Product Requirement Document (PRD)

Project Name: Patient-Trial Matching System

Company: Turmerik

Contributor: Chinmay Dhamapurkar

Table of Contents:

---

1. Objective

The objective of the Patient-Trial Matching System is to match patients with suitable clinical trials based on eligibility criteria using AI-powered matching techniques such as **FuzzyWuzzy** and **Sentence Transformers**. This product is intended for use by medical researchers or healthcare professionals to assist in enrolling eligible patients in active clinical trials.

---

2. Background

Turmerik is developing a system to automate the matching process between patients and clinical trials. The current manual process is tedious and error-prone. This system aims to streamline the matching process using two different algorithms to analyze and match patient attributes with clinical trial eligibility criteria.

---

3. Project Scope

Backend Overview

The backend of this system is designed to scrape clinical trial data from ClinicalTrials.gov, process patient data, and run matching algorithms based on AI techniques. The algorithms match patient attributes like age, gender, and conditions to clinical trial inclusion/exclusion criteria, returning a list of eligible trials for each patient. The system also generates detailed statistics based on the matching results.

UI Overview

The frontend of this system is built using **Streamlit**. The UI provides an intuitive interface for healthcare professionals to select patients, choose a matching technique (FuzzyWuzzy or Sentence Transformers), and view/download the results. It includes real-time loading feedback and data visualization of detailed matching statistics.

---

4. Key Features

Backend:

1. **Clinical Trials Scraping**: Automatically scrapes actively recruiting clinical trials from ClinicalTrials.gov.
2. **Patient Data Processing**: Handles patient data with conditions, demographics, and medical history.
3. **Two Matching Algorithms**:
    - **FuzzyWuzzy** (String-based matching)
    - **Sentence Transformers** (Embedding-based matching for semantic similarity)

4. **Statistics Generation**: Calculates the number of matched trials per patient and condition.

UI:

1. **Patient Selection**: Choose a patient from the dataset for matching.
2. **Matching Technique Selection**: Toggle between FuzzyWuzzy and Sentence Transformers for trial matching.
3. **Results Display**: Show detailed statistics of the matching process.
4. **JSON Download**: Download matched trials in JSON format.
5. **Responsive Feedback**: Displays progress and success/failure messages.

---

5. User Stories

1. **As a Healthcare Researcher**, I want to select a patient from my dataset and find matching clinical trials based on patient attributes.
2. **As a Data Scientist**, I want to choose between different matching algorithms (FuzzyWuzzy or Sentence Transformers) for trial matching.
3. **As a Research Coordinator**, I want to view the statistics of matched trials and download the results in JSON format.
4. **As a Project Manager**, I want to understand the efficiency of different algorithms by comparing the results produced by each technique.

---

6. System Architecture and Flow

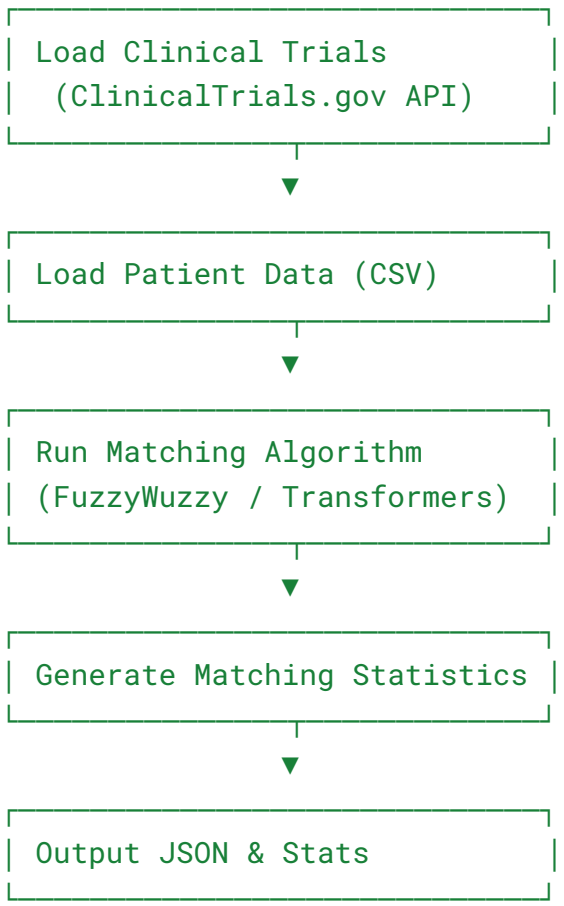
Backend Flow:

1. **Clinical Trial Data Scraping**:
    - Scrapes active trials from ClinicalTrials.gov API.
    - Data is processed into a structured format with necessary attributes (e.g., NCT ID, eligibility criteria).
2. **Patient Data Processing**:
    - Loads and processes patient data from CSV files.
    - Adds key features such as age, gender, and medical conditions.
3. **Matching Algorithms**:

- **FuzzyWuzzy**: Uses string-matching techniques to match patient conditions with trial criteria.
- **Sentence Transformers**: Embedding-based matching for semantic similarity in inclusion/exclusion criteria.

4. **Statistics Generation**:
   - After matching, statistics are generated for each patient, summarizing trial eligibility.

Flowchart for Backend:

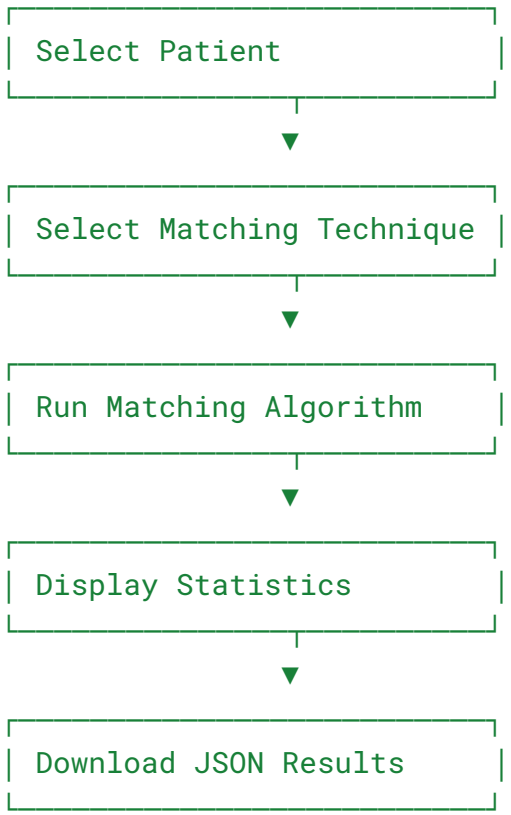


UI Flow:

1. **Patient and Technique Selection**:
   - Select a patient ID and the matching technique.

2. **Matching**:
    - The system runs the chosen algorithm and matches the patient with trials.
3. **Results**:
    - Display the matching statistics and results.
    - Provide a JSON download button.
4. **Progress Feedback**:
    - Provide real-time feedback on loading and matching processes.

Flowchart for UI:



Backend Functionality in Detail

---

1. API Data Scraping

- **Function**: `scrape_clinical_trials()`
- **Description**:
  - The primary role of this function is to scrape **active recruiting clinical trials** from the **ClinicalTrials.gov API**. It does so by issuing a series of **HTTP GET requests** to the API endpoint, specifically requesting data about clinical trials that are currently recruiting patients.
  - The API request contains a filter to ensure that only **recruiting trials** are retrieved and specifies a **JSON response format** to easily process the data.
  - The **maximum number of trials** that can be fetched per API call is **1000**, which is the upper limit imposed by the API. The function handles **pagination** to fetch additional trials when there are more than 1000. The number of pages is dynamically calculated based on the total number of trials requested by the user (e.g., if 3000 trials are needed, the function will fetch 3 pages).
  - **Data Processing**:
    - After receiving the data from the API, the function parses the JSON response and extracts relevant trial information. This includes:
      - **NCT ID**: A unique identifier for each trial.
      - **Title**: A brief title describing the trial.
      - **Condition**: Medical conditions being addressed in the trial.
      - **Inclusion and Exclusion Criteria**: These are crucial for determining whether a patient qualifies for the trial.
      - **Eligibility**: Criteria related to gender and age (e.g., male/female/all, age range).
      - **Status**: To ensure the trial is actively recruiting.
      - **MinAge and MaxAge**: Specifies the age limits of the trial.
  - **Return Value**:
    - The function consolidates this information into a **structured Pandas DataFrame**. This format allows for easy manipulation and analysis in subsequent steps of the process. Each row in the DataFrame represents a trial, with columns corresponding to the attributes mentioned above.
  - **Handling API Errors**:
    - If the API request fails, the function handles errors gracefully by checking the HTTP status code and providing an error message. This ensures that the system can continue working without crashing.

---

2. Patient Data Processing

- **Function**: `load_patient_data()`
- **Description**:
  - This function is responsible for **loading patient data** from pre-existing CSV files. These CSV files contain patient demographics (such as ID, birthdate, gender) and medical history (current and past conditions).
  - **Age Calculation**:
    - One of the key attributes needed for trial eligibility is the **patient's age**. To calculate this, the function converts the **BIRTHDATE** field into a **datetime** format and subtracts it from the current year to compute each patient's age. This is crucial for comparing the patient's age with the **MinAge** and **MaxAge** eligibility criteria of clinical trials.
  - **Medical Condition Processing**:
    - The function then processes the patient's **medical history**. It categorizes conditions into two types:
      - **Past Conditions**: Conditions that the patient previously had but no longer suffers from (determined by checking if the condition has a "STOP" date).
      - **Current Conditions**: Conditions the patient currently has (those with no "STOP" date).
    - This structured medical history is essential for comparing the patient's conditions with the inclusion and exclusion criteria of clinical trials.
  - **Return Value**:
    - The function returns a **merged DataFrame** that contains key patient attributes, including:
      - **ID**: Unique identifier for each patient.
      - **Age**: Age calculated based on the birthdate.
      - **Gender**: Gender of the patient.
      - **Past Conditions**: A list of conditions the patient previously had.
      - **Current Conditions**: A list of conditions the patient is currently suffering from.

---

3. Matching Algorithms

- **FuzzyWuzzy**:
  - This algorithm uses **fuzzy string matching** to compare a patient's medical conditions against the **inclusion and exclusion criteria** of clinical trials.
  - **FuzzyWuzzy** works by calculating the **similarity ratio** between two strings. The ratio represents how similar two strings are on a scale from 0 to 100, where 100 means the strings are identical.

    - When the **FuzzyWuzzy** algorithm is selected, the patient's conditions (both past and present) are compared to the **eligibility criteria** of the trial using the **partial_ratio** function from FuzzyWuzzy.
        - If the similarity score between a patient's condition and a trial criterion exceeds a set threshold (e.g., 50%), it is considered a match.
        - This approach is particularly useful when the wording in the patient's medical history and the trial's criteria are not identical but close enough to imply a match.
- **Sentence Transformers**:
    - Sentence Transformers use **semantic embeddings** to determine how similar two sentences (or phrases) are at a deeper, conceptual level, beyond just string matching. This method is much more robust than FuzzyWuzzy for capturing **meaning-based similarities**.
    - **Sentence-BERT** is used to generate embeddings (numerical representations) for the patient's conditions and the trial's eligibility criteria. These embeddings are then compared using **cosine similarity**, which ranges from -1 to 1, where 1 means the sentences are identical in meaning.
    - When **Sentence Transformers** is selected, the algorithm computes the **cosine similarity** between the patient's conditions and the trial's inclusion/exclusion criteria. A similarity score above a predefined threshold (e.g., 0.5) is considered a match.
    - This method is more sophisticated because it understands the context of the words, allowing it to match conditions and criteria that might use different terminology but have the same meaning (e.g., "hypertension" and "high blood pressure").

---

4. Statistics Generation

- **Function**: `generate_statistics()`
- **Description**:
    - After the matching process is completed, this function is responsible for generating a **detailed statistical summary** of the matching results.
    - For each patient, the function counts:
        - **Total Number of Eligible Trials**: How many trials matched based on the patient's conditions, age, and gender.
        - **Condition Matches**: The number of trials where the patient's conditions matched the trial's criteria (this is tracked specifically to provide insights into how well the patient fits the trial's medical requirements).
    - **Return Value**:
        - The function returns a **Pandas DataFrame** where each row corresponds to a patient, and columns show:
            - **patientId**: The unique identifier for the patient.

        - **number_of_trials**: The total number of trials the patient is eligible for.
        - **conditions_matched**: The number of trials where the patient's conditions match the trial's inclusion/exclusion criteria.
    - **Purpose**:
        - The statistics generated by this function provide actionable insights into how well each patient fits into available clinical trials. These statistics can be useful for healthcare professionals to make informed decisions about which trials a patient should consider.
        - The statistics can also serve as a **performance metric** for the different matching algorithms, helping users understand how each algorithm performs in terms of condition matching.

---

8. UI Functionality

1. Patient Selection:

- **Input**: Dropdown to select a patient from the loaded dataset.
- **Function**: Filters the patient dataset to the selected patient.

2. Technique Selection:

- **Input**: Radio button to choose between FuzzyWuzzy and Sentence Transformers.
- **Function**: Controls which algorithm is used for matching.

3. Matching Execution:

- **Input**: "Run Matching" button.
- **Function**: Initiates the backend process to match patients to trials.

4. Results Display:

- **Output**: Dataframe displaying detailed statistics of the matching process.
- **Function**: Visualizes the number of trials matched to each patient.

5. JSON Download:

- **Input**: Download button.
- **Function**: Allows the user to download the matched trial data as a JSON file.

---

9. Assumptions

1. The patient dataset is well-formatted with necessary columns such as birthdate, gender, and medical history.
2. Clinical trial data can be scraped without API limitations or rate restrictions.
3. The system will handle up to a few thousand patients and trials efficiently.

---

10. Future Considerations

1. **Scaling**: Expand the system to handle larger datasets with thousands of patients and trials.

2. **Enhanced Matching**: Incorporate more advanced NLP techniques for a deeper analysis of trial eligibility.
3. **Role-based Access**: Introduce different user roles (researchers, coordinators) with varying levels of access.
4. **Reporting**: Add more in-depth reporting and analytics on the matching process for administrative purposes.

This PRD outlines the complete system architecture and functionality of the Patient-Trial Matching System developed by Turmerik, covering both backend and UI details. The flowcharts and structure aim to provide a clear vision of how data flows through the system from trial scraping to patient matching and output generation.