**Step-by-Step Guide to Achieve Objective 2: Developing Dynamic QoE Adaptation Strategies in MCC**

**📌 Goal:** Create adaptive systems that optimize QoE by modifying service delivery in real-time based on QoE monitoring data.

**1.1 Identify Key Adaptation Parameters**

✔ **Network-Level Adaptation:** Bandwidth allocation, congestion control, network handover (Wi-Fi to LTE).  
✔ **Application-Level Adaptation:** Video bitrate adjustment, load balancing, task offloading.  
✔ **User-Centric Adaptation:** Personalized content delivery, UI adjustments based on device capabilities.

**🔸 1.2 Define QoE Thresholds for Adaptation**

Define QoE degradation conditions that trigger adaptation. Example:  
✔ **Latency > 200ms:** Switch to a lower bitrate for streaming.  
✔ **Packet loss > 5%:** Trigger redundant data transmission.  
✔ **Low user engagement:** Suggest content adjustments.

**Step 2: Implement Adaptive QoE Control in GCP (Google Cloud Platform)**

📌 **Objective:** Implement real-time adaptive mechanisms that dynamically adjust service parameters based on QoE monitoring data.

**Step-by-Step Deployment of QoE Adaptation in GCP**

✅ **Key Components to Deploy:**  
1️⃣ **Cloud Pub/Sub** → Collects QoE alerts in real-time.  
2️⃣ **Cloud Functions** → Triggers adaptive actions when QoE degrades.  
4️⃣ **Compute Engine / Kubernetes** → Adjusts system resources dynamically.

**🔹 Step 1: Create a Cloud Pub/Sub Topic for QoE Alerts**

📌 **Purpose:** Cloud Pub/Sub acts as a messaging service to **stream real-time QoE data** and trigger adaptation responses.

**1️⃣ Create a Pub/Sub Topic**

1. Go to **Google Cloud Console** → **Pub/Sub**.
2. Click **Create Topic** → Name it "QoE\_alerts".
3. Click **Create Subscription** → Choose **Pull** type for real-time processing.
4. Click **Create**.

A screenshot of a computer

AI-generated content may be incorrect.

✅ **Your system is now ready to collect QoE data from mobile apps, network monitoring tools, and cloud services.**

**🔹 Step 2: Deploy a Cloud Function to Trigger Adaptation**

📌 **Purpose:** When **QoE degradation** is detected, Cloud Functions **automatically adjust** network/application parameters.

**2️⃣ Create a Cloud Function**

1. Navigate to **Cloud Functions** in Google Cloud Console.
2. Click **Create Function** → Name it "QoE\_Adaptation\_Function".
3. Choose **Trigger Type** → Select **Pub/Sub** → Choose the topic "qoe-alerts".
4. Select **Runtime** → Choose **Python 3.9**.
5. Paste the following Python code in the **inline editor**:

python

CopyEdit

import base64

import json

from google.cloud import compute\_v1

# Initialize Compute Engine client

compute\_client = compute\_v1.InstancesClient()

def qoe\_adaptation(event, context):

"""Triggered when QoE drops below threshold."""

message = base64.b64decode(event['data']).decode('utf-8')

qos\_data = json.loads(message)

if qos\_data["latency"] > 200:

print("High latency detected. Optimizing resources...")

optimize\_network\_resources()

def optimize\_network\_resources():

"""Increases instance resources when network performance drops."""

instance\_name = "mcc-instance"

zone = "us-central1-a"

project\_id = "your-project-id"

request = compute\_v1.SetMachineResourcesRequest(

name=instance\_name,

zone=zone,

project=project\_id,

machine\_type="n1-standard-2" # Upgrade instance type dynamically

)

compute\_client.set\_machine\_resources(request=request)

print("Cloud resources optimized based on QoE metrics.")

1. Click **Deploy**.

✅ **Now, when QoE degrades, the Cloud Function automatically scales cloud resources.**

A screenshot of a computer

AI-generated content may be incorrect.

**🔹 Step 3: Train & Deploy AI-Based QoE Prediction Using Vertex AI**

📌 **Purpose:** Instead of reacting after QoE degrades, **use AI to predict issues before they happen** and trigger adaptive strategies.

**3️⃣ Prepare Training Data in BigQuery**

1. Go to **BigQuery** → Create a **dataset** named "QoE\_History".
2. Create a **table** "QoE\_Metrics" with columns:
   * timestamp (TIMESTAMP)
   * latency (INT)
   * jitter (INT)
   * packet\_loss (FLOAT)
   * user\_rating (INT)
   * adaptation\_action (STRING)
3. Load historical QoE data into the table.

A screenshot of a computer

AI-generated content may be incorrect.

**🔹 Step 3.1: Deploy Adaptive Strategies on Cloud**

✅ **Goal:** Ensure **real-time QoE adaptation** by deploying **Cloud Functions**.

**1️⃣ Deploy Cloud Functions for QoE-Based Adaptation**

📌 **Purpose:** Automate QoE-based actions (e.g., **scaling resources, switching video bitrates**) when **QoE degrades**.

**🛠 Steps to Deploy Cloud Function**

1️⃣ **Go to** Google Cloud Console → **Cloud Functions**.  
2️⃣ Click **Create Function** → Name it **"QoE\_Adaptive\_Response"**.  
3️⃣ Choose **Trigger Type: Pub/Sub** → Select the topic **"QoE\_alerts"**.  
4️⃣ Select **Runtime: Python 3.9**.  
5️⃣ Paste the following **Python code**:

python

CopyEdit

import base64

import json

from google.cloud import compute\_v1

# Initialize Compute Engine client

compute\_client = compute\_v1.InstancesClient()

def qoe\_adaptation(event, context):

"""Triggered when QoE drops below threshold."""

message = base64.b64decode(event['data']).decode('utf-8')

qos\_data = json.loads(message)

if qos\_data["latency"] > 200: # Example threshold

print("High latency detected. Optimizing resources...")

optimize\_network\_resources()

def optimize\_network\_resources():

"""Increases instance resources dynamically when network performance drops."""

instance\_name = "mcc-instance"

zone = "us-central1-a"

project\_id = "your-project-id"

request = compute\_v1.SetMachineResourcesRequest(

name=instance\_name,

zone=zone,

project=project\_id,

machine\_type="n1-standard-2" # Upgrade instance type dynamically

)

compute\_client.set\_machine\_resources(request=request)

print("Cloud resources optimized based on QoE metrics.")

6️⃣ Click **Deploy**.  
✅ Now,**Cloud Functions will automatically optimize resources when QoE degrades**.

A screenshot of a computer

AI-generated content may be incorrect.

**🔹 Step 3.2: Evaluate Performance & User Satisfaction**

✅ **Goal:** Ensure that **adaptation strategies improve QoE** by analyzing **performance trends** and collecting **user feedback**.

**3️⃣ Analyze QoE Trends in BigQuery**

📌 **Purpose:** **Track QoE changes over time** to measure the impact of **adaptive strategies**.

**🛠 Steps to Analyze QoE Data**

1️⃣ **Go to** Google Cloud Console → **BigQuery**.  
2️⃣ Select the dataset **QoE\_History** → Click **Query**.  
3️⃣ Run the following **SQL query** to compare **pre- and post-adaptation QoE metrics**:

sql

CopyEdit

SELECT

timestamp,

latency,

jitter,

packet\_loss,

adaptation\_action

FROM `your-project-id.QoE\_History.QoE\_Metrics`

WHERE timestamp > TIMESTAMP\_SUB(CURRENT\_TIMESTAMP(), INTERVAL 24 HOUR)

ORDER BY timestamp DESC;

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**4️⃣ Collect User Feedback via Firebase Surveys**

📌 **Purpose:** Validate **QoE improvements** based on **actual user experience**.

**🛠 Steps to Collect User Feedback**

1️⃣ **Set up Firebase Remote Config** in your mobile app.  
2️⃣ Use **Firebase In-App Messaging** to ask users for feedback when QoE degrades.  
3️⃣ Store responses in **BigQuery** for analysis.

📌 **Example Firebase Realtime Survey Setup**

javascript

CopyEdit

firebase.analytics().logEvent('QoE\_feedback', {

user\_rating: 4, // Example: User rates experience 4/5

issue\_reported: "High latency during video playback"

});

✅ **Expected Outcome:**

* Feedback is stored in **BigQuery**.
* You can analyze trends in **user satisfaction before & after adaptation**.
* **Final Outcome**
* 🚀 **Successfully deployed & evaluated QoE adaptation strategies in Google Cloud!**  
  ✔ **Automated QoE response with Cloud Functions & Pub/Sub.**  
  ✔ **Data-driven validation using BigQuery & Firebase feedback.**

Last step :

**Step-by-Step Guide: Collecting User Feedback via Firebase Surveys & Analyzing in BigQuery**

📌 **Objective:** Capture real-time **user feedback on QoE** using Firebase and store it in BigQuery for trend analysis.

**🔹 Step 1: Set Up Firebase Remote Config in Your Mobile App**

📌 **Purpose:** Firebase Remote Config allows you to control app behavior dynamically without releasing new updates.

**1️⃣ Add Firebase to Your Android/iOS App**

✔ **For Android:**

1. **Go to** Firebase Console → Click **Create Project**.
2. Select **Add App** → Choose **Android**.
3. Enter **Package Name** (e.g., com.example.qoeapp).
4. Download google-services.json and place it in your app’s /app directory.
5. Add the Firebase SDK to build.gradle:

gradle

CopyEdit

dependencies {

implementation platform('com.google.firebase:firebase-bom:32.2.3')

implementation 'com.google.firebase:firebase-analytics'

implementation 'com.google.firebase:firebase-config'

implementation 'com.google.firebase:firebase-messaging'

}

1. Sync and **Run the App**.

✔ **For iOS:**

1. Register the app in Firebase, download GoogleService-Info.plist, and add it to Xcode.
2. Install Firebase SDK using CocoaPods:

bash

CopyEdit

pod 'Firebase/Analytics'

pod 'Firebase/RemoteConfig'

pod 'Firebase/Messaging'

1. Initialize Firebase in AppDelegate.swift:

swift

CopyEdit

import Firebase

FirebaseApp.configure()

**🔹 Step 2: Display In-App Survey When QoE Degrades**

📌 **Purpose:** Show a **feedback prompt** to users when the app detects a QoE issue (e.g., high latency).

**2️⃣ Implement Firebase In-App Messaging**

✔ **Steps to Enable In-App Messaging for Surveys:**

1. **Go to** Firebase Console → Click **Engagement** → Select **In-App Messaging**.
2. Click **Create a Campaign** → Choose **Custom Message**.
3. Enter the **survey message** (e.g., *"How was your video streaming experience?"*).
4. Set **Trigger Condition**:
   * Example: When **QoE drops (latency > 250ms)**, trigger the survey.
5. Click **Save & Publish**.

✔ **Show a Survey Prompt Using Firebase Remote Config**  
📌 **Code to trigger a survey when latency is high:**

javascript

CopyEdit

firebase.remoteConfig().fetchAndActivate()

.then(() => {

const surveyPrompt = firebase.remoteConfig().getValue('qoe\_survey\_enabled');

if (surveyPrompt.asBoolean()) {

showSurveyDialog();

}

});

function showSurveyDialog() {

firebase.analytics().logEvent('QoE\_feedback\_prompt');

alert("How was your experience? Please rate your QoE.");

}

✅ **Expected Outcome:** Users receive a **survey popup** when QoE drops.

**🔹 Step 3: Store User Feedback in BigQuery**

📌 **Purpose:** Store feedback for trend analysis on **QoE improvement over time**.

**3️⃣ Enable Firebase → BigQuery Integration**

✔ **Steps to Connect Firebase Analytics to BigQuery:**

1. **Go to Firebase Console** → Click **Project Settings** → Select **Integrations**.
2. Under **BigQuery**, click **Link** → Enable **Export Firebase Analytics Data to BigQuery**.
3. Select **All Events** (including QoE\_feedback).
4. Click **Save & Confirm**.

**4️⃣ Send User Feedback to Firebase Analytics**

📌 **Modify the survey prompt to send user feedback:**

javascript

CopyEdit

function sendQoEFeedback(rating, issue) {

firebase.analytics().logEvent('QoE\_feedback', {

user\_rating: rating,

issue\_reported: issue

});

}

✅ **Now, QoE feedback is automatically sent to Firebase & stored in BigQuery.**

**🔹 Step 4: Analyze QoE Feedback in BigQuery**

📌 **Purpose:** Identify patterns in **QoE degradation & user satisfaction.**

**5️⃣ Run Queries in BigQuery to Analyze Trends**

✔ **Go to** Google BigQuery Console.  
✔ Click **New Query** and run the following SQL:

sql

CopyEdit

SELECT

event\_bundle\_sequence\_id AS feedback\_id,

event\_timestamp,

event\_name,

user\_properties.value.string\_value AS user\_rating,

user\_properties.value.string\_value AS issue\_reported

FROM `your-project-id.analytics\_your\_project\_id.QoE\_feedback`

WHERE event\_name = 'QoE\_feedback'

ORDER BY event\_timestamp DESC;

✅ **Expected Outcome:** A dataset showing **user ratings, QoE issues, and timestamps**.

**🔹 Step 5: Visualize QoE Feedback Trends**

📌 **Purpose:** Use **Google Looker Studio** or **Grafana** to create reports on **QoE feedback trends**.

✔ **Steps to Visualize in Looker Studio:**

1. **Go to** [Google Looker Studio](https://lookerstudio.google.com/).
2. Click **Create Report** → Connect to **BigQuery Dataset (QoE\_feedback)**.
3. Select **Fields: user\_rating, issue\_reported, timestamp**.
4. Create **Charts & Dashboards** showing:
   * **Average QoE Rating Over Time**
   * **Most Reported Issues** (e.g., latency, video buffering)
   * **QoE Trends Before vs. After Adaptation Strategies**
5. Click **Save & Share Report**.

✅ **Now, you have a real-time dashboard displaying QoE feedback trends!**

**✅ Final Outcome**

🚀 **A fully deployed Firebase-based QoE feedback system with BigQuery analytics!**  
✔ **Surveys trigger automatically when QoE drops.**  
✔ **User responses are stored in BigQuery.**  
✔ **Trends are visualized in Google Looker Studio.**  
✔ **Helps measure QoE improvements over time.**