Phase 4: Development Part 2 Documentation

# Video Upload and On-Demand Playback System Development

1. **Video Upload and Management Interface:**

**i. Video Upload Interface**

In this phase, you'll create a user-friendly interface for uploading and managing videos on your virtual cinema platform. Users should be able to upload their videos, provide necessary information, and manage their uploaded content.

* **Video Upload Interface**: Design an HTML form that allows users to upload videos. This form should include fields for the video's title, description, and the video file itself. Users will fill in these details when uploading their content.
* **Handling the Form Submission**: Use JavaScript to handle the form submission. When the user submits the form, JavaScript collects the data, including the video title, description, and the video file. It then sends this data to the Flask backend for processing.

**ii. Video Management Interface**:

After uploading videos, users need a way to view and manage their content. This part of the project involves creating a page where users can see the videos they've uploaded and take actions like editing or deleting them.

* **Video Management Page**: Design an HTML page where users can see a list of their uploaded videos. This list should be dynamically populated using JavaScript.
* **Dynamically Populating Video List**: Use JavaScript to fetch the user's uploaded videos from the backend and populate the video list. For each video, you can provide options to edit or delete it.

**2. Video Streaming Integration:**

* **IBM Cloud Video Streaming**: Integrate IBM Cloud Video Streaming services into your application. Use the provided APIs and SDKs to facilitate video streaming. Set up the necessary configurations to ensure smooth video playback.
* **User Video Libraries**: Create a system that manages user video libraries. When users upload videos, they should be stored securely on IBM Cloud Object Storage. The database (IBM DB2) should maintain records of the videos, including their storage locations and metadata.

**3. Video Playback Interface:**

Your virtual cinema platform should enable users to select and watch videos. Create a video playback interface that offers a seamless viewing experience.

* **Video Playback Page**: Design an HTML page with a video player that allows users to watch selected videos. The video player should include controls for play, pause, volume, and full screen mode.
* **Handling Video Playback**: Use JavaScript to handle video playback. When a user selects a video to watch, JavaScript dynamically sets the video source (URL) based on the user's selection. This URL points to the video stored on IBM Cloud Object Storage.

**4. Integration with Flask and IBM Services:**

All the data related to video upload, management, and playback needs to be processed by your Flask backend and interact with IBM services.

* **Flask Routes**: In your Flask app, create routes to handle video upload, video management, and video playback. These routes will define how the server processes requests related to these actions.

**5. Search and Filtering Functionality:**

* Implement search and filtering options to help users easily find and access videos. You can use HTML form elements and JavaScript to create search bars, dropdown filters (e.g., by genre or release date), and a search results display.

**6. User Interactions:**

* Enable user interactions with videos, such as likes, comments, and sharing options. You can use JavaScript to create interactive elements that allow users to engage with the content and with each other.

**7. Playlists and User Collections (if applicable):**

* If you plan to incorporate user-generated playlists or collections, design an interface for users to create, manage, and organize their playlists. Use IBM DB2 to store playlist data and link it to the videos.

**8. Real-time Chat (if planned):**

* If real-time chat is part of your project, design and integrate a chat interface that runs in parallel with video playback. Use JavaScript and HTML for the chat functionality and ensure that messages are synchronized with video content.

**9. Video Quality Options:**

* Allow users to select video quality options based on their internet connection and device capabilities. Implement a user-friendly quality selector in the video playback interface.

**10. Testing and Quality Assurance:**

To ensure a smooth and high-quality user experience, rigorous testing and quality assurance are essential.

* **Video Testing**: Thoroughly test the video upload, management, and playback interfaces. Verify video quality, loading times, and compatibility across various devices and browsers.

**11. User Feedback and Usability Testing:**

To continually improve the platform, collect user feedback and conduct usability testing specifically related to video upload and playback features.

**12. Documentation:**

Provide user documentation or tooltips explaining how to upload, manage, and play videos on the platform. This documentation should help users navigate and make the most of these features.

**13. Legal and Copyright Compliance:**

Ensure that video uploads and playback comply with legal requirements, particularly those related to copyrights and licensing. Users should not be able to upload copyrighted material without appropriate permissions.

**14. User Support and Assistance:**

Offer clear error messages and support channels for users encountering issues during video-related actions. Providing support is crucial for a positive user experience.

**Frontend Development:**

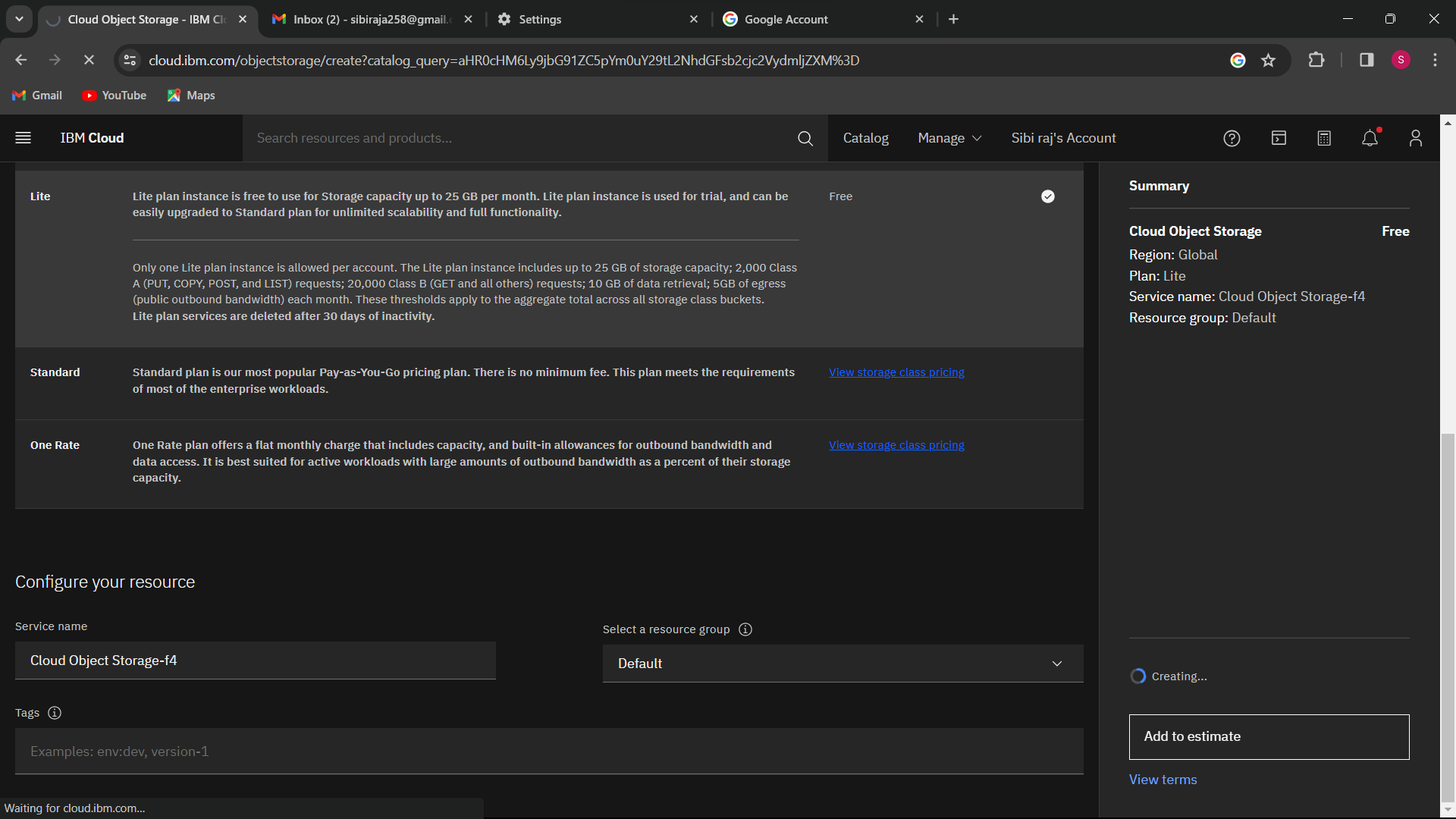
1. **HTML for Video Upload**:
   * Create an HTML page that allows users to upload videos. This page should include a file input for selecting video files and an upload button. Save this file as **upload.html**.
2. **CSS Styling**:
   * Style the video upload page using CSS to make it user-friendly. Save this CSS file as **style.css**.
3. **JavaScript for Video Upload**:
   * Implement JavaScript to handle video uploads. When the user selects a video and clicks the upload button, use JavaScript to capture the selected file and send it to the server for processing.

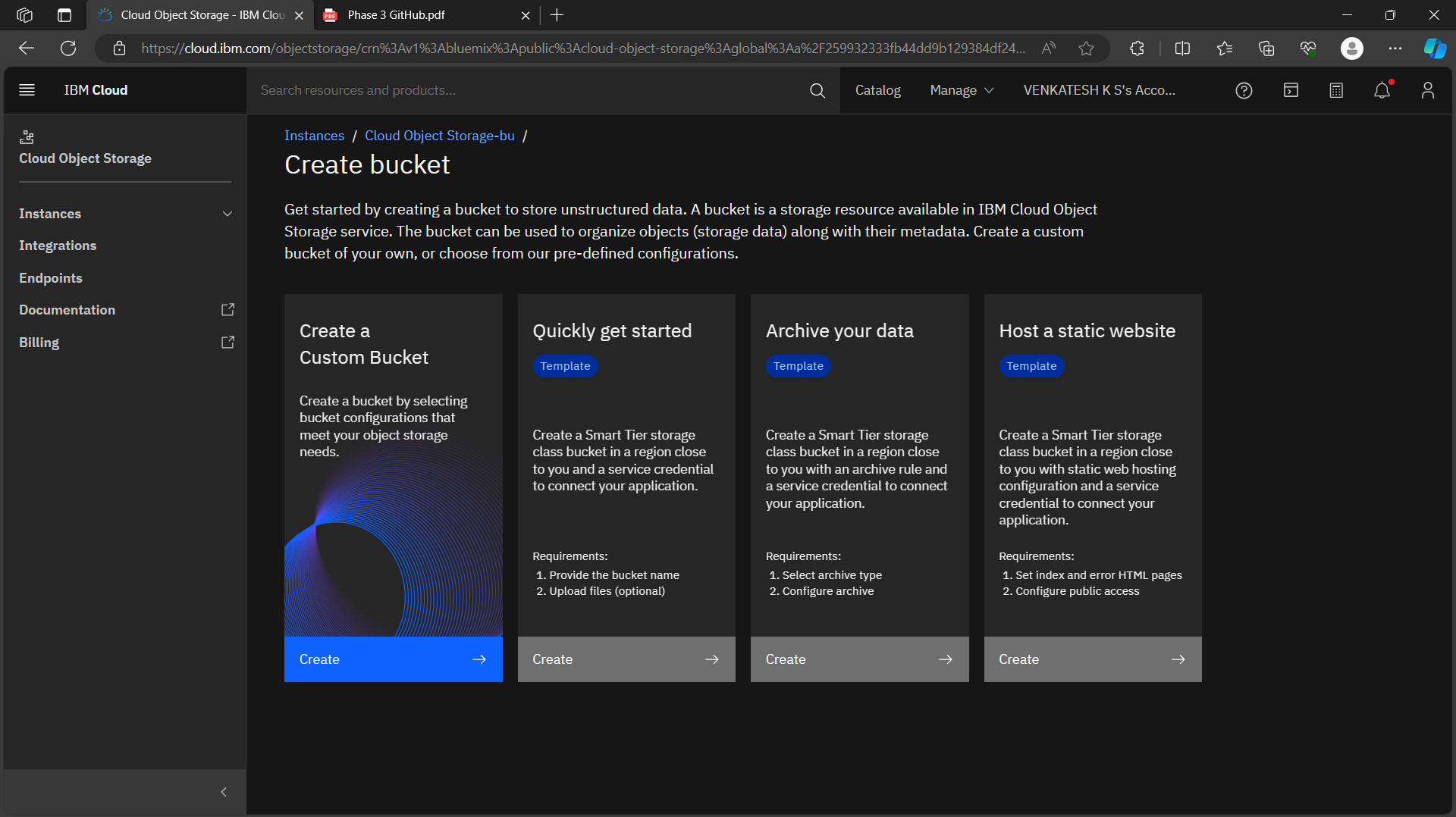
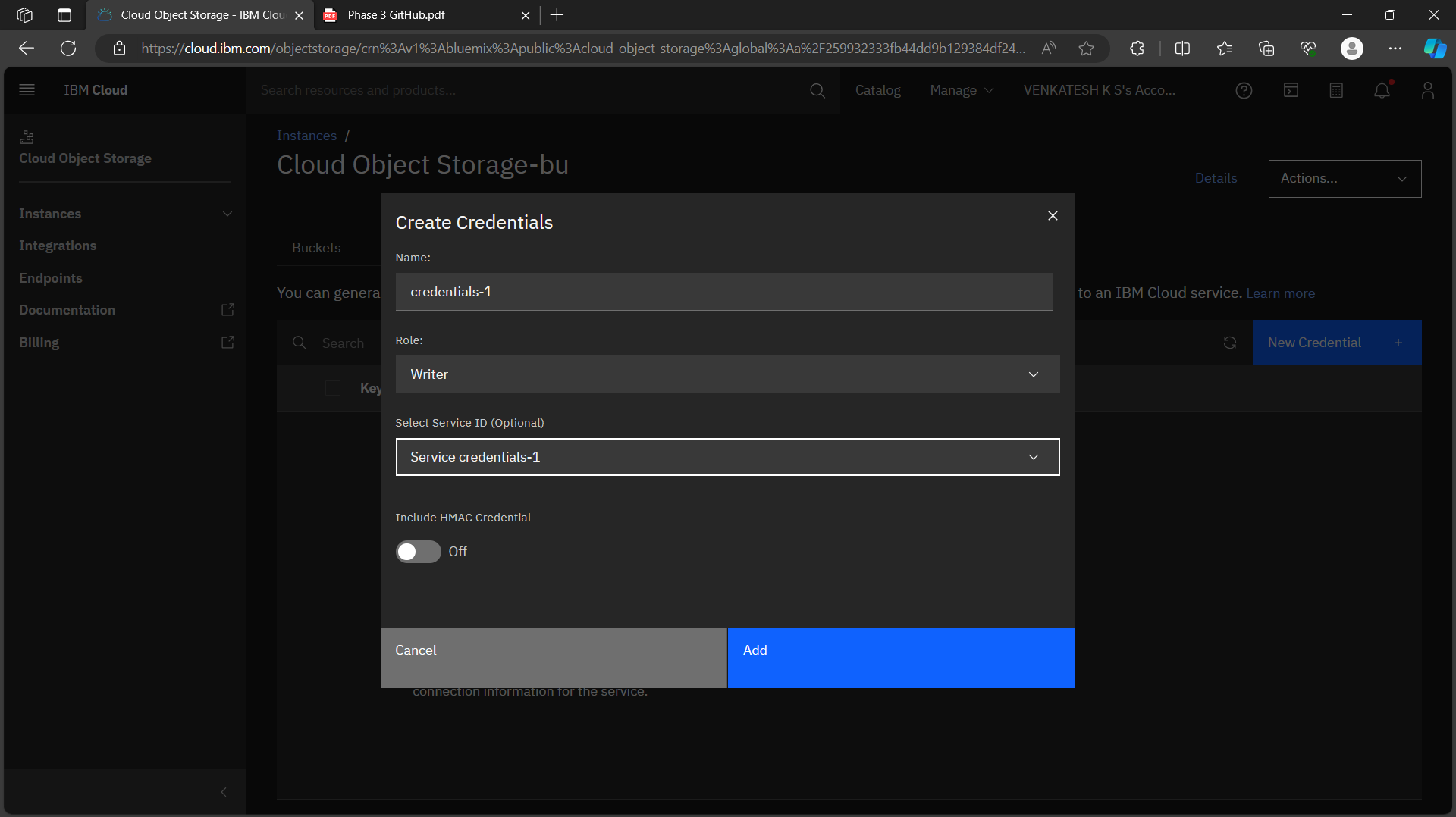
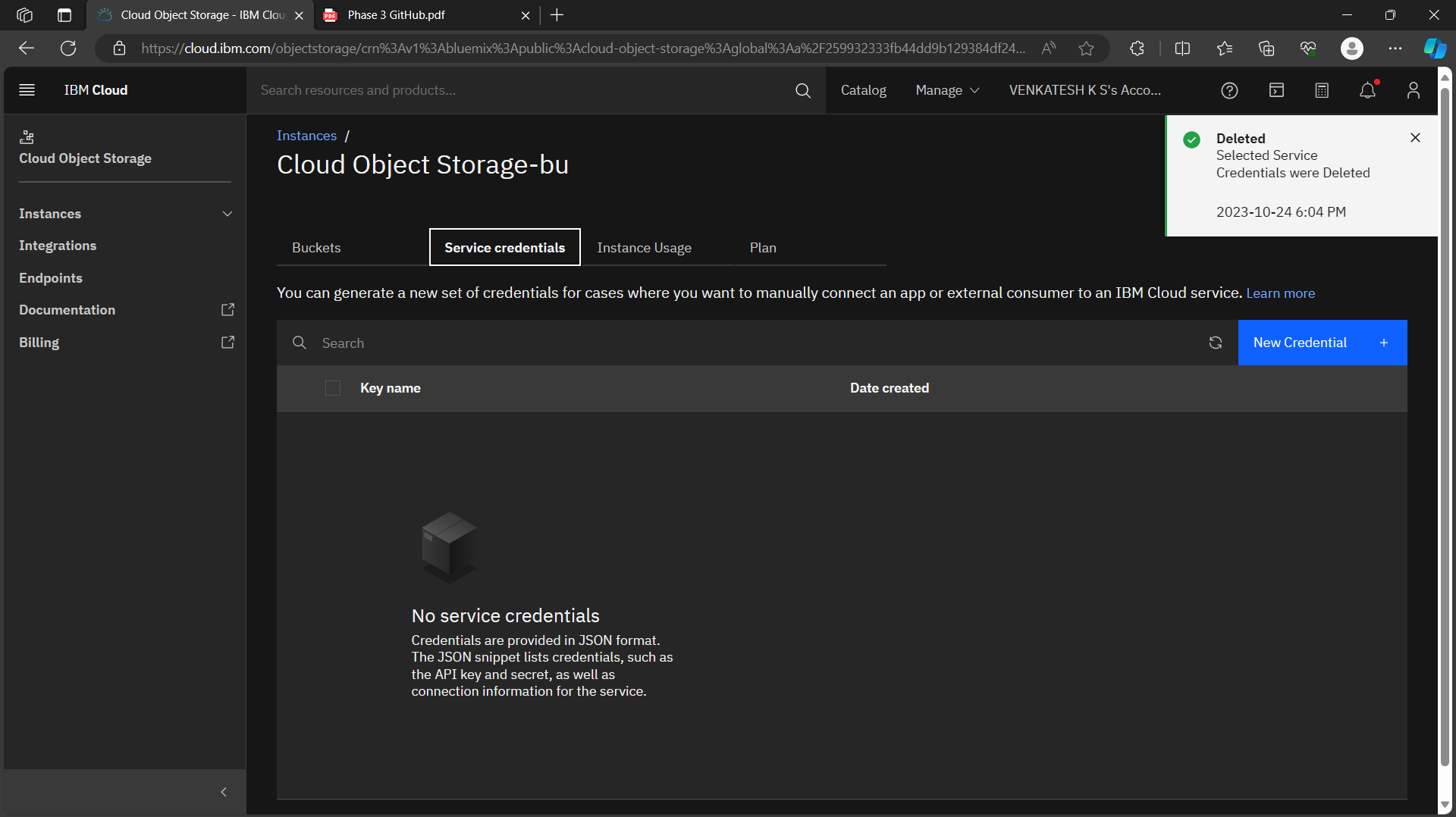
**Backend Development:**

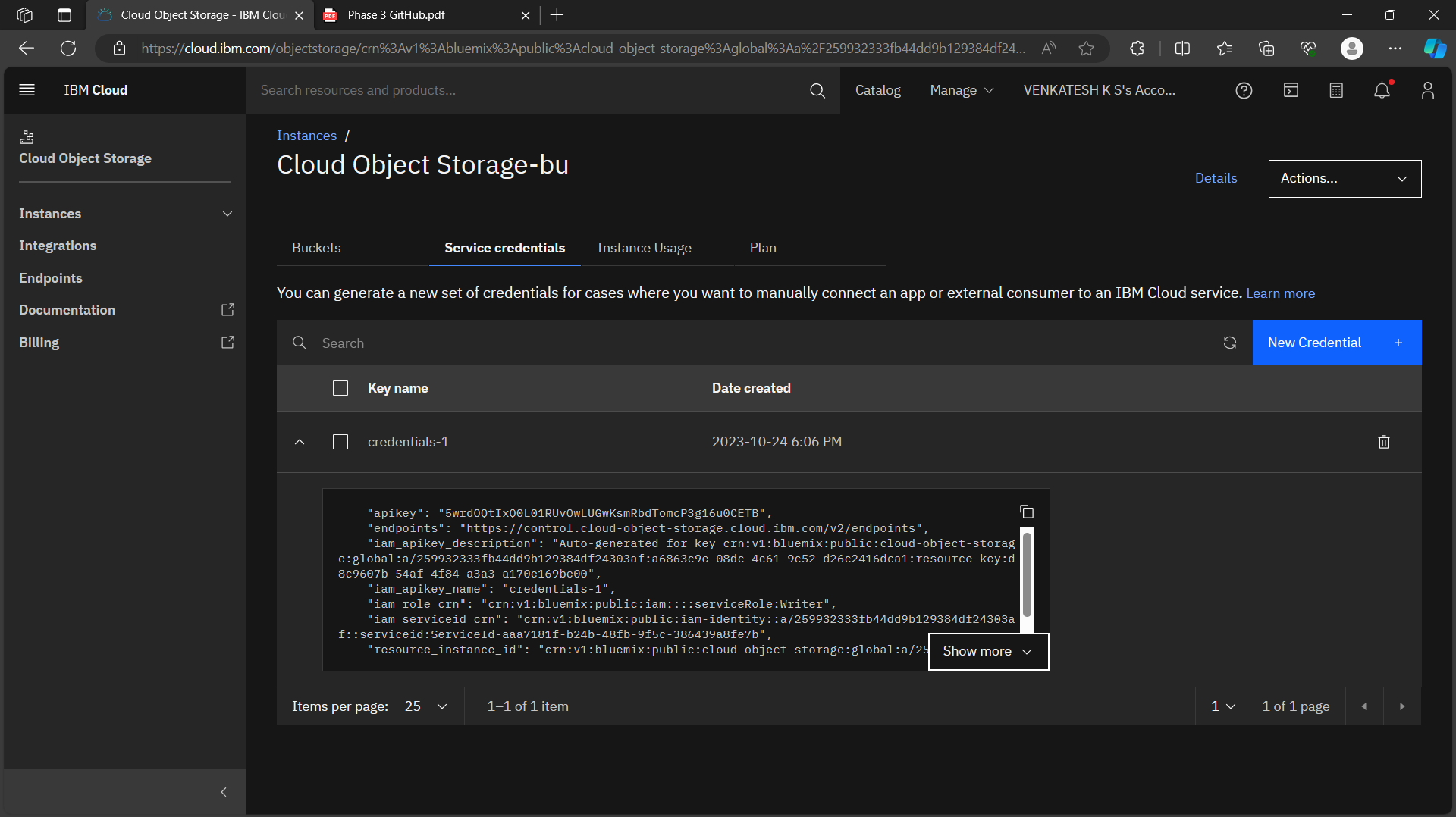
1. **Flask Initialization**:
   * Continue building on your Flask application, which serves as the backend of your virtual cinema platform. In the context of Phase 4, your Flask application should have the following components:
2. **IBM Db2 Integration**:
   * IBM Db2 is used to store video metadata and user information. As part of your backend development, you should establish a connection to your Db2 database, define a schema for video data, and implement CRUD (Create, Read, Update, Delete) operations. Here's how you can set up the IBM Db2 integration:
   * **Connect to IBM Db2**:
     + Use the **ibm\_db** library to connect to your IBM Db2 instance. Provide the necessary database connection details, such as the database name, host, port, protocol, username, and password.
   * **Database Schema Design**:
     + Design your database schema to store video metadata, including video titles, descriptions, and references to video files stored in IBM Cloud Object Storage. You may also store user information, such as the user who uploaded the video.
   * **Upload Video Metadata**:
     + When a video is uploaded, store its metadata in the IBM Db2 database. This includes details like the video title, description, user ID, and a reference to the video file stored in IBM Cloud Object Storage.
   * **Retrieve Video Metadata**:
     + Implement endpoints in your Flask application to retrieve video metadata when a user wants to play a video. The video metadata will include the location of the video in IBM Cloud Object Storage.
3. **IBM Cloud Object Storage Integration**:
   * IBM Cloud Object Storage is used for storing video files securely and efficiently. Your backend should interact with this storage service for video uploads and retrieval. Here's how you can integrate IBM Cloud Object Storage:
   * **Video Upload**:
     + When a user uploads a video, you should use the IBM Cloud Object Storage SDK or API to save the video file to a designated storage container. Ensure that the uploaded video is named uniquely, and organize your storage containers effectively.
   * **Video Retrieval for Playback**:
     + When a user requests to play a video, your backend should fetch the video from IBM Cloud Object Storage. You can use the provided URL or reference to the video in your storage. The backend should then serve this video to the user.
   * **Security and Access Control**:
     + Implement security measures to ensure that only authorized users can upload, access, and play videos. IBM Cloud Object Storage provides access control mechanisms to help secure your data.

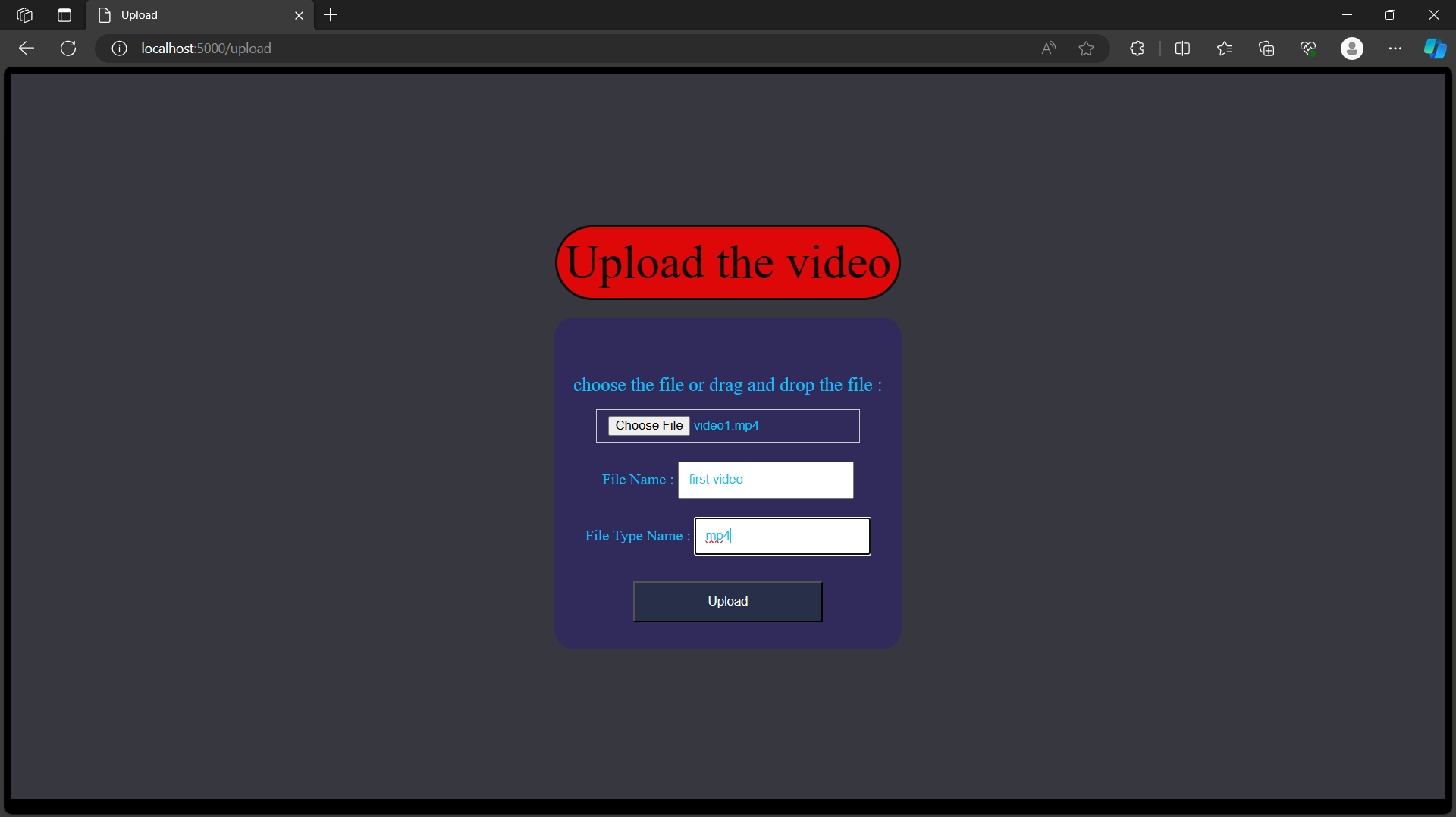
**BASIC DEVELOPMENT OF MY PROJECT:**

1. Login into ibm cloud
2. Goto catalog
3. then select storage section
4. then click on **object storage**
5. then Select free plan and
6. create cloud object storage

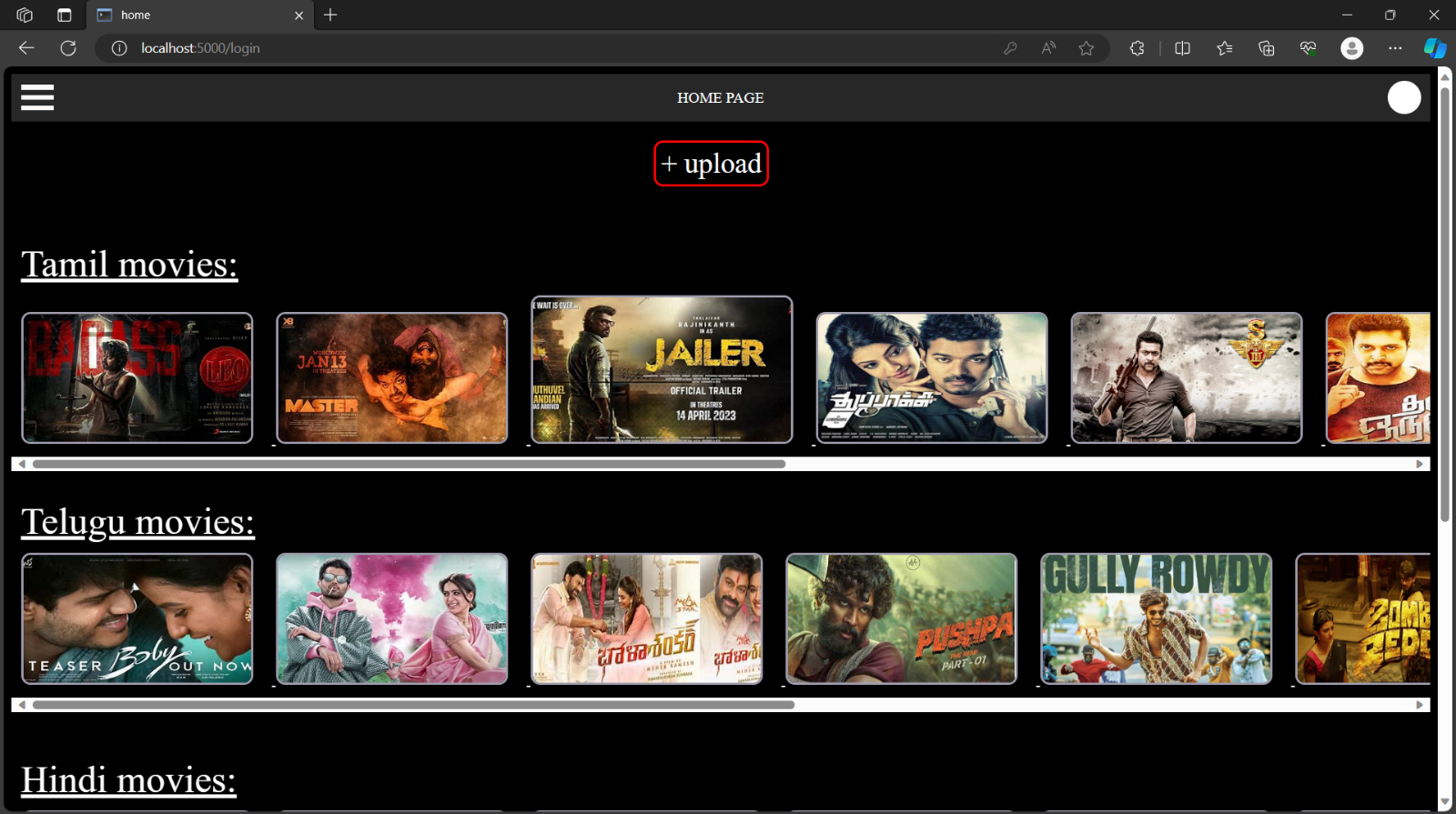


Create bucket  
  
  
Create a service credentials  


Get the details from service credentials like(apikey, endpoint, apikey name)

**Video upload page:** in this page you can select the page in 2 ways  
1 way click on the choose file,  
2 way drag and drop the file ****

**Home page :** its contains the list of thumbnails of videos in ibm object storage, its redirect to video player page when click the thumbnails in home page and play the corresponding video on the video player page



**Video player page and its below contains the chat box:** the video player page is open when the user click on the thumbnails that showing in home page then user redirected to the video player page then the corresponding video will be play an automatically

Below the video the chat box is available: the multiple user are may be watching at the same time then that user can chat to another user 