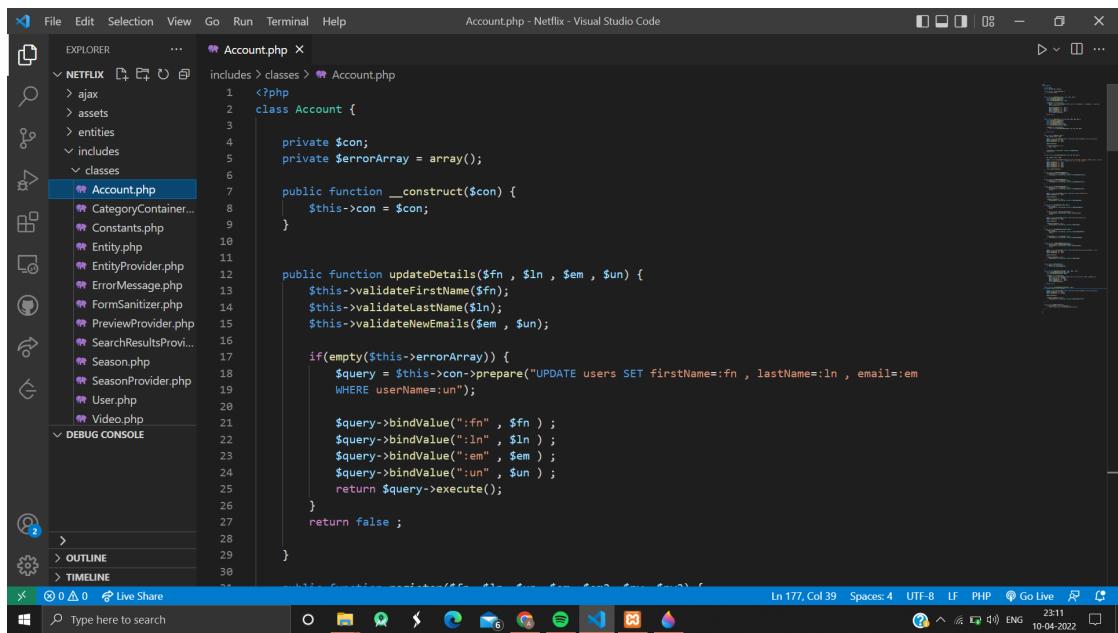


User Case

## 7.5 Entity Relationship Diagram



## 8 Code ScreenShots



The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** Account.php - Netflix - Visual Studio Code.
- Explorer Panel:** Shows the project structure under 'NETFLIX'. The 'Account.php' file is selected in the 'classes' folder.
- Editor Panel:** Displays the PHP code for 'Account.php'. The code includes class definitions, constructor logic, and a method for updating user details using prepared statements.
- Bottom Status Bar:** Shows the current line (Ln 177), column (Col 39), spaces (Spaces: 4), encoding (UTF-8), line feed (LF), PHP, and other status information (23:11, ENG, 10-04-2022).

```
<?php
class Account {
    private $con;
    private $errorArray = array();

    public function __construct($con) {
        $this->con = $con;
    }

    public function updateDetails($fn , $ln , $em , $un) {
        $this->validateFirstName($fn);
        $this->validateLastName($ln);
        $this->validateNewEmails($em , $un);

        if(empty($this->errorArray)) {
            $query = $this->con->prepare("UPDATE users SET firstName=:fn , lastName=:ln , email=:em
                WHERE userName=:un");

            $query->bindValue(":fn" , $fn );
            $query->bindValue(":ln" , $ln );
            $query->bindValue(":em" , $em );
            $query->bindValue(":un" , $un );
            return $query->execute();
        }
        return false ;
    }
}
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Path:** Account.php - Netflix - Visual Studio Code
- Explorer:** Shows the project structure under the NETFLIX folder, including ajax, assets, entities, includes, and classes. The Account.php file is selected.
- Editor:** Displays the PHP code for the Account.php class. The code includes methods for register and login, along with validation logic for first name, last name, user name, email, and password.
- Bottom Bar:** Includes tabs for Live Share, a search bar, and various system icons.

```
public function register($fn, $ln, $un, $em, $em2, $pw, $pw2) {
    $this->validateFirstName($fn);
    $this->validateLastName($ln);
    $this->validateUserName($un);
    $this->validateEmails($em, $em2);
    $this->validatePasswords($pw, $pw2);

    if(empty($this->errorArray)) {
        return $this->insertUserDetails($fn, $ln, $un, $em, $pw);
    }

    return false;
}

public function login($un, $pw) {
    $pw = hash("sha512", $pw);

    $query = $this->con->prepare("SELECT * FROM users WHERE userName=:un AND password=:pw");
    $query->bindValue(":un", $un);
    $query->bindValue(":pw", $pw);

    $query->execute();

    if($query->rowCount() == 1) {
        return true;
    }

    array_push($this->errorArray, Constants::$loginFailed);
    return false;
}
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Path:** Account.php - Netflix - Visual Studio Code
- Explorer:** Shows the project structure under the NETFLIX folder, including ajax, assets, entities, includes, and classes. The Account.php file is selected.
- Editor:** Displays the PHP code for the Account.php class. The code includes methods for insertUserDetails, validateFirstName, validateLastName, validateUserName, validateEmails, and validatePasswords.
- Bottom Bar:** Includes tabs for Live Share, a search bar, and various system icons.

```
private function insertUserDetails($fn, $ln, $un, $em, $pw) {
    $pw = hash("sha512", $pw);

    $query = $this->con->prepare("INSERT INTO users (firstName, lastName, userName, email, password)
VALUES (:fn, :ln, :un, :em, :pw)");
    $query->bindValue(":fn", $fn);
    $query->bindValue(":ln", $ln);
    $query->bindValue(":un", $un);
    $query->bindValue(":em", $em);
    $query->bindValue(":pw", $pw);

    return $query->execute();
}

private function validateFirstName($fn) {
    if(strlen($fn) < 2 || strlen($fn) > 25) {
        array_push($this->errorArray, Constants::$firstNameCharacters);
    }
}

private function validateLastName($ln) {
    if(strlen($ln) < 2 || strlen($ln) > 25) {
        array_push($this->errorArray, Constants::$lastNameCharacters);
    }
}

private function validateUserName($un) {
    if(strlen($un) < 2 || strlen($un) > 25) {
        array_push($this->errorArray, Constants::$userNameCharacters);
    }
}

private function validateEmails($em, $em2) {
    if($em != $em2) {
        array_push($this->errorArray, Constants::$emailCharacters);
    }
}

private function validatePasswords($pw, $pw2) {
    if($pw != $pw2) {
        array_push($this->errorArray, Constants::$passwordCharacters);
    }
}
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Path:** Account.php - Netflix - Visual Studio Code
- Explorer View:** Shows the project structure under the NETFLIX folder, including ajax, assets, entities, includes, and classes. The Account.php file is selected.
- Code Editor:** Displays the PHP code for the Account.php class. The code handles account validation, password hashing, and password update logic.
- Status Bar:** Shows the current line (Ln 177), column (Col 39), spaces (Spaces: 4), encoding (UTF-8), line endings (LF), PHP mode, and a Go Live button.

```
Account.php - Netflix - Visual Studio Code

includes > classes > Account.php
187     array_push($this->errorArray, Constants::$emailsDontMatch);
188     return;
189 }
190
191 if(!filter_var($em, FILTER_VALIDATE_EMAIL)) {
192     array_push($this->errorArray, Constants::$emailInvalid);
193     return;
194 }
195
196 $query = $this->con->prepare("SELECT * FROM users WHERE email=:em");
197 $query->bindValue(":em", $em);
198
199 $query->execute();
200
201 if($query->rowCount() != 0) {
202     array_push($this->errorArray, Constants::$emailTaken);
203 }
204
205 private function validatePasswords($pw, $pw2) {
206     if($pw != $pw2) {
207         array_push($this->errorArray, Constants::$passwordsDontMatch);
208         return;
209     }
210
211     if(strlen($pw) < 5 || strlen($pw) > 25) {
212         array_push($this->errorArray, Constants::$passwordLength);
213     }
214 }

Ln 177, Col 39 Spaces: 4 UTF-8 LF PHP Go Live 23:11 ENG 10-04-2022
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Path:** Account.php - Netflix - Visual Studio Code
- Explorer View:** Shows the project structure under the NETFLIX folder, including ajax, assets, entities, includes, and classes. The Account.php file is selected.
- Code Editor:** Displays the PHP code for the Account.php class, focusing on the updatePassword and validateOldPassword methods. It includes password hashing logic using SHA-512.
- Status Bar:** Shows the current line (Ln 171), column (Col 38), spaces (Spaces: 4), encoding (UTF-8), line endings (LF), PHP mode, and a Go Live button.

```
Account.php - Netflix - Visual Studio Code

includes > classes > Account.php
160 }
161
162 public function updatePassword($oldPw , $pw , $pw2 , $un) {
163     $this->validateOldPassword($oldPw , $un);
164     $this->validatePasswords($pw , $pw2);
165
166     if(empty($this->errorArray)) {
167         $query = $this->con->prepare("UPDATE users SET password=:pw WHERE userName=:un");
168         $pw = hash("sha512" , $oldPw);
169         $query->bindValue(":pw" , $pw );
170         $query->bindValue(":un" , $un );
171         return $query->execute();
172     }
173     return false ;
174 }
175
176 public function validateOldPassword($oldPw , $un) {
177     $oldPw = hash("sha512" , $oldPw);
178
179     $query = $this->con->prepare("SELECT * FROM users WHERE userName=:un AND password=:pw");
180     $query->bindValue(":un" , $un);
181     $query->bindValue(":pw" , $oldPw);
182
183     $query->execute();
184
185     if($query->rowCount() == 0) {
186         array_push($this->errorArray ,Constants::$passwordIncorrect);
187     }
188 }

Ln 171, Col 38 Spaces: 4 UTF-8 LF PHP Go Live 23:11 ENG 10-04-2022
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** CategoryContainers.php - Netflix - Visual Studio Code.
- Explorer:** Shows a tree view of the project structure under the NETFLIX category, including files like Account.php, CategoryContainer.php, Constants.php, Entity.php, EntityProvider.php, ErrorMessage.php, FormSanitizer.php, PreviewProvider.php, SearchResultsProv..., Season.php, SeasonProvider.php, User.php, and Video.php.
- Editor:** Displays the content of CategoryContainers.php. The code includes methods for showing all categories and categories for a specific category ID, along with a method for TV shows categories.
- Bottom Status Bar:** Lines 103, Col 37, Spaces: 4, UTF-8, LF, PHP, Go Live, and a timestamp of 23:11 on 10-04-2022.

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** CategoryContainers.php - Netflix - Visual Studio Code.
- Explorer:** Shows a tree view of the project structure under the NETFLIX category, including files like Account.php, CategoryContainer.php, Constants.php, Entity.php, EntityProvider.php, ErrorMessage.php, FormSanitizer.php, PreviewProvider.php, SearchResultsProv..., Season.php, SeasonProvider.php, User.php, and Video.php.
- Editor:** Displays the content of CategoryContainers.php. The code includes methods for showing all categories and categories for a specific category ID, along with a method for TV shows categories.
- Bottom Status Bar:** Lines 58, Col 6, Spaces: 4, UTF-8, LF, PHP, Go Live, and a timestamp of 23:12 on 10-04-2022.

The screenshot shows the Visual Studio Code interface with the following details:

- File Path:** CategoryContainers.php - Netflix - Visual Studio Code
- Explorer:** Shows the project structure under the NETFLIX folder, including sub-folders like ajax, assets, entities, and classes, along with various PHP files such as Account.php, CategoryContainer.php, Constants.php, Entity.php, etc.
- Editor:** Displays the code for the CategoryContainers.php file. The code includes methods for showing movies categories and a private function for generating category HTML based on SQL data.
- Bottom Status Bar:** Shows the current line (Ln 58), column (Col 6), spaces (Spaces: 4), encoding (UTF-8), line endings (LF), PHP, Go Live, and the date/time (23:12 10-04-2022).

The screenshot shows the Visual Studio Code interface with the following details:

- File Path:** CategoryContainers.php - Netflix - Visual Studio Code
- Explorer:** Shows the project structure under the NETFLIX folder, including sub-folders like ajax, assets, entities, and classes, along with various PHP files such as Account.php, CategoryContainer.php, Constants.php, Entity.php, etc.
- Editor:** Displays the code for the CategoryContainers.php file. The code includes a private function for generating category HTML, which checks if \$tvShows and \$movies are present and retrieves entities accordingly.
- Bottom Status Bar:** Shows the current line (Ln 58), column (Col 6), spaces (Spaces: 4), encoding (UTF-8), line endings (LF), PHP, Go Live, and the date/time (23:12 10-04-2022).

The screenshot shows the Visual Studio Code interface with the Constants.php file open in the editor. The file contains static error messages for various validation checks. The code is as follows:

```
<?php
class Constants {
    public static $firstNameCharacters = "Your first name must be between 2 and 25 characters";
    public static $lastNameCharacters = "Your last name must be between 2 and 25 characters";
    public static $userNameCharacters = "Your userName must be between 2 and 25 characters";
    public static $userNameTaken = "userName already in use";
    public static $emailsDontMatch = "Your emails don't match";
    public static $emailInvalid = "Invalid email";
    public static $emailTaken = "Email already in use";
    public static $passwordsDontMatch = "Passwords don't match";
    public static $passwordLength = "Your password must be between 5 and 25 characters";
    public static $loginFailed = "Your userName or password was incorrect";
    public static $passwordIncorrect = "Your old password is incorrect";
}
>>
```

The screenshot shows the Visual Studio Code interface with the Entity.php file open in the editor. The file defines a class Entity with methods for constructing objects from input arrays and for retrieving entity details. The code is as follows:

```
<?php
class Entity {
    private $con, $sqlData;

    public function __construct($con, $input) {
        $this->con = $con;

        if(is_array($input)) {
            $this->sqlData = $input;
        }
        else {
            $query = $this->con->prepare("SELECT * FROM entities WHERE id=:id");
            $query->bindValue(":id", $input);
            $query->execute();

            $this->sqlData = $query->fetch(PDO::FETCH_ASSOC);
        }
    }

    public function getId() {
        return $this->sqlData["id"];
    }

    public function getName() {
        return $this->sqlData["name"];
    }

    public function getThumbnail() {
        return $this->sqlData["thumbnail"];
    }
}
>>
```

The screenshot shows the Visual Studio Code interface with the Entity.php file open in the main editor. The code implements a getSeasons() method that queries the database for videos based on an entity ID and filters them by season. It then processes the results into seasons and videos arrays.

```
public function getSeasons() {
    $query = $this->con->prepare("SELECT * FROM videos WHERE entityId=:id
        AND isMovie=0 ORDER BY season, episode ASC");
    $query->bindValue(":id", $this->getId());
    $query->execute();

    $seasons = array();
    $videos = array();
    $currentSeason = null;
    while($row = $query->fetch(PDO::FETCH_ASSOC)) {
        if($currentSeason != null && $currentSeason != $row["season"]) {
            $seasons[] = new Season($currentSeason, $videos);
            $videos = array();
        }

        $currentSeason = $row["season"];
        $videos[] = new Video($this->con, $row);
    }

    if(sizeof($videos) != 0) {
        $seasons[] = new Season($currentSeason, $videos);
    }
}

return $seasons;
}
```

The screenshot shows the Visual Studio Code interface with the EntityProvider.php file open in the main editor. This provider class contains a static method getEntities() which constructs a SQL query to select entities from the database. It includes filtering by category ID and ordering by RAND().

```
<?php
class EntityProvider {

    public static function getEntities($con, $categoryId, $limit) {
        $sql = "SELECT * FROM entities ";

        if($categoryId != null) {
            $sql .= "WHERE categoryId=:categoryId ";
        }

        $sql .= "ORDER BY RAND() LIMIT :limit";

        $query = $con->prepare($sql);

        if($categoryId != null) {
            $query->bindValue(":categoryId", $categoryId);
        }

        $query->bindValue(":limit", $limit, PDO::PARAM_INT);
        $query->execute();

        $result = array();
        while($row = $query->fetch(PDO::FETCH_ASSOC)) {
            $result[] = new Entity($con, $row);
        }

        return $result;
    }
}
```

The screenshot shows the Visual Studio Code interface with the file `EntityProvider.php` open. The code implements a static function `getTVShowEntities` that performs a database query to fetch entities from the `entities` table, joining the `videos` table to filter movies. It handles null category IDs and applies a random limit.

```
public static function getTVShowEntities($con, $categoryId, $limit) {
    $sql = "SELECT DISTINCT(entities.id) FROM entities
        INNER JOIN videos ON entities.id = videos.entityId
        WHERE videos.isMovie = 0 ";

    if($categoryId != null) {
        $sql .= " AND categoryId=:categoryId ";
    }

    $sql .= " ORDER BY RAND() LIMIT :limit";

    $query = $con->prepare($sql);

    if($categoryId != null) {
        $query->bindValue(":categoryId", $categoryId);
    }

    $query->bindValue(":limit", $limit, PDO::PARAM_INT);
    $query->execute();

    $result = array();
    while($row = $query->fetch(PDO::FETCH_ASSOC)) {
        $result[] = new Entity($con, $row["id"]);
    }

    return $result;
}
```

The screenshot shows the Visual Studio Code interface with the file `FormSanitizer.php` open. The code defines a class `FormSanitizer` with static methods for sanitizing form inputs: `sanitizeFormString`, `sanitizeFormUserName`, `sanitizeFormPassword`, and `sanitizeFormEmail`. Each method strips tags and replaces whitespace.

```
<?php
class FormSanitizer {

    public static function sanitizeFormString($inputText) {
        $inputText = strip_tags($inputText);
        $inputText = str_replace(' ', '', $inputText);
        $inputText = trim($inputText);
        $inputText = strtolower($inputText);
        $inputText = ucfirst($inputText);
        return $inputText;
    }

    public static function sanitizeFormUserName($inputText) {
        $inputText = strip_tags($inputText);
        $inputText = str_replace(' ', '', $inputText);
        return $inputText;
    }

    public static function sanitizeFormPassword($inputText) {
        $inputText = strip_tags($inputText);
        return $inputText;
    }

    public static function sanitizeFormEmail($inputText) {
        $inputText = strip_tags($inputText);
        $inputText = str_replace(' ', '', $inputText);
        return $inputText;
    }
}
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Path:** PreviewProvider.php - Netflix - Visual Studio Code
- Explorer:** Shows the project structure under the NETFLIX folder, including classes like Account.php, CategoryContainer.php, Constants.php, Entity.php, EntityProvider.php, ErrorMessage.php, FormSanitizer.php, PreviewProvider.php, SearchResultsProvider.php, Season.php, SeasonProvider.php, User.php, and Video.php.
- Editor:** Displays the PHP code for PreviewProvider.php. The code includes methods for creating preview videos for TV shows and movies, as well as a category preview video.
- Status Bar:** Shows the current line (Ln 8, Col 37), spaces (Spaces: 4), encoding (UTF-8), line endings (LF), PHP, Go Live, and the date/time (22:13 10-04-2022).

The screenshot shows the Visual Studio Code interface with the following details:

- File Path:** PreviewProvider.php - Netflix - Visual Studio Code
- Explorer:** Shows the project structure under the NETFLIX folder, including classes like Account.php, CategoryContainer.php, Constants.php, Entity.php, EntityProvider.php, ErrorMessage.php, FormSanitizer.php, PreviewProvider.php, SearchResultsProvider.php, Season.php, SeasonProvider.php, User.php, and Video.php.
- Editor:** Displays the PHP code for PreviewProvider.php, focusing on the creation of category preview videos and a general preview video.
- Status Bar:** Shows the current line (Ln 35, Col 5), spaces (Spaces: 4), encoding (UTF-8), line endings (LF), PHP, Go Live, and the date/time (22:13 10-04-2022).

The screenshot shows the Visual Studio Code interface with the following details:

- File Path:** includes > classes > PreviewProvider.php
- Code Content (PreviewProvider.php):**

```
61     return "<div class='previewContainer'>
62         <img src='$thumbnail' class='previewImage' hidden>
63
64         <video autoplay muted class='previewVideo' onended='previewEnded()'>
65             <source src='$preview' type='video/mp4'>
66         </video>
67
68         <div class='previewOverlay'>
69             <div class='mainDetails'>
70                 <h3>$name</h3>
71
72                 <div class='buttons'>
73                     <button><i class='fas fa-play'></i> Play</button>
74                     <button onclick='volumeToggle(this)'><i class='fas fa-volume-mute'></i></button>
75                 </div>
76             </div>
77         </div>
78     </div>";
79
80
81 }
82
83
84
85
86
87
88 public function createEntityPreviewSquare($entity) {
89     $id = $entity->getId();
90     $thumbnail = $entity->getThumbnail();
91     $name = $entity->getName();
92
93     return "<a href='entity.php?id=$id'>
94         <div class='previewContainer small'>
95             <img src='$thumbnail' title='$name'>
96         </div>
97     </a>";
98 }
99
100 private function getRandomEntity() {
101
102     $entity = EntityProvider::getEntities($this->con, null, 1);
103
104     return $entity[0];
105 }
106
107 ?>
```

- Status Bar:** Ln 35, Col 5 | Spaces: 4 | UTF-8 | LF | PHP | Go Live | 22:13 | ENG | 10-04-2022

The screenshot shows the Visual Studio Code interface with the following details:

- File Path:** includes > classes > PreviewProvider.php
- Code Content (PreviewProvider.php):**

```
87
88     public function createEntityPreviewSquare($entity) {
89         $id = $entity->getId();
90         $thumbnail = $entity->getThumbnail();
91         $name = $entity->getName();
92
93         return "<a href='entity.php?id=$id'>
94             <div class='previewContainer small'>
95                 <img src='$thumbnail' title='<span>$name</span>'>
96             </div>
97         </a>";
98     }
99
100    private function getRandomEntity() {
101
102        $entity = EntityProvider::getEntities($this->con, null, 1);
103
104        return $entity[0];
105    }
106
107 ?>
```

- Status Bar:** Ln 35, Col 5 | Spaces: 4 | UTF-8 | LF | PHP | Go Live | 22:13 | ENG | 10-04-2022

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** SearchResultsProvider.php - Netflix - Visual Studio Code.
- Explorer:** Shows a tree view of the project structure under the NETFLIX folder, including ajax, assets, entities, includes, and classes. The SearchResultsProvider.php file is selected.
- Editor:** Displays the PHP code for the SearchResultsProvider class. The code defines a constructor that takes a database connection and a user name, and a getResults method that returns HTML for preview categories.
- Bottom Bar:** Includes tabs for EntityProvider.php, ErrorMessage.php, FormSanitizer.php, PreviewProvider.php, and SearchResultsProvider.php. It also shows status information like Line 1, Column 1, Spaces: 4, UTF-8, CRLF, PHP, and a timestamp of 22:13 on 10-04-2022.

This screenshot is similar to the first one but shows code completion suggestions for the \$entity variable in the \$entitiesHtml .= \$previewProvider->createEntityPreviewSquare(\$entity); line. The suggestions include 'Category', 'Genre', 'Movie', and 'Show'.

The code completion interface shows the following suggestions:

- Category
- Genre
- Movie
- Show

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** Season.php - Netflix - Visual Studio Code.
- Explorer:** Shows a tree view of the project structure under the 'NETFLIX' folder:
  - ajax
  - assets
  - entities
  - includes
    - classes
      - Account.php
      - CategoryContainer.php
      - Constants.php
      - Entity.php
      - ErrorProvider.php
      - ErrorMessage.php
      - FormSanitizer.php
      - PreviewProvider.php
      - SearchResultsProvider.php
      - Season.php**
      - SeasonProvider.php
      - User.php
      - Video.php
- Editor:** Displays the PHP code for the Season.php class.

```
1 <?php
2 class Season {
3
4     private $seasonNumber, $videos;
5
6     public function __construct($seasonNumber, $videos) {
7         $this->seasonNumber = $seasonNumber;
8         $this->videos = $videos;
9     }
10
11    public function getSeasonNumber() {
12        return $this->seasonNumber;
13    }
14
15    public function getVideos() {
16        return $this->videos;
17    }
18
19 }
20 ?>
```
- Bottom Status Bar:** Ln 1, Col 1, Spaces: 4, UTF-8, LF, PHP, Go Live, 23:14, ENG, 10-04-2022.
- Search Bar:** Type here to search.

## 9 Future Scope

Currently our project is constrained by Cost and Time factors in development. But in the future, with enough efforts it can be developed and more improved.

- 1. Desktop Application Software** Now, we have implemented our Streaming platform as Website. But in the future it can be ported in to as application software in Windows, Linux or Mac, etc.
- 2. Mobile Application** It can be developed into mobile application for more ease of usage.
- 3. More available Content** With enough resources we can provide our user more cheap content to watch with smooth user experience.

# 10 Conclusion

We learned quite a few new things through the project.

**Problem Identification and Analysis:** From the state of the art survey, we identified a need of a more user friendly cheap alternative to OTT platforms. We attempted to recreate a more popular site with less cost.

**Knowledge of Software Development Techniques:** We discovered and learned various SDLC (Software Development Life Cycle) and Testing models.

**Collaboration:** We learned to collaborate between two developers working on same project

**Project Planning and Workflow:** We learned how to planned a project beforehand and follow accordingly with selected model.

**Project Documentation:** We created Gantt diagrams and Workflow diagrams for suitable planning of the project. We developed UML diagrams for suitable maintaince of the project.

**Project Completion within Deadline:** We successfully created a complete Online Streaming Site.

## 11 GitHub Repository Link

Project GitHub Link : <https://github.com/amanjpatil/Notflix>