Project Planning and Management

1. Project Proposal:

Project Overview:

The Healthcare Predictive Analytics project aims to enhance healthcare decision-making by developing a machine learning model that predicts patient risks and health outcomes. This predictive system will support healthcare professionals in assessing patient conditions, allocating resources efficiently, and identifying emerging health trends.

Objectives:

- Improve patient risk prediction through data-driven insights.
- Identify patterns in health metrics to assist in proactive healthcare management.
- Develop a scalable and deployable predictive model.
- Ensure model reliability through MLOps practices.

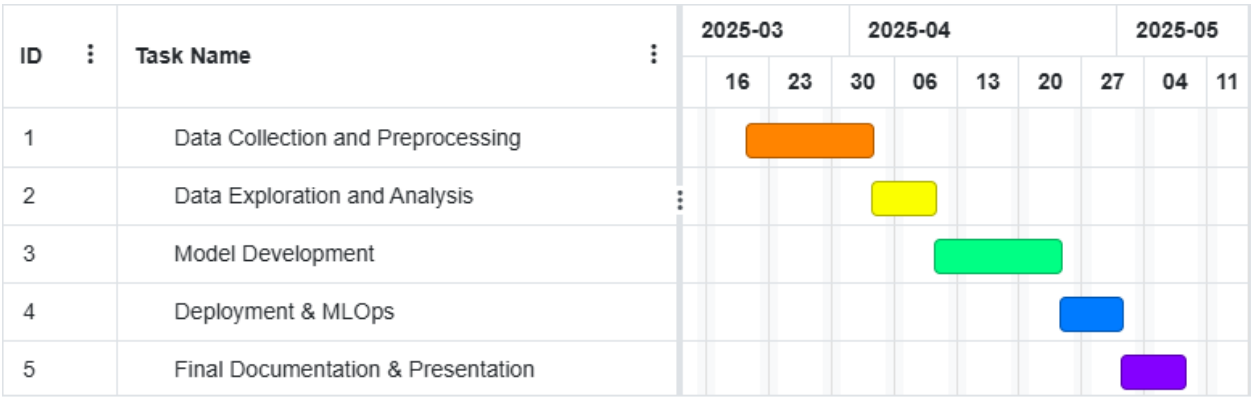
Scope:

- Data collection, cleaning, and preprocessing.
- Exploratory data analysis and visualization.
- Machine learning model selection, training, and optimization.
- Deployment of the predictive model as a web service.
- Ongoing model monitoring and performance evaluation.

---

2. Project Plan:

Timeline:



3. Milestones and Deliverables:

1-Milestone 1: Data Collection, Exploration, and Preprocessing (Weeks 1-2)

Tasks:

- **Collect healthcare datasets (e.g., patient records, clinical data, test results).**
- **Perform Exploratory Data Analysis (EDA) to identify trends, missing values, and outliers.**
- **Preprocess data (handling missing values, encoding categorical variables, normalization).**

2-Milestone 2: Data Analysis & Feature Engineering (Weeks 3-4)

Tasks:

- **Perform statistical analysis (correlation, hypothesis testing).**
- **Engineer new features (e.g., risk factors, trends in medical history).**
- **Visualize insights using heatmaps, scatter plots, and dashboards**

3-Milestone 3: Model Development & Optimization (Weeks 5-7)

Tasks:

- **Select machine learning models (Logistic Regression, Random Forest, Neural Networks).**
- **Train models and evaluate using metrics like Precision**
- **Optimize hyperparameters using Grid Search/Random Search.**

4-Milestone 4: Deployment & Monitoring (Weeks 8-9)

Tasks:

- **Deploy the model.**
- **Monitor model performance over time (detect data drift).**

5- Milestone 5: Final Documentation & Presentation (Weeks 10-12)

Tasks:

- **Summarize findings, methodologies, and challenges.**
- **Create an engaging PowerPoint presentation for stakeholders.**
- **Discuss future improvements (e.g., adding more patient data, refining model accuracy).**

---

4. Task Assignment and Roles:

Team Member	Role
Omar Khaled Ashour	
Jana Khaled Beshir	
Sara Samy El Wakeel	
Sara Aly Ahmed	
Mariam Taher Ahmed	
Mariam Mostafa Abdallah	

---

5. Risk Assessment and Mitigation Plan:

Risk	Mitigation Strategy
Data availability issues	Use publicly available healthcare datasets as a backup
Model performance below expectations	Use multiple models and optimize hyperparameters
Deployment Challenges	Conduct extensive testing before deploying to production

6. Key Performance Indicators (KPIs):

- Model accuracy: >85% for predictions.
- API response time: <500ms.
- System uptime: 99%.
- Stakeholder adoption rate: 80% of target users utilize the system.

- False positive/negative rates: Minimized through ongoing model monitoring.

Requirements Gathering

1. Stakeholder Analysis:

a. Doctors need accurate patient risk predictions and intuitive dashboards.
b. Hospital administrators need resource optimization insights and cost management.
c. Patients need personalized health insights and transparency in risk predictions.
d. IT and DevOps teams need secure and scalable deployment with easy integration.

2. User Stories and Use Cases:

- *User story 1:* As a doctor, I want to input patient data and receive risk predictions to make informed treatment decisions.
- *User story 2:* As a hospital administrator, I need trend reports on patient outcomes to optimize resource allocation.
- *User story 3:* As an IT manager, I want to ensure the model remains updated and performs well over time.

Use Case Example:

- Title: Predicting Patient Readmission Risk
- Actors: Doctor, patient, predictive model
- Description: A doctor inputs a patient's recent health data, and the model provides a probability score for readmission within 30 days. Based on this score, the doctor can adjust treatment plans accordingly.

---

3. Functional Requirements:

- **Data Input:** Accepts patient records, clinical test results, demographics.
- **Predictive Analytics:** Generates risk scores for various health conditions.
- **Visualization:** Provides interactive dashboards for trend analysis.

- **Model Performance Tracking:** Logs predictions, accuracy, and drift detection.
- **API Integration:** Allows external healthcare systems to access predictions.

---

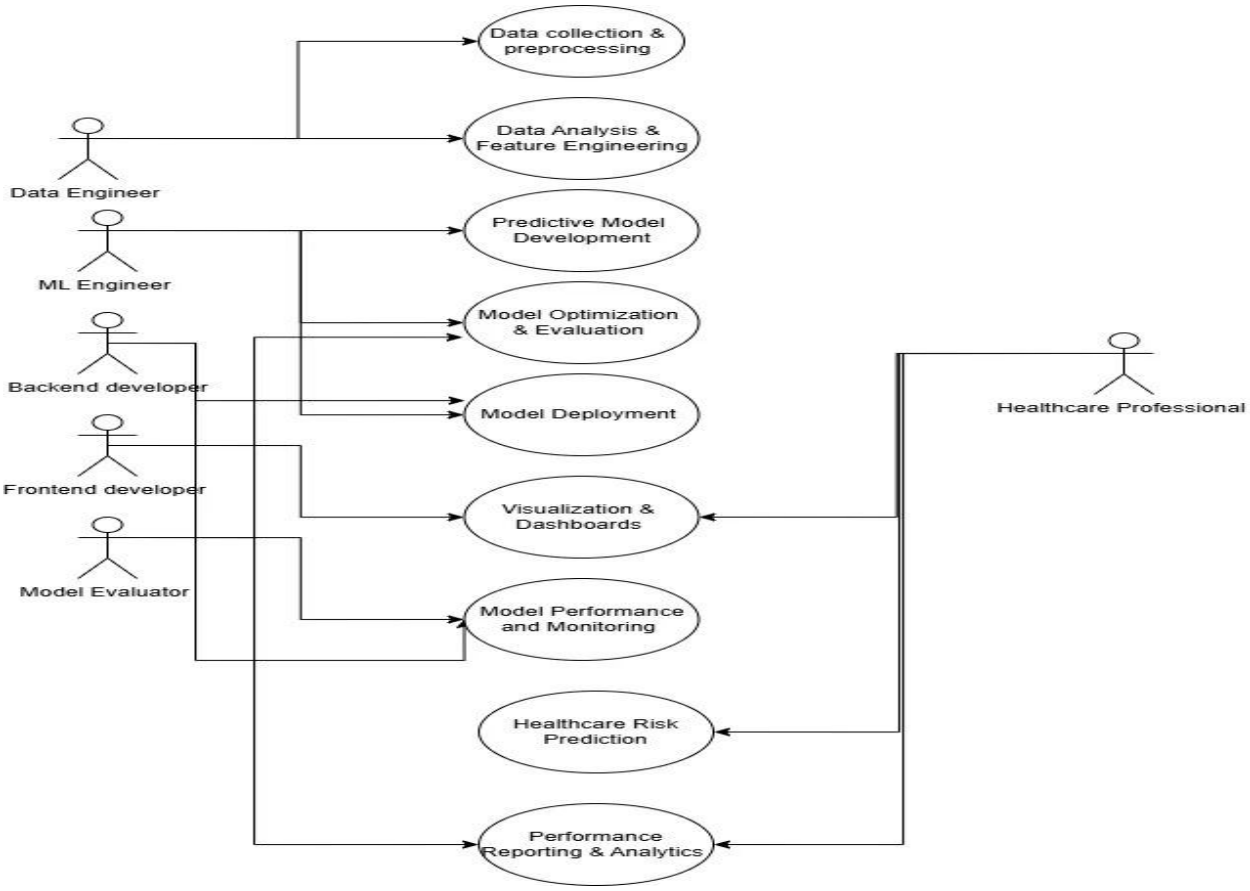
4. Non-Functional Requirements:

- **Performance:** Predictions within 500ms per request.
- **Security:** Data encryption, role-based access control.
- **Usability:** User-friendly interface for doctors and administrators.
- **Reliability:** System uptime of 99%.
- **Scalability:** Cloud-based deployment to handle high loads.

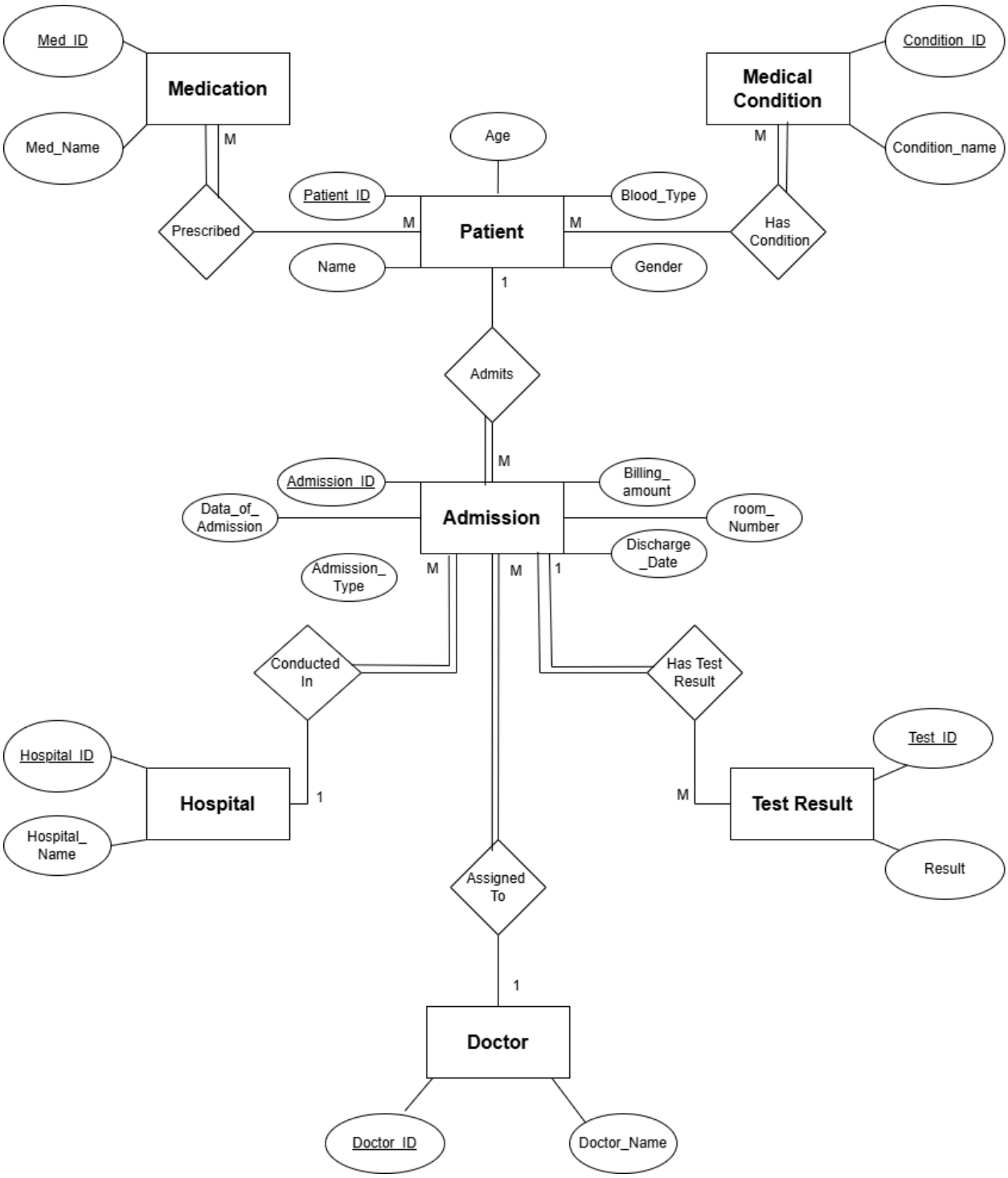
System Analysis and Design

1. Database Design and Data Modeling:

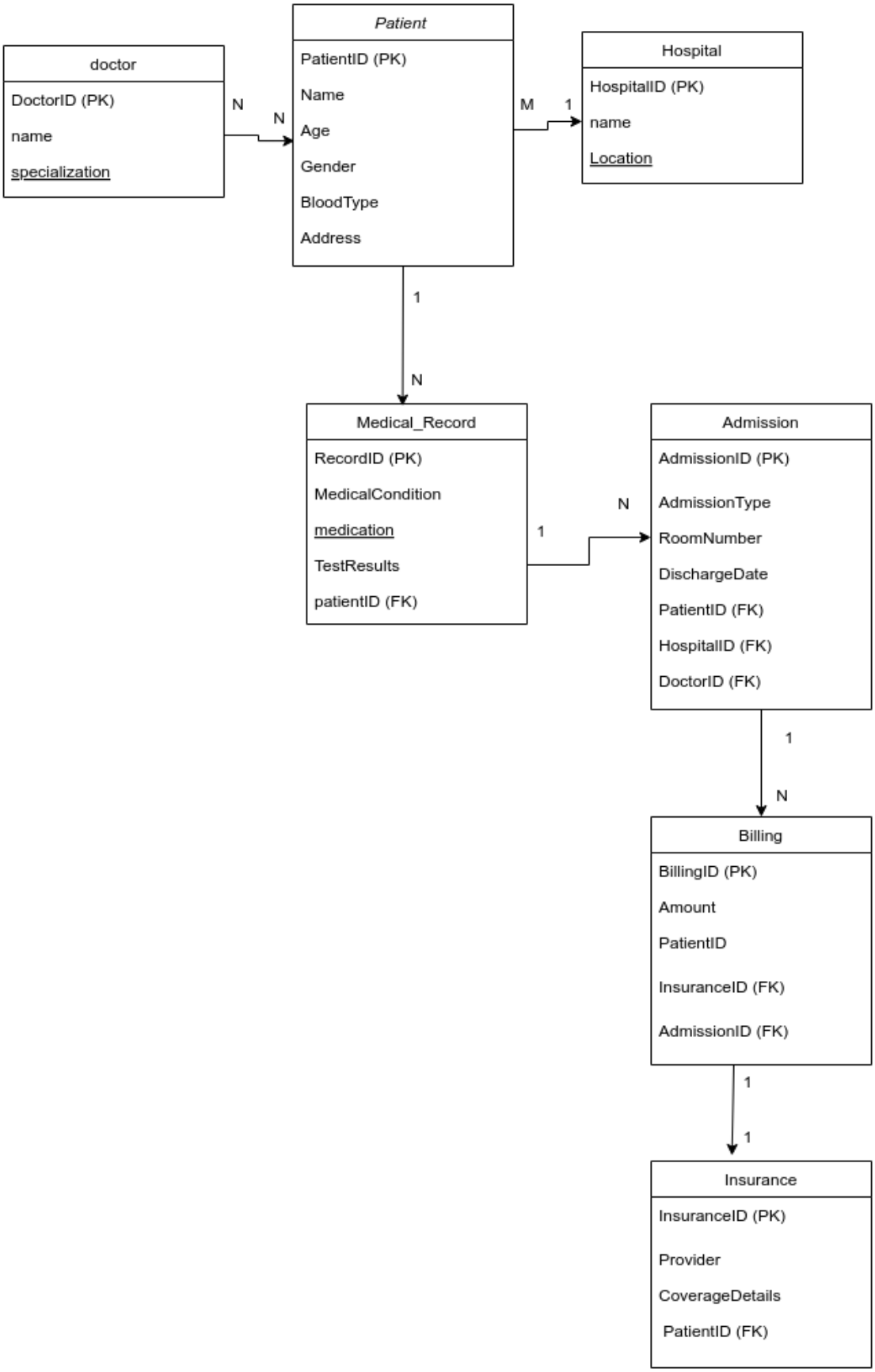
Use-Case Diagram:



Entity Relationship Diagram (ERD):

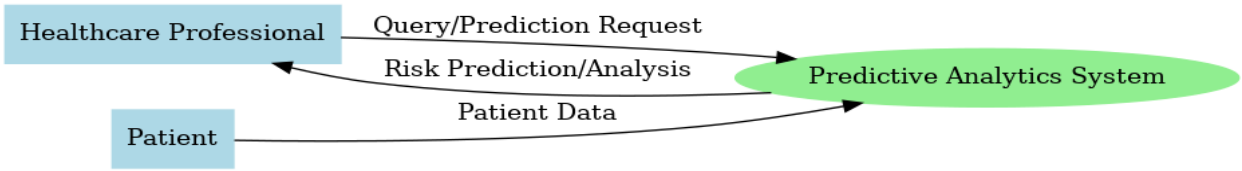


Logical and Physical schema:



2. Data Flow and System Behavior:

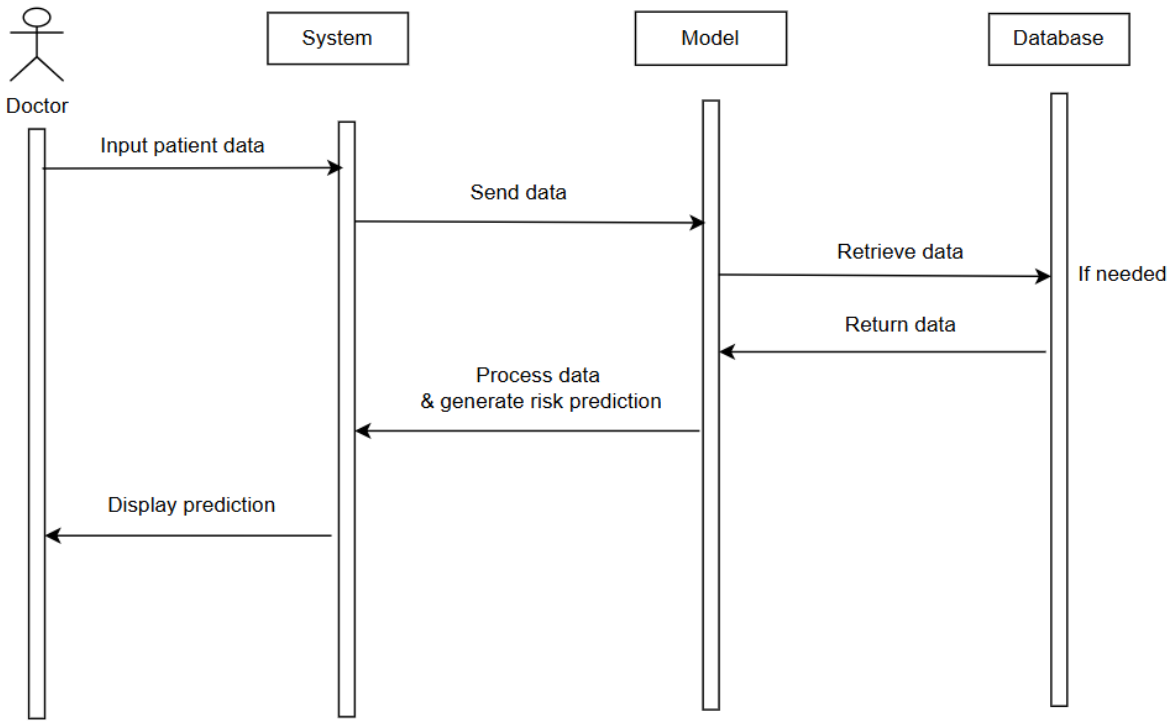
- ***Data Flow Diagrams***
    - Context-Level DFD:



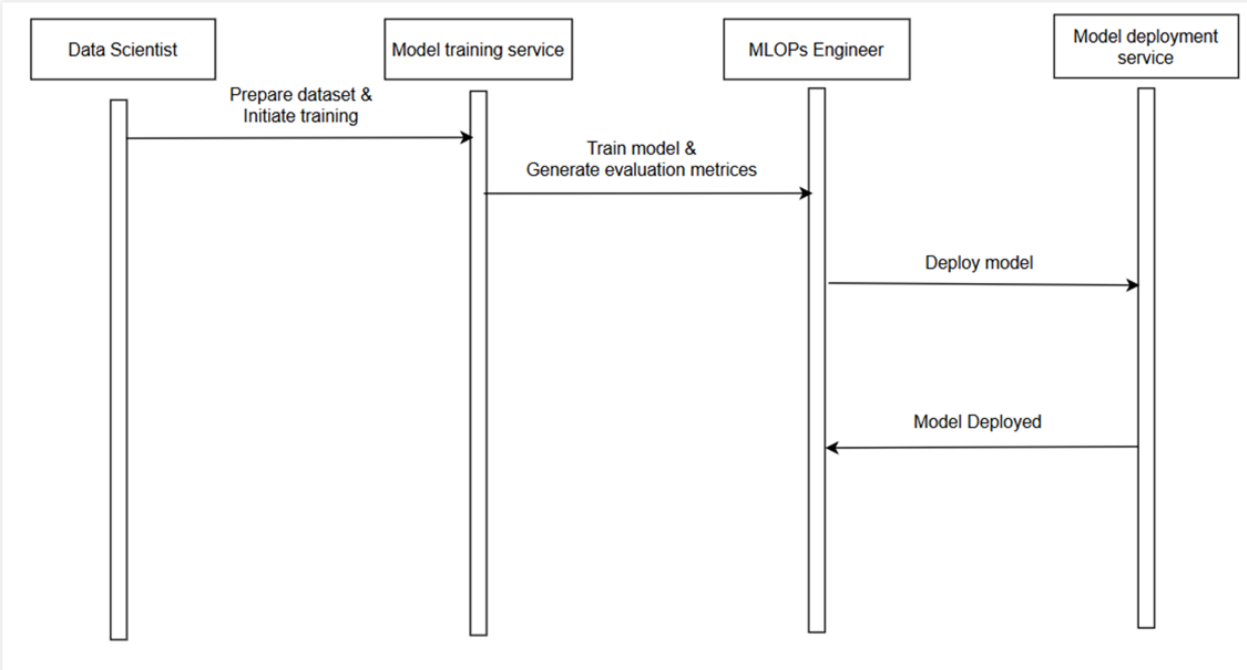
    - Detailed-Level DFD:

Sequence Diagrams

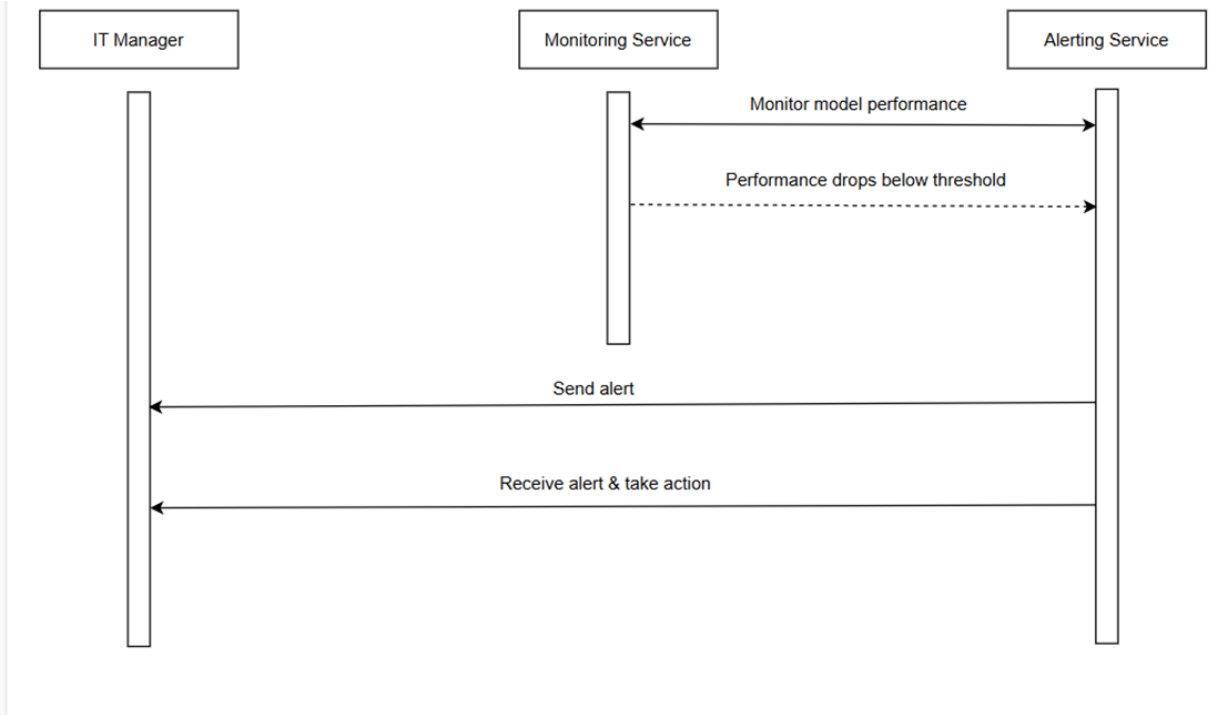
1. Patient Risk Prediction:



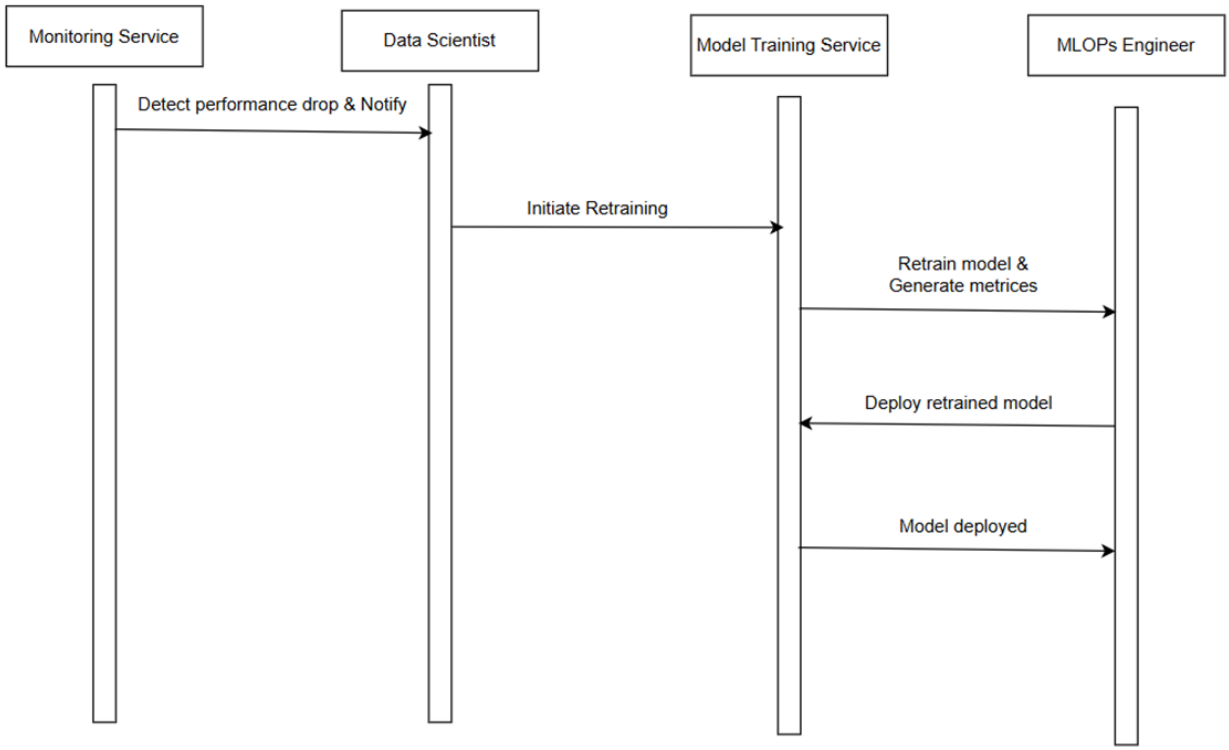
2. Model Training and Deployment:



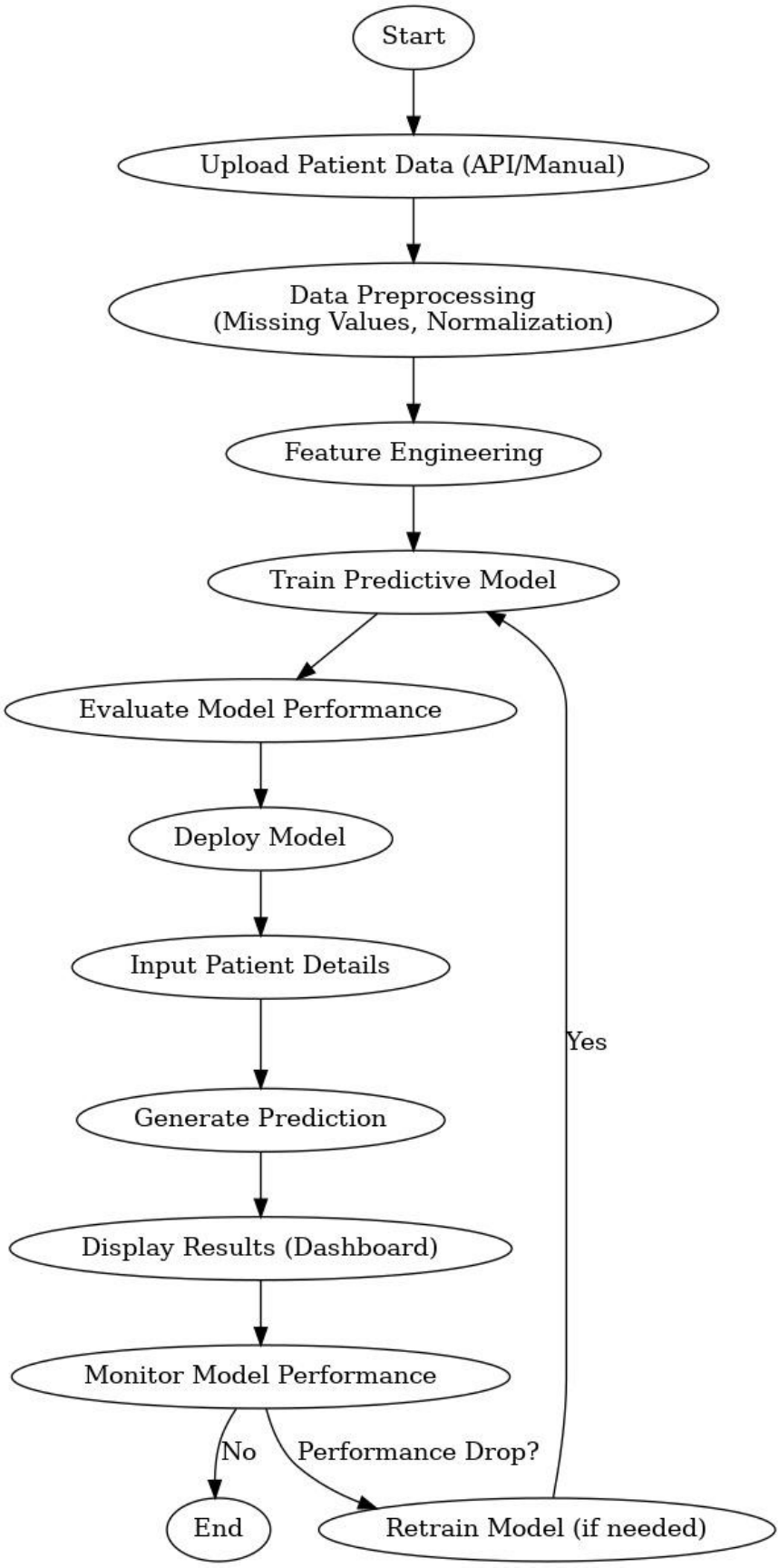
3. System Monitoring and Alerting:



4. Model Retraining:

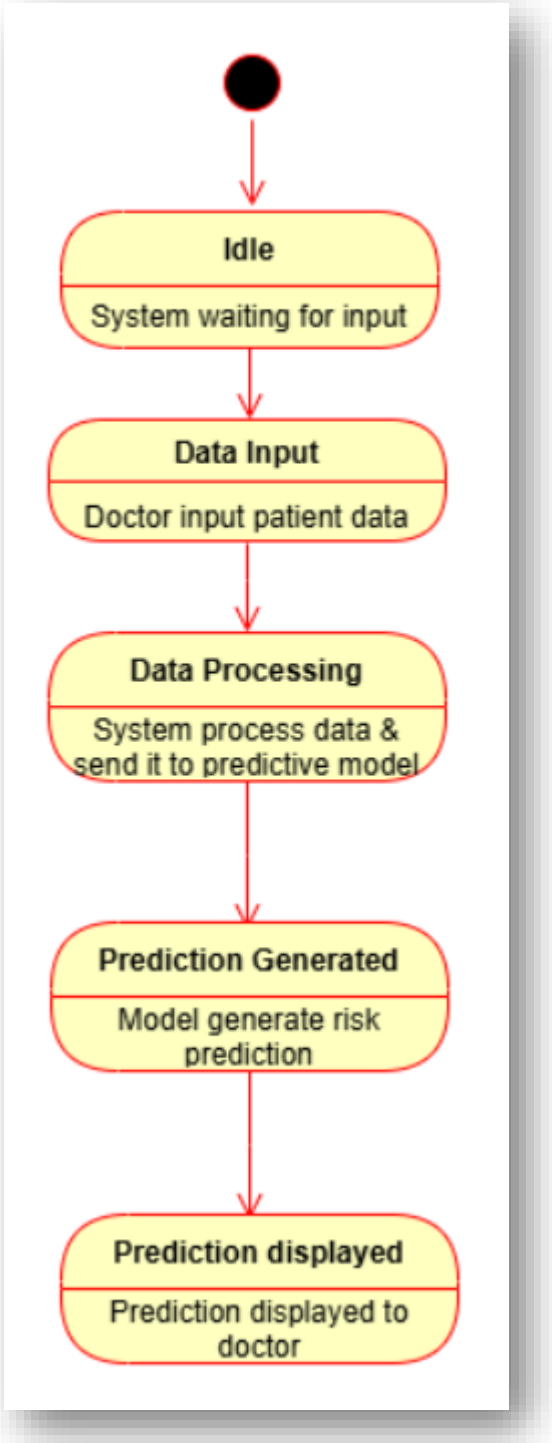

Monitoring Service
Data Scientist
Model Training Service
MLOPs Engineer
Detect performance drop & Notify
Initiate Retraining
Retrain model &
Generate metrices
Deploy retrained model
Model deployed

Activity Diagram

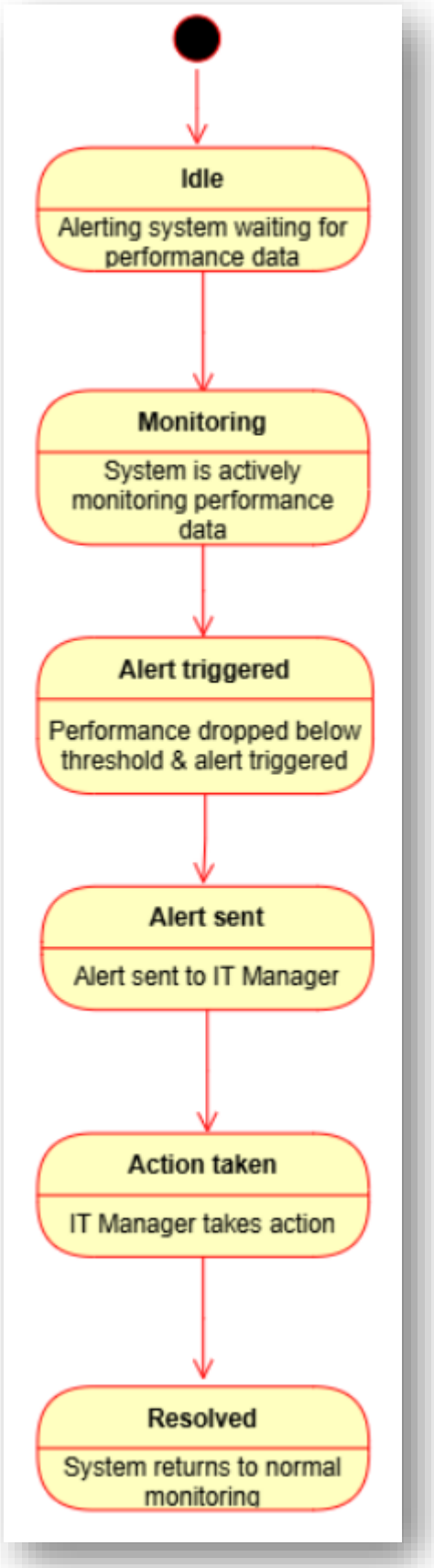


State Diagrams

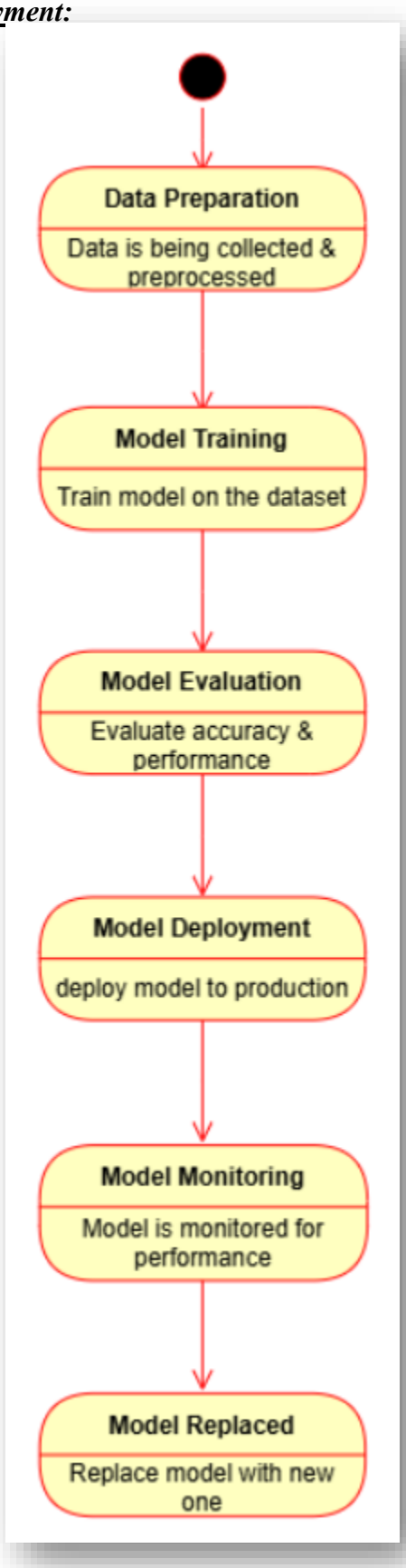
1. ***Patient Risk Prediction :***



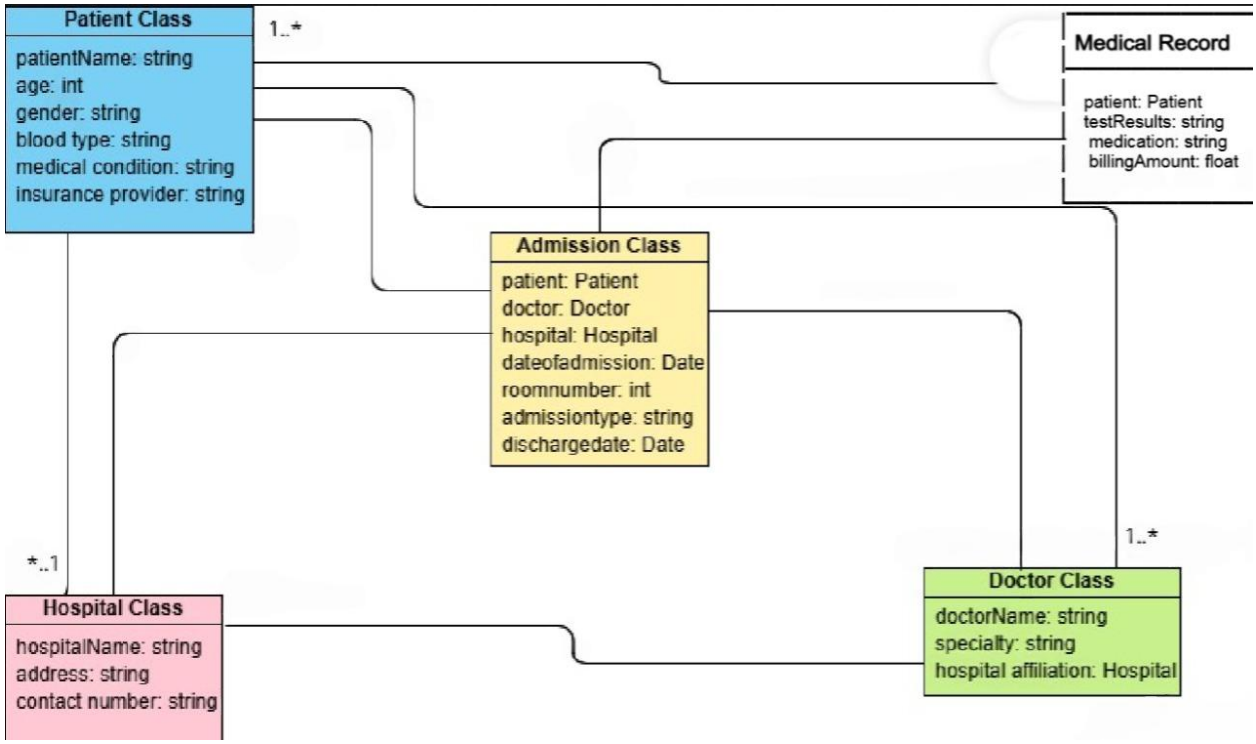
2. ***Alerting System:***



3. ***Model Training and Deployment:***



Class Diagram:



UI/UX Design & Prototyping

Wireframes & Mockups

1. Login Screen:
    a. Simple and clean login interface with fields for username and password.
    b. Option to reset password.
    c. Login button and sign-up link.
2. Patient Data Entry Screen:
    a. Form to input patient details (age, medical history, test results, etc.).
    b. Submit button to send data for prediction.
    c. Input validation and clear button.
3. Prediction Results Screen:
    a. Displays risk prediction and insights.
    b. Graphical representation of patient risk over time.
    c. Option to download or print the report.
4. Model Monitoring Screen:
    a. Real-time monitoring of model performance.
    b. Metrics such as accuracy, recall, and model drift.

UI/UX Guidelines

1. Design Principles:
    a. Minimalistic and clean design to reduce cognitive load.
    b. Focus on simplicity and easy navigation.
2. Color Scheme:
    a. Primary: #4CAF50 (Green) for positive actions and success messages.
    b. Secondary: #FFC107 (Amber) for warnings.
    c. Error: #F44336 (Red) for critical alerts.
    d. Neutral: #E0E0E0 (Light Gray) for backgrounds.
3. Typography:
    a. Headers: Roboto Bold, 24px
    b. Subheaders: Roboto Medium, 18px
    c. Body Text: Roboto Regular, 14px
4. Accessibility Considerations:
    a. High contrast between text and background for readability.
    b. Keyboard navigability and screen reader compatibility.
    c. Clear labeling and tooltips for interactive elements.

System Deployment & Integration

Technology Stack :

- Backend: Python (Django), TensorFlow for model inference
- Frontend: React.js for interactive user interfaces
- Database: SQL for storing patient data and predictions
- Deployment Platform: Docker containers hosted on AWS EC2
- Monitoring: Prometheus and Grafana for real-time performance monitoring

Deployment :

- AWS EC2 Instance: Hosts the backend (Django API) and model server
- Docker Containers: Isolate backend services and model serving
- SQL Database: Stores patient records and predictions
- React Frontend: Deployed as a static site served via NGINX
- Monitoring Services: Prometheus for metrics, Grafana for visualization

Components:

1. User Interface (Frontend):
    a. Built with React.js, allows users to enter patient data and view predictions.
2. Backend API (Django):
    a. Handles data processing and communicates with the model server.
3. Prediction Engine (Model Server):
    a. Uses trained ML models to generate risk predictions.
4. Database (SQL):
    a. Stores patient data, predictions, and logs for monitoring.
5. Monitoring System (Prometheus & Grafana):
    a. Tracks model performance, system uptime, and data processing metrics.
6. Deployment Infrastructure (AWS EC2 with Docker):
    a. Ensures scalability and reliability of services.