Software Requirement Specifications

AI Disease Prediction System

Submitted by

Arooj Nayyar (F22BINFT1E02006)

Submitted to

Mr. Faisal Shahzad

Department of Information Technology

Faculty of Computing

The Islamia University of Bahawalpur

Revision History

Date	Description	Author	Comments

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date

Supervised by

Mr.FaisalShahzad

Signature______ ___

Summary

The MediCare AI Disease Prediction System is a comprehensive web-based platform designed to connect users, healthcare seekers, and administrators to facilitate seamless and transparent health risk assessments using artificial intelligence and machine learning algorithms. The application enables users to input their health parameters through intuitive dynamic forms, which the system utilizes to predict potential disease risks for six major health conditions: Diabetes, Cardiovascular Disease, Kidney Disease, Breast Cancer, Depression, and Obesity.

The primary goal of the application is to ensure early health awareness, provide detailed probability-based risk assessments, and offer comprehensive information about diseases including their causes, symptoms, prevention methods, and treatment options. The system bridges the gap between users and healthcare awareness by providing AI-powered predictions that can help individuals make informed decisions about seeking professional medical advice.

Key functionalities of the system include:

Users: - Register and create personal accounts on the platform - Log in securely to access personalized features - Input health parameters through disease-specific dynamic forms - Receive instant disease risk predictions with detailed probability scores - View risk levels categorized as Low (0-30%), Medium (31-60%), or High (61-100%) - Access comprehensive disease information including: - Detailed disease descriptions and causes - Risk factors and symptoms - Prevention guidelines and lifestyle recommendations - Treatment options and medical interventions - View complete prediction history with timestamps and results - Track health trends over time through historical data - Submit inquiries and feedback through the contact form - Access FAQ sections for common questions

Admin: - Manage the entire platform through Django’s powerful admin panel - Configure and manage ML model versions dynamically including: - Upload new trained models in multiple formats (Pickle, Keras, ONNX, PyTorch) - Define feature schemas for each model - Set feature types (numeric, categorical) - Configure dropdown options for categorical features - Activate/deactivate models without code changes - Update disease information content through the built-in CMS: - Disease Information sections - Prevention Guidelines - Treatment Options - Manage site-wide settings: - Site name, tagline, and branding - Contact information (email, phone, address) - Social media links (Facebook, Twitter, LinkedIn, Instagram) - Footer content and medical disclaimers - View and respond to contact messages from users - Manage FAQ sections and About page content - Monitor prediction logs and system activities - Maintain detailed records of all predictions and user interactions

The system aims to provide transparency, ease of use, and an efficient mechanism for bridging the gap between users and healthcare awareness through AI-powered disease predictions. It serves as an educational and awareness tool while clearly communicating that predictions are not medical diagnoses and users should consult healthcare professionals for medical advice.

Contents

1. Introduction

1.1 Purpose

The purpose of the **MediCare AI Disease Prediction System** is to build a reliable, intelligent, and user-friendly web platform that empowers individuals to assess their potential health risks using advanced machine learning algorithms. In the modern era, a large portion of the population avoids regular medical checkups due to limited time, financial constraints, or simply lack of awareness. This system fills that critical gap by offering instant, accessible, and data-driven health risk predictions for major diseases like Diabetes, Cardiovascular Disease, Kidney Disease, Breast Cancer, Depression, and Obesity.

The system allows users to input their health parameters through dynamic, disease-specific forms. These inputs are processed through trained ML models that return prediction results including probability scores, risk-level classification, and meaningful recommendations. Users are transparently guided about their health conditions with clear explanations rather than complicated medical terminologies.

On the administrative side, the system provides full control through a comprehensive dashboard where administrators can upload and configure ML models, update disease content, manage site settings, and monitor system performance. The purpose is not only to predict diseases but also to educate users, enhance awareness, and steer them towards timely professional medical consultation.

1.1.1 Goals

The primary goals of MediCare AI include:

- Providing users with **accurate, AI-driven prediction results** along with probability percentages and risk classification.
- Delivering **structured, reliable, and up-to-date information** about diseases, their symptoms, causes, preventive measures, and treatment options.
- Ensuring **transparent health guidance** by explaining prediction logic, limitations, and recommended next steps in an easy-to-understand manner.
- Supporting administrators in efficiently managing all system content, ML models, and configurations without requiring technical expertise.
- Creating a system that motivates health-conscious individuals to take **proactive measures** and track their health trends over time.

1.1.2 Objectives

The main objectives of the system are:

- To enable users to **securely input** their health parameters through dynamic forms aligned with ML feature schemas.
- To process user inputs through trained ML models and provide results including:

  - At-risk / Not-at-risk prediction
  - Probability score (0–100%)
  - Risk level (Low, Medium, High)
  - Personalized recommendations
- To maintain detailed **prediction history** that helps users track changes in their health over time.
- To offer administrators powerful tools for:
  - ML model versioning
  - Feature schema configuration
  - CMS updates
  - Site management
  - Monitoring activities and logs
- To bridge the gap between the population and healthcare awareness through accessible, AI-powered insights.

1.2 Scope

The scope of the MediCare AI Disease Prediction System is to provide a centralized, web-based platform that allows users to evaluate disease risks using advanced machine learning predictions. It covers: user registration, login, input validation, prediction processing, probability calculation, CMS-based disease information, and historical data tracking.

The system operates as a fully online platform, accessible on any modern browser. It also includes a complete admin panel for configuring ML models, updating site content, managing disease pages, and monitoring internal system logs and prediction activity. The scope strictly focuses on delivering highly accurate predictions and comprehensive disease information without extending into external healthcare integrations, telemedicine services, or medical diagnostics.

1.2.1 In-Scope

The following items are included within the project boundaries:

- User registration, login, password reset, and profile management
- Prediction modules for: Diabetes, CVD, Kidney Disease, Breast Cancer, Depression, Obesity
- Dynamic form generation based on feature schemas
- Real-time input validation
- Instant prediction results with risk-level classification
- Detailed disease information, prevention guidelines, and treatment options
- Complete prediction history with filtering
- Admin panel features such as:
  - ML model upload
  - Feature schema configuration
  - Version management
  - CMS content updates

    - Site configuration (branding, links, contact info)
    - Communication management

1.2.2 Out-of-Scope

The following functionalities are explicitly not included in the current version:

- Integration with hospital EHR systems
- Wearable device integration (Fitbit, Apple Watch)
- Telemedicine or video consultation features
- Appointment booking
- Pharmacy or prescription systems
- Insurance integration
- Mobile apps (iOS/Android)
- Real-time health monitoring
- Multi-language translations
- Advanced analytics dashboards or ML training interfaces

1.3 Product Perspective

MediCare AI is designed as a standalone, independent, and self-contained system that does not rely on any external healthcare service or third-party integration for its core prediction operations. This independence ensures maximum data privacy, cost efficiency, and reliability.

The system revolves around a structured architecture that includes:

Presentation Layer

- HTML/CSS/JS
- Dynamic content rendered from CMS
- Form inputs and validation

Application Layer

- Django-based logic
- Routing, session handling, admin operations

ML Prediction Module

- Model loading
- Feature validation
- Prediction inference
- Probability extraction

Data Layer

- SQLite/PostgreSQL database
- Storage of user data, predictions, CMS content, settings

1.3.1 Integration within the System

The system integrates multiple modules into a unified workflow:

- **User Module → ML Prediction Module:**
  User inputs flow directly to ML engine for prediction processing.
- **User Module → CMS Module:**
  Users access detailed disease information retrieved from the CMS.
- **Admin Module → Database & ML Engine:**
  Admin actions such as content editing or model activation are instantly reflected across the system.
- **CMS Module → User Interface:**
  Updated content is displayed dynamically without code changes.

1.4 User Characteristics

The system is designed for multiple categories of users, each with different skill levels and responsibilities.

1.4.1 Admin

Admins possess moderate to advanced technical knowledge and handle:

- ML model uploads
- Schema configuration
- Content editing
- Monitoring logs
- Managing communication

1.4.2 Data Entry Operators

These users assist in updating disease content, FAQs, images, and other CMS sections using easy-to-use interfaces. Their technical expertise is basic to moderate.

1.4.3 Customers (General Users)

Users are health-conscious individuals with basic web skills who visit the platform to check their disease risk level, view probability scores, and learn about health conditions. They rely on straightforward UI and visualized results.

1.4.4 System Developers / IT Support

These users manage deployment, ML model placement, backend configurations, debugging, system logs, and server setup. They require advanced technical expertise in Django, ML processing, and databases.

1.5 Similar Apps and Systems / Literature Review

1.5.1 Existing Apps and Systems

The SRS compares MediCare AI with other healthcare platforms such as:

- Ada Health
- WebMD
- Babylon Health
- Google Health Studies
- K Health

File analysis shows that these platforms either use rule-based systems, conversational symptom checkers, or subscription-based consultation models — none provide multi-disease, parameter-based ML predictions integrated with CMS + admin model control like MediCare AI.

1.5.2 Literature Review

The system differentiates itself by presenting:

- Multi-disease ML predictions
- Probability scoring
- Admin-controlled model versioning
- Structured CMS health content

It addresses gaps found in traditional tools by adding transparency, model flexibility, and disease-specific prediction workflows.

1.6 Proposed Technologies

1.6.1 Programming Language

The system primarily uses **Python**, selected for its strong ecosystem of machine learning frameworks, numerical computation libraries, data processing tools, and compatibility with Django. Python enables efficient handling of ML models, JSON schemas, prediction logic, and error handling.

1.6.2 Frameworks and Libraries

Based on file content:

- **Django** — primary framework for backend logic, admin panel, API routing.
- **Bootstrap 5** — responsive UI framework.
- **NumPy, Pandas, Scikit-Learn** — used for data preprocessing, model I/O, and prediction.
- **TensorFlow/Keras / PyTorch / ONNX Runtime** — for loading various ML model formats.
- **Joblib** — efficient model serialization.

1.6.3 Database Management

The system uses:

- **SQLite** during development for simplicity
- **PostgreSQL** for production-level reliability

The database stores:

- User accounts
- Prediction history
- CMS content (disease info, prevention, treatment)
- Site settings
- Model configurations

1.6.4 Front-End Technologies

- HTML5, CSS3
- Bootstrap Grid System
- JavaScript ES6 for form validation, AJAX calls, and dynamic updates
- Rich text rendering for CMS content

1.6.5 Data Handling and Caching

- Joblib for model caching
- Schema-based validation
- Type conversion, ordering, and preprocessing based on JSON schema

1.6.6 Cloud Hosting and Deployment

- **Gunicorn** (WSGI server)
- **Nginx** reverse proxy
- Hosting platforms such as **Heroku**, **PythonAnywhere**, or **AWS**

2. Requirements

2.1 Functional Requirements

Functional requirements describe in detail **what the system must do**, how users will interact with the system, and how the internal components must respond. These requirements originate from the system's core functionality: prediction, content delivery, user interaction, and admin control.

2.1.1 Product Recommendation Generation

The MediCare AI system's primary function is **AI-driven disease prediction**. Even though the heading mentions "Recommendation Generation", your file specifies that the system generates **risk predictions** and provides **health recommendations** after prediction.

The system must enable users to input their medical or lifestyle parameters into dynamic input forms that are fully generated using the **ML feature schema** defined by the administrator. These forms automatically adapt to the selected disease, ensuring users always see the correct fields required for a specific model.

Key characteristics of this functionality include:

- The form dynamically adjusts based on field types such as **numeric inputs**, **categorical dropdowns**, and **boolean selections**.
- All fields are validated client-side and server-side to avoid incorrect data being sent to the ML model.
- Once validated, the system sends the parameters to the backend prediction API where the **active ML model** processes the input and generates a prediction.
- The prediction result includes:
  - **Risk classification** (At Risk / Not At Risk)
  - **Probability score** (0–100%)
  - **Interpretation** (Low, Medium, High risk)
  - **System-generated recommendations** based on risk level
  - **Disclaimer text**, ensuring that the system does not replace medical consultation

This functionality ensures that every user receives **accurate, personalized, and medically aligned risk insights**.

2.1.2 Feedback Collection

Your system supports a **feedback mechanism** allowing users to interact with the administrators. The system must allow users to submit feedback, ask medical-related questions, or report issues through a dedicated Contact or Feedback Form. This module ensures effective communication between the user and admin.

Core requirements include:

- Users must provide **name**, **email**, and **message** fields.
- Input validation ensures that the email is correctly formatted and the message field is meaningful.
- Each feedback entry is stored in the database and is forwarded to the Admin Panel.
- Administrators should be able to:
  - View all submitted messages
  - Filter messages (latest, unread, read)
  - Mark messages as reviewed
  - Respond externally through official email channels

The goal is to create a seamless feedback loop, enhancing user trust and helping admins gather user insights.

2.1.3 Admin Management Panel

This is one of the most critical components of your system — a fully integrated **administrative dashboard** for non-technical and technical users.

The system must provide administrators with full control over ML models, content, website information, user communication, and operational configuration.

Your file describes the following detailed admin capabilities:

(I) ML Model Management

Admins can:

- Upload ML model files in multiple formats: **.pkl**, **.joblib**, **.h5**, **.onnx**, **.pt**, **.pth**
- Store multiple versions of each disease prediction model
- Activate or deactivate models
- Maintain version history for tracing changes
- Ensure only one active model exists per disease

(II) Feature Schema Configuration

Admins can:

- Define input fields for each ML model through a **JSON schema**
- Set field types (numeric, categorical, boolean)
- Add dropdown options for categorical data
- Reorder features to match ML model requirements
- Quickly update schema when new model versions are uploaded

(III) CMS (Content Management System)

Admins can manage:

- Disease information pages
- Symptoms, causes, prevention, treatment
- Homepage banners, prediction cards, hero section
- FAQ and About page content
- Team members page
- Images, icons, and layout text

The CMS ensures that the entire platform can be updated without modifying any code.

(IV) Site Settings Management

Admins can modify:

- Application name
- Tagline
- Contact email and phone
- Footer text
- Social media links
- SEO tags

(V) User Communication Management

Admins can view and process all user messages submitted via the contact form.

Overall, the Admin Panel is responsible for **keeping the system accurate, updated, and well-maintained**.

2.1.4 Real-Time Recommendations

The system enhances user experience by providing **real-time, meaningful, AI-based guidance** immediately after prediction results are generated.

This module includes:

(I) Interactive Prediction Dashboard

- Displays **probability bar** with color-coded risk levels
- Shows **risk badges/labels** such as Low, Medium, High
- Provides **textual explanations** of what the result means
- Highlights **critical warnings** for high-risk users

(II) Automated Recommendations Based on Risk

- **Low Risk:** Healthy lifestyle encouragement
- **Medium Risk:** Suggest monitoring symptoms, adjusting lifestyle
- **High Risk:** Immediate consultation suggestion, warning messages

(III) Prediction History Management

Users can:

- Review past predictions
- Filter by date and disease
- Open detailed view of each record
- Understand long-term health trends

All predictions are stored with:

- Parameters entered
- Probability result
- Final risk label
- Timestamp
- Disease type

This ensures that the entire recommendation cycle is data-driven, transparent, and helpful to the user.

2.2 Non-Functional Requirements

Non-functional requirements define the **quality standards** of the system — performance, usability, security, efficiency, scalability, and maintainability.

Your file contains rich non-functional requirement details which are expanded below.

2.2.1 Performance Requirements

The system must be optimized for fast execution and smooth user interactions.

Key performance expectations include:

- **Prediction Response Time:**
  The ML engine should return results within **1 second**, ensuring real-time feedback.
- **Page Load Time:**
  Website pages should load within **2 seconds** under standard network conditions.
- **Concurrent Users:**
  The system must support **100+ simultaneous active sessions** without performance degradation.

- **Lightweight ML Models:**
  Serialization using joblib/ONNX ensures efficient loading times.

This ensures smooth experience for both regular users and administrators.

2.2.2 Availability

The system must be available and reliable throughout the day.

- **Uptime Requirement:**
  99.9% system uptime ensures minimal downtime.
- **CMS Content Sync:**
  Admin updates (disease content, site settings) must reflect on the frontend within **1 minute**.
- **Database Connectivity:**
  The system must maintain consistent and stable connections for prediction logging, content retrieval, and schema queries.

The availability requirement guarantees a stable system for both user prediction activities and admin updates.

2.2.3 Security Requirements

Security is a foundational aspect of the system considering sensitive health-related data.

The system ensures:

(I) User Authentication Security

- Encrypted passwords using Django’s hashing system
- Secure login session handling
- CSRF protection
- Strong password policies

(II) System Data Protection

- SQL injection prevention
- Input validation
- Access control at admin endpoints
- Session timeout to prevent unauthorized access

(III) ML Model Protection

- ML files stored safely in backend
- Only authenticated admins can upload or replace models
- No direct public URL access

This ensures that health data, models, and system configurations remain secure.

2.2.4 Usability Requirements

The system must be usable by individuals with minimal technical knowledge.

- **Responsive UI:**
  Based on Bootstrap; adapts to devices of all screen sizes.
- **Form Usability:**
  Clean, minimalistic forms; real-time validation; tooltips for field hints.
- **Accessibility:**
  Clear typography, color contrast, focus indicators, alt text for images.
- **Clarity in Results:**
  Color-coded risk bars and simple language explanations ensure users understand their health status instantly.

The goal is to make the platform intuitive and easy for everyone.

2.2.5 Maintainability

Maintainability ensures the system can grow and stay healthy over time:

- **Modular Code Architecture**
  ML module, CMS module, admin module, and UI components are separated.
- **Admin-Driven Updates**
  Non-technical admins can update content and ML models without developer support.
- **Version Control**
  ML models store version numbers; upgrading does not break previous configuration.
- **Extensibility**
  New diseases can be added by uploading a new ML model and schema — no coding required.

This structure ensures long-term system stability.

2.2.6 Compatibility

- **Browser Compatibility:**
  Works on Chrome, Firefox, Edge, Safari.
- **Device Compatibility:**
  Supports desktops, laptops, tablets, and smartphones.
- **Platform Independence:**
  Runs on Linux-based servers; UI loads on any OS with a modern browser.

Ensures maximum reach across user devices.

2.2.7 Data Integrity and Accuracy

The system ensures that all prediction results and input data are accurate, consistent, and complete.

- ML models must return reliable and predictable output.
- Input values must strictly match schema data types.
- Stored history must preserve full detail.
- Database must not lose or corrupt prediction logs.

Accuracy is crucial for health predictions; data integrity ensures output trustworthiness.

2.2.8 Backup and Recovery

- **Regular Database Backups:**
  Daily or weekly backups must be maintained to avoid data loss.
- **Data Recovery:**
  In case of crash or corruption, recovery time should be minimal to restore system operations quickly.
- **Disaster Prevention Measures:**
  Secure hosting, monitored uptime, predictable load handling.

This ensures that the system remains stable even in unexpected situations.

2.2.9 Environmental Requirements

- **Server OS:**
  Ubuntu 20.04+ recommended.
- **Runtime:**
  Python 3.10+
  Django with Gunicorn + Nginx
- **Hosting Platform:**
  Compatible with cloud environments like AWS, Heroku, DigitalOcean, PythonAnywhere.

These environmental specifications ensure optimal performance in production.

3. USE CASES AND FLOW OF PROCESSES

This chapter explains how all users interact with the system, how internal processes flow, and how each diagram represents system behavior. Every use case and diagram has been rewritten into readable, clean, professional paragraph form with placeholders for the images you already generated.

3.1 System Level Use Case Diagram

The System Level Use Case represents the three major actors and their interactions with the MediCare AI system: **Users**, **Admins**, and the **ML Prediction Engine**.

Users interact with the system by registering, logging in, entering health parameters, submitting prediction requests, reviewing disease information, accessing FAQs, submitting messages through the contact form, and viewing their previous prediction history.

Admins manage the system using the Admin Panel. They upload and activate ML models, configure the feature schema required for dynamic forms, update disease content, handle system-wide settings, maintain FAQs, view contact messages, and monitor system activity or prediction logs.

The ML Prediction Engine handles all prediction-related operations. It validates the user inputs, loads the correct model based on the selected disease, executes the prediction, calculates probability and risk level, and returns a well-structured response for display.

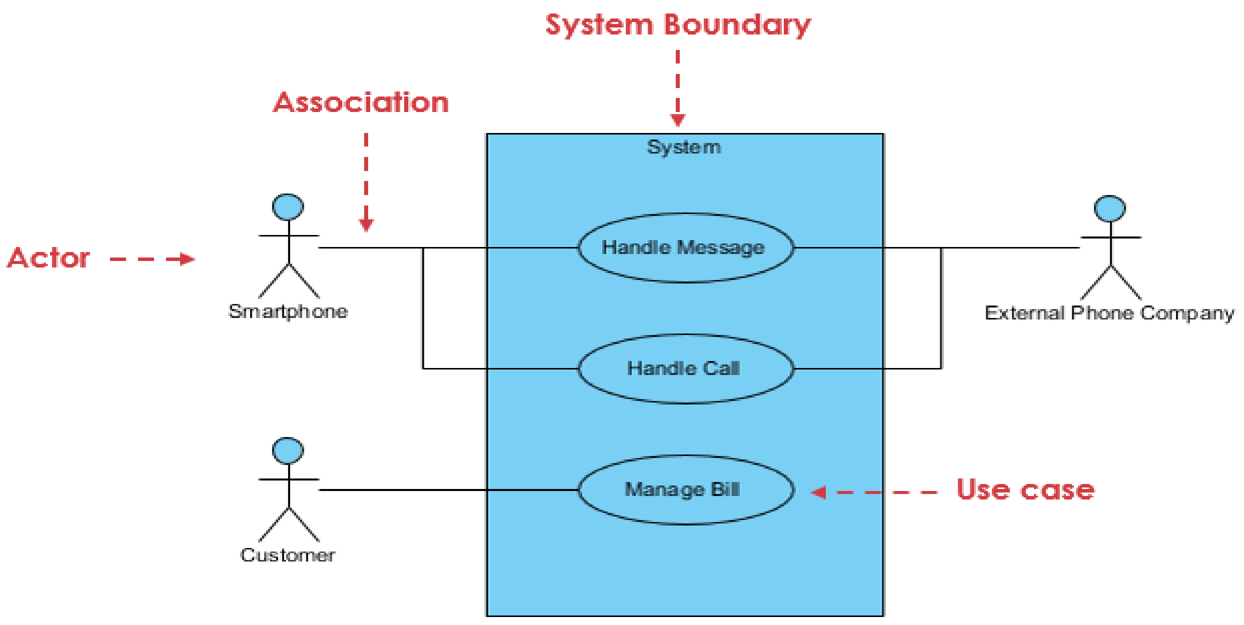


Figure 3.1 System Level Use Case Diagram

3.2 User Use Cases

This section explains how general users interact with the system in daily operations.

Use Case — User Registration

A new user creates an account by entering the required details. The system validates the input, stores the user profile securely, and enables personalized access to the platform including prediction history.

Use Case — User Login

Returning users log in with their credentials. After authentication, the system redirects them to a personalized dashboard where they can access prediction tools and previous history.

Use Case — Request Disease Prediction

Users choose a disease and fill out a dynamic, schema-driven form. The system validates all fields and forwards the input to the ML Prediction Engine. The engine calculates results including probability score, risk classification, and interpretation text, which the system then displays to the user.

Use Case — View Disease Information

Users browse educational pages that describe diseases, symptoms, causes, prevention tips, and treatment guidance. These pages are controlled by the admin-based CMS and retrieved dynamically.

Use Case — View Prediction History

Users can revisit previous predictions. The system retrieves all prediction data associated with the user and allows filtering by disease or date. Detailed prediction information is displayed for each entry.

Use Case — Submit Contact Form

Users can raise questions or share feedback. They enter their name, email, and message. The system validates the details and stores the message in the database for review by the admin.

3.3 Admin Use Cases

This section explains how administrators manage the system using the Admin Panel.

Use Case — Upload ML Model

Administrators upload machine learning model files in supported formats such as .pkl, .onnx, .h5, or .pt. The system verifies the file and stores it, making the model available for activation.

Use Case — Configure Feature Schema

Admins define and edit the feature schema responsible for form generation. They specify field types, names, input validation rules, and dropdown options. The system uses this schema to create dynamic prediction forms.

Use Case — Activate/Deactivate Model

Admins can activate one specific model version for each disease. When activated, the system immediately begins using that version for predictions. Older versions remain stored for reference.

Use Case — Update Disease Content

Admins manage all disease-related content including symptoms, prevention, causes, treatment guidelines, and supportive images. Updated content appears instantly on the user side.

Use Case — Manage Site Settings

Admins update global settings such as website name, tagline, contact details, footer text, and social media links. The system applies these settings across the platform.

Use Case — View Contact Messages

Admins review messages submitted by users through the contact form. Messages can be filtered, marked as read, and used for external follow-up.

3.4 System Use Cases

System use cases describe internal system-level operations that are not directly visible to end users.

Use Case — Load Active ML Model

Whenever a prediction is requested, the system automatically loads the active model for the selected disease from the stored model list.

Use Case — Validate Feature Schema

Before prediction, the system validates user inputs against the active schema to ensure type accuracy, required fields, and correct ordering.

Use Case — Generate Prediction Response

The system sends validated inputs to the ML engine, receives the prediction output, calculates risk and probability, formats the result, stores it in history, and displays it to the user.

3.5 System Architecture Diagram

The architecture shows how different layers interact within the system. Users interact with the **Frontend Interface** built with HTML, CSS, JavaScript, and Bootstrap. Their requests travel to the **Django Application Layer**, which handles routing, logic, validation, and invoking the ML Prediction Engine. The Admin Panel is integrated into this layer and interacts with the **CMS Module**.
All content, prediction history, user records, and model metadata are stored in the **Database Layer**.

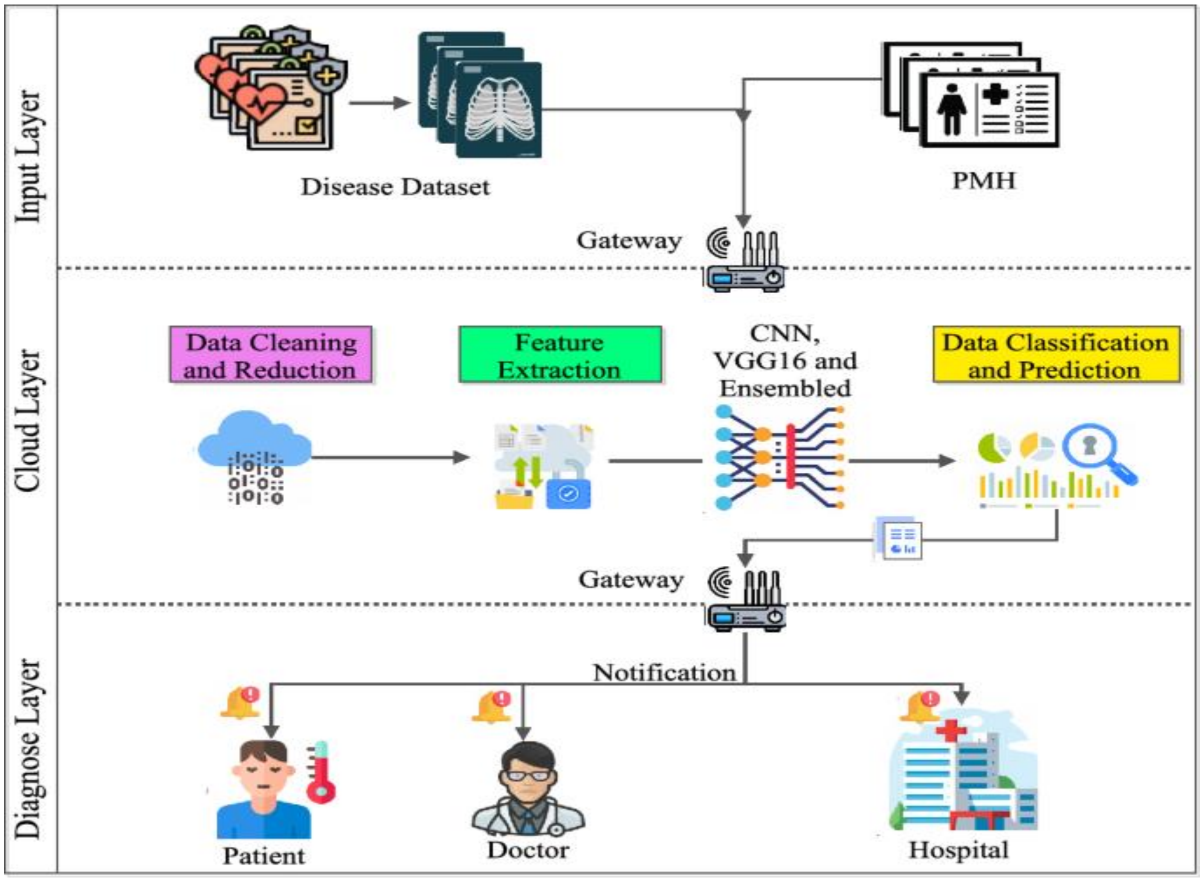


Figure 3.5 System Architecture Diagram

3.6 Entity Relationship Diagram

The ERD shows the relationships between system entities:

- A **User** can generate many **Predictions**.
- An **Admin** manages multiple **ML Model Versions**.
- Each **Disease Info** entry contains several **Content Sections** such as symptoms and prevention.
- **Contact Messages** store user inquiries.

The ERD ensures structured, relational data storage.

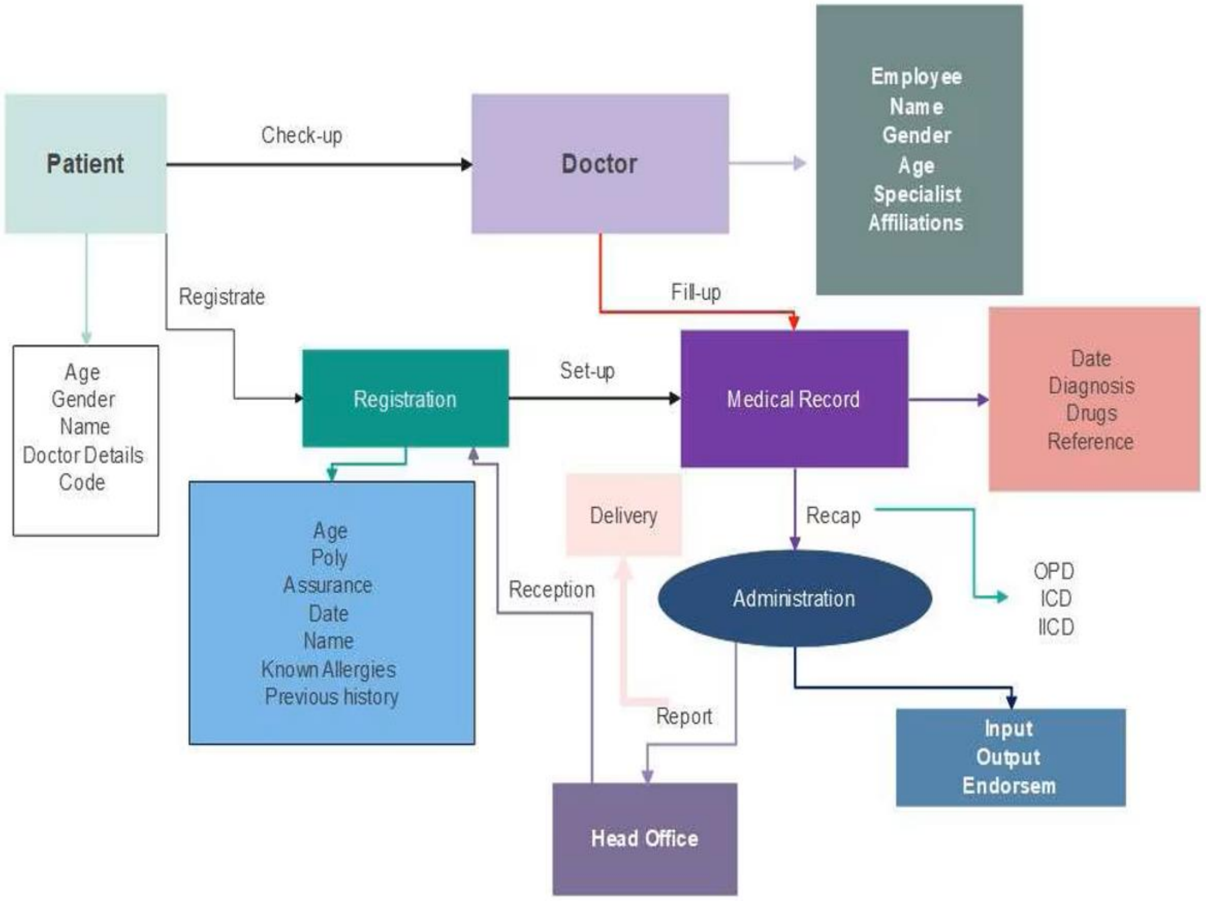
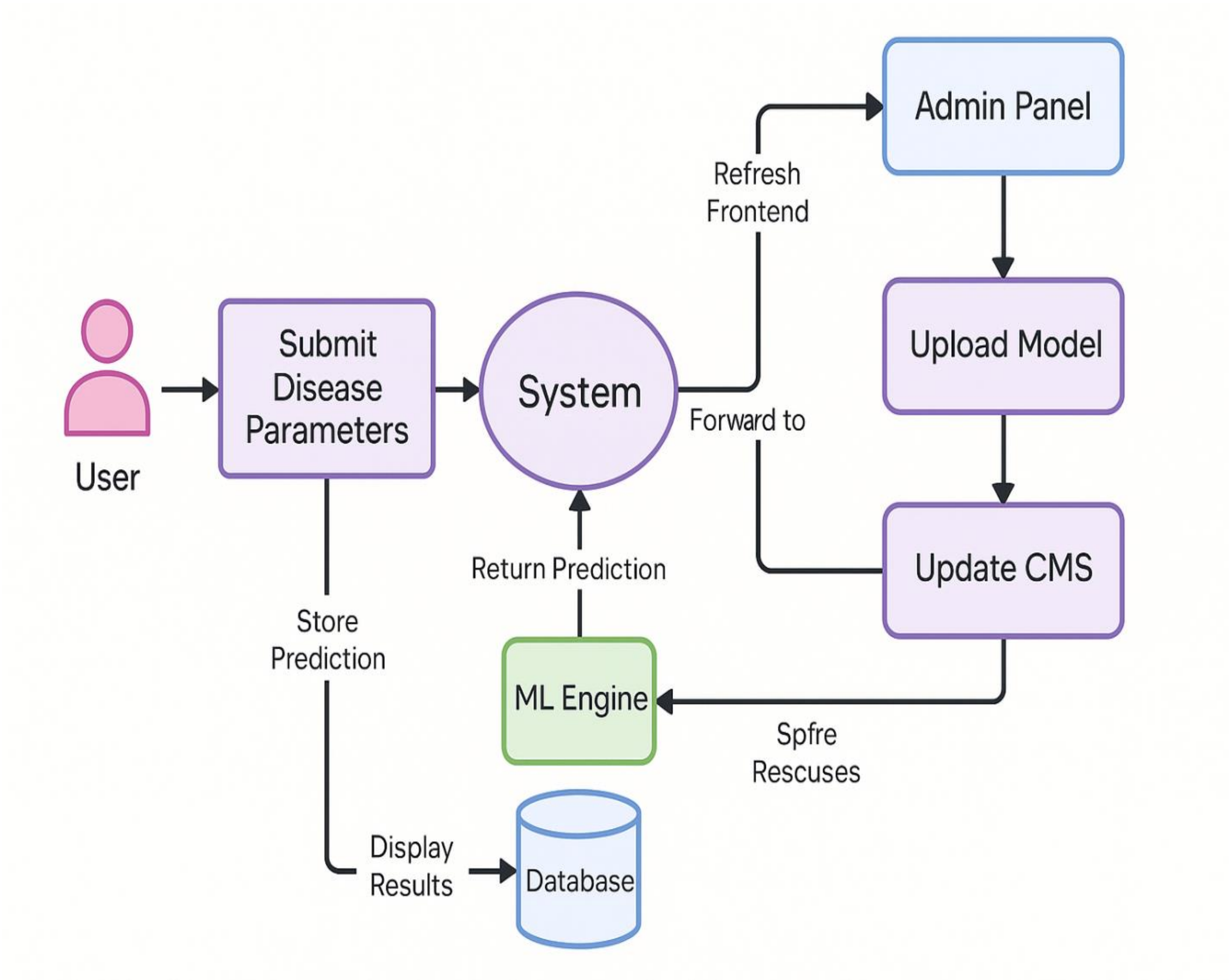


Figure 3.6 Entity Relationship Diagram

3.7 Data Flow Diagrams (DFD)

The DFD explains how data moves through the system:

- Users submit parameters → the system validates them → forwards to the ML Engine.
- ML Engine processes data → returns prediction → system stores history → displays results.
- Admin actions (content changes, model uploads) go through the Admin Panel → stored into the database → reflected on the frontend.



3.7 Data Flow Diagrams (DFD

3.8 Sequence Diagrams

The sequence diagram shows the timeline of interactions:

1. User opens prediction page → System loads schema.
2. User submits form → API validates → ML Engine predicts.
3. System formats and returns results → UI displays them to the user.

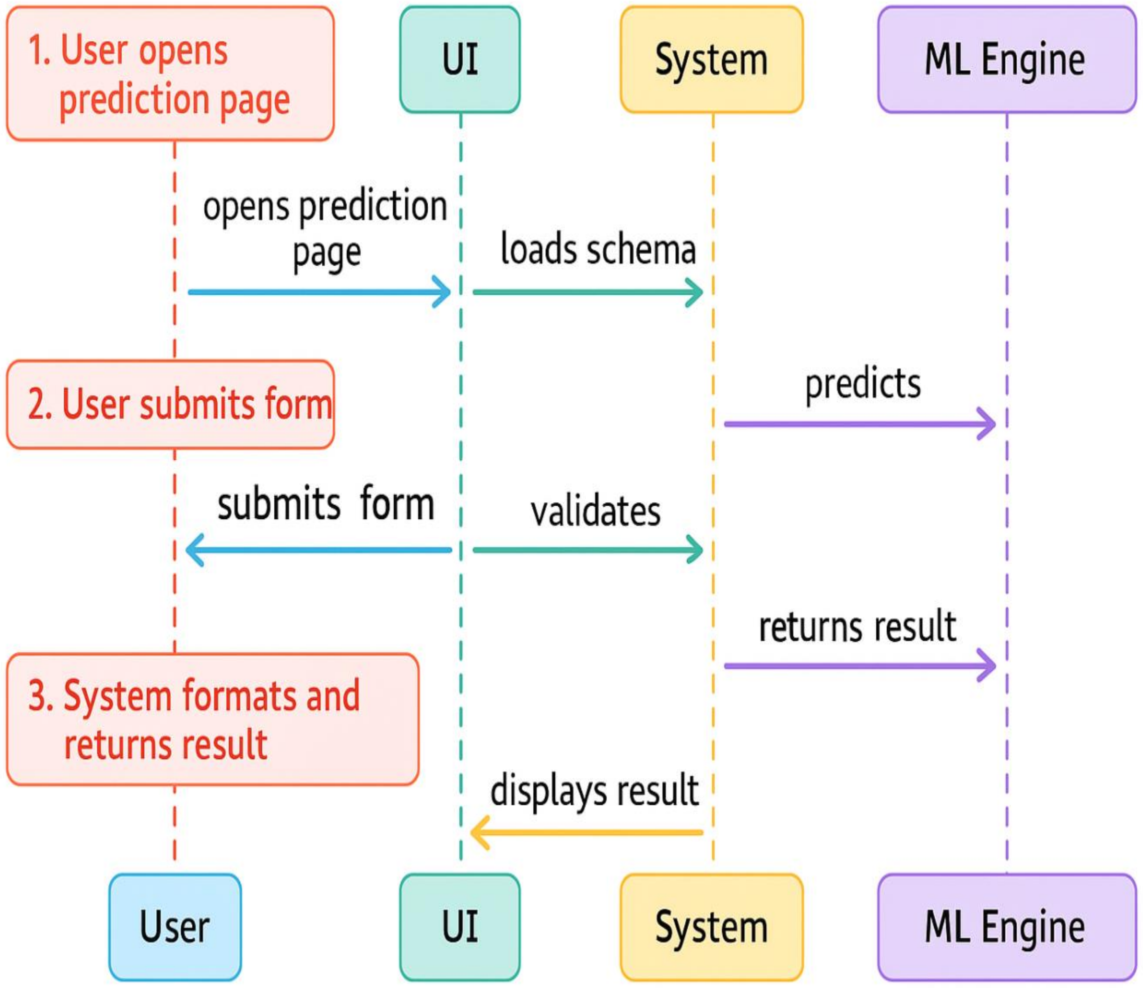


Figure 3.8 Sequence Diagrams

3.9 Class Diagrams

The class structure includes:

- User
- Prediction
- MLModelVersion
- DiseaseInfo
- ContentSection
- ContactMessage

These classes reflect system data structures used throughout the application.

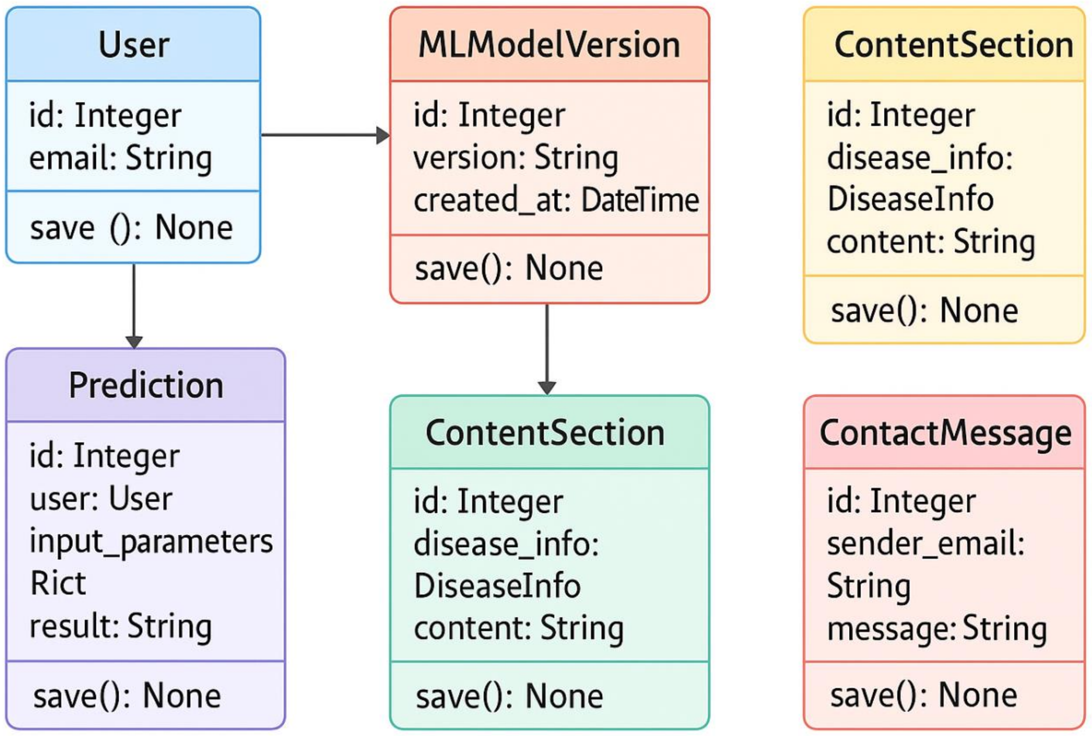


Figure 3.9 Class Diagrams

3.10 Activity Diagrams

The user activity diagram outlines:
Opening the form → entering data → validation → prediction → viewing results → saving history → end of flow.

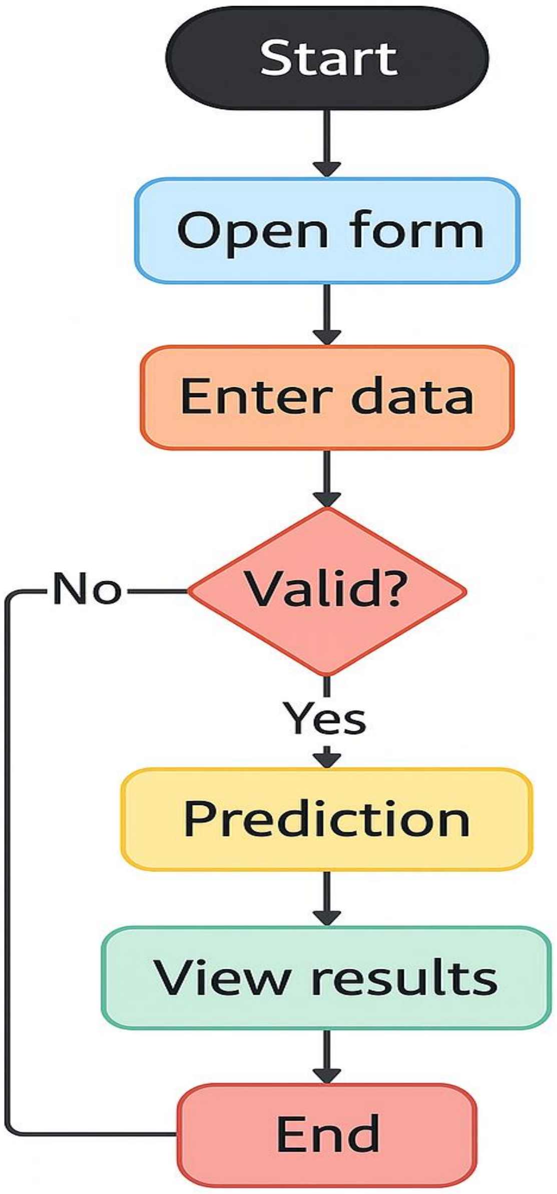


Figure 3.10 Activity Diagrams

3.11 Admin Activity Diagram

The admin activity flow includes logging in → editing CMS or settings → uploading models → saving updates → system applying changes across the platform.

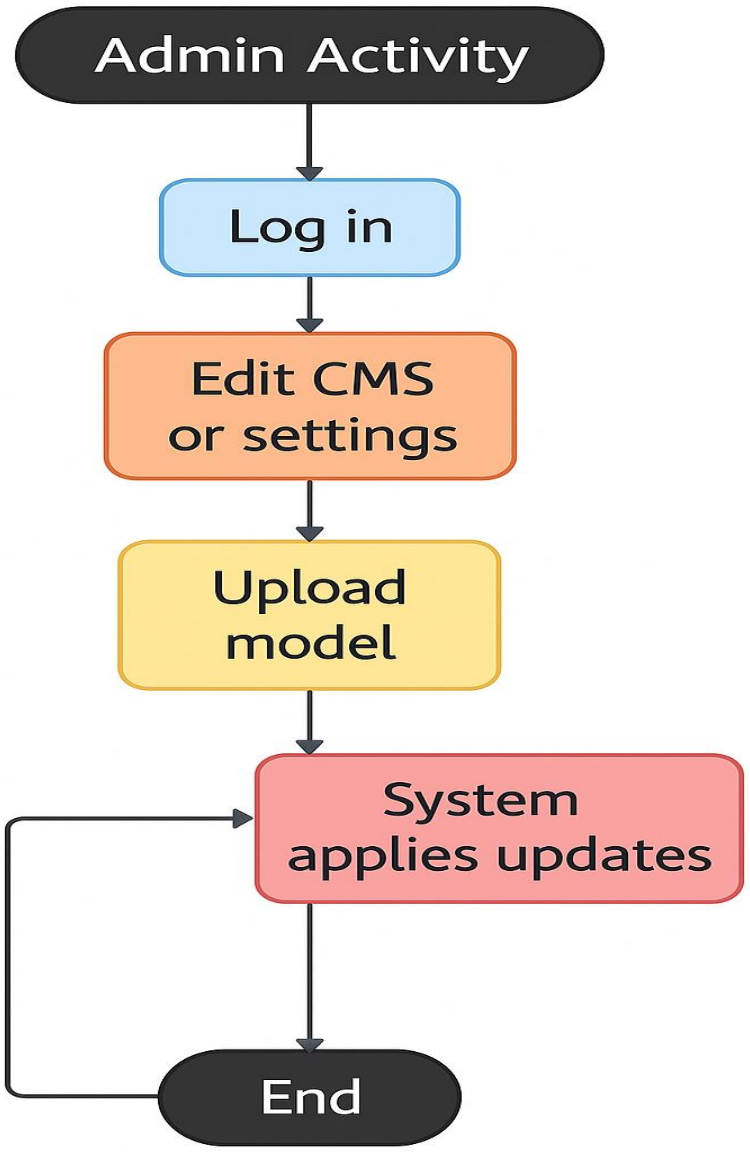
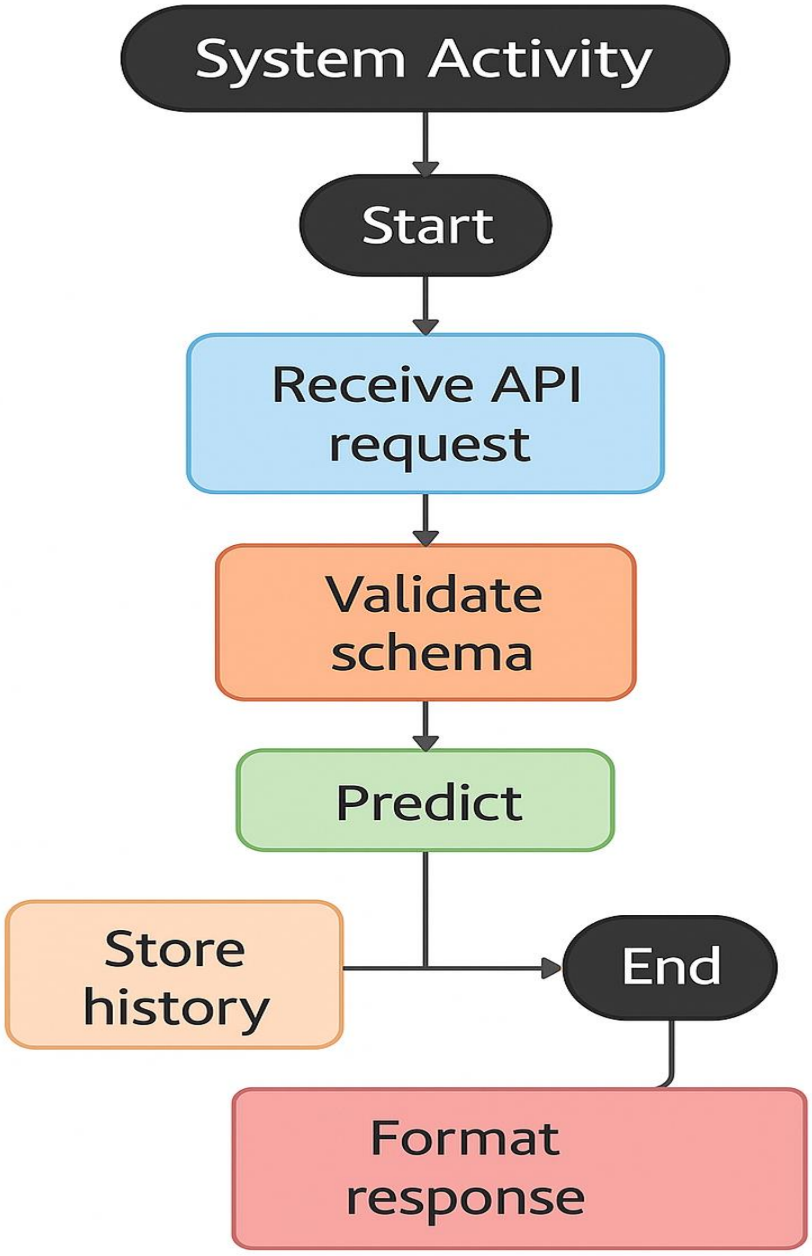


Figure 3.11 Admin Activity Diagram

3.12 System Activity Diagram

The system activity diagram shows the internal operational cycle:
Receiving API request → validating schema → loading model → predicting → formatting response → storing history → sending results back.



3.12 System Activity Diagram

4.1 References

4.1 Technical Documentation

4.1.1 Django Documentation

- Official Django web framework documentation
- Django is a high-level Python web framework that enables rapid, secure, and scalable web application development using an MVC-style architecture.
- URL: https://docs.djangoproject.com/
- Version: 4.2 / 5.0

4.1.2 Python Documentation

- Official Python programming language documentation
- URL: https://docs.python.org/3/
- Version: 3.10+

4.1.3 Scikit-learn Documentation

- Python machine learning library documentation
- Scikit-learn is a Python machine learning library that provides efficient tools for data preprocessing, classification, regression, clustering, and model evaluation built on NumPy, SciPy, and Matplotlib.
- URL: https://scikit-learn.org/stable/documentation.html
- Version: 1.3+

4.1.4 TensorFlow / Keras Documentation

- Deep learning framework API documentation
- **TensorFlow/Keras is an open-source deep learning framework that provides scalable tools and APIs for building, training, and deploying neural network models across CPUs, GPUs, and distributed systems.**
- URL: https://www.tensorflow.org/api_docs
- Version: 2.13+

4.1.5 NumPy Documentation

- Numerical computing library documentation
- NumPy is a fundamental Python library for high-performance numerical computing that provides support for large multidimensional arrays, matrices, and optimized mathematical operations.
- URL: https://numpy.org/doc/
- Version: 1.24+

4.1.6 Bootstrap Documentation

- CSS framework for responsive UI
- **Bootstrap is a responsive front-end framework that provides pre-built CSS and JavaScript components for quickly designing modern, mobile-first web interfaces.**
- URL: https://getbootstrap.com/docs/5.0/
- Version: 5.3

4.1.7 Font Awesome Documentation

- Icon library documentation
- URL: https://fontawesome.com/docs
- Version: 6.4

4.2 Medical References

4.2.1 World Health Organization (WHO)

- Global health information and statistics
- URL: https://www.who.int/

4.2.2 American Diabetes Association

- Diabetes research & health guidelines
- URL: https://www.diabetes.org/

4.2.3 American Heart Association

- Cardiovascular disease guidelines & prevention
- URL: https://www.heart.org/

4.2.4 National Kidney Foundation

- Kidney disease information
- URL: https://www.kidney.org/

4.2.5 American Cancer Society

- Cancer awareness, screening & treatment
- URL: https://www.cancer.org/

4.2.6 National Institute of Mental Health (NIMH)

- Mental health research
- URL: https://www.nimh.nih.gov/

4.2.7 Centers for Disease Control and Prevention (CDC)

- Public health and disease prevention
- URL: https://www.cdc.gov/

4.3 Similar Systems References

4.3.1 Ada Health

- AI-powered health assessment system
- URL: https://ada.com/

4.3.2 WebMD

- Health information & symptom checker
- URL: https://www.webmd.com/

4.3.3 Babylon Health

- Digital health & medical consultation platform
- URL: https://www.babylonhealth.com/

4.3.4 K Health

- AI-powered health assessment platform
- URL: https://khealth.com/

4.4 Standards and Guidelines

4.4.1 IEEE 830-1998

- Software Requirements Specification standard

4.4.2 WCAG 2.1

- Web Content Accessibility Guidelines
- URL: https://www.w3.org/WAI/WCAG21/quickref/

4.4.3 OWASP Top 10

- Web application security risk guidelines
- URL: https://owasp.org/www-project-top-ten/

4.4.4 PEP 8

- Python Style Guide
- URL: https://peps.python.org/pep-0008/

4.5 Research Papers

4.5.1 Machine Learning in Healthcare

- ML-based healthcare research
- Sources: PubMed, IEEE Xplore, arXiv