ORM Project Documentation

Project Overview

Front-end (Angular)

- **User Invoice Upload:**
  - Enable users to effortlessly upload invoices.
- **Manipulation of the Extracted Information:**
  - Provide CRUD (Create, Read, Update, Delete) operations for the extracted information.

Back-end (Node.js with Express and JSON)

- **Data Extraction:**
  - Utilize the CaaS API for efficient data extraction from both images and PDFs.
- **Data Storage in JSON Format:**
  - Store the extracted data along with payment status in a structured JSON format.
- **CRUD Operations for Invoices:**
  - Implement functionalities for creating, reading, updating, and deleting invoices.

MVP (Minimum Viable Product)

Objectives

The MVP aims to deliver a functional core of the ORM project with essential features that allow users to upload, extract, view, and manipulate invoice data extracted.

Features

1. **User Interface (UI) for Invoice Upload:**
    - Simple and intuitive UI for users to upload invoice images and PDFs.
2. **Data Extraction:**
    - Integration with CaaS API for extracting data from uploaded invoices.
3. **Data Storage:**
    - Store extracted data in JSON format.
4. **CRUD Operations:**
    - Basic CRUD operations to manage invoice data:
        - **Create:** Allow users to add new invoice data extraction .
        - **Read:** Retrieve and display all stored extracted data from invoices.
        - **Update:** Modify existing invoice data fields.
        - **Delete:** Remove extracted data from the storage json.

Workflows

1. **Invoice Upload Workflow:**
    - User uploads an invoice file.
    - The file is sent to the backend for data extraction using the CaaS API.
    - Extracted data is stored in JSON format .
2. **Data Retrieval Workflow:**
    - User requests to view all invoices data extracted .
    - Backend retrieves and sends a list of all stored datas.
3. **Data Update Workflow:**
    - User selects an invoice and updates specific fields.
    - Backend processes the update request and modifies the data accordingly.
4. **Data Deletion Workflow:**

- User selects an invoice data to delete.
- Backend processes the deletion request and removes the invoice data from storage.

Technical Requirements

- **Front-end:**
  - Angular framework for building a responsive UI.
  - Forms and validation for invoice uploads and data manipulation.
- **Back-end:**
  - Node.js with Express for server-side logic and API endpoints.
  - Integration with CaaS API for data extraction.
  - JSON format for data storage.

Representative: Kanmegne and Mihail

Overview

Kanmegne and Mihail are responsible for overseeing the development and implementation of the ORM project. They ensure that the project meets its goals and timelines, and they coordinate between the front-end and back-end teams to integrate all functionalities seamlessly.

Responsibilities

- **Mihail:**
  - Supervises the front-end development.
  - Ensures the user interface is user-friendly and functional.
  - Coordinates with the back-end team to ensure smooth data handling.
- **Kanmegne andre:**
  - Leads the back-end development.
  - Oversees the implementation of the CaaS API and CRUD operations.

- Ensures the data storage and retrieval processes are efficient and secure.

This documentation outlines the project's architecture, API endpoints, MVP features, and key representatives, providing a comprehensive guide for developers and stakeholders involved in the ORM project.