growsari

Developer Assessment Exam

Instructions

This is a 3 part exam where your technical skills across the web development stack will be assessed. Each part builds upon the previous part so make sure you create them chronologically. Not all items are required. Optional items are marked but keep in mind that your proficiency will be determined by the points you have earned through this exam.

You are free to use any language and framework that you think is necessary to carry out the requirements.

Commit your work in your personal Github account. Make sure it is public so we can see it. When you are done, send in a link of the repository.

Good Luck!

Points Summary

		Required Items	**Optional Items**
Part 1: API			
	A. REST API	30 points	25 points
	B. GraphQL API - *Optional*		25 points
Part 2: UI			
	A. SPA	40 items	10 points
	B. PWA - *Optional*		25 points
Part 3: Docker			
	Docker - *Optional*		30 points

Total Possible Points: 185 points

growsari

Developer Assessment Exam

Part 1: API *(30 points, 50 points optional)*

A. REST API

Create a JSON REST API with the endpoints listed below. As a general guide, take note of the following constraints and considerations:

- Use proper HTTP headers and response codes
- All ID's are auto-generated and in UUID v1 format
- All timestamps are auto-generated and in ISO8601 format
- Token is in JWT format
- Encrypt password
- Include error checks
- Users can only update and delete topics that they have created
- Topics are sorted alphabetically
- Messages are sorted in reverse chronological order

1. **User Registration** *(5 points)*

   Endpoint: `POST /user/register`

   Request:
   ```
   {
     "email": "email@address.com",
     "name": "Name",
     "password": "password
   }
   ```

   Response:
   ```
   {
     "id": "d6e2769b-1f71-4ec0-b241-775654b858cb",
     "name": "Name",
     "email": "email@address.com",
     "created_at": "2019-07-11T01:54:00+0000",
     "updated_at": "2019-07-11T01:54:00+0000"
   }
   ```

2. **User Login** *(5 points)*

   Endpoint: `POST /user/login`

   Request:
   ```
   {
     "email": "email@address.com",
     "password": "password"
   }
   ```

   Response:

```
{
  "token": "eyJhbGcipXVCJ9.eyJzdWIiOiIxMjM0NTIiwiaWF0IfQ.SflKxwRJSMeKKF2QT4fwpMeJf36Pc"
}
```

3. **Create Topic** *(3 points)*

   Endpoint: `POST /topic`

   Request:

```
{
  "token": "eyJhbGcipXVCJ9.eyJzdWIiOiIxMjM0NTIiwiaWF0IfQ.SflKxwRJSMeKKF2QT4fwpMeJf36Pc",
  "subject": "Topic 1",
  "description": "This is the description for Topic 1"
}
```

   Response:

```
{
  "id": "a5ce5d94-2509-47ce-883e-8120d9170b6f",
  "subject": "Topic",
  "description": "This is the description for Topic 1",
  "created_by": "d6e2769b-1f71-4ec0-b241-775654b858cb",
  "updated_by": "d6e2769b-1f71-4ec0-b241-775654b858cb",
  "created_at": "2019-07-11T01:54:00+0000",
  "updated_at": "2019-07-11T01:54:00+0000"
}
```

4. **Update Topic** *(4 points)*

   Endpoint: `PATCH /topic/{id}`

   Request:

```
{
  "token": "eyJhbGcipXVCJ9.eyJzdWIiOiIxMjM0NTIiwiaWF0IfQ.SflKxwRJSMeKKF2QT4fwpMeJf36Pc",
  "subject": "Topic One",
  "description": "This is the description for Topic One"
}
```

   Response:

```
{
  "id": "a5ce5d94-2509-47ce-883e-8120d9170b6f",
  "subject": "Topic One",
  "description": "This is the description for Topic One",
  "created_by": "d6e2769b-1f71-4ec0-b241-775654b858cb",
  "updated_by": "d6e2769b-1f71-4ec0-b241-775654b858cb",
  "created_at": "2019-07-11T01:54:00+0000",
  "updated_at": "2019-07-11T02:33:00+0000"
}
```

5. **Delete Topic** *(3 points)*

   Endpoint: `DELETE /topic/{id}`

Request:

```
{
  "token": "eyJhbGcipXVCJ9.eyJzdWIiOiIxMjM0NTIiwiaWF0IfQ.SflKxwRJSMeKKF2QT4fwpMeJf36Pc"
}
```

Response:

```
{
  "success": true
}
```

6. **Create Message in a Topic** *(3 points)*

Endpoint: `POST /topic/{id}/message`

Request:

```
{
  "token": "eyJhbGcipXVCJ9.eyJzdWIiOiIxMjM0NTIiwiaWF0IfQ.SflKxwRJSMeKKF2QT4fwpMeJf36Pc",
  "message": "This is a message in topic 1"
}
```

Response:

```
{
  "id": "e0823261-20ee-4c71-a9be-81aa33005855",
  "topic_id": "a5ce5d94-2509-47ce-883e-8120d9170b6f",
  "message": "This is a message in topic 1",
  "created_by": "d6e2769b-1f71-4ec0-b241-775654b858cb",
  "updated_by": "d6e2769b-1f71-4ec0-b241-775654b858cb",
  "created_at": "2019-07-11T01:54:00+0000",
  "updated_at": "2019-07-11T01:54:00+0000"
}
```

7. **Retrieve all topics** *(3 points)*

Endpoint: `GET /topics`

Request:

```
{
  "token": "eyJhbGcipXVCJ9.eyJzdWIiOiIxMjM0NTIiwiaWF0IfQ.SflKxwRJSMeKKF2QT4fwpMeJf36Pc"
}
```

Response:

```
{
  "data": [
    {
      "id": "a5ce5d94-2509-47ce-883e-8120d9170b6f",
      "subject": "Topic One",
      "description": "This is the description for Topic One",
      "created_by": "d6e2769b-1f71-4ec0-b241-775654b858cb",
      "updated_by": "d6e2769b-1f71-4ec0-b241-775654b858cb",
      "created_at": "2019-07-11T01:54:00+0000",
```

```
      "updated_at": "2019-07-11T02:33:00+0000"
    }
  ]
}
```

8. **Retrieve all messages in a topic** *(4 points)*

   Endpoint: `GET/topic/{id}/messages`

   Request:

```
{
  "token": "eyJhbGcipXVCJ9.eyJzdWIiOiIxMjM0NTIiwiaWF0IfQ.SflKxwRJSMeKKF2QT4fwpMeJf36Pc",
}
```

   Response:

```
{
  "data": [
    {
      "id": "e0823261-20ee-4c71-a9be-81aa33005855",
      "topic_id": "a5ce5d94-2509-47ce-883e-8120d9170b6f",
      "message": "This is a message in topic 1",
      "created_by": "d6e2769b-1f71-4ec0-b241-775654b858cb",
      "updated_by": "d6e2769b-1f71-4ec0-b241-775654b858cb",
      "created_at": "2019-07-11T01:54:00+0000",
      "updated_at": "2019-07-11T01:54:00+0000"
    }
  ]
}
```

Bonus Optional Items:

- Add pagination to **`GET /topics`** and **`GET /topic/{id}/messages`**. *(5 points)*
- For delete topic, instead of physically deleting the record, add a field named "**`deleted_at`**" which serves a flag for soft-deleted records. These soft deleted records should not show up during retrieval. *(5 points)*
- Add a mechanism in such a way that only one token is active and valid for a single user. When a user logs in while there is an active token, the current token will be invalidated and the newly generated token will be the only valid token. *(5 points)*
- Create unit tests for each endpoint with at least 75% code coverage. *(10 points)*

growsari

Developer Assessment Exam

B. GraphQL API (Optional)

Add a GraphQL layer in front of your REST API so that the interface becomes GraphQL instead of REST. Use the following schema. **(25 points)**

```
type User {
  id: ID!
  name: String!
  email: String!
  created_at: String!
  updated_at: String!
}

type Topic {
  id: ID!
  subject: String
  created_by: User!
  updated_by: User!
  created_at: String!
  updated_at: String!
  messages: [Message]
}

type Message {
  id: ID!
  message: String!
  created_by: User!
  updated_by: User!
  created_at: String!
  updated_at: String!
}
```

```
type Mutation {

  user_register(
    email: String!
    name: String!
    password: String!
  ): User

  user_login(
    email: String!
    password: String!
  ): String

  topic_create(
     token: String!
     subject: String!
     description: String!
  ): Topic

  topic_update(
     token: String!
     id: ID!
     subject: String
     description: String
  ): Topic

  topic_delete(
     token: String!
     id: ID!
  ): Boolean

  message_create(
     token: String!
     topic_id: ID!
     message: String!
  ): Message

}

type Query {

   topics(
     token: String!
   ): [Topic]

   topic(
     token: String!
     id: ID!
  ): Topic
}
```

growsari

Developer Assessment Exam

Part 2: UI

A. SPA

Create a single page application (SPA) where the endpoints you created in Part 1 are utilized. The following views should be present:

1. User registration ***(5 points)***
2. User authentication (login and logout) ***(6 points)***
3. Create topic ***(5 points)***
4. Update topic ***(5 points)***
5. Delete topic ***(4 points)***
6. List all topics ***(5 points)***
7. Add a message to a topic ***(5 points)***
8. List all messages in a topic ***(5 points)***

Make sure of the following considerations:

- Error feedback
- Page responsiveness
- Intuitive UI flow

Bonus Item:
If you have enabled pagination for retrieval of topics and messages in your API, employ **either** of the following in your web app:

- Paginator ***(5 points)***
- Infinite scrolling ***(10 points)***

B. PWA (Optional)

Turn your application into a progressive web application in such a way that topics and messages can still be created even when there is no network connection. Use a service worker to do this. “Cached” topics and messages will be sent to the server once a network connection is available . ***(25 points)***

Part 3: Docker (Optional)

Dockerize your whole setup so that the whole thing could be run with a "**`docker-compose up`**" command. At the minimum, the setup should contain the following containers: ***(30 points)***

- Database
- API
- UI