# CLOUD APPLICATION AND DEVELOPMENT(CAD)

## Project: Media Streaming with IBM Cloud Video Streaming

### Phase 3: Development Part 1

Start Building the virtual cinema platform using IBM Cloud Video Streaming. By defining Platform definition and User Interface Design. Also discuss about the User Registration and Authentication.

## Introduction:

Creating a virtual cinema platform involves several components, and it's a complex task that requires various technologies. So we define the detailed procedure for developing a Virtual Cinema Platform using IBM Cloud Video Streaming.

## Platform Definition and User Interface Design:

Here, we consider the simpler platforms to build the respective application as,

- Using Python and Flask for the Backend,
- Using HTML and CSS for the Frontend,
- Using JavaScript for handling client-side interactions.

### Platform Features:

The below features are basics to a Video Streaming Application,

1. **User Registration and Authentication-** Authentication is handled using IBM Cloud Video Streaming credentials and optionally integrated with a user authentication system.

2. **Video Streaming and Management-** Supports live video streaming and on-demand playback and Video metadata, such as title, description, and thumbnail, can be edited.

3. **Real-Time Chat-** Integrated real-time chat for users to engage in discussions while watching videos to Enhances the social aspect of the virtual cinema experience.

4. **On-Demand Playback-** With On-Demand, viewers can pause, play, fast-forward, rewind, and re-watch the show On-Demand as much as they would like to watch.

5. **User Feedback-** Provides feedback messages for successful actions or errors.

## Intuitive User Interface Design:

For an Intuitive User Interface to the Virtual Cinematic Platform, we list out some ideas to give better User Interface (UI).

First we build our application into pages which includes:

- **Home Page**

  -contains Navigation links (My Videos, Playlists, Profile, Logout)

  -Thumbnails of Videos selection allows users to explore videos by categories or genres.

- **Video Playback Page**

  -Displays the video player with controls for play, pause, volume, and full-screen mode. Includes video title, description, and uploader information.

  -Real-time chat panel for users to engage in conversations.

- **User Profile Page**

  -Includes Profile information's like user statistics, such as the number of uploaded videos and playlists.

  -User's playlists with the option to create, edit, or delete.

- **Upload Video Page**

  -Allows users to select a video file for upload which includes fields for title, description, and thumbnail selection.

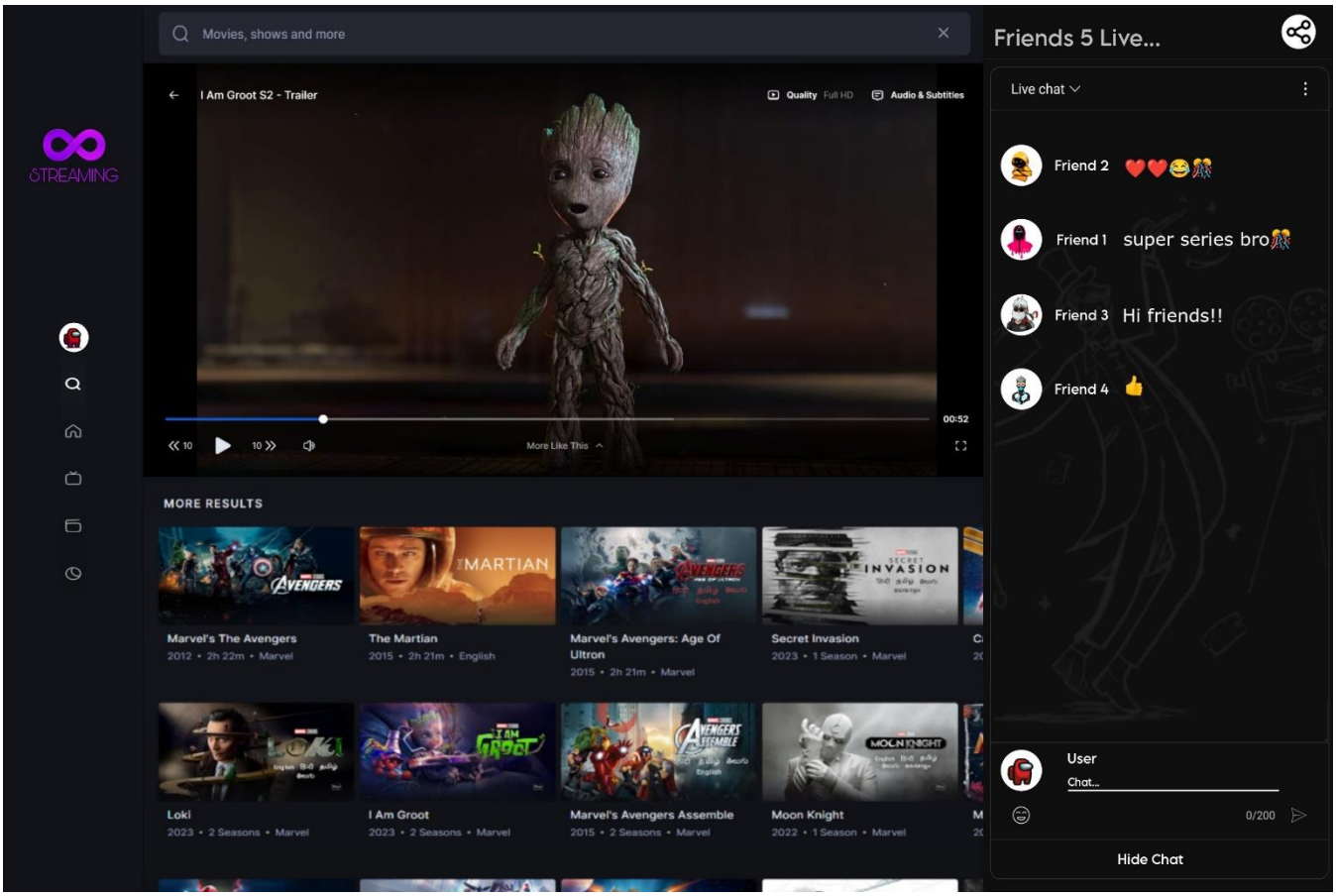
- **Playlist Page**

  -Allows users to add, view or manage videos to existing playlists or create new ones.

Our Design Principles is to be simple and user_friendly navigation with a clear menu structure. Design a responsive layout for a seamless experience on different devices. Design intuitive controls for video playback and interaction like engaging visuals such as thumbnails and cover pages. Mainly to provide feedback messages and visual cues for user actions.

We creating this application by HTML and CSS for a few key pages in the virtual cinema platform: the home page, the video playback page, the user profile page, and the upload video page.

Please note that this is only the sample code snippets for the implementation of Virtual Cinema Platform integrating with IBM Cloud.




## User Registration and Authentication:

In this development part we set up user registration and authentication mechanisms to ensure secure access to the platform. As we referred from the web resource here are the common procedures.

# User Registration and Authentication Setup Procedure

### 1. Package Installation:

- Begin by installing the required packages for your Flask application. Execute the following command in your terminal or command prompt:

**Bash**

```bash
pip install Flask Flask-Login Flask-WTF Flask-Bcrypt
```

### 2. Flask App Configuration:

- Create a new file named app.py to configure your Flask application. Import necessary modules, set a secret key, and define your database configuration. Here's a simplified snippet:

**Python**

```python
from flask import Flask, render_template, redirect, url_for, flash
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager, UserMixin, login_user,
login_required, logout_user, current_user
from flask_bcrypt import Bcrypt
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField
from wtforms.validators import DataRequired, Length, EqualTo

app = Flask(   name   )
app.config['SECRET_KEY'] = 'your_secret_key' app.config
['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///site.db' db = SQLAlchemy(app)
bcrypt = Bcrypt(app) login_manager = LoginManager(app) login_manager.login_view = 'login'

```

### 3. User Model Definition:

- Define the User model in python file to represent users in the database. Include fields like id, username, email, and password.

### 4. Form Creation:

- Create registration and login forms using Flask-WTF. These forms include fields for username, email, password, and confirm_password.

5. **User Registration Route:**

- Implement a route in app.py for user registration (/register). Handle form submissions, hash passwords with Flask-Bcrypt, and store user information in the database.

6. **User Login Route:**

- Set up a user login route (/login). Validate user credentials, log the user in using Flask-Login, and redirect to the main page upon successful login.

7. **User Logout Route:**

- Create a user logout route (/logout). Log the user out using Flask-Login and redirect to the main page.

8. **Application Execution:**

- Run your Flask application using the command flask run in the terminal or command prompt. Access the application at http://127.0.0.1:5000/.

By following these simplified steps, users can seamlessly register, log in, and log out of your Flask application. Customize routes, templates, and additional features based on your specific application requirements. `