Media Streaming with IBM Cloud Video Streaming

Phase 3: Development Part 1

In this phase, starting to build virtual cinema platform using IBM Cloud Video Streaming. Defining the platform's features and designing an intuitive user interface and setting up user registration and authentication mechanisms to ensure secure access to the platform.

The project involves creating a virtual cinema platform using IBM Cloud Video Streaming. The objective is to build a platform where users can upload and stream movies and videos on-demand. This project encompasses defining the virtual cinema platform, designing the user interface, integrating IBM Cloud Video Streaming services, enabling on-demand video playback, and ensuring a seamless and immersive cinematic experience.

Virtual Cinema platform using IBM Cloud Streaming:

Platform Features:

- Movie catalogue with titles, descriptions, and ratings.
- User profiles with preferences and watch history.
- Virtual screening rooms for different movies.
- Chat or interaction features for viewers.
- Secure payment integration for rentals or purchases.
- Reviews and ratings system.
- Admin panel for content management.
- Analytics for user behaviour and movie popularity.

Design User Interface:

- Create wireframes and mock ups for the user interface (UI) to visualize how users will interact with your platform.
- Focus on creating an intuitive and user-friendly interface to ensure a seamless experience.

Select IBM Cloud Video Streaming:

- Choose the specific services and features from IBM Cloud Video Streaming that will support your streaming needs.
- Configure and set up your video streaming infrastructure.

User Registration and Authentication:

- Implement a user registration system to allow users to create accounts.
- Utilize authentication mechanisms to secure access to the platform.
- You can consider options like email/password, social media login, or single sign-on (SSO) depending on your target audience and security requirements.

Database Setup:

- Set up a database to store user profiles, movie information, and user preferences.
- Choose a database service that suits your needs, such as IBM Cloud Databases.

Frontend and Backend Development:

- Develop the frontend of your platform based on the UI design.
- Implement the backend to handle user authentication, movie catalogue, streaming functionality, and user interactions.
- Choose appropriate technologies and frameworks for both frontend and backend development.

Secure Streaming:

- Ensure secure video streaming by using encryption and access controls.
- Integrate with IBM Cloud Video Streaming's security features for content protection.

Payment Integration:

- If you plan to charge for movie rentals or purchases, integrate a secure payment gateway to handle transactions.

Testing and Quality Assurance:

- Thoroughly test your platform for usability, security, and performance.
- Conduct beta testing with a select group of users to gather feedback.

Deployment and Scaling:

- Deploy your platform to a production environment, and set up necessary scaling mechanisms to handle increased traffic.

User Support and Feedback:

- Provide customer support channels and gather user feedback to continuously improve the platform.

Marketing and Promotion:

- Develop a marketing strategy to attract users to your virtual cinema platform.

Designing user interface

Understanding Users:

- Before starting design, understand target audience and their preferences. What are their needs and expectations when using a virtual cinema platform?

Clear Navigation:

- Create a clear and straightforward navigation structure. Use menus, tabs, or a sidebar to help users find their way around the platform easily.

Consistent Layout:

- Maintain a consistent layout throughout the platform. Users should know where to find common elements like the search bar, user profile, and movie categories.

Responsive Design:

- Ensure UI to be responsive and adapts to different screen sizes and devices. This is essential for users accessing your platform from various devices like smartphones, tablets, and desktops.

Visually Pleasing Design:

- Use a visually appealing color scheme that complements your brand and creates an inviting atmosphere. Incorporate high-quality images and graphics.

User-Friendly Forms:

- If there is registration or payment forms, make them user-friendly with clear labels, error messages, and validation to prevent user frustration.

Effective Search and Filters:

- Implement a robust search feature and filters that allow users to quickly find movies by genre, release date, or other criteria.

User Feedback:

- Include features for user feedback, such as star ratings and reviews, to help users make informed decisions.

Progress Indicators:

- When users initiate actions like renting a movie, provide clear progress indicators to keep them informed about the process.

User Profile and Settings:

- Create a user profile section where users can manage their preferences, settings, and view their watch history.

Intuitive Play Controls:

- Design intuitive play controls for watching movies. Users should be able to play, pause, skip, and adjust volume easily.

Accessibility Features:

- Implement accessibility features to accommodate users with disabilities, such as screen readers and keyboard navigation.

Testing and User Feedback:

- Conduct usability testing with real users to gather feedback on the platform's UI. Make improvements based on their insights.

Keep Load Times Short:

- Ensure that the platform loads quickly to keep users engaged and minimize frustration.

Mobile-First Approach:

- Given the increasing use of mobile devices, consider a mobile-first approach in your design to ensure a seamless experience on smaller screens.

Branding and Consistency:

- Maintain a consistent branding throughout the platform. Use logos, icons, and color schemes that reinforce your platform's identity.

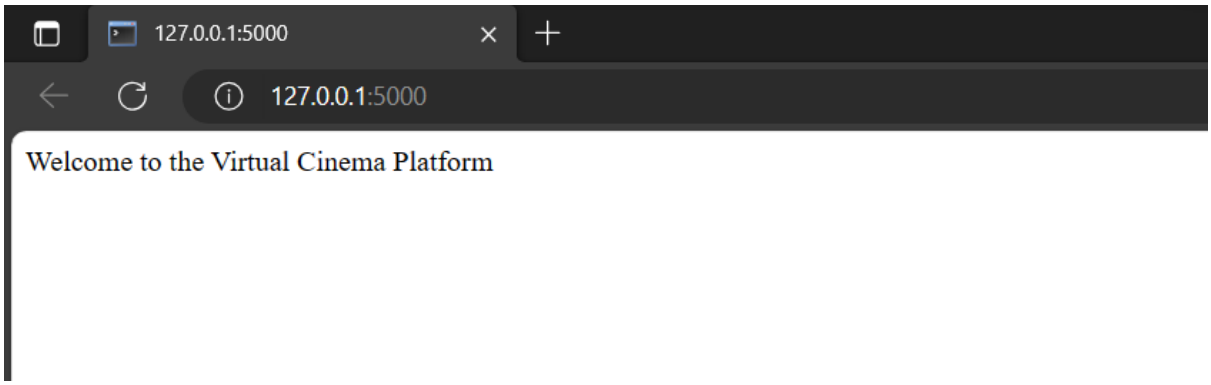
Setting up user registration and authentication:

app.py

```
from flask import Flask, render_template, request, redirect, url_for, session

from flask_sqlalchemy import SQLAlchemy

from database import db

from flask.cli import with_appcontext

app = Flask(__name__)
```

```
app.secret_key = '20062002'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///virtual_cinema.db'
db = SQLAlchemy(app)
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    password = db.Column(db.String(100), nullable=False)
@app.cli.command('init-db')
@with_appcontext
def init_db():
    db.create_all()
@app.route('/')
def home():
    return 'Welcome to the Virtual Cinema Platform'
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        user = User(username=username, password=password)
        db.session.add(user)
        db.session.commit()
        return redirect(url_for('login'))
    return render_template('register.html')
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        user = User.query.filter_by(username=username, password=password).first()
        if user:
```

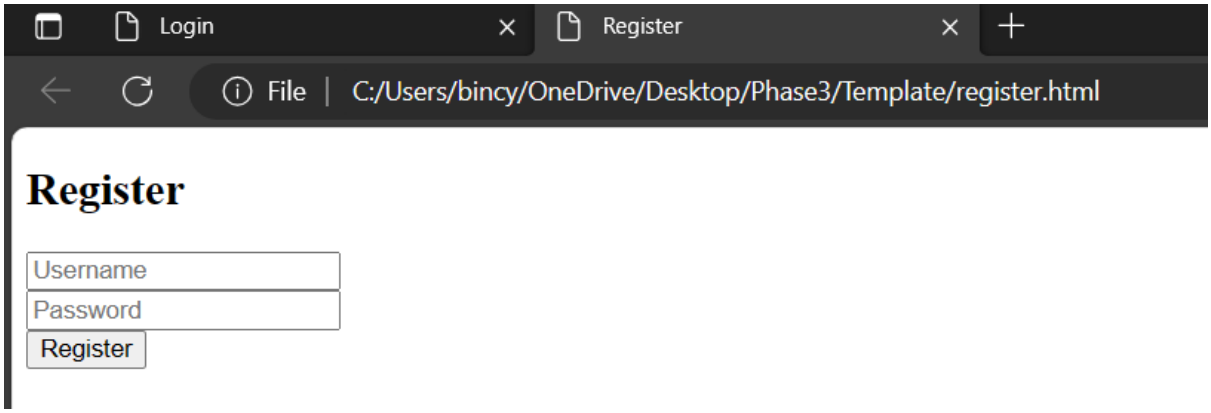
```
            session['user_id'] = user.id
            return 'Logged in as ' + user.username
        else:
            return 'Invalid username or password'
    return render_template('login.html')
if __name__ == '__main__':
    with app.app_context():
        app.run(debug=True)
```



Templates for register and login:

register.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Register</title>
</head>
<body>
    <h2>Register</h2>
    <form method="POST">
        <input type="text" name="username" placeholder="Username" required><br>
        <input type="password" name="password" placeholder="Password" required><br>
        <input type="submit" value="Register">
    </form>
</body>
</html>
```



login.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
    <h2>Login</h2>
    <form method="POST">
        <input type="text" name="username" placeholder="Username" required><br>
        <input type="password" name="password" placeholder="Password" required><br>
        <input type="submit" value="Login">
    </form>
</body>
</html>
```