Developer Assessment

Title:
Full Stack Development Assessment (C# .NET & Angular)

Purpose

This assessment is designed to evaluate:

- Your ability to **learn new technologies quickly** (C#/.NET and Angular).
- Your **problem-solving skills** in both backend and frontend development.
- Your capability to **debug broken code**.
- Your **approach to writing basic tests**.
- Your **understanding of general development principles**.

Prior experience with C# or Angular is **not required** — you are encouraged to research and learn as you go.

Scenario

You have joined a small development team tasked with creating a **Task Manager** web application. The application will allow users to:

1. Add tasks with a **title** and **priority**.
2. View a list of existing tasks.
3. Mark a task as completed.
4. Filter tasks by priority or completion status.

The system will be built using:

- **Backend:** C# .NET Web API
- **Frontend:** Angular

Your tasks will include **building**, **debugging**, and **testing** this application.

Part 1 – Environment Setup

You must install the following tools before beginning:

1. **.NET SDK** – Download here
2. **Node.js** – Download here
3. **Angular CLI** – Install via terminal:

   ```
   npm install -g @angular/cli
   ```

4. **Git** – Download here

   Verify installation by running in your terminal:

```
dotnet --version
node -v
ng version
git --version
```

Part 2 – Application Development

Backend Requirements (C# .NET API)

- **Endpoints to Implement:**
  - `GET /tasks` – Retrieve all tasks.
  - `POST /tasks` – Create a new task. Must reject empty titles.
  - `PUT /tasks/{id}` – Mark a task as completed.
- **Data Storage:**
  - Use **in-memory storage** (a list in code).
- **Validation:**
  - Task title cannot be empty.
  - Priority must be one of: High, Medium, Low.

Frontend Requirements (Angular)

- **Pages/Components:**
  - **Task List Page** – Displays tasks, filterable by priority and completion.
  - **Task Form** – Allows creating a new task.
- **Functionality:**
  - Connect to backend API using Angular's `HttpClient`.
  - Apply filters on the frontend without reloading the page.

Part 3 – Debugging Challenge

You will be given:

- `buggy_backend.cs`
- `buggy_frontend.ts`

These files contain **intentional bugs** (syntax errors, incorrect logic, API mismatches). Your job is to:

1. Fix them so the application works.
2. Document your changes in `DEBUG_NOTES.txt` with:
   - **Problem Identified**
   - **How You Fixed It**

Part 4 – Testing

- **Backend:** Create a simple unit test that verifies adding a task works correctly.
- **Frontend:** Create a basic test that confirms tasks are displayed after retrieval.

Part 5 – General Development Questions

Answer these in `answers.txt`:

1. Explain the difference between an API integration error and a logic error.
2. What steps would you take to troubleshoot a non-responsive API call?
3. Why is it important to separate frontend and backend responsibilities?
4. Name a testing framework or tool you have used or researched and describe its purpose.
5. How do you ensure your code remains maintainable over time?

Submission Requirements

Submit via GitHub:

- **Source Code:** Backend and frontend in separate folders.
- **README.md:** Instructions to run the project locally.
- **DEBUG_NOTES.txt**
- **answers.txt**
- **Test Files**

Presentation Requirement:

After submitting your work, you will be expected to present your approach, solutions, and findings in a short session. Be prepared to explain your code, debugging fixes, and reasoning clearly.

Originality Notice:

You may use AI tools, but your submission must be your own work. Turnitin will check for plagiarism and AI-generated content. AI content should not exceed** 40%**.

Submissions above this may be reviewed or disqualified.