Name:Farhana Akter

ID : 2212084642

Group member: 01

1.Technologies Used

Frontend:

- **React.js:** Used to build a dynamic user interface for the web application.
- **Axios:** Handles HTTP requests to communicate with the backend API.
- **Tailwind CSS & DaisyUI:** Used for styling and responsive design.

Backend:

- **Express.js:** Provides a RESTful API to handle file uploads, data extraction, and form filling.
- **Multer:** Handles file uploads efficiently.
- **Tesseract.js:** Performs OCR (Optical Character Recognition) to extract text from images and scanned documents.
- **PDFKit:** Generates downloadable PDF files with auto-filled form data.
- **pdf-parse:** Extracts text from uploaded PDF documents.
- **Sharp:** Processes and extracts images from uploaded documents.

3. System Design & Architecture

Workflow:

1. **User Uploads Documents:** The user uploads a document (PDF, image, or scanned file).
2. **Backend Processing:**
   - If the document is a PDF, text extraction is done using `pdf-parse`.
   - If the document is an image, `Tesseract.js` performs OCR to extract text.
   - If the document contains an image (like a photo in an NID), `Sharp` extracts and processes it.
3. **Form Auto-Filling:** The extracted text is parsed and mapped to predefined form fields.
4. **Downloadable File Generation:** The completed form is converted into a downloadable PDF using `PDFKit`.

4. Algorithms Implemented

File Upload Handling:

- `Multer` processes file uploads and stores them temporarily for processing.

OCR & Text Extraction:

- `pdf-parse` extracts raw text from PDFs.
- `Tesseract.js` performs OCR on images and scanned text.
- Regex-based parsing identifies key information (Name, DOB, ID Number, etc.).

Auto-Filling Forms:

- Extracted data is mapped to specific form fields using a JSON schema.

PDF Generation:

- The filled form is converted into a structured PDF document with `PDFKit`.

5. Testing Strategy

Unit Testing:

- Test individual functions such as file upload, text extraction, and form population.
- Ensure OCR and PDF parsing return accurate results.

Integration Testing:

- Validate the end-to-end workflow: Upload → Extract Data → Auto-Fill → Download.
- Ensure API endpoints respond correctly to various document types.

User Testing:

- Upload different document formats (PDF, JPG, PNG) and verify extracted data accuracy.
- Test usability and response time of the application.