Full Stack Technical Assessment: Job Offer Management Module with DocuSign Integration

Task Description

Generate a lightweight enterprise-grade web application that streamlines the job offer and contract signing process. The system should enable HR or business teams to easily prepare a contract, preview it, and send it electronically for signature, with providing real-time visibility into signing status.

The application should allow users to:

- Fill out a form to generate a job offer or contract PDF
- Preview the generated PDF before sending
- Send the PDF for signature using a simulated DocuSign API
- View and track the signing status in real time

Features to Implement

1. Frontend

Prototype URL:

https://www.figma.com/design/YiEqxcFaEUwNj2rPENarO6/Untitled?node-id=1-2407&m=dev

- A form to collect job offer details, including:
    - Recipient name
    - Recipient email
    - Job offer letter content (rich text input that defines the body of the offer letter)
- A button to **"Generate Offer Letter"**
- A button to preview the generated PDF (using PDF.js or similar)
- After generating the PDF, a **"Send for Signature"** button should appear
- **Validations:** All fields must be required, and email must be in valid format
- **Error messages** should be clear and consistent

UI Hint: Implement the design based on the provided Figma link

2. Backend

- API to receive form data and **generate a job offer PDF**
- Endpoint to return the generated PDF for preview
- Simulated **DocuSign API integration**:
    - Create an envelope
    - Add recipient and generated PDF
    - Send for signing using the recipient's email
- Endpoint to check signing status
- Save metadata (recipient info, status, etc.) to a **SQL Server** database

3. Bonus (Optional but Appreciated)

- Webhook to receive updates from the simulated DocuSign API and update status in real time.
- Admin interface or logs to view signature history.

Key Requirements

1. **Modularity & Reusability**
    - Design the component to be easily reusable across different parts of a large enterprise application.
    - Utilize Angular's modularity features effectively (e.g., separate modules, services, shared components).
2. **Programming Fundamentals & Design Patterns**
    - **Clean Code:** Write well-structured, readable, and maintainable code, adhering to Angular best practices and coding standards.
    - **Error Handling:** Implement graceful error handling for validation, dynamic loading issues, and other potential runtime problems.
    - **State Management:** Clearly define how state (e.g., form data, signing status) is managed and updated.
3. **Output / Data Handling:**
    - Upon successful completion of the upload and "Send for Signature" action, the component should emit or return the consolidated data, including:
        - Recipient name and email

    - File name or identifier
    - Simulated DocuSign envelope ID
    - Submission timestamp
    - Signature status (e.g., "sent", "completed")
- Ensure emitted data is clean, structured, and can be consumed easily by a parent component or stored in state.
- Implement consistent output handling for success and failure scenarios.

Core Technologies

- Backend: .NET Core (C#)
- Frontend: Angular 12+
- Styling: Tailwind CSS (preferred), or bootstrap
- Integration: Simulated DocuSign API structure

Deliverables

- Link to a GitHub repo with the full source code.
- A README.md with:
    - Instructions to set up and run the app locally
    - Any environment variables needed (e.g., DocuSign simulation)
    - Screenshots or a short Loom video demo (optional but welcome)

Timeframe

- 3 to 5 days

Evaluation Criteria

- Code quality and structure
- Working simulated DocuSign integration
- Angular modularity and clean coding
- Validation and error handling
- Clear documentation and setup instructions
- Accurate implementation of UI based on provided Figma design (pixel-perfect where applicable)