Senior Backend Engineer – Technical Assessment

Overview:

You are being tasked with building a small, functional backend application that interacts with a relational database (e.g., PostgreSQL) or document store (e.g., MongoDB), exposes an API, and integrates with an LLM agent using an MCP-compatible design to accomplish a real-world scenario.

You may choose **NodeJS (Typescript preferred)** or **Python**. You may use whatever server based packages you see fit (ex. Express, flask, nestJS, etc)

Time: This assessment should take you 3 – 5 hours

Deliverables:

- A GitHub repository with:
- API source code
- README with setup and explanation
- Integration with an MCP agentic workflow
- An architecture diagram (drawn via draw.io, or hand drawn image accepted)

Scenario:

You are building the backend for a Project Tracker used by AI agents to log tasks, fetch updates, and analyze progress based on a user query to a LLM Chatbot. This system will enable a project manager to quickly retrieve in flight project statuses, filter for specific project details/tasks and make updates/manage ongoing work items. This project is broken into 2 parts, first, you must create the backend service API that interacts with the project database. Second, this API must be built into an MCP tool that would be able to be connected to an LLM.

Part 1: Project Tracker API with Query Features

Functional Requirements:

Build a RESTful API to complete the following project tasks:

1. Create a new project with the fields: name, description, start_date, end_date, and status
2. Add a task to a project with: title, assigned_to, status, due_date, project_id
3. Retrieve a project and its associated tasks
4. Query/filter projects by fields

Technical Requirements:

- Use an in-memory or real DB (eg. PostgreSQL, MongoDB, etc)
- Add schema validation and error handling

Bonus:

- Add caching capabilities for frequently queried data
- Add Swagger or other API documentation
- Add unit tests & well structured & documented code (in line comments)

Part 2: Agentic Workflow with Model Context Protocol

Extend the API you've built to support an agent-based interaction, where an AI agent can interact with your API to fetch project data and execute actions based on context.

Requirements:

- Build a basic MCP-compatible tool or function that can interact with an agentic workflow to complete the following:
    - Accepts a prompt or plan (e.g., "Show me all overdue tasks assigned to Bob")
    - Calls your Project Tracker API internally
    - Returns structured results usable by an LLM (e.g., JSON)

- Describe your internal prompt engineering technique and design
- Include at least one example of a user prompt, and simulate the full flow:
- Prompt → MCP → API → Structured Result → Agent response

Bonus:

- Add details on how you would safeguard your LLM inputs and add guardrails to keep the model on track

Architecture Diagram

Draw and submit a rough architecture diagram that includes:
- API layer
- Database (SQL or NoSQL)
- Agentic integration with MCP
- Optional: any pub/sub, background jobs, or observability tools you'd consider for production

Use tools like Lucidchart, draw.io, or submit a hand drawn image
Submission Instructions

- Email your GitHub repo link to Christopher.ortiz@rbc.com & erick.chin@rbc.com
- Ensure your README explains how to run the app and test endpoints
- Include screenshots or link to diagrams