King Fahd University of Petroleum and Minerals

Computer Engineering Department

COE 453: Cloud and Edge Computing

Term 232

**Homework 3**

(Due on 19-Mar. @ 11:59 pm)

**Objectives:**

1. Getting familiar with the Google Cloud Software Development Kit (SDK)
2. Using the following GCP services: API Gateway, Cloud Functions, and Cloud Run
3. Getting familiar with POSTMAN and using it for API testing
4. Designing, architecting, and testing the calculator example on GCP
5. Practicing the methodology of REST API and GraphQL API design (from user requirements to API design)

**Submission:**

Insert your answers into the tables provided at the end of each task in this file. Then, submit this file.

**Task 1: Architecting a Web application on GCP using the API gateway and Cloud functions**

Develop a calculator Web application that supports the four arithmetic operations (+, -, \*, /) on integer numbers. Each arithmetic operation is performed on a dedicated Cloud function. Create the following four HTTP APIs on the API gateway:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | **HTTP API** | **JSON Object in HTTP Request** | **JSON Object in HTTP Response** |
| + | Post /add | {  “X”: 3,  “Y”: 4  }  **Note: Values of X and Y are input by the user. The values shown above are just an example.** | {  “X”: 3,  “Y”: 4,  “Result”: 7  } |
| - | Post /sub |
| / | Post /div |
| \* | Post /mul |

**Note: I talked with dr. Yahya and he said that change the GET request to POST request in both task01 and task02**

Your Web application architecture should contain the following five services: (1) Gateway, (2) Add, (3) Sub, (4) Mul, and (5) Div.

Use POSTMAN to test your Web application. The Github repository should contain all files used in developing the Web application (API gateway file, Cloud function files, …. etc)

**Submission:**

|  |  |
| --- | --- |
| API Gateway URL | https://task01-1l1wcchl.uc.gateway.dev |
| Github Repository Link | https://github.com/MohammedThan/COE453-Homework03 |

***Note:*** *Grader will use POSTMAN for testing your Web applicaiton. Your submitted API gateway URL will be used as a base URL (e.g., your\_url\_gateway\_api/add). The grader will use the JSON file given above in the HTTP request. Your service should return the JSON file given above in the HTTP response.*

**Task 2: Architecting a Web application on GCP using the API gateway and Cloud Run**

Your team has decided to re-deploy the calculator Web application on Cloud Run using containers. In this case, each arithmetic operation is performed on a container.

Use POSTMAN to test your new architecture. The Github repository should contain all files used in developing the Web application (Server files, Docker files, …. etc)

***Note:*** *You will have to create a new API gateway for this task.*

**Submission:**

|  |  |
| --- | --- |
| API Gateway URL | https://task02-1l1wcchl.uc.gateway.dev |
| Github Repository Link | https://github.com/MohammedThan/COE453-Homework03 |

***Note:*** *Grader will use POSTMAN for testing your Web applicaiton. Your submitted API gateway URL will be used as a base URL (e.g., your\_url\_gateway\_api/sub). The grader will use the JSON file given above in the HTTP request. Your service should return the JSON file given above in the HTTP response.*

***For this task, please have a look at Question 5 in the sample midterm exam on Blackboard.***

**Task 3: Designing REST APIs and then Translating them into GraphQL APIs**

You are going to develop a book library application for some client. You have identified the following user requirements:

1. The administrator shall be able to list all library users.
2. The administrator shall be able to list all books in the library.
3. The administrator shall be able to list all books borrowed by a specific user.
4. The administrator shall be able to list all library users with books that are overdue.
5. The administrator shall be able to send an email to a specific library user.

(a) Extract the resources from every requirement.

|  |  |
| --- | --- |
| **Requirement No.** | **Resources** |
| 1 | List of library users. |
| 2 | List of library books. |
| 3 | List of books borrowed by a user. |
| 4 | List of library users with overdue books. |
| 5 | Send email to a library user. |

All the answers below depend on the database structure, so I will assume that we are using SQL database with 3 tables:

|  |  |  |
| --- | --- | --- |
| User | Books | borrowBook |
| { "id": "", "name": ""," email":"" } | { "id": "", "title": "", "author": "" } | {"id": "","userId": "","bookId": " ", "borrowDate":""," returnDate":"" } |

(b) Design the REST APIs that satisfy all the requirements.

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement No.** | **Endpoint**  **(HTTP Verb + Route)** | **JSON File in HTTP Request**  **(Use Correct Format)** | **JSON File in HTTP Response**  **(Use Correct Format)** |
| 1 | GET /users | N/A | { "users": [ { "id": "", "name": "" } ] } |
| 2 | GET /books | N/A | { "books": [ { "id": "", "title": "", "author": "" } ] } |
| 3 | GET /users/:userId/borrowed-books | N/A (we will pass the userId in the URL) | {"data":{"userId","books":[{"id": "", "title": "", "author": "" } ] }} |
| 4 | GET /users/with-overdue-books | N/A (we will pass the userId in the URL) | { "users": [ { "id": "", "name": "","email":"" } ] } |
| 5 | POST /services/send-user-email/:userId | { "subject": "", "message": "" } | { "message": "Email sent successfully" } |

(c) You are required to transform the REST APIs designed in part (b) into GraphQL.

1. Define the schema

|  |  |
| --- | --- |
| **Object Type** | **Query / Mutation Type** |
| type User { id:ID!,  name: String,  email: String!  }  type Book {  id: ID!  title: String!  author: String!  }  type BorrowedBook {  id: ID!  user: User  book: Book  borrowDate: String!  returnDate: String!  } | type Query {  users: [User!]!  books: [Book!]!  userBorrowedBooks(userId: ID!): [Book] usersWithOverdueBooks: [User]  }  type Mutation {  sendUserEmail(userId: ID!, subject: String!, message: String!): String!  } |

1. Define the query / mutation that fulfills each requirement.

|  |  |
| --- | --- |
| **Requirement No.** | **Query / Mutation** |
| 1 | query{  users{  id  name  email  }  } |
| 2 | query{  books{  id  title  author  }  } |
| 3 | query{  userBorrowedBooks(userId:2){  id  title  author  }  } |
| 4 | query{  usersWithOverdueBooks {  id  name  email  }  } |
| 5 | mutation {  sendUserEmail(userId: "1", subject: "Hello", message: "This is a test email.")  } |