System Analysis & Design

# Problem Statement & Objectives

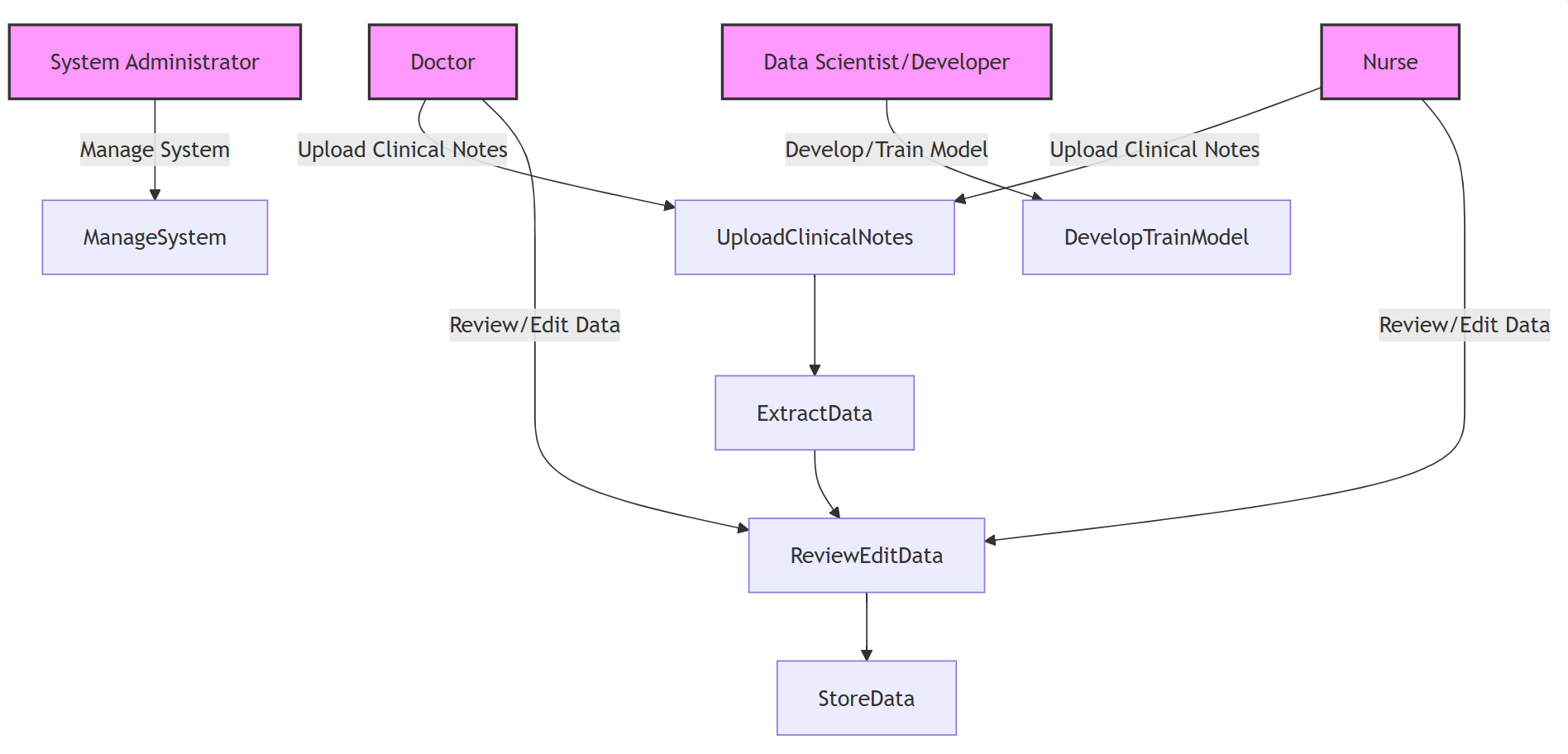
**Problem Statement:** The goal of the project is to create an automated medical information extraction system using Natural Language Processing (NLP) to process and extract meaningful data from unstructured clinical notes. Healthcare professionals spend a significant amount of time manually reviewing clinical notes, which can lead to errors, delays, and inefficiencies in patient care.

**Project Objectives:**

1. Develop an NLP model to accurately extract medical entities such as diagnoses, medications, treatments, allergies, and procedures from clinical texts.
2. Create a user-friendly interface for healthcare professionals to interact with the system, review extracted data, and ensure accuracy.
3. Integrate the system with hospital Electronic Health Records (EHR) for seamless data updates.
4. Improve efficiency and accuracy in clinical decision-making by reducing manual data entry and potential errors.

# 2. Use Case Diagram & Descriptions

## 2.1 Use Case Diagram



**Actors:**

1. **Doctor** – Interacts with the system to upload clinical notes and review extracted medical data.
2. **Nurse** – Assists in uploading clinical notes and verifies extracted data.
3. **System Administrator** – Manages system settings, access permissions, and user roles.
4. **Patient** – The source of clinical data, benefiting indirectly from improved care.
5. **Data Scientist/Developer** – Works on the model development, training, and system maintenance.

## 2.2 Use Case Descriptions

Use Case 1: Upload Clinical Note

- Actor: Doctor, Nurse

- Description: User uploads clinical note into the system.

- Preconditions: User must be logged in.

- Postconditions: Clinical note is uploaded to the system.

Use Case 2: Process Clinical Note

- Actor: System

- Description: System processes the uploaded clinical note and extracts relevant information.

 **Review and Edit Extracted Data:** Doctors or nurses review and edit the extracted data.

 **Store Extracted Data:** The system stores the data in the database and updates the patient's record.

# 3. Functional & Non-Functional Requirements

## 3.1 Functional Requirements

1. Automatically extract medical entities (e.g., diagnoses, medications) from clinical notes.
2. Allow users to review and edit the extracted data.
3. Integrate with Electronic Health Records (EHR) systems for data synchronization.
4. Generate reports based on the extracted medical data.

## 3.2 Non-Functional Requirements

• Performance: System should process data within 5 seconds for files up to 5000 words.

• Security: All data should be encrypted at rest and in transit.

• Reliability: The system should maintain 99.9% uptime.

# 4. Software Architecture

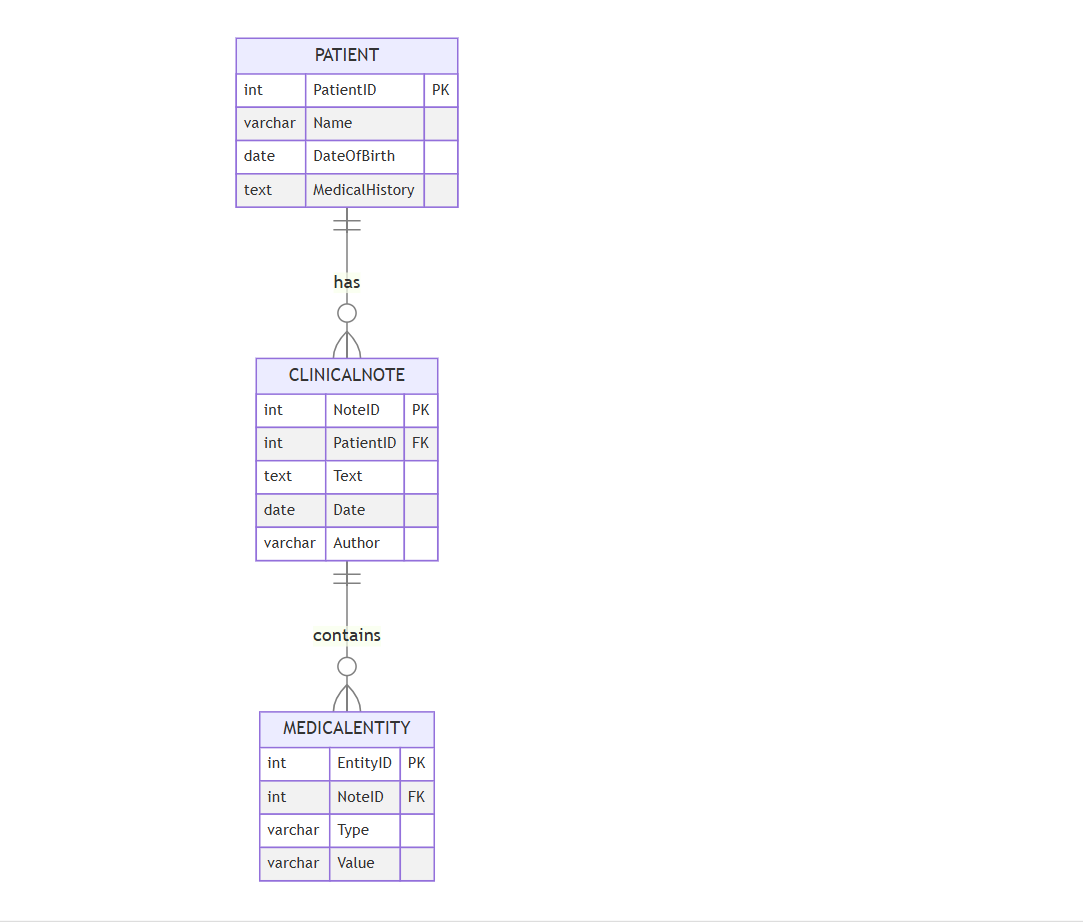
## 4.1 High-Level Design

Overview: A high-level description of the system components and how they interact.

Architecture Style: Microservices / Modular Design (Based on your architecture decision).

# 5. Database Design & Data Modeling

## 5.1 Entity-Relationship Diagram (ERD)



## 5.2 Logical & Physical Schema

• Tables: List out key tables such as `Patient`, `ClinicalNote`, and `MedicalEntity`.

• Attributes: Describe attributes of each table (e.g., `PatientID`, `NoteID`, `Text`, etc.).

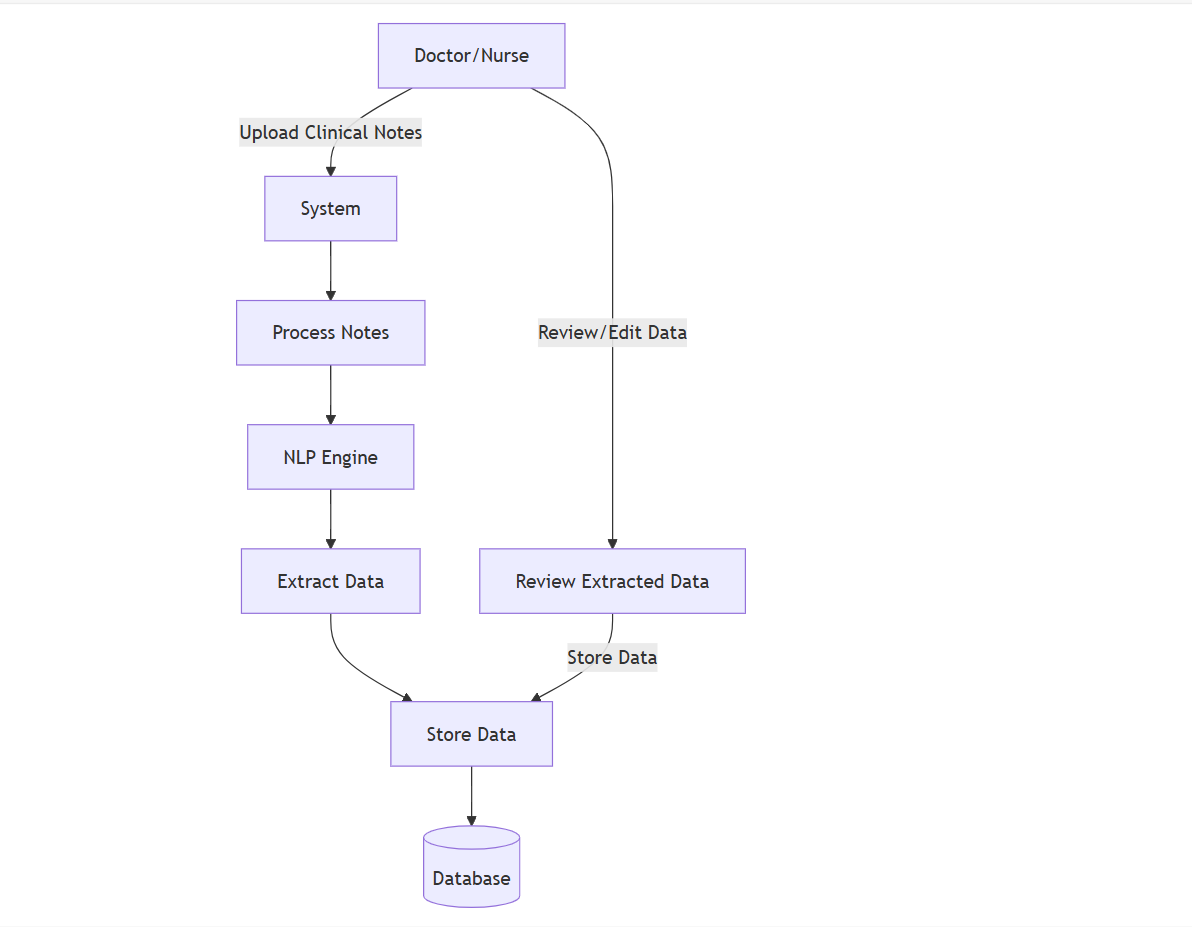
• Keys: Primary and foreign key relationships.

# 6. Data Flow & System Behavior

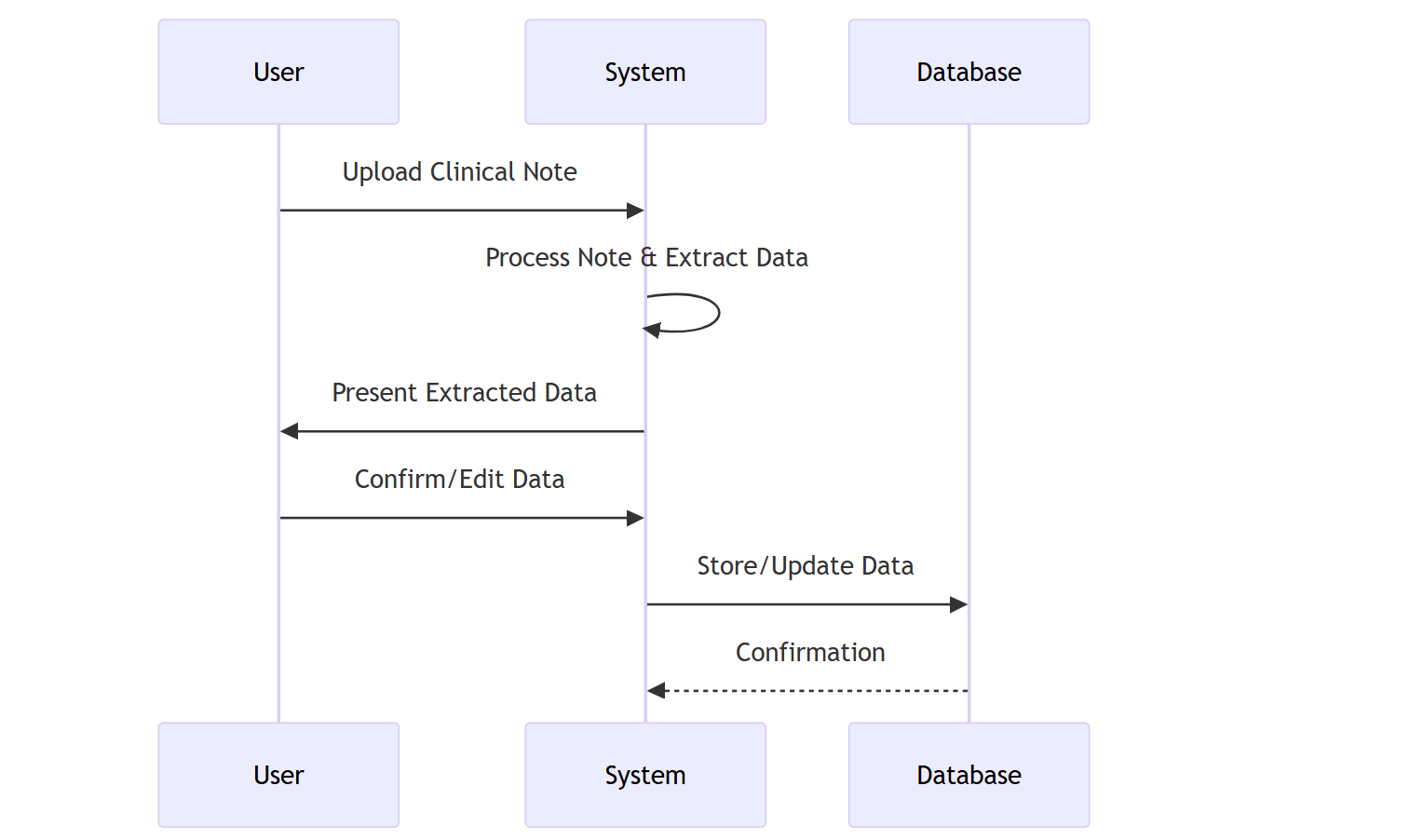
## 6.1 Data Flow Diagram (DFD)

#### 1. DFD (Data Flow Diagram)

* **Context-Level Diagram:**
  + The user (Doctor/Nurse) uploads clinical notes into the system. The system processes these notes using the NLP engine and stores the extracted data in the database.
* **Detailed-Level Diagram:**
  + Shows the internal processes for text extraction, data storage, and user interaction.



## 6.2 Sequence Diagram



# 8. System Deployment & Integration

## 8.1 Technology Stack

• Backend: Python (Flask/FastAPI), NLP libraries (SpaCy, NLTK)

• Frontend: React, Angular

• Database: PostgreSQL, MongoDB