Project Title: Media Streaming with IBM cloud Streaming.

The Project phase-4 is about the continuing building the platform by integrating video streaming services and enabling on-demand playback and Implementing the functionality for users to upload their movies and videos to the platform and to Integrate IBM Cloud Video Streaming services to enable smooth and high-quality video playback.

Integrating video streaming services and enabling on-demand playback is a great way to enhance your platform. To achieve this, you can follow these steps:

1. Choose a Video Streaming Service: Select a reliable video streaming service like IBM Cloud Video or other popular options such as Vimeo, YouTube, or Amazon Web Services (AWS) Elemental MediaLive
2. Set Up an Account: Create an account on the chosen video streaming service and obtain necessary API keys or credentials.
3. Integrate the Service: Integrate the chosen video streaming service's API into your platform. This usually involves adding code that allows you to upload, store, and retrieve videos.
4. Implement User Uploads: Create a user-friendly interface for users to upload their videos or movies. This interface should support various video formats and sizes.Implement validation and security measures to ensure that uploaded content is safe and adheres to platform guidelines.
5. Video Encoding: Depending on the service, you might need to encode videos into various formats and bitrates to ensure smooth playback on different devices and connection speeds.
6. Storage: Store the uploaded videos securely, either on your servers or using cloud storage services.
7. IBM Cloud Video Integration: If you're using IBM Cloud Video, follow their documentation and guidelines to integrate their services for high-quality video playback.
8. Implement On-Demand Playback: Develop a user interface that allows users to browse, search, and select videos for on-demand playback.
9. Streaming Optimization: Ensure that videos are streamed efficiently using adaptive streaming technologies to adjust video quality based on the user's internet connection.
10. Content Delivery Network (CDN): Use a CDN for efficient content distribution to reduce load times and ensure low-latency streaming.
11. Monetization Options: If applicable, consider integrating payment gateways and subscription models for premium content.
12. Testing and Quality Assurance: Thoroughly test the platform to ensure a seamless user experience. Check for video quality, load times, and platform performance.
13. Scalability: Design the platform to handle increased traffic and video uploads as it grows.
14. User Feedback: Collect user feedback and continuously improve the platform based on their suggestions and needs.
15. Legal and Copyright Considerations: Be mindful of copyright and licensing issues when users upload their content. Implement content reporting and removal mechanisms.
16. User Engagement: Implement features like comments, likes, and sharing options to enhance user engagement.

These are the basic steps to perform the given task.

Here is some basic code for implementing our project.

```
# Import necessary libraries
Import ibm_boto3
Import ibm_s3transfer
From flask import Flask, request, render_template
From ibm_botocore.client import Config
# Initialize your IBM Cloud Video Streaming credentials
Api_key = 'your_api_key'
Service_instance_id = 'your_service_instance_id'
Auth_endpoint = 'https://iam.cloud.ibm.com/identity/token'
Service_endpoint = 'https://api.video.cloud.ibm.com'
Bucket_name = 'your_bucket_name'
# Create an IBM Cloud Video Streaming client
Cos = ibm_boto3.client('s3',
   Ibm_api_key_id=api_key,
   Ibm_service_instance_id=service_instance_id,
   Ibm_auth_endpoint=auth_endpoint,
   Config=Config(signature_version='oauth'),
   Endpoint_url=service_endpoint
)
# Initialize your web application
App = Flask(__name)
# HTML form for video uploads
@app.route('/upload', methods=['POST'])
Def upload_video():
   If 'video' not in request.files:
      Return "No video file provided"
 Video_file = request.files['video']
   If video_file.filename == '':
      Return "No selected video file"
```

```
 # Upload video to IBM Cloud Object Storage
    Cos.upload_fileobj(video_file, bucket_name, video_file.filename)
  Return “Video uploaded successfully!”
# HTML template for on-demand video playback
@app.route(‘/play/<video_name>’)
Def play_video(video_name):
    Video_url = cos.generate_presigned_url(
        ClientMethod=’get_object’,
        Params={‘Bucket’: bucket_name, ‘Key’: video_name}
    )
  Return render_template(‘video_player.html’, video_url=video_url)
If __name__ == ‘__main__’:
    App.run()
```

Make sure to install the necessary Python libraries (Flask, ibm_boto3, ibm_s3transfer) and replace ‘your_api_key’, ‘your_service_instance_id’, and ‘your_bucket_name’ with you’re actual IBM Cloud Video Streaming credentials. Additionally, you need to create the HTML templates for the upload form and video player.