Backend Engineer Assignment

Assignment overview

The goal of this assignment is to implement a backend solution for a simple blogging platform. It uses a custom REST API for all requests. Your assignment is to implement only a backend solution and provide an open API, which can be consumed by any client. You don't need to work on the frontend.

General functionalities

- Create, read, update and delete blog post
- List and filter blog posts by tag
- List of all tags in the system
- Create, read and delete comments from a blog post (no updating is required)

General notes

- **Some notes in this text are required and some are there to help you find your way faster.**
- The point of the exercise is to showcase to us your approach to **understanding assignment requirements**, coding, problem-solving and writing well designed, testable, efficient code by using best software development practices.
- Feel free to use Node.js (TypeScript), Python or .NET (C#) to build your API, it's your choice. Use the latest LTS versions.
- For the database, you can go with SQLite or any other database engine.
- Seed the database with some initial data **(required).**
- Please keep in mind that we need to run your project so make it easier for us and document all requirements and dependencies needed. Optionally use docker containers to run the project API and database.
- Below you will find a detailed specification for your API: JSON Objects returned by API and Endpoints.
- Integrate Swagger or similar API documentation tool into your project, this will help you to present and test your APIs in a simple and comprehensive way (**required**).
- Don't forget to use the final product yourself to test it and be sure it's working as it should. To test your API you can use tools like Postman or other.
- Include at least one unit test in your project to demonstrate how testing works (full testing coverage is not required!)

Delivery notes

- We need to run and test the API, write detailed instructions in the README.md.
- We only accept links to public Git repositories which we can check out. Please do not send archived files.
- The deadline to submit your app is 7 days after accepting the assignment.
- If you think there is something more detailed you need to tell us please put it all into the README.md and we will check it out there.

API Specification

JSON Objects returned by API:

- Make sure the right content type is correctly returned, like `Content-Type: application/json; charset=utf-8`

Single Blog Post

```
{
  "blogPost": {
    "slug": "augmented-reality-ios-application",
    "title": "Augmented Reality iOS Application",
    "description": "Rubicon Software Development and Gazzda furniture are proud to launch an augmented reality app.",
    "body": "The app is simple to use, and will help you decide on your best furniture fit.",
    "tagList": ["iOS", "AR"],
    "createdAt": "2018-05-18T03:22:56.637Z",
    "updatedAt": "2018-05-18T03:48:35.824Z"
   }
}
```

Multiple Blog Posts

```
{
  "blogPosts": [{
    "slug": "augmented-reality-ios-application",
    "title": "Augmented Reality iOS Application",
    "description": "Rubicon Software Development and Gazzda furniture are proud to launch an augmented reality app.",
    "body": "The app is simple to use, and will help you decide on your best furniture fit.",
```

```
    "tagList": ["iOS", "AR"],
    "createdAt": "2018-05-18T03:22:56.637Z",
    "updatedAt": "2018-05-18T03:48:35.824Z"
  }, {
    "slug": "augmented-reality-ios-application-2",
    "title": "Augmented Reality iOS Application 2",
    "description": "Rubicon Software Development and Gazzda furniture are proud to launch an augmented reality app.",
    "body": "The app is simple to use, and will help you decide on your best furniture fit.",
    "tagList": ["iOS", "AR", "Gazzda"],
    "createdAt": "2018-04-18T03:22:56.637Z",
    "updatedAt": "2018-04-18T03:48:35.824Z"
  }],
  "postsCount": 2
}
```


List of tags


```
{
  "tags": [
    "iOS",
    "Android"
  ]
}
```


Single Comment


```
{
  "comment": {
    "id": 1,
    "createdAt": "2018-04-18T03:22:56.637Z",
    "updatedAt": "2018-04-18T03:22:56.637Z",
    "body": "Great blog."
  }
}
```


Multiple Comments

```
{
  "comments": [{
    "id": 1,
    "createdAt": "2018-04-18T03:22:56.637Z",
    "updatedAt": "2018-04-18T03:22:56.637Z",
    "body": "Great blog."
  }]
}
```

Endpoints:

- Please note that API uses **slug** instead of IDs**.** Slug is a **“**URL friendly” version, used instead of IDs to identify blog post. Slugs should be entirely lowercase, with accented characters replaced by letters from the English alphabet and whitespace characters replaced by a dash or an underscore to avoid being encoded. Punctuation marks are generally removed.

Get Blog Post

```
GET /api/posts/:slug
```

Will return a single blog post.

List Blog Posts

```
GET /api/posts
```

Returns most recent blog posts by default, optionally provide `tag` query parameter to filter results. Query Parameters:

Filter by tag:

```
?tag=AngularJS
```

Will return multiple blog posts, ordered by most recent first.

Create Blog Post

```
POST /api/posts
```

Example request body:

```
{
  "blogPost": {
    "title": "Internet Trends 2018",
    "description": "Ever wonder how?",
    "body": "An opinionated commentary, of the most important presentation
of the year",
    "tagList": ["trends", "innovation", "2018"]
  }
}
```

Will return a blog post.
Required fields: title, description, body.
Optional fields: tagList as an array of strings.

Update Blog Post

```
PUT /api/posts/:slug
```

Example request body:

```
{
  "blogPost": {
    "title": "React Why and How?"
  }
}
```

Returns the updated blog post.
Optional fields: title, description, body
The **slug** also gets updated when the title is changed.

Delete Blog Post

```
DELETE /api/posts/:slug
```

Get Tags

```
GET /api/tags
```

Returns a list of all tags in the database.

Add Comments to a blog post

```
POST /api/posts/:slug/comments
```

Example request body:

```
{
  "comment": {
    "body": "Nice explanation."
  }
}
```

Returns the created comment

Required field: `body`

Get Comments from a blog post

```
GET /api/posts/:slug/comments
```

Returns multiple comments for a given blog post.

Delete Comment

```
DELETE /api/posts/:slug/comments/:id
```