Project Title	HealthCare
Technologies	Machine Learning Technology
Domain	AI and ML

Problem Statement:

For healthcare companies like Apollo Hospitals Enterprise Limited to plan effectively and operate efficiently, they must understand their patients' disease conditions and see how those conditions progress over time. To segment patients by their disease conditions and severity of illnesses, the healthcare company's management team approached their AI team for machine learning applications (MLOps).In a meeting with the management team of the healthcare company, it was suggested that the machine learning model be trained and tested with a variety of data sets as needed. As a result, business users should be able to upload training data to MLOps and select features through the user interface (UI). Users should also be able to upload and preview test data to test the model. Explanations AI functionality should be implemented by MLOps to help business users understand what the model outcomes mean. To simplify and better understand model outcomes, business users requested visual data analysis functionality.

Dataset:

https://www.kaggle.com/datasets/tomaslui/healthcare-dataset

Project Evaluation metrics:

Sn.No	Develiable	Format
1	Document	Link: https://github.com/Subi13/RIT-HACK
2	Code	• Code should be written in a modular manner to make sure it isorganized and easy to maintain. • Prioritize safety in your code to prevent harm. • Ensure your code is testable at the code level to facilitate effectivetesting. • Maintainability is crucial, so design your code to handle growthas your project expands. • Aim for portability, ensuring your code worksconsistently acrossdifferent environments and operating systems.
		• Host your code on GitHub for version control and collaboration. • Set your GitHub repository to be public to allow others to reviewyour code. • Create a comprehensive README file for each project youdevelop. • Include essential information in the README, such as basicworkflows and project execution instructions. • Adhere to Python's coding standards outlined in PEP 8: link
3	User Interface	User interfaces should be included in your model testing. Anythingwill be fine with us. For example, Flask, Streamlit, etc.
4	Deployment	The deployment process for your code can be done using any cloudplatform like:AWS, GCP, Azure and Streamlit Share.

Code:

```
from flask import Flask, render_template, request, redirect, url_for, flash, jsonify
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager, UserMixin, login_user, login_required, current_user,
logout_user
import pandas as pd
```

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from io import BytesIO, StringIO
import base64
from werkzeug.security import generate_password_hash, check_password_hash

app = Flask(_name)
app.config['SECRET_KEY'] = 'your_secret_key' # Change this to a secure secret key
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///data.db' # Use SQLite for
simplicity
db = SQLAlchemy(app)

# User model for database
class User(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    password = db.Column(db.String(120), nullable=False)

# Configure login manager
login_manager = LoginManager()
login_manager.login_view = "login"
login_manager.init_app(app)

@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))

# Global variables to store the model and training data
model = None
X_train = None
y_train = None

# Error handling for invalid data uploads
def handle_invalid_data_upload(file):
    if file is None or file.filename == '':
        return True
    return False

# Route to the main page
@app.route('/')
def home():
    return render_template('index.html', user=current_user)

# User login
```

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form.get('username')
        password = request.form.get('password')
        user = User.query.filter_by(username=username).first()
        if user and check_password_hash(user.password, password):
            login_user(user)
            flash('Login successful', 'success')
            return redirect(url_for('home'))
        else:
            flash('Login failed. Check your username and password.', 'danger')
    return render_template('login.html')

# User logout
@app.route('/logout')
@login_required
def logout():
    logout_user()
    flash('You have been logged out', 'success')
    return redirect(url_for('home'))

# Route to upload and train the model
@app.route('/train_model', methods=['POST'])
@login_required
def train_model():
    global model, X_train, y_train

    # Get the uploaded training data
    uploaded_file = request.files.get('file')

    if handle_invalid_data_upload(uploaded_file):
        return "Invalid or missing training data file"

    try:
        # Read and preprocess the training data
        data = pd.read_csv(uploaded_file)
        X = data.drop('target', axis=1)
        y = data['target']

        # Train a simple model (Random Forest for demonstration)
        X_train, _, y_train, _ = train_test_split(X, y, test_size=0.2, random_state=42)
        model = RandomForestClassifier(n_estimators=100, random_state=42)
        model.fit(X_train, y_train)

        return "Model trained successfully!"
```

```
except Exception as e:
    return f"An error occurred while training the model: {str(e)}"
```

Deepsphere.

AI

```
@app.route('/predict', methods=['POST'])
@login_required
```

```
def predict():
    if model is None:
        return "Model is not trained yet"

    uploaded_file = request.files.get('file')

    if handle_invalid_data_upload(uploaded_file):
        return "Invalid or missing test data file"

    try:
        data = pd.read_csv(uploaded_file)
        X_test = data

        predictions = model.predict(X_test)

        return "Predictions: " + str(predictions)
    except Exception as e:
        return f"An error occurred while making predictions: {str(e)}"

@app.route('/visualizations')
@login_required
def generate_visualizations():
    if X_train is None or y_train is None:
        return "No training data available for visualization"

    try:
        plt.figure(figsize=(8, 6))
        class_counts = y_train.value_counts()
        class_counts.plot(kind='bar')
        plt.title('Distribution of Classes')
        plt.xlabel('Class')
        plt.ylabel('Count')
        img = BytesIO()
        plt.savefig(img, format='png')
        img.seek(0)
        plt.close()
        plot_url = base64.b64encode(img.read()).decode()

        return render_template('visualizations.html', plot_url=plot_url)
    except Exception as e:
        return f"An error occurred while generating visualizations: {str(e)}"

if__name__== '_main_':
    db.create_all()
    app.run(debug=True)
```