*King Fahd University of Petroleum and*

*Minerals*

*Computer Engineering Department*

*COE 558: Cloud and Edge Computing*

*Project Final Submission*

***Bookshelf Manager***

A black text on a white background

Description automatically generated

**Team Member :**

**Abdulaziz Alshehri (201776310)**

**MOHAMMED ALRABRABAH (201049840)**

**Computer Engineering Department**

**COE 558: Cloud and Edge Computing**

**Final Course Project**

**Term 231**

|  |  |
| --- | --- |
| **Project Title** | **Bookshelf Manager** |
| **Team Members**  **(Name + ID)** | **Abdulaziz Alshehri (201776310)**  **MOHAMMED ALRABRABAH (201049840)** |
| **Project Description** | **The Bookshelf Manager is a sophisticated web application designed to empower users in efficiently organizing and overseeing their personal book collections. Utilizing React for the frontend, GraphQL serves as the API gateway, and MongoDB handles data storage. This project incorporates a serverless function and a microservice to enhance functionality. The Single Page Application (React) allows users to seamlessly view, add, edit, and delete books through a clean and intuitive UI. The GraphQL API Gateway acts as the entry point for the mongoDB, managing queries and mutations related to book management. The services include serverless functions triggered by events, such as adding and deleting books. The microservice, responsible for managing books in the database, facilitates editing and viewing functionalities. MongoDB serves as the storage solution, storing comprehensive information about books, including titles, authors, number of pages and the book id .** |
| **My feedback on your course project proposal** | **No feedback in the proposal** |
| **Services** | |  |  |  | | --- | --- | --- | | **Service Name** | **URL** | **Description** | | BookManager Web Page (microservice) | <https://bookmanager-lwup2vslqa-uc.a.run.app/> | The BookManager Web Page microservice serves as a centralized hub for managing bookshelves, facilitating seamless Create, Update, Read, and Delete (CURD) operations. Interacting with various services and microservices, this web page empowers users to efficiently organize and manipulate their book collections. Its integration capabilities enable a dynamic and user-friendly experience, ensuring a comprehensive bookshelf management solution within a larger ecosystem of interconnected services. | | AddBook (serverless) | <https://bookfunctions-80xq521x.uc.gateway.dev/AddBook> | The AddBook serverless service streamlines the process of book addition by enabling users to submit a user-input JSON via a GraphQL mutation. Leveraging the efficiency of serverless architecture, this microservice seamlessly integrates with MongoDB, ensuring a swift and scalable solution for adding books to your collection. | | DeleteBook (serverless) | <https://bookfunctions-80xq521x.uc.gateway.dev/DeleteBook> | The DeleteBook serverless microservice simplifies the book deletion process by allowing users to submit a book ID via a GraphQL mutation, seamlessly interfacing with MongoDB. | | EditBook (microservice) | <https://editbook-lwup2vslqa-uc.a.run.app/update> | A micro-service that allow the user to mutate updating the book details by interacting with the mongoDB with QraphQL interface. | | ViewBook (microservice) | <https://viewbookservice-lwup2vslqa-uc.a.run.app> | A microservice that been called as the user launch the webpage to retrieve all the books from the MongoDB. | | MongoDB | <https://europe-west1.gcp.realm.mongodb.com/api/client/v2.0/app/bookshelf-enhmm/graphql> | A noSQL database service provided in the cloud to allow users to read, add , edit and delete books from/ to collections. | |
| **REST APIs** | Document all REST endpoints in your architecture (not only the REST endpoints for your services)   |  |  | | --- | --- | | **Endpoint (HTTP Verb + Path)** | **JSON File** | | POST / <https://bookfunctions-80xq521x.uc.gateway.dev/AddBook> | {  "BookId": "123456",  "Author": "John Doe",  "NumberOfPages": 300,  "BookName": "The Book Title"  } | | POST / <https://bookfunctions-80xq521x.uc.gateway.dev/DeleteBook> | {  "BookId": "123456"  } | | GET / <https://viewbookservice-lwup2vslqa-uc.a.run.app> | NA | | POST / <https://editbook-lwup2vslqa-uc.a.run.app/update> | {  "BookId": "123456",  "Authoer": "New Author Name",  "NumberOfPages": 250,  "BookName": "Updated Book Title"  } | |
| **GraphQL API** | |  | | --- | | **Object Type** | | type Book {  BookId: ID!  Authoer: String  NumberOfPages: Int  BookName: String  } | | **Query Type** | | type Query {  getAllBooks: [Book]  getBookById(query: BookQuery!): Book  } | | **Mutation Type** | | type Mutation {  insertOneBookShelfC(data: BookInput!): Book  updateOneBookShelfC(query: BookQuery!, set: BookInput!): Book  deleteOneBookShelfC(query: BookQuery!): Book  } | | **Queries** | | query {  bookShelfCS {  Authoer  BookId  BookName  NumberOfPages  }  } | | **Mutations** | | mutation {  updateOneBookShelfC(  query: { BookId: "${req.body.BookId}" }  set: {  BookId: "${req.body.BookId}",  Authoer: "${req.body.Authoer}",  NumberOfPages: ${req.body.NumberOfPages},  BookName: "${req.body.BookName}"  }  ) {  Authoer  BookId  BookName  NumberOfPages  }  }  mutation {  insertOneBookShelfC(  data: {  BookId: "${req.body.BookId}",  Authoer: "${req.body.Authoer}",  NumberOfPages: ${req.body.NumberOfPages},  BookName: "${req.body.BookName}"  }  ) {  Authoer  BookId  BookName  NumberOfPages  }  }  mutation {  deleteOneBookShelfC(  query: { BookId: "${req.body.BookId}" }  ) {  Authoer  BookId  BookName  NumberOfPages  }  } | |
| **Workload Characterization**  **(Bonus)** | |  |  |  | | --- | --- | --- | | **REST Endpoint** | **Average Response Time** | **Test sceenshot** | | POST / <https://bookfunctions-80xq521x.uc.gateway.dev/AddBook> | 15 S |  | | POST / <https://bookfunctions-80xq521x.uc.gateway.dev/DeleteBook> | 15 S |  | | GET / <https://viewbookservice-lwup2vslqa-uc.a.run.app> | 16 S |  | | POST / <https://editbook-lwup2vslqa-uc.a.run.app/update> | 39 S |  | | **GraphQL** | **Average Response Time** |  | | For each query (Query 1, Query 2, …) |  |  | | For each mutation (Mutation 1, Mutation 2, …) |  |  | |

**Challenge**

During the course of my project, dealing with Cross-Origin Resource Sharing (CORS) emerged as a significant challenge. CORS is a security feature implemented by web browsers to restrict web pages from making requests to a different domain than the one that served the web page. This restriction is in place to prevent potentially malicious activities, such as unauthorized data access or cross-site request forgery. While CORS enhances web security, it can pose obstacles when developing web applications that need to interact with APIs hosted on different domains. Configuring CORS involves defining server-side rules that specify which domains are permitted to access the resources of a particular web application or API. Navigating these configurations, ensuring seamless communication between different origins, and troubleshooting potential issues added a layer of complexity to the project. Overcoming CORS challenges required a thorough understanding of the security model and meticulous attention to configuration details to ensure the smooth functioning of the web application and its interactions with external APIs.

**Solution**

To address CORS issues while developing or testing a web application, you can use browser extensions that allow you to temporarily disable CORS restrictions. One such popular extension is "Allow CORS: Access-Control-Allow-Origin."

Here's how you can use this extension:

1. Download the Extension:

Visit the Chrome Web Store and search for "Allow CORS: Access-Control-Allow-Origin" or simply click on the following link to go directly to the extension page:

<https://chrome.google.com/webstore/detail/allow-cors-access-control/lhobafahddgcelffkeicbaginigeejlf>

2. Install the Extension:

Click on the "Add to Chrome" button to install the extension. Confirm the installation when prompted.

3. Enable the Extension:

Once installed, you'll see the extension icon added to your browser's toolbar. To enable it, click on the icon. The icon should change color, indicating that CORS is now temporarily disabled.

4. Verify CORS Bypass:

Open your web application or make API requests that were initially causing CORS issues. With the extension enabled, the requests should now go through without encountering CORS restrictions.