The Loan Eligibility Engine (SDE Intern Backend)

Mission: Your challenge is to design, build, and deploy an automated system that ingests user data, discovers personal loan products from public websites, matches users to eligible products, and notifies them. This project tests your ability to integrate cloud services with a powerful, self-hosted workflow automation engine. This assignment is objective-based; while we provide a framework, we highly value creative solutions and architectural decisions that improve performance, reduce cost, and demonstrate a deeper understanding of the problem space.

Core Technologies:

- **Backend Language:** Python or Go
- **Cloud Platform:** AWS (Lambda, RDS for PostgreSQL, SES for email). You are encouraged to use the **AWS Free Tier**. API Gateway is optional.
- **AI & Workflow:**
    - A self-hosted **n8n** instance for workflow automation.
    - A free-tier API from **Google (Gemini)** or **OpenAI (GPT)** for specific AI tasks.
- **Deployment:** Serverless Framework (for AWS resources) & Docker Compose (for n8n)

Functional Requirements

1. Scalable User Data Ingestion

- **Bulk CSV Upload:** Design a robust and scalable mechanism for uploading a CSV file containing user data (e.g., `user_id`, `email`, `monthly_income`, `credit_score`, `employment_status,age`). **Note:** A direct upload through a standard Lambda-backed API Gateway endpoint can hit request size and timeout limits. We are looking for a more professional, event-driven pattern.
- **Data Persistence:** Once uploaded, the data must be parsed and stored in a dedicated table within an Amazon RDS for PostgreSQL database.

2. The n8n Automation Engine

You must set up and configure a self-hosted n8n instance using Docker. All core business logic will be orchestrated through n8n workflows. You will need to design and build the following interconnected workflows:

- **Workflow A: Loan Product Discovery (The Web Crawler)**
    - **Goal:** To autonomously find and store personal loan product information.
    - **Trigger:** This workflow should run on a schedule (e.g., once daily).
    - **Process:**
        1. Crawl a list of public financial websites (you can choose 2-3 relevant sites).

2. Extract key details for personal loan products, such as `product_name`, `interest_rate`, and the **eligibility criteria** (e.g., minimum monthly income, required credit score range).
3. Store this structured data in a new `loan_products` table in your PostgreSQL database. The workflow must be robust enough to handle changes in website layouts.

- **Workflow B: User-Loan Matching (The Eligibility Filter)**
  - **Goal:** To match users from your database with the loan products they are likely eligible for.
  - **Trigger:** This workflow should be triggered via a webhook after a new CSV of users has been successfully processed.
  - **Process:**
    1. Fetch the newly added user records from the PostgreSQL database.
    2. Fetch the entire list of available loan products and their criteria from the database.
    3. Implement the core matching logic within n8n.
    4. For every successful match, store the result (e.g., linking `user_id` to `product_id`) in a `matches` table.
  - **Creative Challenge: The Optimization Treasure Hunt**
    1. **Problem:** Filtering thousands of users against dozens of loan products can be slow. Using a powerful LLM (like Gemini/GPT) to evaluate every single user-product pair would be extremely slow and expensive.
    2. **Your Task:** Design and implement a multi-stage, optimized filtering pipeline. Show your creativity here. How can you intelligently reduce the number of candidates before potentially using an LLM for more nuanced, qualitative checks? (e.g., a fast SQL-based pre-filter, followed by a more complex logic check in n8n, or another creative approach).
- **Workflow C: User Notification**
  - **Goal:** To inform users about the loan products they have been matched with.
  - **Trigger:** This workflow should run after the matching process (Workflow B) is complete.
  - **Process:**
    1. For each user who has new matches, construct a personalized email.
    2. Using the **AWS SES** node in n8n, send the email to the user, listing the loan products they may qualify for.

3. User Interface

- Provide a minimalistic UI that allows for the initial upload of the user data CSV file. This is the primary interaction point for starting the entire pipeline.

Deliverables

1. **GitHub Repository:** A private repository containing all source code (Lambda functions, etc.) and configuration files, with `saurabh@clickpe.ai` and `harsh.srivastav@clickpe.ai` invited as collaborators.
2. **Infrastructure & Automation Files:**
    - A `serverless.yml` file to deploy all required AWS resources.
    - A `docker-compose.yml` file to easily launch your self-hosted n8n instance.
    - The JSON files for each of your n8n workflows.
3. **Comprehensive Documentation (README.md):**
    - **Architecture Diagram:** A clear diagram showing how AWS services and the n8n workflows all interact.
    - **Setup & Deployment:** Step-by-step instructions to deploy the AWS stack and launch the n8n container. This must include how to configure all necessary credentials (AWS, database) within n8n.
    - **Design Decisions:** A brief explanation of your workflow design, web crawling strategy, and especially your solution to the **Optimization Treasure Hunt**.
4. **Demonstration Video (5-10 minutes):**
    - A walkthrough of your n8n workflows, explaining the logic of each node.
    - A live, end-to-end demonstration of the entire pipeline.
    - Display a final, received email generated by the system.

Evaluation Criteria

Your submission will be evaluated based on the following criteria:

Category (Max Points)	Description
n8n Workflow Design & Automation (30)	Sophistication, robustness, and efficiency of your n8n workflows. Creative use of n8n nodes for crawling, data manipulation, and logic. Your solution to the **Optimization Treasure Hunt** will be a key factor.
Cloud Architecture & Integration (20)	Quality of the AWS serverless design. Seamless and secure integration between AWS services and the self-hosted n8n instance. Scalability of the data ingestion method.

Backend Functionality & Code Quality (30)	Correct implementation of the data ingestion pipeline. Clean, modular, and well-commented code.
Documentation & Presentation (20)	Clarity of the architecture diagram and `README.md`. Quality and comprehensiveness of the video demonstration.
Total (100)	

user.csv data download it

https://drive.google.com/file/d/1KBaZRkfr7T2kVBKSr_2fr7v2tTe6f8qu/view?usp=sharing