In the Name of God

Book Store

Supervisor:
Sugi Moto Co

Developer:
Marzieh Bagherinia

Summer 2024

Table of Contents

0) Part 0: Setup the Project

We do this part in 2 steps, the first is creating and settting up the Supabase project and the second is creating a NodeJs project for implementing the project's necessary REST APIs:

0.1) Supabase Project

For passing this part, we do the following steps:

a) We create a project in Supabase website

b) We save the generated project's **URL** and **Anon_Key** which are located in **Project Setting** tab in Supabase dashboard

c) We choose a preferred authentication provider, means **Email**, in **Authentication** tab in Supabase dashboard

d) We disable the **Confirm Email** option in above selected **Email** authentication provider for handling the sending email rate limit in Supabase free plan

0.2) NodeJs Project

For passing this part, we do the following steps:

a) We create a Github repository for storing the project's necessary Rest APIs in NodeJs framework

b) The mentioned APIs contain two aspects of project: authentication and the books CRUD

c) The authentication part contains signup and login with email/password APIs

d) The books part conatins index books API

e) All implemented APIs are located in a Postman collection which exists in the following path in the project's GitHub repository:

```
./documentations/book_store.postman_collection
```

1) Database Design

Related to the problem description, we should create **authors** and **books** tables and then, for project initialization, we should insert multiple sample records to each table; we do these tasks by running the SQL queries in SQL CLI in **SQL Editor** tab in Supabase dashboard:

1.1) Authors Table

a) Creating Table's Query:

```
CREATE TABLE authors (
    author_id serial PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    country VARCHAR(255) NOT NULL
);
```

b) Inserting Record's Query:

```
INSERT INTO authors (name, country) VALUES
('George Orwell', 'UK'),
('J.K. Rowling', 'UK'),
('Ernest Hemingway', 'USA');
```

1.2) Books Table

a) Creating Table's Query:

```
CREATE TABLE books (
    book_id serial PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    author_id INT REFERENCES authors(author_id),
    price DECIMAL(10, 2) NOT NULL,
    publish_date DATE NOT NULL
);
```

b) Inserting Record's Query:

```
INSERT INTO books (title, author_id, price, publish_date) VALUES
('1984', 1, 19.99, '1949-06-08'),
('Harry Potter and the Philosophers Stone', 2, 9.99, '1997-06-26'),
('The Old Man and the Sea', 3, 12.99, '1952-09-01');
```

2) RESTful Edge Function

Related to the problem description, we should implement an API for indexing book's records; we can implemnt this task using two methods, but note that we prefer to use the second method, because we have more capability for developing and expanding the project in an independent project rather than the edge functions:

2.1) Edge Functions:

For passing this part, we do the following steps:

a) Install the Deno and Supabase CLI tools

b) Run the **supabase init** command to creating the supabase project environment

c) Run the **supabase functions new books** command in created supabase project directory

d) The index.ts file added to supabase/functions/books path and we should add the API controller implementation to it; this implementation is located in the following path in the project's GitHub repository:

`./edge_functions/supabase/functions/books/index.ts`

e) Run the **supabase start** command

f) Run the **supabase functions serve** command

g) Execute the prepared API using the following CURL:

```
curl --request 'http://localhost:54321/functions/v1/books' \
  --header 'Authorization: Bearer SUPABASE_ANON_KEY' \
  --header 'Content-Type: application/json''
```

2.2) Rest API

For passing this part, we do the following steps:

a) Add the index books API implementation to NodeJS project which created at section 0; the controller logic is similar ro above edge function and its implementation is located in the following path in the project's GitHub repository:

`./routes/api/v1/books/index.js`

b) Execute the prepared API using the following CURL:

```
curl --location 'http://localhost:3000/api/v1/books/' \
  --header 'Authorization: Bearer SUPABASE_ANON_KEY' \
  --header 'Content-Type: application/json''
```

3) Advanced SQL Queries

Related to the problem description, we should implement the SQL query for three cases; the implementation and its explanations for each case is as follows:

3.1) Case 1

a) Query Implementation:

```sql
SELECT
  a.name,
  COUNT(b.book_id) AS book_count
FROM
  authors a
JOIN
  books b ON a.author_id = b.author_id
GROUP BY
  a.name
HAVING
  COUNT(b.book_id) > 5;
```

b) Query Explanation:

This query joins the authors and books tables on the author_id column. It groups the results by author name and uses the HAVING clause to filter out authors who have published 5 or fewer books.

3.2) Case 2

a) Query Implementation:

```sql
SELECT
  a.country,
  AVG(b.price) AS average_price
FROM
  authors a
JOIN
  books b ON a.author_id = b.author_id
GROUP BY
  a.country;
```

b) Query Explanation:

This query joins the authors and books tables on the author_id column. It groups the results by the authors' country and calculates the average book price for each country.

3.3) Case 3

a) Query Implementation:

```sql
SELECT
  b.title,
  a.name AS author_name,
  b.price,
  b.publish_date
FROM
  books b
JOIN
  authors a ON b.author_id = a.author_id
WHERE
  EXTRACT(YEAR FROM b.publish_date) = 2023
ORDER BY
  b.price DESC;
```

b) Query Explanation:

This query joins the books and authors tables on the author_id column. It filters the books by the specified publication year (2023 in this case) and sorts the results by price in descending order.

4) Additional documents

To run the project, we need to do the following steps:

a) Clone the project from Github and add the .env file to it. A sample .env file located in the following path in the project's GitHub repository:

```
./documentations/.env
```

b) Run the **npm install** command in project's directory for installing the project dependencies

c) Run the **npm run dev** or **node index.js** for running the project (the default port is 3000)

d) Test the APIs (both auth and books APIs) using Postman collection which located in the following path in the project's GitHub repository:

```
./documentations/book_store.postman_collection
```