JUNIOR DEVELOPER TECHNICAL TEST

Overview

The primary aim of this test is to assess your skills in Node.js, your ability to communicate your thoughts and decisions clearly, and your proficiency in managing time effectively. The test is divided into three sections: Coding Tasks, Written Questions, and Time Management Task.

Part 1: Coding Tasks (60 Points)

Task 1.1: Create a Basic REST API (20 Points)

Create a simple REST API using Express.js that performs CRUD operations for managing a list of books. Each book will have the following attributes:

- id: A unique identifier for the book
- title: The title of the book
- author: The author of the book
- publishedDate: The date when the book was published

Commit your code to a GitHub repository and share the link.

Task 1.2: Implement Authentication (15 Points)

Add JWT authentication to the above API. Only authenticated users should be able to perform CRUD operations.

Task 1.3: Connect to Database (15 Points)

Use MongoDB to store the book data. Connect your Node.js application to MongoDB and perform the CRUD operations on the database.

Task 1.4: Error Handling (10 Points)

Implement basic error handling in your application. Return appropriate status codes and messages when an error occurs.

Part 2: Written Questions (20 Points)

Question 2.1: Explain your decisions (10 Points)

Write a brief markdown document explaining the following:

- Why did you structure your code the way you did?
- What would you do differently in a production environment?

Question 2.2: Code Review (10 Points)

Here is a small snippet of Node.js code:

```
app.post('/user', function (req, res) {
  let user = req.body;
  if (user.age < 21) {
    res.status(400).send("Too young");
  } else {
    db.addUser(user);
    res.status(200).send("User added");
  }
});
```

Identify any issues or improvements that could be made to this code and explain them.

Part 3: Time Management Task (20 Points)

Task 3.1: Prioritization and Planning (20 Points)

You have been given the following tasks to complete:

1. Fix a critical bug in the login module.
2. Develop a new feature that has been highly requested by clients.
3. Document the API you developed in Task 1.
4. Optimize the database queries in an existing module.

Write a plan detailing how you would prioritize and tackle these tasks within a week. Include a rough timeline and reasoning for your decisions.

Submission Instructions

1. Complete all tasks and questions within 3 hours.
2. Push your code for Part 1 to a GitHub repository.
3. Provide your written answers and plan for Part 2 and Part 3 in a markdown file within the same repository.