

1. Introduction

1. Introduction

1.1 Institute Profile

Institute of Management and Career Courses (IMCC) is a premier Management Institute, established in 1983 by Maharashtra Education Society (MES) for providing quality education and technical expertise at the Post Graduation Level in the Fields of Computers and Management. The Institute is recognized by SPPU under Section 46 of Pune University Act, 1974 and Section 85 of Maharashtra University Act, 1994 and Approved by AICTE New Delhi to conduct MCA and MBA programmes. The Institute is located at 131, Mayur Colony, Kothrud, Pune-411038 having 30,000 sq. ft-built area & totally independent campus. IMCC is recognised as a Ph.D. Research Centre under the Faculty of Management, SPPU. IMCC has 38 years standing & it is well-known for its conducive educational atmosphere. IMCC focuses on the all-round development of its students. Thus, apart from excellence in academics, students develop their inner potential by way of active participation in co-curricular & extra-curricular activities. IMCC has developed excellent rapport with Industry by way of Guest Lectures, Seminars, Workshops, Industrial Visits & Placements. The main motto of the Institute is to instill the concepts of total personality development in the students. The emphasis is laid on 'Teacher Disciple Relationship in place of Boss Subordinate relationship at their assignments. Institute produces the new breed of professionals, who's deeds will speak and there

could be no requirement of pomposity. MES's Institute of Management & Career Courses (IMCC) is dedicated to empowering students through high-quality postgraduate education. We believe in fostering professionals who are not only knowledgeable but also action-oriented, driven by the motto "Facta Non Verba" (Actions Speak Louder Than Words). Our curriculum emphasizes total personality development, fostering a unique "Teacher-Disciple Relationship" that builds strong mentoring bonds.

1.2 Abstract:

The proposed project is a web-based healthcare management system designed to facilitate user registration, profile management, health issue tracking, disease prediction, and interaction between users and doctors. The system consists of three main modules: User Signup, Doctor Signup, and Login.

1. User Signup:

Users can register by providing personal details such as Aadhar number, name, date of birth, gender, mobile number, email address, physical address, health insurance number, and password. Upon successful registration, users can access various functionalities including profile management and health issue tracking.

2. User Module:

Once logged in, users have access to their profiles where they can view and update personal information. They can also add current health issues including disease name, symptoms, and medicines. Additionally, users can maintain their medical history and utilize the system's disease prediction feature. Users input five symptoms, and the system predicts the likelihood of having a particular disease along with suggested medicines, doctors, and diet plans based on the prediction.

3. Doctor Signup:

Doctors can sign up for the system to access patient data and perform various actions. They can search for patients and view their health card data, including current disease names, symptoms, medicines, and detection dates. Doctors can also perform tasks similar to those available to patients, facilitating seamless communication and collaboration between users and healthcare providers.

The proposed system aims to streamline healthcare management by providing a user-friendly interface for users to track their health issues, predict diseases, and receive personalized medical recommendations. Moreover, it offers doctors access to comprehensive patient data for efficient diagnosis and treatment planning. This project contributes to improving healthcare accessibility and enhancing patient-doctor communication in a digital environment.

1.3 Existing System and Need for System

The existing healthcare management systems may lack certain features such as comprehensive user profiles, disease prediction capabilities, and seamless interaction between users and healthcare providers. Traditional systems might not provide personalized medical recommendations based on user data or offer predictive analytics for disease diagnosis.

Need for System:

1. **Comprehensive User Profiles:** The current systems may not offer detailed user profiles encompassing personal information, medical history, and current health issues. There is a need for a system that allows users to maintain and update their profiles easily.
2. **Disease Prediction:** Traditional systems may lack predictive analytics for disease diagnosis based on user input. Introducing a feature that predicts diseases based on symptoms can aid in early detection and prevention of illnesses.
3. **Seamless Interaction:** Existing systems may not facilitate efficient communication between users and healthcare providers. Implementing a system that enables users to consult with doctors, receive medical advice, and access personalized treatment plans can enhance healthcare accessibility and quality.

4. Personalized Medical Recommendations: The current systems may not offer personalized medical recommendations tailored to individual user profiles and predicted diseases. There is a need for a system that provides users with customized treatment plans, including medicines, suggested doctors, and dietary advice.

5. Improved Patient Experience: Self-service signup: Patients can register conveniently without relying on administrative assistance. Centralized health record: All health information (profile, medical history, current health issues) is readily accessible in a single location. Disease prediction: Early detection of potential health concerns empowers patients to take proactive steps towards better health.

1.4 Scope of System

The scope of the proposed healthcare management system encompasses several key features and functionalities to meet the needs of users and healthcare providers. The system aims to provide a comprehensive platform for managing user health information, facilitating disease prediction, enabling seamless interaction between users and doctors, and offering personalized medical recommendations. The scope includes but is not limited to the following components:

1. User Registration and Profile Management:

- Users can register with the system by providing necessary details such as personal and health insurance information.
- Users can log current health issues including disease names, symptoms, and prescribed medicines.
- The system allows users to track the progression of health issues over time and maintain a record of past health conditions.

2. Disease Prediction:

- Users can input symptoms, and the system utilizes predictive analytics to assess the likelihood of specific diseases based on the input.
- The system provides users with the percentage probability of having the predicted disease and offers suggestions for further actions.

3. Personalized Medical Recommendations:

- Based on user profiles, health issues, and predicted diseases, the system generates personalized medical recommendations. Recommendations may include prescribed medicines, suggested doctors for consultation, and dietary advice tailored to individual health needs.

1.5 Operating Environment - Hardware and Software

Hardware Requirements:

1. Server:

- Processor: Multi-core processor (e.g., Intel Core i5 or equivalent)
- RAM: Minimum 4 GB (8 GB or more recommended for better performance)
- Storage: Sufficient storage space for database and application files (SSD recommended for faster data access)
- Network: Stable internet connection for hosting the application and facilitating user interactions

2. Client Devices:

- Personal Computers, Laptops, Tablets, or Smartphones with modern web browsers (e.g., Google Chrome, Mozilla Firefox, Safari, Microsoft Edge) for accessing the web-based application.

Software Requirements:

1. Server-Side:

- Operating System: Linux (e.g., Ubuntu, CentOS), Windows Server, or macOS Server
- Database: PostgreSQL, MySQL, SQLite (for development)
- Backend Framework: Django (Python-based web framework)
- Python: Latest version of Python (e.g., Python 3.8 or higher)
- Virtual Environment: Virtualenv or Anaconda (recommended for managing Python dependencies)

- Other Dependencies: Django REST framework, NumPy, pandas, scikit-learn (for predictive analytics), etc.

2. Client-Side:

- Operating System: Windows, macOS, Linux, Android, iOS
- Web Browser: Latest versions of Google Chrome, Mozilla Firefox, Safari, Microsoft Edge, etc.

3. Development Tools (Optional for Development and Deployment):

- Integrated Development Environment (IDE): PyCharm, Visual Studio Code, Sublime Text, etc.
- Version Control: Git

1.6 Brief Description of Technology Used

1.6.1 Operating Systems Used:

- Windows:
- Windows Server: The system can be deployed on Windows Server operating system for hosting the web application and managing server-side operations. Windows Server provides a robust platform for running web servers, databases, and other necessary services required for the healthcare management system.

1.6.2 Technology Stack for Web Development and Database:

1) Python Django for Web Development:

- Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It follows the model-template-view (MTV) architectural pattern, which facilitates the development of scalable and maintainable web applications. Django provides built-in features for handling authentication, URL routing, form processing, and database management, making it well-suited for building complex web applications like the healthcare management system.

2) Frontend Technologies: HTML, CSS, JavaScript

- HTML is the standard markup language used to structure content on web pages. HTML elements are used to define the structure of a webpage, including headings, paragraphs, lists, forms, and more.

- CSS (Cascading Style Sheets): CSS is used for styling HTML elements, controlling layout, colors, fonts, and other visual aspects of a webpage. CSS rules consist of selectors and declarations, defining how selected elements should be styled.
- JavaScript: JavaScript is a high-level, interpreted programming language that adds interactivity and dynamic behavior to web pages. JavaScript can manipulate the HTML DOM (Document Object Model), handle user events, perform form validation, make AJAX requests, and more.

3) Asynchronous Task Queue: Celery with Redis

Celery: Celery is a distributed task queue that is used for asynchronous processing in web applications. Celery supports distributed task execution, task scheduling, periodic tasks, retries, and monitoring.

Redis: Redis is an open-source, in-memory data structure store that acts as a message broker for Celery in your Django project. Redis provides fast read and write operations, data persistence, pub/sub messaging, and support for various data types.

4) Machine Learning Models for Prediction : Machine learning models are used for predicting diseases based on symptoms input by users.

- Data Collection and Preparation: Historical medical data is collected and preprocessed to train the machine learning models.
- Model Training: Machine learning algorithms such as decision

trees, random forests, support vector machines, or deep learning models are trained on the prepared dataset.

- Model Evaluation: The trained models are evaluated using metrics such as accuracy, precision, recall, and F1 score to assess their performance.
- Deployment: Once trained and evaluated, the models are deployed in the application to make predictions based on user input.

5) Django Database: Django supports various relational databases, including SQLite (default), PostgreSQL, MySQL, and Oracle.

ORM Integration: Django's ORM abstracts database interactions, allowing developers to work with database models using Python classes and methods.

Data Integrity: Django enforces data integrity constraints such as unique constraints, foreign key relationships, and field constraints.

Query Optimization: Django provides query optimization techniques to improve database performance, including query caching, indexing, and prefetching.

2. Proposed System

2. Proposed System

2.1 Study of Similar Systems

Several existing healthcare management systems offer functionalities similar to those proposed in the project:

1. Epic Systems' Electronic Health Records (EHR) platform: Epic's EHR system provides comprehensive patient profiles, health issue tracking, and interactive features for both patients and healthcare providers. It offers predictive analytics tools for disease management and personalized treatment recommendations.
2. Cerner's Health Information Exchange (HIE) platform: Cerner's HIE platform facilitates secure exchange of health information among different healthcare organizations. It includes features for patient registration, profile management, and disease prediction using advanced analytics.
3. IBM Watson Health: IBM Watson Health offers solutions for personalized medicine, disease prediction, and patient engagement. Its cognitive computing capabilities enable data-driven insights for healthcare providers and patients, supporting personalized treatment plans and predictive analytics.

Research papers and academic studies also provide insights into similar projects:

1. "Predictive Analytics in Healthcare: A Review of Current Trends and Future Directions" by Li et al. (2018): This paper reviews the application of predictive analytics in healthcare, including disease prediction models and personalized medicine approaches.

2.2 Feasibility Study

Technical Feasibility:

- a. Platform: The system will be developed using Python Django framework which is well-suited for web applications and provides robust security features.
- b. Database: Utilize a relational database management system (RDBMS) like PostgreSQL to store user and health-related data securely.
- c. APIs: Integration of APIs for disease prediction and suggesting doctors based on symptoms can be achieved using third-party services like Infermedica API.
- d. Scalability: The system is designed to handle a large number of users and data transactions efficiently.
- e. Security: Implementation of security measures such as encryption, secure authentication, and role-based access control to protect sensitive user data.

Operational Feasibility:

- a. User Adoption: The system offers a user-friendly interface for both users and doctors, making it easy for them to adopt and utilize the platform.
- b. Training: Minimal training will be required for users and doctors to understand the functionalities of the system due to its intuitive design.
- c. Integration: The system can be integrated with existing healthcare systems to streamline data sharing and interoperability.

Economic Feasibility:

- a. Development and Maintenance Costs: The development cost includes expenses of software development, database setup, API integration, and testing along with server hosting.

2.3 Objectives of Proposed System

1. **Efficient Health Information Management:** Create a centralized platform for users to manage their health information, including personal details, medical history, and current health issues, ensuring easy access and updates.
2. **Disease Prediction and Prevention:** Implement predictive analytics to analyze user-input symptoms and predict the likelihood of specific diseases. This aims to facilitate early detection and preventive measures.
3. **Enhanced Patient-Doctor Communication:** Enable seamless interaction between users and healthcare providers (doctors) for consultations, medical advice, and treatment recommendations, improving patient-doctor communication and collaboration.
4. **Personalized Medical Recommendations:** Provide personalized treatment recommendations based on user profiles, health issues, and predicted diseases, including prescribed medicines, suggested doctors, and dietary advice tailored to individual health needs.
5. **Data Security and Privacy:** Ensure secure storage, retrieval, and management of user health data in compliance with privacy regulations (e.g., GDPR, HIPAA), prioritizing data security and confidentiality.

2.4 Users of System

1) Patients:

- Individuals seeking healthcare services and managing their own health information.
- Patients can register, create profiles, input health data, track health issues, receive disease predictions, and access personalized medical recommendations.

2) Doctors/Healthcare Providers:

- Healthcare professionals responsible for diagnosing and treating patients.
- Doctors can access patient profiles, view health information, provide medical advice, prescribe treatments, and communicate with patients through the system.

3. Analysis and Design

3. Analysis and Design

3.1 System Requirements

Functional Requirements:

1. User Registration and Authentication:

- Users should be able to register with the system using their personal information and create unique user accounts.
- The system should authenticate users securely using credentials such as username/email and password.

2. User Profile Management:

- Users should be able to create, view, update, and delete their profiles.
- Profile information should include personal details, medical history, and contact information.

3. Health Information Management:

- Users should be able to input, view, update, and delete their health information, including current health issues, medications, and allergies.
- The system should support the management of health records, including lab results, diagnostic reports, and treatment history.

4. Disease Prediction and Analysis:

- The system should provide a disease prediction feature based on user-input symptoms and predictive analytics algorithms.
- Users should be able to view the probability of having specific diseases and receive suggested actions or recommendations.

5. Patient-Doctor Communication:

- Users should be able to communicate with healthcare providers (doctors) through messaging or virtual consultations.
- Doctors should have access to patient profiles and health information for effective communication and consultation.

6. Personalized Medical Recommendations:

- The system should provide personalized medical recommendations based on user profiles, health issues, and predicted diseases.
- Recommendations may include prescribed medications, suggested doctors, and dietary advice tailored to individual health needs.

7. Data Security and Privacy:

- The system should ensure the security and privacy of user health information through encryption, access control, and compliance with relevant regulations (e.g., GDPR, HIPAA).
- User consent should be obtained for data collection, processing, and sharing.

Non-Functional Requirements:

1. Performance:

- The system provides fast response times for user interactions, queries, and data retrieval.
- It is capable of handling multiple concurrent users without significant performance degradation.

2. Scalability:

- The system is scalable to accommodate an increasing number of users, data volumes, and system loads over time.
- It support horizontal and vertical scaling to handle growing demands.

3. Reliability:

- The system is reliable and available 24/7 with minimal downtime for maintenance or upgrades.

4. Usability:

- The system is having a user-friendly interface with intuitive navigation and clear instructions.
- It is accessible to users with varying levels of technical proficiency and disabilities.

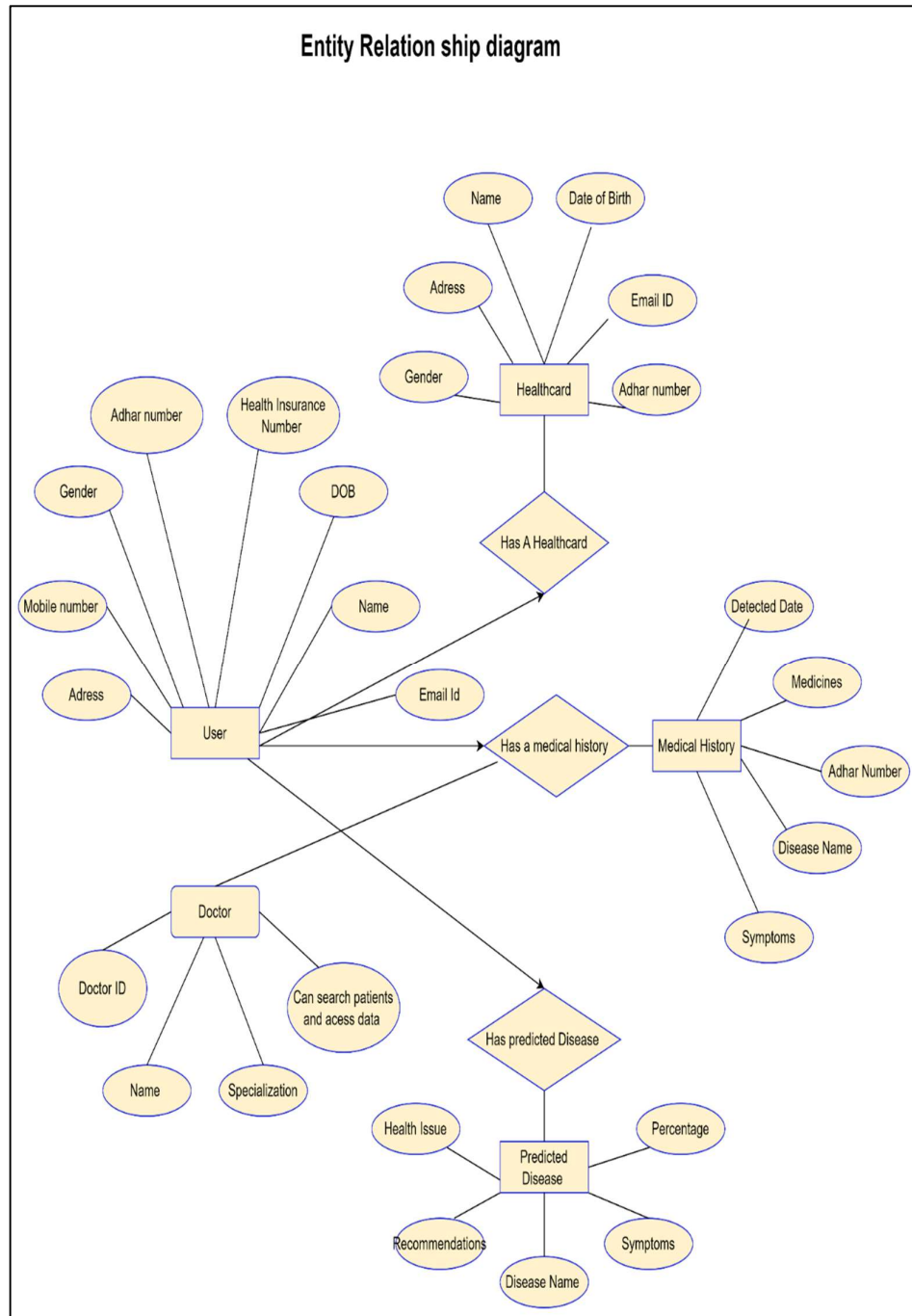
5. Compatibility:

- The system is compatible with a wide range of devices, OS , and web browsers.

6. Security:

- The system implements robust security measures to protect against unauthorized access,

3.2 Entity Relationship Diagram



3.3 Table Structure

1) User Table (users)

Field Name	Data Type	Size (Optional)	Constraints
id	INT	PRIMARY KEY	Unique identifier for the user (auto-increment)
aadhar_number	VARCHAR (12)	NOT NULL UNIQUE	Unique 12-digit Aadhaar number (if applicable)
name	VARCHAR (255)	NOT NULL	User's full name
date_of_birth	DATE	NOT NULL	User's date of birth
gender	ENUM ('Male', 'Female', 'Other')	NOT NULL	User's gender
mobile_number	VARCHAR (15)	NOT NULL UNIQUE	Unique mobile number
email_id	VARCHAR (255)	NOT NULL UNIQUE	Unique email address
address	TEXT		User's address

health_insurance_no	VARCHAR (255)		Health insurance number (if applicable)
password_hash	VARCHAR (255)	NOT NULL	Securely hashed password using a strong hashing algorithm
created_at	DATETIME	NOT NULL DEFAULT CURRENT_TIMESTAMP	Date and time the user account was created

2. Doctor Table (doctors)

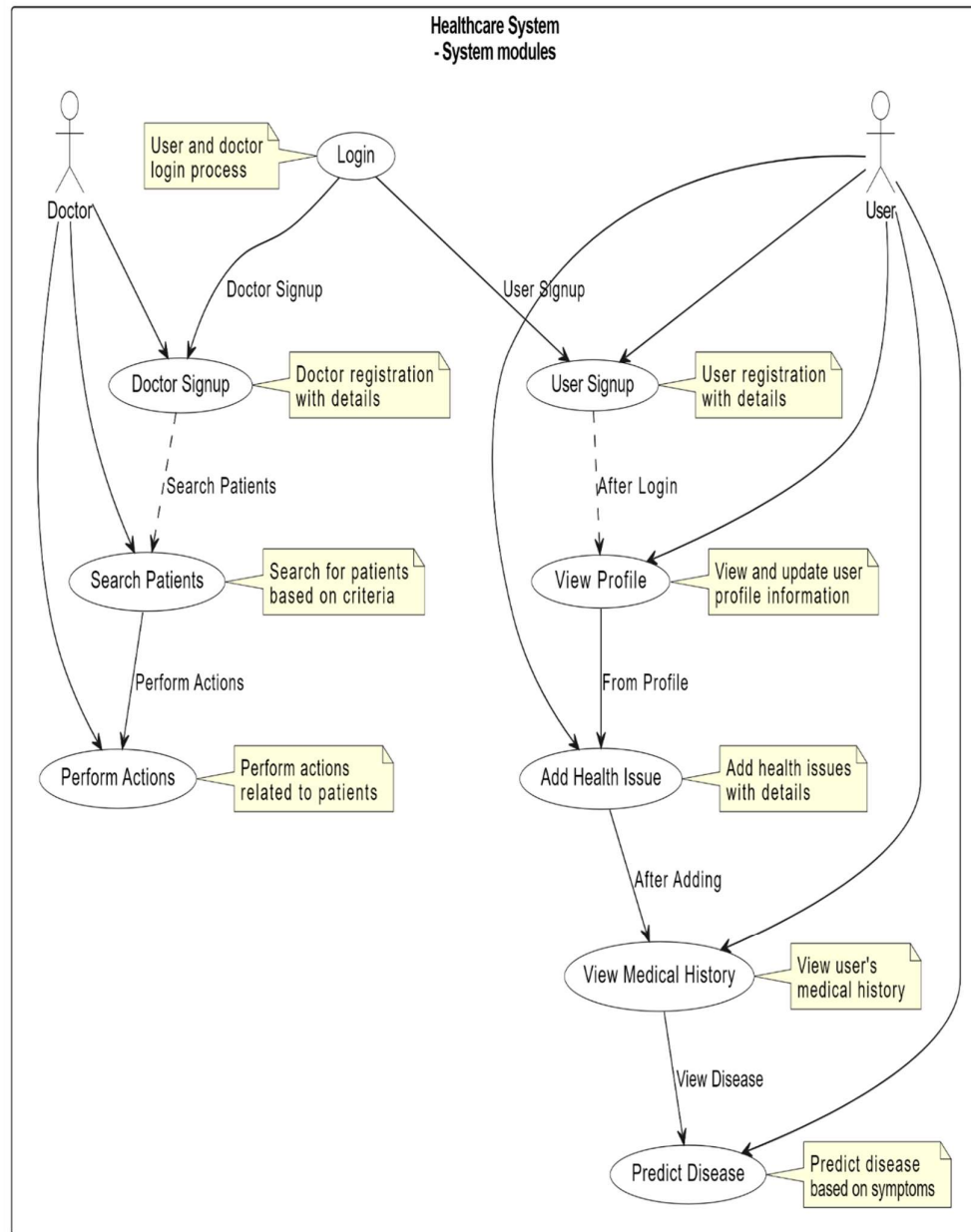
Field Name	Data Type	Size (Optional)	Constraints
id	INT	PRIMARY KEY	Unique identifier for the doctor (auto-increment)
name	VARCHAR (255)	NOT NULL	Doctor's full name
specialization	VARCHAR (255)		Doctor's specialization(s)
registration_number	VARCHAR (255)	NOT NULL UNIQUE	Unique doctor registration number
mobile_numb	VARCHAR	NOT NULL	Unique mobile number

er	(15)	UNIQUE	
email_id	VARCHAR (255)	NOT NULL UNIQUE	Unique email address
password_hash	VARCHAR (255)	NOT NULL	Securely hashed password using a strong hashing algorithm
created_at	DATETIME	NOT NULL DEFAULT CURRENT_TIMESTAMP	Date and time the doctor account was created

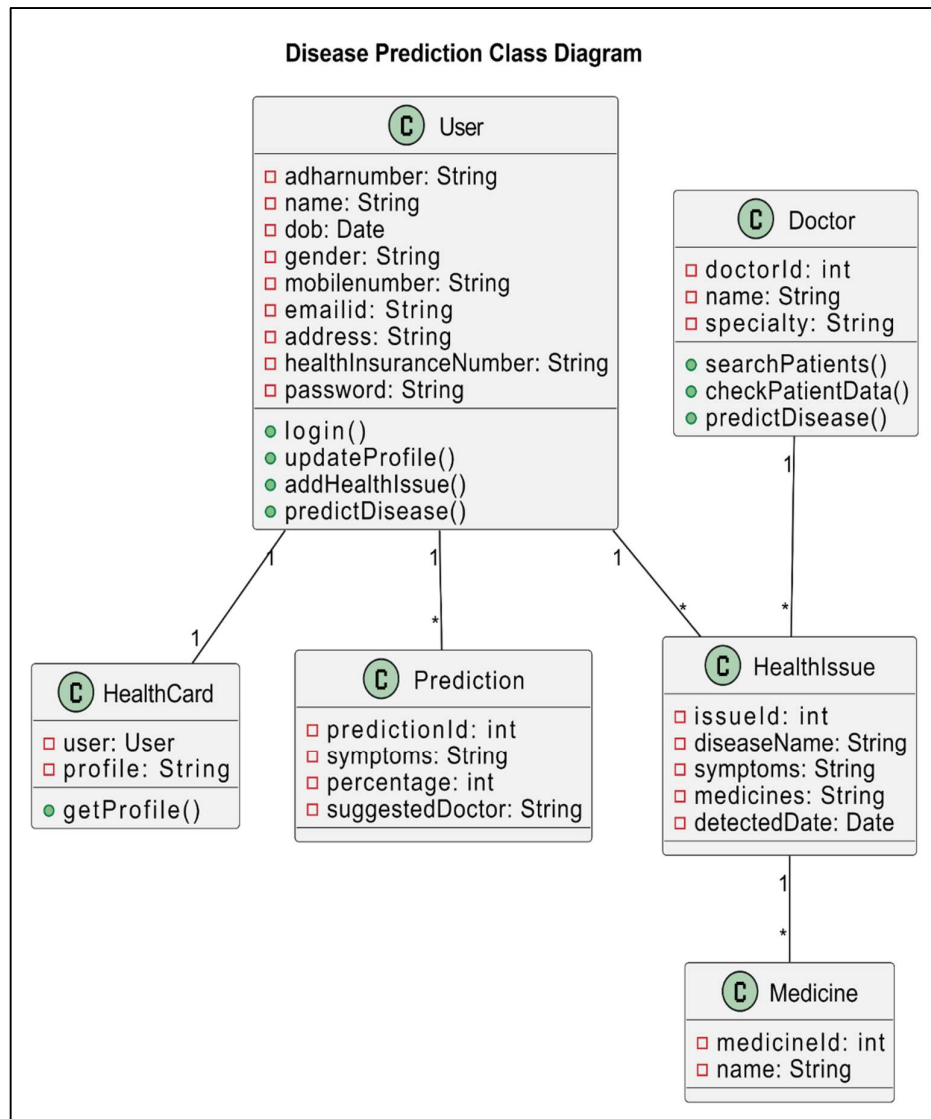
3. Health Card Table (health_cards)

Field Name	Data Type	Size	Constraints
user_id	INT	NOT NULL	Foreign key referencing the users.id
current_disease	VARCHAR (255)	NOT NULL	Name of the current disease (if any)
symptoms	TEXT	NOT NULL	Description of the current symptoms (if any)
medicines	TEXT	NOT NULL	List of prescribed medications (if any)
detected_date	DATE	NOT NULL	Date the current disease was detected (if any)
medical_history	TEXT	NOT NULL	User's medical history

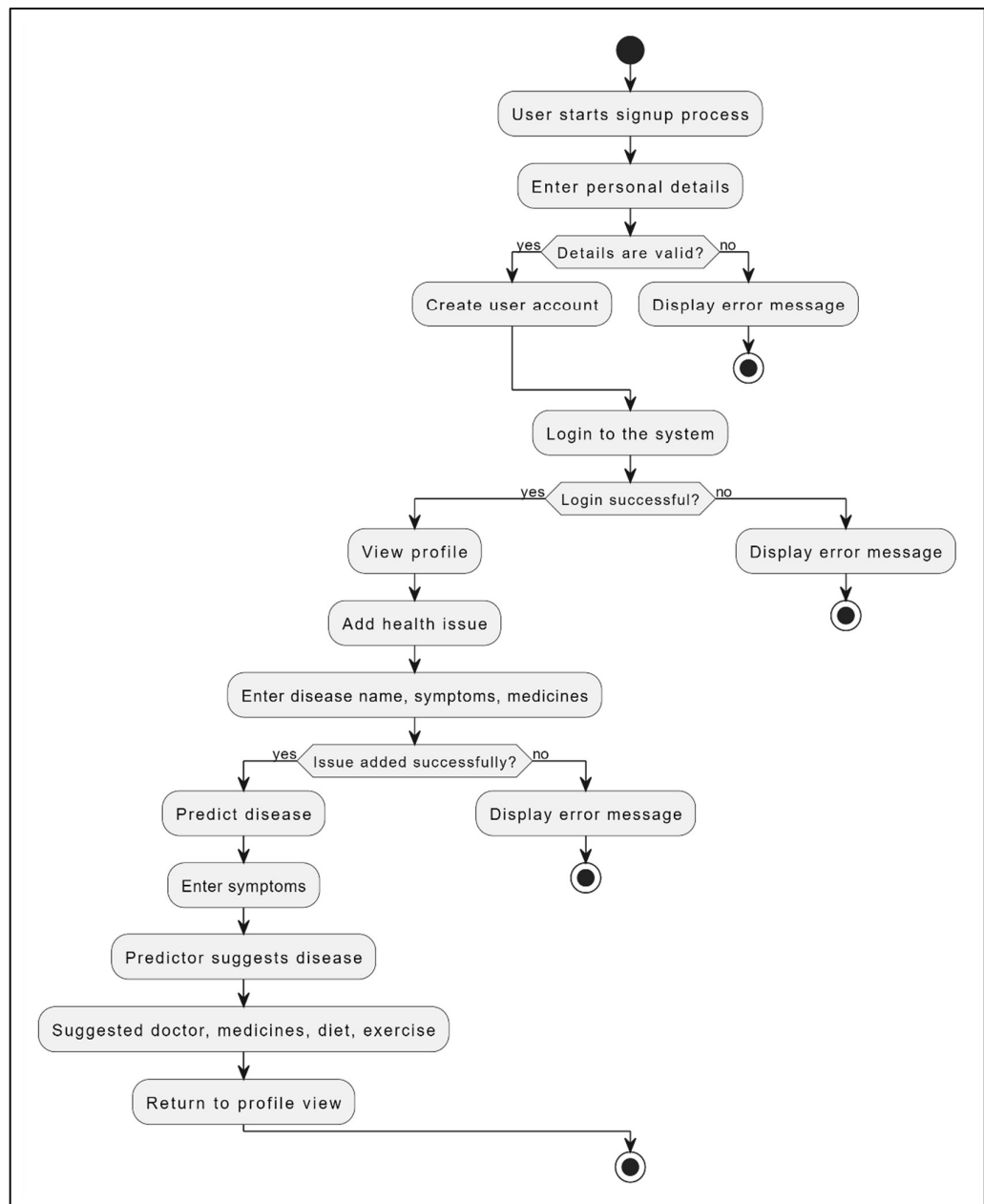
3.4 Use Case Diagrams



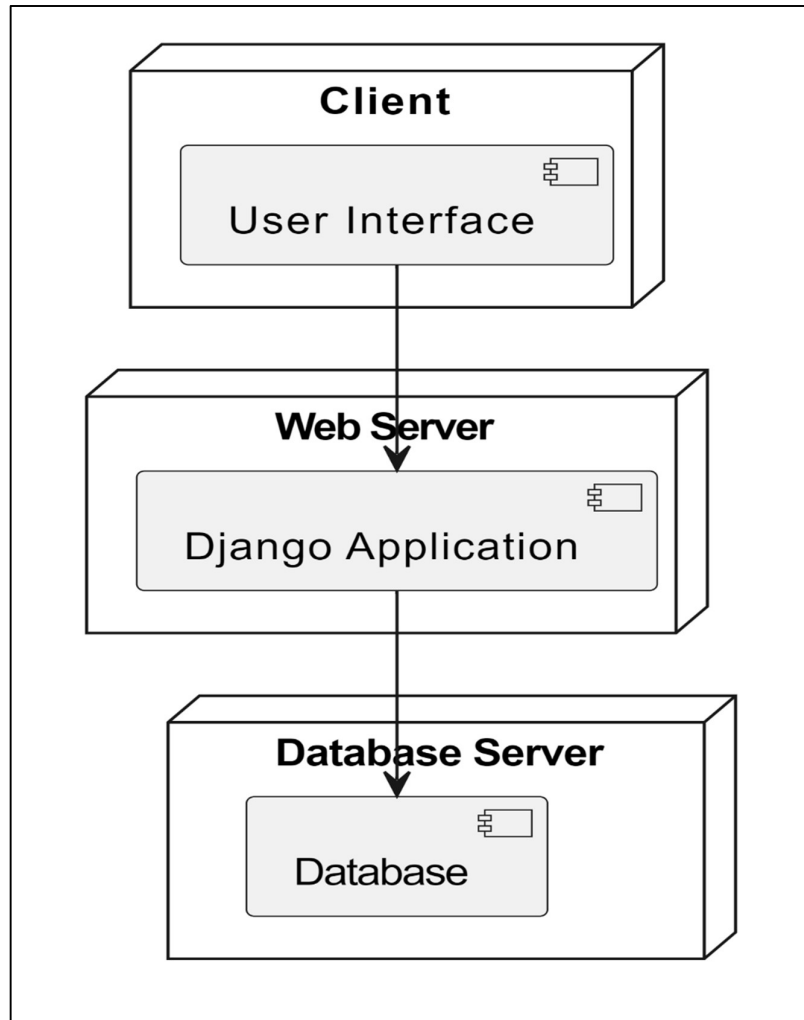
3.5 Class Diagram



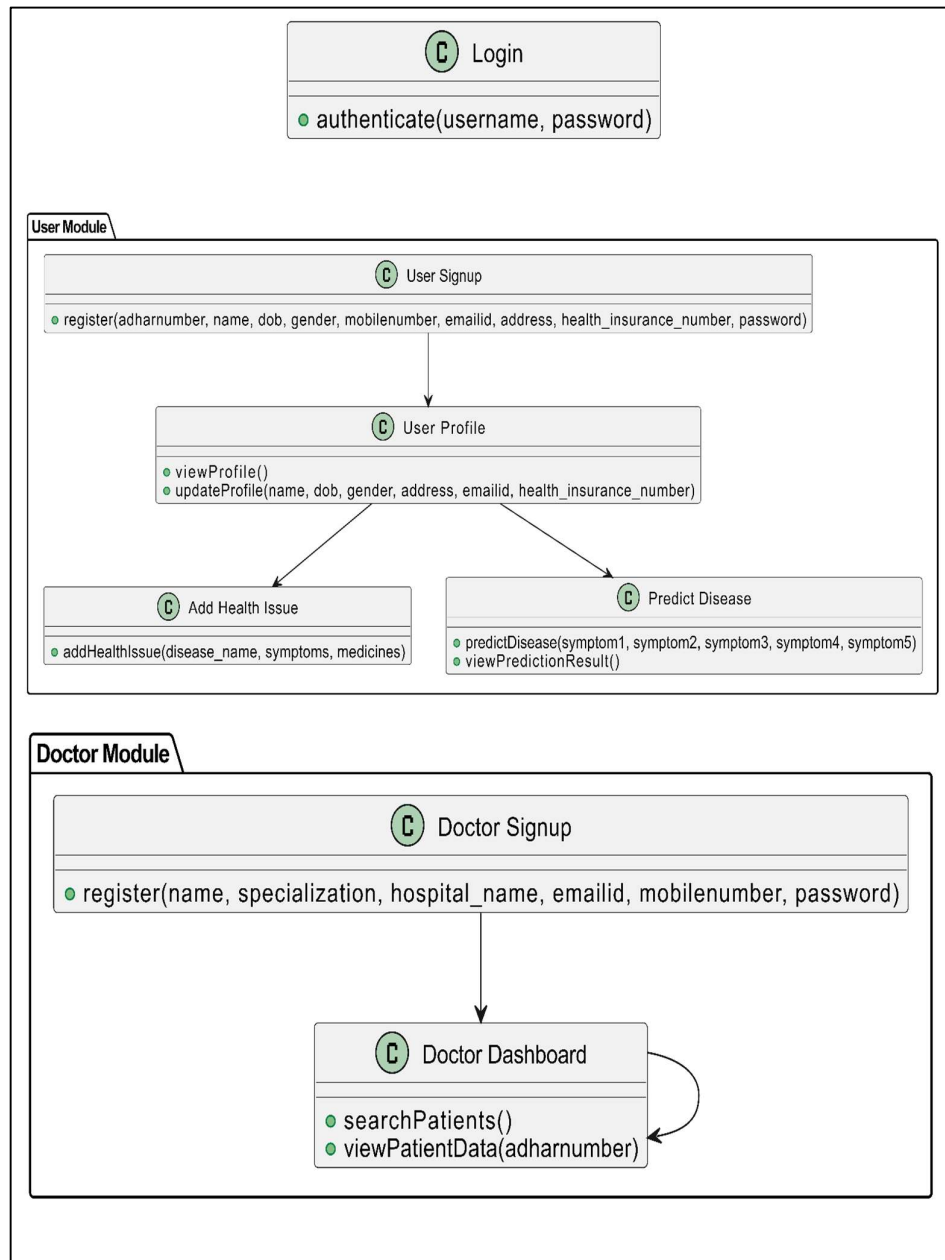
3.6 Activity Diagram



3.7 Deployment Diagram

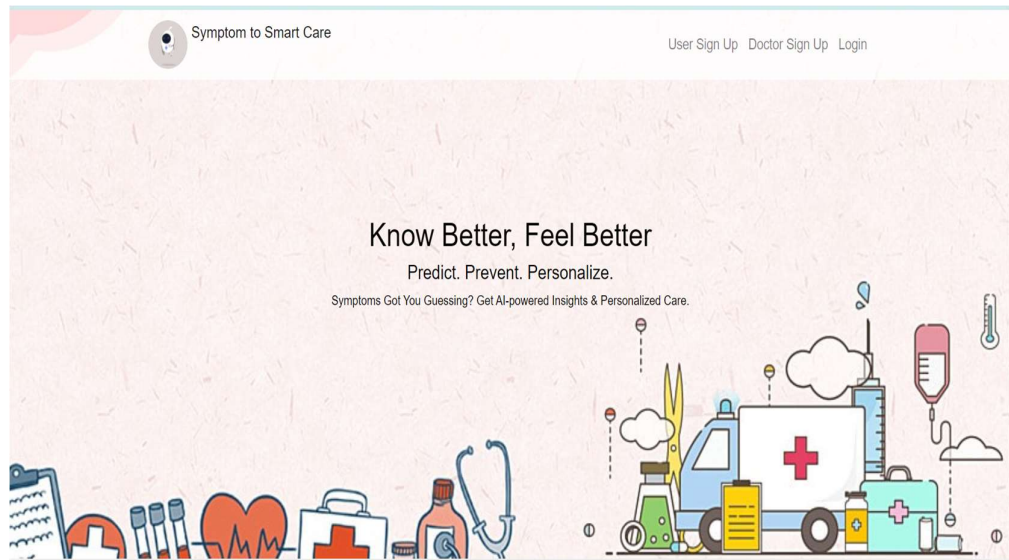


3.8 Module Hierarchy Diagram



3.9 Sample Input and Output Screens

1) Home page



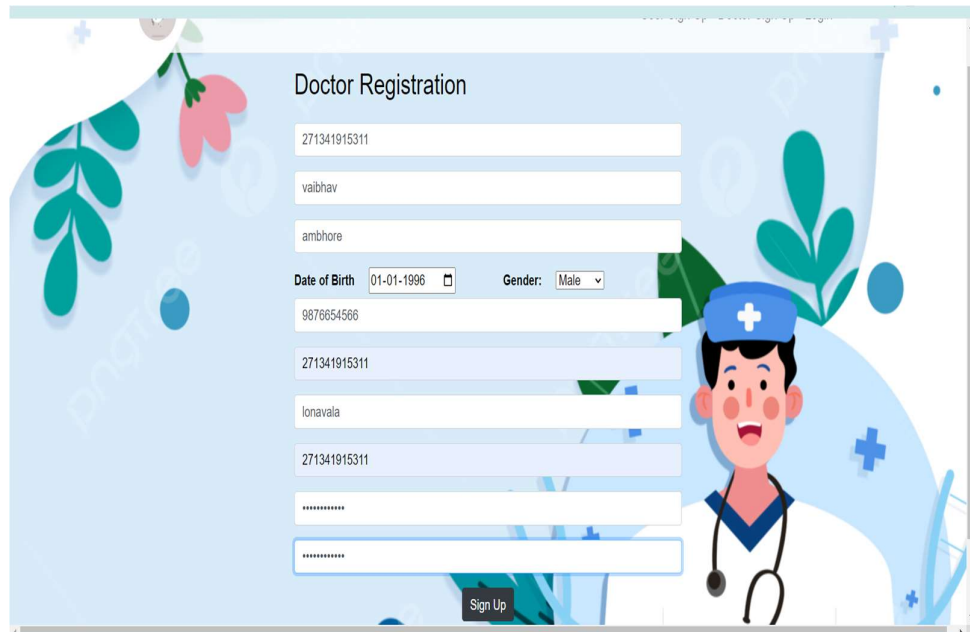
2) User Registration

The screenshot displays the 'Registration form for user' interface. The form is set against a light blue background with medical-themed illustrations, including a thermometer on the left and a doctor on the right. The form fields are as follows:

- Adhar card Number as your medical ID
- First Name
- Last Name
- Date of Birth (dd-mm-yyyy) and Gender (Male)
- Mobile Number
- Email ID
- Addresss
- Health insurance number
- Password (masked with asterisks)
- Confirm Password

A 'Sign Up' button is located at the bottom of the form. Below the button, a link states 'Do you already have an account? [Login here](#)'.

3) Doctor Registration



Doctor Registration

271341915311

vaibhav

ambhore

Date of Birth: 01-01-1996 Gender: Male

9876654566

271341915311

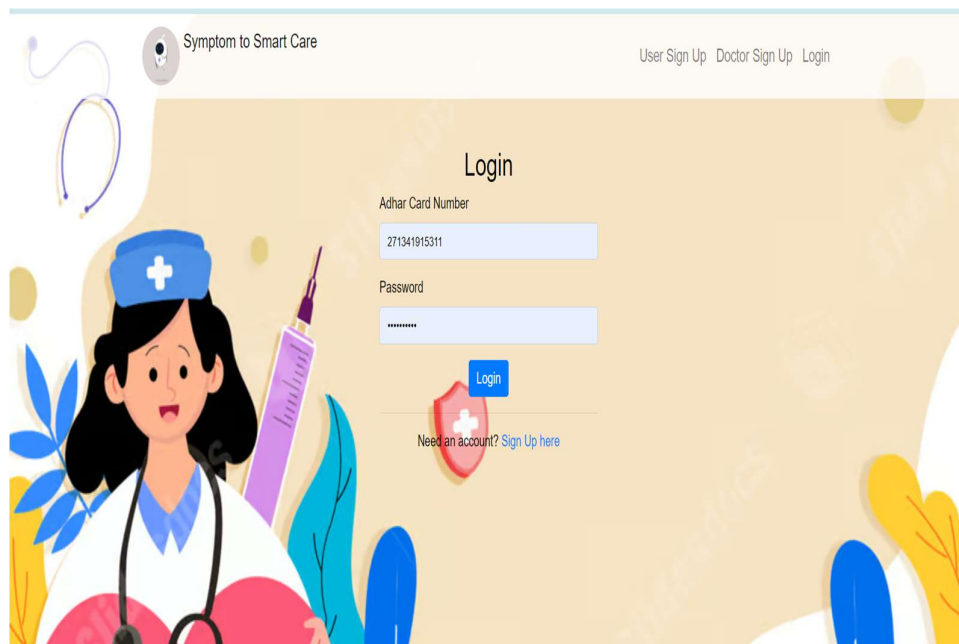
lonavala

271341915311

Sign Up

The registration form is titled 'Doctor Registration'. It contains several input fields for personal and professional details. The first field contains the number '271341915311'. The second field contains the name 'vaibhav'. The third field contains the location 'ambhore'. The 'Date of Birth' field is set to '01-01-1996' and the 'Gender' dropdown is set to 'Male'. Below these are fields for a phone number ('9876654566'), an email address ('271341915311'), a location ('lonavala'), another email address ('271341915311'), and two password fields, both masked with '*****'. A 'Sign Up' button is located at the bottom right of the form. The background features a light blue color with decorative green leaves and a pink flower. On the right side, there is a cartoon illustration of a male doctor wearing a blue cap with a white cross, a white lab coat, and a stethoscope.

4) Login Module



Symptom to Smart Care

User Sign Up Doctor Sign Up Login

Login

Adhar Card Number

271341915311

Password

Login

Need an account? Sign Up here

The login module is titled 'Login'. At the top left, there is a logo and the text 'Symptom to Smart Care'. At the top right, there are links for 'User Sign Up', 'Doctor Sign Up', and 'Login'. The main form area has a title 'Login' and two input fields: 'Adhar Card Number' (containing '271341915311') and 'Password' (masked with '*****'). A blue 'Login' button is positioned below the password field. Below the button, there is a red shield icon with a white cross and the text 'Need an account? Sign Up here'. The background is a light beige color with decorative yellow and blue leaves. On the left side, there is a cartoon illustration of a female doctor wearing a blue cap with a white cross, a white lab coat, and a stethoscope. A large purple syringe is also visible in the background.

5) Profile

Symptom to Smart Care [Your Profile](#) [Current Disease](#) [Medical History](#) [Add Health Issue](#) [Predict Disease](#) [Logout](#)

Name	aditi palange
Adhar Card Number	271341915311
Gender	male
Date of Birth	June 5, 2023
Address	Municipal housing soc,opp to bank of baroda,near urban spices,bul 3,Ionavala
Mobile Number	+91 9067843285
Email ID	aditipalange9883@gmail.com
Health Insurance Number	271341915311

[Download Health Card](#)

6) current disease

Symptom to Smart Care [Your Profile](#) [Current Disease](#) [Medical History](#) [Add Health Issue](#) [Predict Disease](#) [Logout](#)

6 Current Diseases

malaria - headache body main : [cccc]
malaria - headache pain in body : [paracetamol]
Drug Reaction - [shivering, 'fatigue', 'bur... : [Disprine]
Impetigo - [nodal skin eruptions]: [Take opinion from doctor]
Impetigo - [] : [Take opinion from doctor]
Common Cold - Runny nose, sore throat, cough... : [Decongestants, cough suppress...]

7) add health issue

The screenshot shows the 'Add Health Issue' form in the 'Symptom to Smart Care' application. The form is titled 'Health Issue' and contains three input fields: 'Disease name' with the value 'Common Cold', 'Symptoms' with the value 'Runny nose, sore throat, cough, congestion', and 'Medicines' with the value 'Decongestants, cough suppressants, pain relievers'. A 'Save' button is located below the 'Medicines' field. The background features medical illustrations like a stethoscope, syringe, and pills. The top navigation bar includes links for 'Your Profile', 'Current Disease', 'Medical History', 'Add Health Issue', 'Predict Disease', and 'Logout'.

Symptom to Smart Care | Your Profile | Current Disease | Medical History | Add Health Issue | Predict Disease | Logout

Health Issue

Disease name
Common Cold

Symptoms
Runny nose, sore throat, cough, congestion

Medicines
Decongestants, cough suppressants, pain relievers

Save

8) predict disease

The screenshot shows the 'Predict Disease' form in the 'Symptom to Smart Care' application. The form is titled 'Predict Health issue from Symptoms' and contains five dropdown menus for symptoms: 'Symptom 1: skin_rash', 'Symptom 2: itching', 'Symptom 3: cold_hands_and_feets', 'Symptom 4: weight_gain', and 'Symptom 5: vomiting'. A 'Predict Disease' button is located below the dropdowns. The background features medical illustrations like pills and a large red cross. The top navigation bar includes links for 'Your Profile', 'Current Disease', 'Medical History', 'Add Health Issue', 'Predict Disease', and 'Logout'.

Symptom to Smart Care | Your Profile | Current Disease | Medical History | Add Health Issue | Predict Disease | Logout

Predict Health issue from Symptoms

Symptom 1 : skin_rash

Symptom 2 : itching

Symptom 3 : cold_hands_and_feets

Symptom 4 : weight_gain

Symptom 5 : vomiting

Predict Disease

9) predicted Result

The screenshot shows a web application interface with a navigation bar at the top containing links: 'Symptom to Smart Care', 'Your Profile', 'Current Disease', 'Medical History', 'Add Health Issue', 'Predict Disease', and 'Logout'. The main content area is titled 'Predicted Disease from Symptoms'. It contains a text input field for symptoms with the value '[skin rash', 'itching', 'cold hands and feets', 'weight gain', 'vomiting]'. Below this, it displays 'Predicted Disease (Percentage of having this disease: 53.05 %)' as 'Fungal infection'. A 'Medicine' section suggests 'Take opinion from doctor'. At the bottom, it provides suggestions: 'Suggested doctor for predicted disease: Dr. shastri', 'Suggested Exercise for predicted disease: Yoga', and 'Suggested Diet for predicted disease: High Protein, low carbohydrates'. A blue 'Save' button is located at the bottom right of the form.

10) Generated Heathcard

The health card features a header with two blue caduceus symbols flanking the text 'Smart Health Card'. Below the header are two horizontal bars, one orange and one green. The patient's details are listed in a plain font: Name: aditi palange, Address: Muncipal housing soc, opp to bank of baroda, near t, Date of Birth: 2023-06-05, Mobile: 9067843285, Email: aditipalange9883@gmail.com, Health Insurance Number : 271341915311, and Adhar Card number : 271341915311. At the bottom, a blue banner contains the text 'HEALTH IS WEALTH' in white capital letters.

4. Coding

4. Coding

4.1 Algorithms

1. User Signup:

1. User enters registration details (Aadhaar number, name, DOB, etc.).
2. System validates user input (e.g., email format, Aadhaar number length).
3. System checks for duplicate Aadhaar number in the database.
4. If valid, system creates a new user record in the database with hashed password.
5. System sends a confirmation email or message to the user.

2. User Login:

1. User enters email and password.
2. System retrieves user record from the database based on email.
3. System verifies password by comparing the hashed input with the stored hash.
4. If login is successful, system generates a session token and stores it on the user's side (e.g., cookie).
5. System redirects the user to the appropriate dashboard (user profile).

3. User Profile:

1. System retrieves user data from the database based on the session token.
2. System displays the user's profile information (name, DOB, address, etc.).
3. System may display health card information if applicable.

4. Add Health Issue:

1. User enters new health issue details (disease name, symptoms, medicines).
2. System validates user input.
3. System creates a new health issue record in the database linked to the user.

5. Disease Prediction:

1. User enters five symptoms for prediction.
2. System utilizes a pre-trained disease prediction model (if applicable). This could involve:
3. Matching user symptoms against a symptom database.
4. Using machine learning algorithms trained on historical data.
5. Interacting with an external disease prediction API.
6. System calculates a prediction percentage for potential diseases.

6. Doctor Dashboard:

1. Doctor can search for patients based on criteria (name, Aadhaar number, etc.).
2. System retrieves patient data from the database based on search results (applying access control).
3. Doctor can view patient health card information (current disease, symptoms, medicines, medical history).
4. Doctor can potentially utilize the disease prediction model for patients.

4.2 Code snippets

1) views.py file

```
from django.shortcuts import render, redirect,
get_object_or_404
from django.contrib.auth.forms import UserCreationForm,
AuthenticationForm
from django.contrib.auth.models import User
from django.db import IntegrityError
from django.contrib.auth import login, logout, authenticate
from .forms import TodoForm
from .models import Todo
from django.utils import timezone
from django.contrib.auth.decorators import login_required
from .symptoms_list import *
from .models import User_info
from PIL import ImageFont, ImageDraw, Image
from django.core.files.storage import default_storage
from django.core.files.base import ContentFile, File
from django.http import HttpResponse
import os
from .CNN_TESTING import get_result

def home(request):
    return render(request, 'todo/home.html')

class convert_to_class:
```

```

def __init__(self,a,b):
    self.user_identity =
self.user_id_to_search = b
@login_required
def check_detail(request):
    if request.method == 'GET':
        user_info = User_info.objects.all()
        user_identity = []

        user_id_to_search = []

        for u in user_info:
            if u.role == "User":
                user_identity.append(u.first_name + " " +
u.last_name + " {" + str(User.objects.all().filter(id =
u.user_id)[0]) + "}")
                user_id_to_search.append(User.objects.all().
.filter(id = u.user_id)[0].id)
                print(user_id_to_search)
                print(user_identity)

        data_information = []
        for i in range(len(user_id_to_search)):
            data_information.append(convert_to_class(user_i
dentity[i],user_id_to_search[i]))

        return render(request,
'todo/check_detail.html',{'data_information':data_informati
on})

```

```

else:
    patient_id = request.POST['patient']
    print(patient_id)
    user_info = User_info.objects.all().filter(user_id
= int(patient_id))
disease_info = Todo.objects.all().filter(user_id =
int(patient_id))
    return render(request,
'todo/check_detail.html',{'user_info':user_info,'disease_in
fo':disease_info})

@login_required
def your_profile(request):

    user_info = User_info.objects.all().filter(user_id =
request.user.id)
    user_adhar_num = User.objects.all().filter(id =
request.user.id)[0]

    if request.method == 'GET':
        return render(request,
'todo/your_profile.html',{'user_info':user_info})
    else:
        print(user_info[0].first_name)
        print(User.username)
        image = Image.open("HEALTH_CARD.png")
        draw = ImageDraw.Draw(image)
        font = ImageFont.truetype("arial.ttf", 15)

```

```

        draw.text((10, 250), "Name: " +
user_info[0].first_name + " " + user_info[0].last_name ,
font=font, fill='#000000')
        draw.text((10, 280), "Address: " +
user_info[0].Address, font=font, fill='#000000')
        draw.text((10, 310), "Date of Birth: " +
str(user_info[0].dob), font=font, fill='#000000')
        draw.text((10, 340), "Mobile: " +
user_info[0].mobile, font=font, fill='#000000')
        draw.text((10, 370), "Email: " +
user_info[0].Email_ID, font=font, fill='#000000')
        draw.text((10, 400), "Health Insurance Number : " +
user_info[0].hin, font=font, fill='#000000')
        draw.text((10, 430), "Adhar Card number : " +
str(user_adhar_num), font=font, fill='#000000')
        image = image.convert('RGB')
        path = "todo/Health_Card.pdf"
        image.save(path)
        if os.path.exists(path):
            with open(path, 'rb') as fh:
                response = HttpResponse(fh.read(),
content_type="application/vnd.ms-excel")
                response['Content-Disposition'] = 'inline;
filename=' + os.path.basename(path)
            return response

        return render(request,
'todo/your_profile.html',{'user_info':user_info})

```



```

def signupuser(request):

    if request.method == 'GET':
        return render(request, 'todo/signupuser.html',
{'form':UserCreationForm()})
    else:
        if request.POST['password1'] ==
request.POST['password2']:
            try:
                user =
User.objects.create_user(request.POST['adhar_num'],
password=request.POST['password1'],is_staff=False)
                user.save()
                user_info = User_info()
                user_info.user = user
                user_info.first_name =
request.POST['first_name']
                user_info.last_name =
request.POST['last_name']
                user_info.dob = request.POST['dob']
                user_info.gender = request.POST['gender']
                user_info.mobile =
request.POST['Mobile_Number']
                user_info.Email_ID =
request.POST['Email_ID']
                user_info.Address = request.POST['Address']
                user_info.hin = request.POST['hin']
                user_info.role = "User"
                user_info.save()

```

```

login(request, user)

        return render(request, 'todo/home.html')
    except IntegrityError:
        return render(request,
'todo/signupuser.html', {'form':UserCreationForm(),
'error':'That adhar card number has already been taken.
Please check your adhar card number carefully'})
    else:
        return render(request, 'todo/signupuser.html',
{'form':UserCreationForm(), 'error':'Passwords did not
match'})

def doctor_signupuser(request):
    if request.method == 'GET':
        return render(request,
'todo/doctor_signupuser.html', {'form':UserCreationForm()})
    else:
        if request.POST['password1'] ==
request.POST['password2']:
            try:
                user =
User.objects.create_user(request.POST['adhar_num'],
password=request.POST['password1'],is_staff=True)
                user.save()

                user_info = User_info()
                user_info.user = user

```

```

        user_info.first_name =
request.POST['first_name']
        user_info.last_name =
request.POST['last_name']
        user_info.dob = request.POST['dob']
        user_info.gender = request.POST['gender']
        user_info.mobile =
request.POST['Mobile_Number']
        user_info.Email_ID =
request.POST['Email_ID']
        user_info.Address = request.POST['Address']
        user_info.hin = request.POST['hin']
        user_info.role = "Doctor"
        user_info.save()

        login(request, user)

        return render(request, 'todo/home.html')
    except IntegrityError:
        return render(request,
'todo/doctor_signupuser.html', {'form':UserCreationForm(),
'error':'That username has already been taken. Please
choose a new username'})
    else:
        return render(request,
'todo/doctor_signupuser.html', {'form':UserCreationForm(),
'error':'Passwords did not match'})

```

```

def loginuser(request):

    if request.method == 'GET':
        return render(request, 'todo/loginuser.html',
{'form':AuthenticationForm()})
    else:
        user = authenticate(request,
username=request.POST['username'],
password=request.POST['password'])
        if user is None:
            return render(request, 'todo/loginuser.html',
{'form':AuthenticationForm(), 'error':'Username and
password did not match'})
        else:
            login(request, user)
            return redirect('currenttodos')

@login_required
def logoutuser(request):
    if request.method == 'POST':
        logout(request)
        return redirect('home')

@login_required
def createtodo(request):
    if request.method == 'GET':
        return render(request, 'todo/createtodo.html',
{'form':TodoForm()})
    else:
        try:

```

```

        form = TodoForm(request.POST)
        newtodo = form.save(commit=False)
        newtodo.user = request.user
        newtodo.save()
    return redirect('currentttodos')
except ValueError:
    return render(request, 'todo/createtodo.html',
{'form':TodoForm(), 'error':'Bad data passed in. Try
again.'})

@login_required

def predict_disease(request):
    return render(request, 'todo/predict_disease.html',
{'form':TodoForm(), 'symptoms':symptoms})

@login_required
def predicted_results(request):
    if request.method == 'GET':
        sym1 =request.GET['sys1']
        sym2 =request.GET['sys2']
        sym3 =request.GET['sys3']
        sym4 =request.GET['sys4']
        sym5 =request.GET['sys5']

        syms = [sym1,sym2,sym3,sym4,sym5]
        symptoms = []
        for sym in syms

```

```

if sym != 'none':
    symptoms.append(sym.replace('_', ' '))

    symptoms_inserted = syms
    result, acc, doctor_info, exercise_info, diet_info, medicine_info = get_result(symptoms_inserted)
    print(result)

    return render(request,
'todo/predicted_result.html', {'form': TodoForm(), 'symptoms':
symptoms, 'disease': result, 'doctor': doctor_info, 'excercise':
exercise_info, 'accuracy': acc, 'diet': diet_info
, 'medicine': medicine_info})
    else:
        try:
            form = TodoForm(request.POST)
            newtodo = form.save(commit=False)
            newtodo.user = request.user
            newtodo.save()
            return redirect('currentttodos')
        except ValueError:
            return render(request, 'todo/createtodo.html',
{'form': TodoForm(), 'error': 'Bad data passed in. Try
again. '})
@login_required
def currentttodos(request):
    todos = Todo.objects.filter(user=request.user,
datecompleted__isnull=True)

```

```

        return render(request, 'todo/currenttodos.html',
                        {'todos': todos})

@login_required
def completedtodos(request):
    todos = Todo.objects.filter(user=request.user,
                                datecompleted__isnull=False).order_by('-datecompleted')
    return render(request, 'todo/completedtodos.html',
                  {'todos': todos})

@login_required
def viewtodo(request, todo_pk):
    todo = get_object_or_404(Todo, pk=todo_pk,
                             user=request.user)

    if request.method == 'GET':
        form = TodoForm(instance=todo)
        return render(request, 'todo/viewtodo.html',
                      {'todo': todo, 'form': form})
    else:
        try:
            form = TodoForm(request.POST, instance=todo)
            form.save()
            return redirect('currenttodos')
        except ValueError:
            return render(request, 'todo/viewtodo.html',
                          {'todo': todo, 'form': form, 'error': 'Bad info'})

@login_required
def completetodo(request, todo_pk

```

```

        todo = get_object_or_404(Todo, pk=todo_pk,
user=request.user)
        if request.method == 'POST':
            todo.datecompleted = timezone.now()
            todo.save()
            return redirect('currenttodos')
@login_required
def deletetodo(request, todo_pk):
    todo = get_object_or_404(Todo, pk=todo_pk,
user=request.user)
    if request.method == 'POST':
        todo.delete()
        return redirect('currenttodos')

```


5. Testing

5. Testing

5.1 Test Strategy

Test strategies are high-level plans that define how you'll approach testing a software application. They outline the overall testing methodology, scope, and resources needed to achieve your testing goals.

Strategies:

- 1) Combine Multiple Levels: Use unit, integration, functional, and potentially end-to-end testing for comprehensive coverage.
- 2) Mock Dependencies: Isolate components for unit testing (avoid modifying real data).
- 3) Security Focus: Test authentication, authorization, and input validation to prevent vulnerabilities.
- 4) Error Handling: Ensure proper handling of unexpected situations (database errors, API failures).

Objectives:

1. Verify Functionality: Ensure all features work as intended from user and doctor perspectives.
2. Data Integrity: Make sure user and health data are stored and accessed correctly.
3. Usability: Test that the application is easy to navigate and use for both users and doctors.
4. Security: Guarantee that the application is protected a

5.2 Unit Test Plan

1. Objectives:

Verify the functionality of individual code units (functions, modules) according to specifications.

Identify and fix coding errors and logic issues within isolated units.

2. Techniques:

Utilize a testing framework like pytest

Functionality	Test Case	Expected Outcome
User Signup	Valid signup with all required fields	Successful user creation and account activation
User Signup	Missing required field (e.g., name)	Error message indicating missing field
User Signup	Invalid email format	Error message indicating invalid email format
Login	Login with correct credentials	Successful login and access to user profile
Login	Login with incorrect password	Error message indicating incorrect password
Disease Prediction (if implemented)	Predict disease with known symptom combination	Prediction result with a percentage and emphasis on consulting a professional

5.3 Acceptance Test Plan

1. Introduction

This acceptance test plan outlines the procedures for User Acceptance Testing of the healthcare management system . It aims to ensure the system meets user expectations and functional requirements before deployment.

Functionality	Test Case	Expected Outcome	Pass/Fail
User Signup	New user registers with valid data	Successful account creation and activation	Pass
User Login	Existing user logs in with correct credentials	Successful login and access to user profile	Pass
Add Health Issue	User adds a new health issue with disease name, symptoms, and medications	Health issue added to user's medical history	Pass
Edit & Delete Health Issue	User edits an existing health issue and deletes another	Updated and deleted information reflected in user's medical history	Pass
Disease Prediction	User inputs specific symptoms	Prediction result displayed with a percentage and a clear message emphasizing the limitations	Pass

5.4 Test Case / Test Script

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail
TC-001	User Signup - Valid Data	1. Access the signup page.	1. User account successfully created and activated.	1. User account created, but activation email not sent.	Pass
		2. Enter valid information in all required fields (name, email, password, etc.).	2. Confirmation email sent to the registered email address (optional).	2. Confirmation email received by the user.	Pass
		3. Click "Submit" button.			
TC-002	User Signup - Missing Required Field	1. Access the signup page.	1. Error message displayed indicating missing field.	1. No error message displayed.	Fail
		2. Leave a	2. User unable	2. User able to	Fail

		required field (e.g., name) blank.	to submit the signup form.	submit the signup form despite missing field.	
		3. Click "Submit" button.			
TC-003	User Signup - Invalid Email Format	1. Access the signup page.	1. Error message displayed indicating invalid email format.	1. No error message displayed.	Fail
		2. Enter invalid email format (e.g., "username" instead of "[email address removed]").	2. User unable to submit the signup form with invalid email.	2. User able to submit the signup form with invalid email.	Fail
		3. Click "Submit" button.			
TC-004	User Login - Correct	1. Access the login page.	1. Successful login and	1. Successful login, but	Fail

	Credentials		redirection to user profile.	redirected to an error page.	
		2. Enter valid username and password.	2. Session established and user granted access to their profile.	2. User unable to access profile after login.	Fail
		3. Click "Login" button.			
TC-005	User Login - Incorrect Username	1. Access the login page.	1. Error message displayed indicating invalid username.	1. No error message displayed.	Fail
		2. Enter an invalid username that does not exist.	2. User unable to log in.	2. User able to attempt login with nonexistent username.	Fail
		3. Click "Login" button.			
TC-006	User Login - Incorrect Password	1. Access the login page.	1. Error message displayed indicating	1. No error message displayed.	Fail

			incorrect password.		
		2. Enter the correct username but an incorrect password.	2. User unable to log in. Implement account lockout after multiple attempts (optional).	2. User able to attempt login with incorrect password indefinitely.	Fail
		3. Click "Login" button.			

5.5 Defect report / Test

Defect ID	Module	Description	Severity	Priority
DR-001	User Signup	Aadhaar number validation might not be strict enough (length, format)	Medium	High
DR-002	User Signup	Name field might not have proper validation (empty, length restrictions)	Low	Medium
DR-003	User Signup	Date of birth validation might not catch future dates or invalid formats	Medium	High
DR-004	User Signup	Mobile number validation might not be strict enough (length, format)	Low	Medium
DR-005	User Signup	Email format validation might not be implemented	Medium	Medium
DR-006	User Signup	Signup doesn't handle missing required fields gracefully	Medium	High
DR-007	User Signup	Duplicate Aadhaar number check might not be implemented (if applicable)	High	High
DR-008	User Signup	Password validation might not enforce minimum length requirements	Medium	High
DR-009	User Signup	Password validation might not reject weak passwords (lacking complexity)	Medium	High
DR-010	Doctor Signup	Signup doesn't handle missing required fields (e.g., license number)	Medium	High
DR-011	Doctor Signup	Duplicate doctor credentials check might not be implemented (if applicable)	Medium	High

DR-012	Login	Login might not handle incorrect username/email or password gracefully	Medium	High
DR-013	Login	Login might not handle non-existent user credentials gracefully	Medium	Medium
DR-014	Login	Session management might not be implemented or might have issues (creation, maintenance, expiration)	Medium	High
DR-015	Disease Prediction	Disease prediction might not handle unknown symptoms gracefully (e.g., no recommendation)	Medium	Medium

6. Limitations of Proposed System

6. Limitations of Proposed System

1. Data Security and Privacy:

- **Breaches:** The system will store sensitive user information, including health profiles and potentially medical history. Mitigating data breaches will be crucial. Robust security measures like encryption and access controls are essential.
- **Compliance Challenges:** The system will need to comply with various data privacy regulations like HIPAA (Health Insurance Portability and Accountability Act) in the US or GDPR (General Data Protection Regulation) in the EU. Navigating these regulations can be complex.

2. Accuracy and Reliability:

- **Limitations of Technology:** Machine learning models are not perfect. Factors like limited training data or inherent complexity of medical diagnosis can affect accuracy.
- **Scalability Issues:** As the user base and disease data grow, the system may require adaptation and ongoing maintenance to ensure its effectiveness.

3. Ethical Considerations:

- **Bias in Predictions:** The model's predictions could be biased if the training data itself reflects societal or historical biases. Continuous monitoring and mitigation strategies are needed.
- **Over-reliance on Technology:** The system should be used as a tool to complement medical expertise, not replace it. Patients should be encouraged to consult doctors for diagnosis and treatment.

4. Additional Limitations:

- Maintenance: The system will require ongoing maintenance to ensure it remains functional and up-to-date.
- User Adoption: Patients and doctors may be hesitant to use a new system, especially if concerns about data privacy or accuracy exist. Building trust and promoting the system's benefits will be important.
- Cost: Developing and maintaining the system, including data security measures and model updates, can be expensive. Cost-effectiveness needs to be considered.

7. Proposed Enhancements

7. Proposed Enhancements

- 1) Go beyond 5 symptoms: Allow users to input a wider range of symptoms to improve prediction accuracy.
- 2) Symptom severity: Enable users to indicate the severity of each symptom (e.g., mild, moderate, severe) for more nuanced analysis.
- 3) Timeframe: Capture the timeframe of symptom onset (e.g., recent, ongoing, chronic) to provide context for potential diagnoses.
- 4) Integrate user profile: Factor in user demographics, medical history, and allergies to tailor medication suggestions.
- 5) Dosage considerations: While dosages should not be provided automatically, the system could offer general recommendations for exploring different dosage options with a doctor.
- 6) E-prescriptions: Explore integrating secure e-prescription functionality within the system (subject to legal and regulatory compliance).
- 7) Integrate nutritional databases: Link the system to nutritional databases to provide personalized dietary recommendations based on predicted diseases, user preferences, and potential dietary

restrictions.

8) Tailored exercise plans: Offer adaptable exercise plans considering disease implications, user fitness levels, and physical limitations. This could involve collaborating with fitness professionals or integrating with reputable exercise planning platforms.

9) Educational resources: Provide links to informative resources about healthy eating, exercise regimes, and disease management to enhance user knowledge and empower informed decision-making.

10) Enhanced data protection: Employ robust encryption techniques to safeguard user information.

User control over data: Empower users to manage their data access and consent preferences.

Compliance with regulations: Ensure adherence to relevant data privacy regulations like HIPAA or GDPR.

11) Personalized health analytics: Develop capabilities to analyze user health trends over time and provide proactive recommendations for disease prevention and well-being.

8. Conclusion

8. Conclusion

This healthcare management system, while providing a foundation for managing personal health data and communication with doctors, offers further potential for enhancements. By focusing on **security, privacy, user experience, doctor functionality, accessibility, and ethical considerations**, the system can evolve into a more robust, reliable, and user-centric platform. Implementing the proposed enhancements and addressing potential ethical considerations will contribute to a more informed and empowered healthcare experience for users. In this project, we have developed a comprehensive healthcare management system using Django. The system caters to two main user roles: users (patients) and doctors. Users can sign up, log in, manage their profiles, add health issues, and even predict diseases based on symptoms. On the other hand, doctors can sign up, log in, access patient data, and also utilize the disease prediction functionality.

Throughout the development process, we have successfully implemented various features such as user authentication, profile management, database integration, and predictive analysis. The system provides a user-friendly interface for both users and doctors to interact seamlessly and efficiently manage healthcare-related tasks.

9. Bibliography

9. Bibliography

1. Predictive Analytics in Healthcare: A Review of Current Trends and Future Directions by Li et al. (2018)

(<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6857503/>)

2. Telemedicine and Its Role in Revolutionizing Healthcare Delivery by Bashshur et al. (2018)

(<https://www.ncbi.nlm.nih.gov/books/NBK459384/>)

3. Government health agencies like the Centers for Disease Control and Prevention (CDC) (<https://www.cdc.gov/>), National Institutes of Health (NIH) (<https://www.nih.gov/>), or Mayo Clinic (<https://www.mayoclinic.org/>)

4) Healthcare IT publications (HIMSS, AHIMA)

5) Government healthcare websites (e.g., HHS)

10. Appendix

10. Appendix

1. Data Collection and Preparation:

- **Real-world Data:** Large datasets are required to train the machine learning model. This data comes from electronic health records, clinical trials, and medical research.
- **Data Features:** Each data point includes a set of features (symptoms, test results, demographics) and a corresponding diagnosis.
- **Data Cleaning and Preprocessing:** The data is cleaned to remove errors and inconsistencies, and formatted for the machine learning algorithm.

2. Machine Learning Algorithms:

- **Common Algorithms:** Logistic Regression, Naive Bayes, Decision Trees, Support Vector Machines (SVMs), Random Forests are some commonly used algorithms for disease prediction.
- **Model Training:** The chosen algorithm is trained on a portion of the data. The model learns to identify patterns between symptoms and diseases.
- **Model Testing:** The trained model is evaluated on a separate data set to assess its accuracy and generalizability.

3. Application Development:

- **User Interface:** The application is designed to allow users to input their symptoms.
- **Model Integration:** The trained machine learning model is integrated into the application.
- **Output and Disclaimers:** The application should display the predicted disease(s) along with clear disclaimers about limitations and the importance of consulting a doctor for a proper diagnosis.

4. Important Considerations:

- **Accuracy and Limitations:** Machine learning models are not perfect and can have limitations. Their accuracy depends on the quality of the training data and the complexity of the disease being predicted.
- **Not a Substitute for Medical Diagnosis:** These applications are intended to be an initial screening tool, not a replacement for a doctor's diagnosis. It is crucial to consult a medical professional for any health concerns.
- **Medical Disclaimer:** The application should have a clear disclaimer stating that it is for informational purposes only and should not be used for self-diagnosis or treatment.

11. User Manual

11. User Manual

1) Registration form for user and doctor

User Signup page will take inputs like below and would need following validations :

1. Adhar number (only 12 Digits and should be unique)
2. Name (Only characters allowed)
3. Date of birth and Gender (To be selected from options)
4. Mobile number (only 10 digits t be entered)
5. Email id (Should be as per the standard email id format)
6. Health Insurance number (A 10 digit unique integer)

The Doctor Registration would also be same but have an additional unique License ID of 10 digits

The screenshot shows a web browser window with a light blue background. On the left, there is a blue stethoscope icon and a grey first aid kit with a red cross. On the right, there is a cartoon illustration of a female doctor with brown hair, wearing a white lab coat and a blue stethoscope. The main content area is titled 'Registration form for user'. The form contains the following fields: 'Adhar number' (pre-filled with '271341915311'), 'First Name', 'Last Name', 'Date of Birth' (with a dropdown menu showing 'dd-mm-yyyy'), 'Gender' (a dropdown menu showing 'Male'), 'Mobile Number', 'Email ID', 'Address', 'Health insurance number', and 'Confirm Password' (with a masked input '*****'). A dark grey 'Sign Up' button is located at the bottom right of the form.

2) login page : If you already have an account then click on login and put the credentials having adhar card number and password

Symptom to Smart Care

User Sign Up Doctor Sign Up Login

Login

Adhar Card Number
271341915311

Password

Login

Need an account? [Sign Up here](#)

3) Your Profile module : This module will display the profile information of user in tabular form.

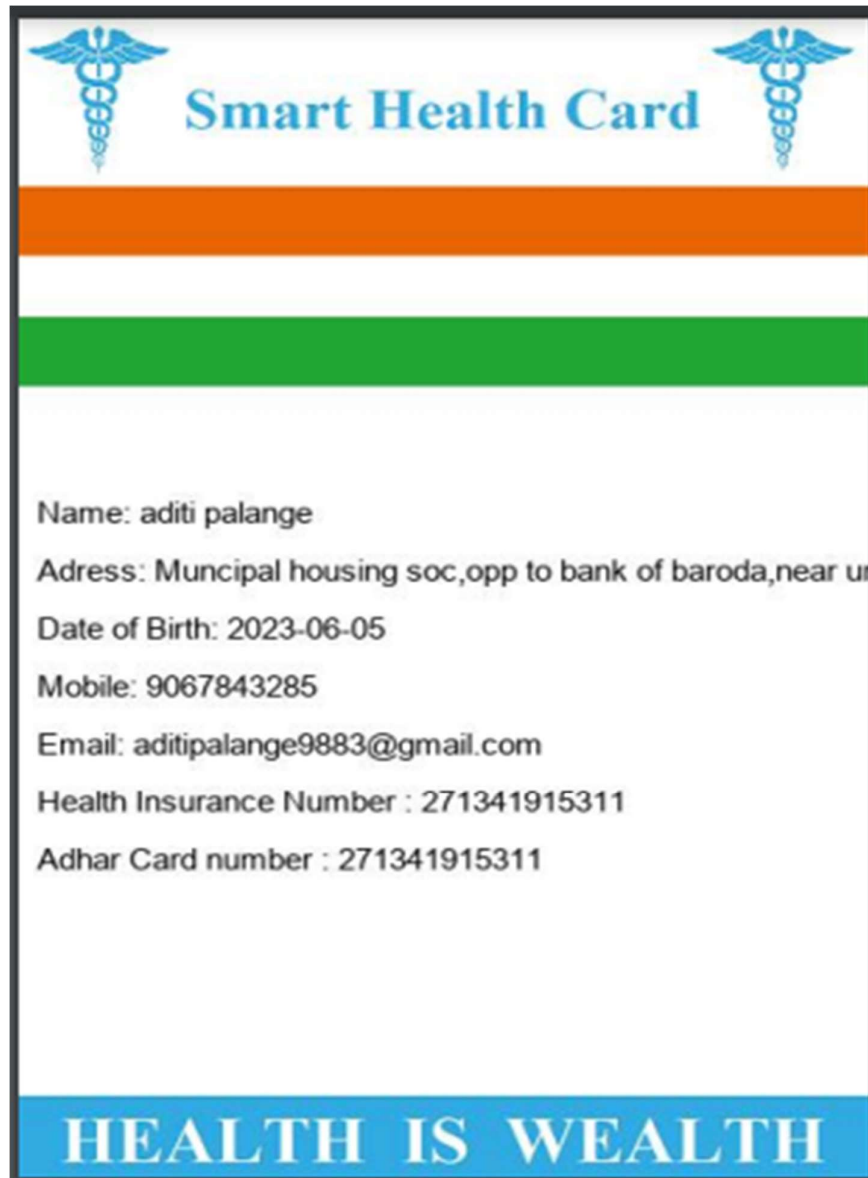
Symptom to Smart Care

Your Profile Current Disease Medical History Add Health Issue Predict Disease Logout

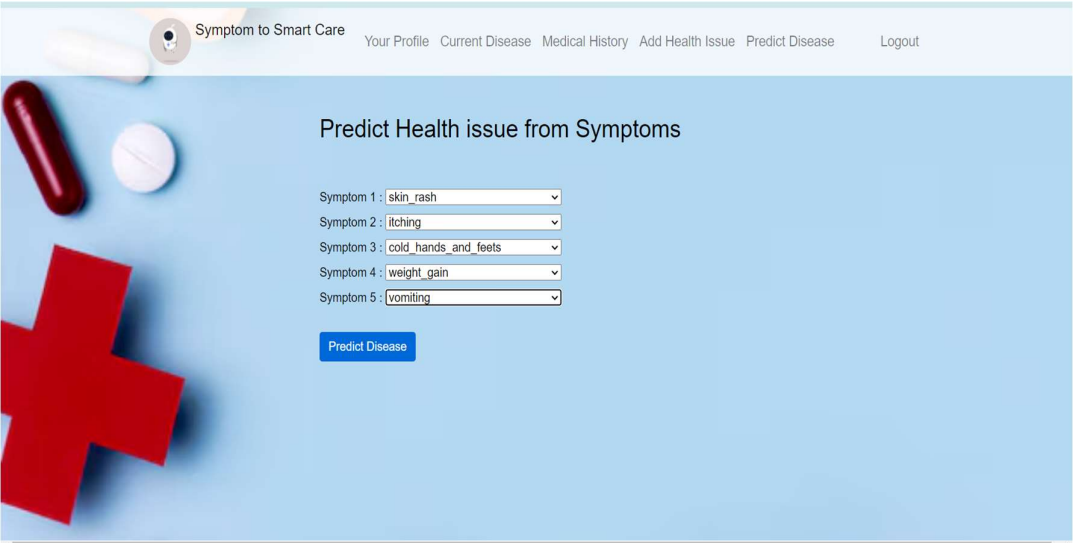
Name	aditi palange
Adhar Card Number	271341915311
Gender	male
Date of Birth	June 5, 2023
Address	Municipal housing soc, opp to bank of baroda, near urban spices, bul 3, lonavala
Mobile Number	+91 9067843285
Email ID	aditipalange9883@gmail.com
Health Insurance Number	271341915311

Download Health Card

4) Generated healthcard : after clicking on download health card a health card having user information would get downloaded .




5) Predict Disease: User can predict the disease by giving 5 symptoms out of which atleast 3 are mandatory to predict the disease .



The screenshot shows a web application interface titled "Symptom to Smart Care". The navigation bar includes links for "Your Profile", "Current Disease", "Medical History", "Add Health Issue", "Predict Disease", and "Logout". The main heading is "Predict Health issue from Symptoms". Below this, there are five dropdown menus for symptoms: Symptom 1 (skin_rash), Symptom 2 (itching), Symptom 3 (cold_hands_and_feets), Symptom 4 (weight_gain), and Symptom 5 (vomiting). A blue "Predict Disease" button is located below the dropdowns. The background features a large red cross and several pills.

6) Prediction Result : The predicted result screen displays the symptoms given along with the predicted disease and also the percentage of having this disease.

Adittionally it also suggest the specialized doctors treating the disease and medicines required to cure the disease along with the personalized diet plan and exercises.

Symptom to Smart Care

[Your Profile](#) [Current Disease](#) [Medical History](#) [Add Health Issue](#) [Predict Disease](#) [Logout](#)

Predicted Disease from Symptoms

Symptoms

[skin rash, 'itching', 'cold hands and feels', 'weight gain', 'vomiting']

Predicted Disease (Percentage of having this disease: 53.05 %)

Fungal infection

Medicine

Take opinion from doctor

Suggested doctor for predicted disease: **Dr. shastri**

Suggested Exercise for predicted disease: **Yoga**

Suggested Diet for predicted disease: **High Protein, low carbohydrates**

Save