Bdeo

Full Stack Developer - Technical Case

1. Purpose

The objective is to evaluate your ability to build scalable applications, the use of modern design patterns, and your efficiency in an AI-first environment. Special emphasis will be placed on the use of SDD (Spec-Driven Development) and the implementation of reactive interfaces.

2. The Challenge: Claims Management System

You will develop a tool for claim managers that allows managing claims with multiple associated damages

2.1 Data Model

- Claim (Main Entity): Title, description, status (**Pending, In Review, Finished**), and Total Amount (calculated).
- Damage (Associated Entity): Part, severity (low, mid, high), image URL, and Price (numeric).

2.2 Front Views

- Claim List
- Claim Detail

  Requirement: In the claim detail view, a table must be displayed showing the associated damages and a total price representing the sum of the damage amounts. The option to add damages directly within this view must be provided.

2.3 Business Rules and Logic

- Form Validation: When creating a damage, all fields (description, severity, image, and price) are mandatory.
- Frontend Reactivity: In the claim detail view, the Total Amount must automatically update in real-time when adding, deleting, or modifying the price of any damage.
- Status Restrictions:
  - Damages can only be managed (added/deleted/edited) if the claim is Pending.
  - A claim with at least one damage with high severity requires a general description of more than 100 characters to be able to switch to Finished status.

3. Mandatory Technical Requirements

3.1 Architecture (Backend & Frontend)

- Dependency Injection: Mandatory use of DI in both Angular and the Backend to ensure decoupling and testability.
- Design Patterns: Mandatory use of at least 1 design pattern applied to business logic.
- Frontend (Angular): Use of reactive forms and efficient state management for total calculation.
- Backend (Node.js/TypeScript): RESTful API with type validation and persistence in MongoDB.

3.2 Quality and Testing

- Coverage >95%: It is essential to achieve coverage greater than 95% in Backend unit tests.
- Integration Tests: Validate that the total amount calculation on the server matches the sum of its damages.

4. SDD and AI Methodology

- SDD (Preferred): It is recommended to define the OpenAPI contract before starting development to align the AI with the data model.
- Use of AI: You can use Copilot, Cursor, or similar tools. You must include an AI_LOG.md file detailing how you used these tools to generate the high-coverage tests and how you supervised the frontend reactive logic.

5. Evaluation Criteria

- Frontend Skills: Implementation of reactive logic (totals) and form validation.
- Backend Skills: Correct application of Dependency Injection and data consistency.
- Testing Quality: Rigor in tests to achieve 95% coverage.
- Critical Thinking (AI): Ability to correct AI-generated code in complex business rules.

6. Submission

- Deadline: 3-4 days.
- Format: Git repository (monorepo) with README.md detailing the configuration and the command for the coverage report.