FINANCIO WEB DEV TECHNICAL ASSESSMENT v3.0

Candidate Name	
Date	

Overview

This assessment evaluates your ability to develop a full-stack invoice management feature using Angular for the front-end and Laravel for the back-end. It also tests your understanding of API development, integration, best coding practices, automated testing, and the ability to properly document and share necessary setup files.

Submission Instructions:

1. Submit both the Front-End and Back-End assessments separately.
2. Ensure source code is well-documented and follows best coding practices.
3. Make the repository public for evaluation.
4. Use a cloud-based version control system like GitHub or Bitbucket.
5. Include the .env file, database migration/seed scripts, and PHPUnit test execution scripts.

Front End Assessment (Angular):

1. Setup Angular Project
    a. Initialize a new Angular project.
    b. Ensure proper folder structure and modular organization.
2. List Invoices
    a. Create a component or page to display all invoices.
    b. Fetch data from the backend API and display it in a structured format with pagination.
3. Create Invoice
    a. Implement a form to create a new invoice.
    b. Ensure form validation (e.g., required fields, numeric validation for amounts, date pickers for invoice dates).
    c. Support adding multiple line items within the invoice.
4. Show Invoice Details
    a. Implement a detailed view for a single invoice.
    b. Display all invoice details, including line items and total calculations.
5. Edit Invoice
    a. Create a mechanism to modify and update invoices.
    b. Implement validation and ensure changes are correctly reflected in the system.
6. Delete Invoice
    a. Implement functionality to delete an invoice.
    b. Add a confirmation prompt before deletion.
7. Source Code Submission
    a. Submit the Angular project source code to a public repository (e.g., GitHub, Bitbucket).

Back End Assessment (Laravel with API):

1. Setup Laravel Project

    a. Initialize a new Laravel project.
    b. Ensure proper project structure and best practices.
    c. Share the .env file with appropriate configurations.
2. Setup Database
    a. Create tables for invoices and invoice_items with necessary relationships.
    b. Use appropriate data types (e.g., VARCHAR for customer name, DECIMAL for amounts, DATE for invoice date).
    c. Provide database migration and seed scripts to populate sample data.
    d. Example Fields for Invoice:
        i. number (VARCHAR, must be unique per customer per year)
        ii. date (DATE)
        iii. reference (VARCHAR)
        iv. customer_name (VARCHAR)
    e. Example Fields for Invoice Lines:
        i. id (INT, Primary Key)
        ii. product_name (VARCHAR)
        iii. unit_price (DECIMAL)
        iv. quantity (INT)
        v. total_amount (DECIMAL)
3. List Invoices via API
    a. Implement an API endpoint to retrieve and list all invoices.
    b. Support pagination and sorting.
4. Create Invoice via API
    a. Implement an API endpoint to create a new invoice with multiple line items.
    b. Validate input data before insertion.
    c. Ensure invoice_number is unique per customer per year.
    d. Calculate invoice totals and store them appropriately.
5. Show Invoice Details via API
    a. Implement an API endpoint to retrieve a specific invoice by ID.
    b. Include associated line items in the response.
6. Edit Invoice via API
    a. Implement an API endpoint to update an existing invoice and its line items.
    b. Ensure validation and update confirmation.
7. Delete Invoice via API
    a. Implement an API endpoint to delete an invoice.
    b. Add soft delete functionality to prevent accidental data loss (optional)
8. Unit Testing (PHPUnit)
    a. Write PHPUnit test cases for each API endpoint.
    b. Ensure test coverage includes:
    c. Successful invoice creation, retrieval, updating, and deletion.
    d. Validation failures and expected error responses.
    e. Provide scripts or instructions to execute PHPUnit tests.
    f. Run tests and ensure all pass before submission.
9. Source Code Submission
    a. Submit the Laravel project source code to a public repository (e.g., GitHub, Bitbucket).

Front End Assessment (Angular):

Criteria	Description	Evaluation

Angular Project Setup	Proper setup and organization of the project.	/5
List Invoices	Functional and visually structured invoice list.	/10
Create Invoice	Working form with validation and submission.	/10
Show Invoice Details	Functional detailed view of an invoice.	/10
Edit Invoice	Update functionality with validation.	/10
Delete Invoice	Invoice deletion with confirmation.	/10
Code Quality	Well-documented, clean, and maintainable code.	/5
Submission Format	Proper submission as repository.	/5
Total Marks (Front End)	**Front-End Score**	**/65**

Back End Assessment (Laravel with API):

Criteria	Description	Evaluation
Laravel Project Setup	Proper setup and organization of the project.	/5
Database Setup	Schema design, migrations, and seed data.	/10
List Invoices via API	Functional API endpoint for listing invoices.	/10
Create Invoice via API	Functional API endpoint for inserting invoices.	/10
Show Invoice via API	Functional API endpoint for viewing an invoice.	/10
Edit Invoice via API	Functional API endpoint for updating invoices.	/10
Delete Invoice via API	Functional API endpoint for deleting invoices.	/10
Unit Testing (PHPUnit)	Valid test cases and execution scripts.	/10
Code Quality	Clean, maintainable, and well-documented code.	/5
Submission Format	Proper submission as repository.	/5
Total Marks (Back End)	**Back-End Score**	**/85**

Overall Assessment:

Combined Total Marks (Front End + Back End):	**/150**