COLLEGE CODE: 3105

COLLEGE NAME: DHANALAKSHMI SRINIVASAN COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT: B.tech information technology

STUDENT NM-ID:
e1cf9bc29af6a1aec
34560e0087d96d

ROLL NO : 310523205012

DATE: 14/05/2025

TECHNOLOGY-PROJECT NAME: AI-Powered Healthcare Assistant

SUBMITTED BY:

Your Name :
Alice
Soundariya
.A

Phase 5: Project Demonstration & Documentation

Title: AI-Powered Healthcare Assistant

Abstract:

The **AI-Powered Healthcare Assistant** project is designed to transform healthcare accessibility through **artificial intelligence (AI), natural language processing (NLP), and Internet of Things (IoT) technologies**. This innovative system integrates **advanced AI models** for symptom diagnosis, **real-time health data collection from wearable IoT devices**, and **secure, decentralized medical data management**, ensuring **scalability and seamless integration** with **Enterprise Resource Planning (ERP) systems**.

Demonstration Components:

- System Walkthrough → Interactive presentation of chatbot interactions, AI health assessments, and user-friendly functionality.

- AI Diagnosis Accuracy → Showcase real-time symptom analysis and personalized treatment recommendations, backed by AI predictions.

- IoT Integration → Display wearable health data collection, including heart rate, oxygen levels, and temperature tracking.

- Performance Metrics → Demonstrate response times, multi-user scalability, and AI efficiency in high-load conditions.

- Security & Privacy → Explain blockchain-based encryption and permission controls for secure medical data management.

---

s.no	Topic	pg.no
1.	Project Documentation	4
2.	Final refinements	4-5
3.	Project Handover & Future Works	5-6
4.	Final Project	7
5.	Source code	8-14

Project Documentation:

Overview:

Detailed technical documentation ensures that the AI system is well-documented for future scalability, improvements, and troubleshooting.

Documentation Sections:

- System Architecture → Illustrate AI workflow, chatbot processes, IoT data integration, and security protocols with diagrams.
- Code Documentation → Provide source code explanations, including AI training scripts, API integrations, and encryption handling.
- User Guide → Explain step-by-step interactions, enabling users to understand diagnostics and AI-driven health suggestions.
- Administrator Guide → Detail system maintenance, performance optimization, and data security management.
- Testing Reports → Include performance benchmarks, stress test results, and security audits to validate robustness.

Final Refinements:

Overview:

The system undergoes a final feedback round to refine its functionality, usability, and reliability before full implementation.

- Stakeholder Review → Gather insights from healthcare experts, developers, and test users via structured surveys.
- Performance Bottleneck Analysis → Identify and resolve AI misdiagnoses, response delays, and integration challenges.
- Final Testing Iteration → Conduct post-feedback adjustments, ensuring high usability and optimal AI performance.

Outcome:

A structured, well-documented final report ensures transparency, project completion validation, and future work recommendations.

Project Handover & Future Works:

Overview:

The AI-Powered Healthcare Assistant project reaches a pivotal stage where the focus shifts from development to structured planning for its real-world implementation. This phase ensures that the system is effectively handed over to stakeholders with clear guidelines for future advancements, scalability, and integration possibilities.

Final Handover Components:

1.Future Scalability Recommendations

- Expand AI capabilities to support advanced diagnostics and personalized health insights.
- Introduce multilingual AI interaction, increasing accessibility for global users.
- Enhance interoperability with healthcare provider systems, enabling seamless integration with hospital databases and telemedicine platforms.

2.System Maintenance Guide

- Establish protocols for regular security updates, ensuring compliance with data protection standards.
- Implement continuous AI model refinement, incorporating new medical research for improved accuracy.
- Develop structured data management procedures, ensuring efficient health record storage, retrieval, and encryption.

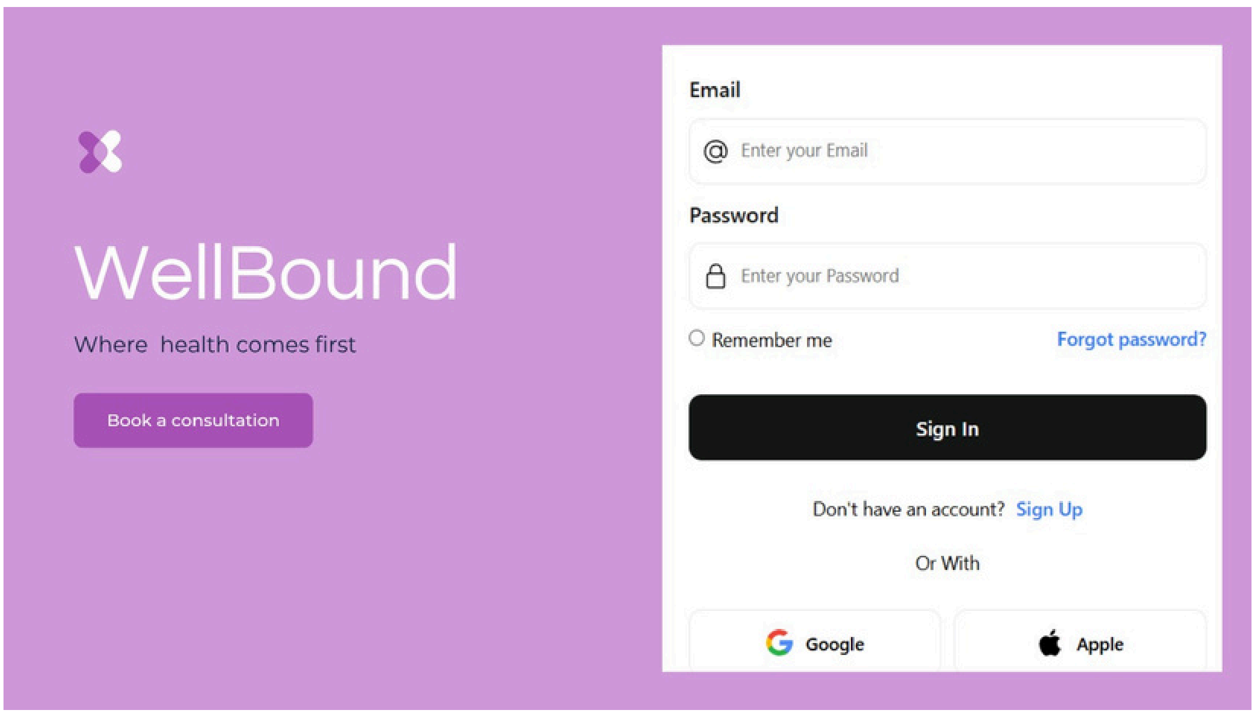
3.Next-Phase Roadmap:

Focus on AI self-learning mechanisms, enabling autonomous adaptation to evolving medical datasets and symptoms.

- Strengthen real-world deployment strategies, facilitating seamless hospital adoption and telehealth compatibility.
- Enhance user experience optimization, refining chatbot responsiveness and predictive analytics for personalized recommendations.

Outcome:

This phase lays the groundwork for real-world implementation, ensuring that stakeholders receive a well-structured framework for future scalability, performance improvements, and integration planning. The system's next iteration will be shaped by collaborative feedback, technological advancements, and industry-driven AI innovations.



```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import joblib

# Function to analyze medical images (placeholder for actual image processing)
def analyze_medical_image(image):
    # Placeholder for image analysis logic
    # In a real scenario, this would involve using libraries like OpenCV or TensorFlow
    return np.random.rand(1, 2)  # Simulated feature extraction

# Function to recommend personalized treatment plans
def recommend_treatment(patient_data):
    # Load pre-trained model
    model = joblib.load('treatment_model.pkl')

    # Prepare data for prediction
    features = np.array(patient_data).reshape(1, -1)

    # Predict treatment
    treatment = model.predict(features)
    return treatment[0]

# Example usage
if __name__ == "__main__":
```

```html
<form class="form">
    <div class="flex-row">
      <div>
      <input type="checkbox">
      <label>Remember me </label>
      </div>
      <span class="span">Forgot password?</span>
    </div>
    <button class="button-submit">Sign In</button>
    <p class="p">Don't have an account? <span class="span">Sign Up</span>

    </p><p class="p line">Or With</p>

    <div class="flex-row">
      <button class="btn google">
        <svg version="1.1" width="20" id="Layer_1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0p
<path style="fill:#FBBB00;" d="M113.47,309.408L95.648,375.94l-65.139,1.378C11.042,341.211,0,299.9,0,256
    c0-42.451,10.324-82.483,28.624-117.732h0.014l57.992,10.632l25.404,57.644c-5.317,15.501-8.215,32.141-8.215,49.456
    C103.821,274.792,107.225,292.797,113.47,309.408z"></path>
<path style="fill:#518EF8;" d="M507.527,208.176C510.467,223.662,512,239.655,512,256c0,18.328-1.927,36.206-5.598,53.451
    c-12.462,58.683-45.025,109.925-90.134,146.187l-0.014-0.014l-73.044-3.727l-10.338-64.535
    c29.932-17.554,53.324-45.025,65.646-77.911h-136.89V208.176h138.887L507.527,208.176L507.527,208.176z"></path>
<path style="fill:#28B446;" d="M416.253,455.624l0.014,0.014C372.396,490.901,316.666,512,256,512
    c-97.491,0-182.252-54.491-225.491-134.681l82.961-67.91c21.619,57.698,77.278,98.771,142.53,98.771
    c28.047,0,54.323-7.582,76.87-20.818L416.253,455.624z"></path>
<path style="fill:#F14336;" d="M419.404,58.936l-82.933,67.896c-23.335-14.586-50.919-23.012-80.471-23.012
    c-66.729,0-123.429,42.957-143.965,102.724l-83.397-68.276h-0.014C71.23,56.123,157.06,0,256,0
    C318.115,0,375.068,22.126,419.404,58.936z"></path>

</svg>

        Google

      </button><button class="btn apple">
<svg version="1.1" height="20" width="20" id="Capa_1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y=
```

```html
</svg>

        Google

      </button><button class="btn apple">
<svg version="1.1" height="20" width="20" id="Capa_1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y

        Apple

</button></div></form>
</body>
</html>
```

```css
.form {
  display: flex;
  flex-direction: column;
  gap: 10px;
  background-color: #ffffff;
  padding: 30px;
  width: 450px;
  border-radius: 20px;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;
}

::placeholder {
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;
}

.form button {
  align-self: flex-end;
}

.flex-column > label {
  color: #151717;
  font-weight: 600;
}

.inputForm {
  border: 1.5px solid #ecedec;
  border-radius: 10px;
  height: 50px;
  display: flex;
  align-items: center;
  padding-left: 10px;
  transition: 0.2s ease-in-out;
}

.input {
  margin-left: 10px;
```

```css
.input:focus {
  outline: none;
}

.i
Specifies how flex items are placed in the flex container, by setting the direction of the flex container's main axis.
}  (Edge 12, Firefox 20, Safari 9, Chrome 29, IE 11, Opera 12)
   Syntax: row | row-reverse | column | column-reverse
.f
   MDN Reference
  flex-direction: row;
  align-items: center;
  gap: 10px;
  justify-content: space-between;
}

.flex-row > div > label {
  font-size: 14px;
  color: black;
  font-weight: 400;
}

.span {
  font-size: 14px;
  margin-left: 5px;
  color: #2d79f3;
  font-weight: 500;
  cursor: pointer;
}

.button-submit {
  margin: 20px 0 10px 0;
  background-color: #151717;
  border: none;
  color: white;
  font-size: 15px;
```

```
  cursor: pointer;
}

.button-submit:hover {
  background-color: #252727;
Click to collapse the range.

.p {
  text-align: center;
  color: black;
  font-size: 14px;
  margin: 5px 0;
}

.btn {
  margin-top: 10px;
  width: 100%;
  height: 50px;
  border-radius: 10px;
  display: flex;
  justify-content: center;
  align-items: center;
  font-weight: 500;
  gap: 10px;
  border: 1px solid #ededef;
  background-color: white;
  cursor: pointer;
  transition: 0.2s ease-in-out;
}

.btn:hover {
  border: 1px solid #2d79f3;
  ;
}
```

```python
import numpy as np
import pandas as pd
import os
import pydicom
import joblib
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from cryptography.fernet import Fernet

# Function to load DICOM images and preprocess them
def load_dicom_images(directory):
    images = []
    for filename in os.listdir(directory):
        if filename.endswith('.dcm'):
            ds = pydicom.dcmread(os.path.join(directory, filename))
            images.append(ds.pixel_array / np.max(ds.pixel_array))
                # Normalized Image Data
    return np.array(images)

# Function to encrypt patient data securely
def encrypt_data(data, key):
    f = Fernet(key)
    encrypted_data = f.encrypt(data.encode())
    return encrypted_data
```

```python
    encrypted_data = f.encrypt(data.encode())
    return encrypted_data

# Generate encryption key (should be securely stored for decryption)
key = Fernet.generate_key()

# Function to generate synthetic patient data
def generate_patient_data(num_samples):
    data = {
        'age': np.random.randint(20, 80, num_samples),
        'genetics': np.random.choice(['A', 'B', 'C'], num_samples),
        'history': np.random.choice(['None', 'Diabetes',
            'Hypertension'], num_samples),
        'real_time_data': np.random.rand(num_samples)
    }
    return pd.DataFrame(data)
```

```python
# Generate patient data and encrypt it
patient_data = generate_patient_data(1000)
encrypted_patient_data = encrypt_data(patient_data.to_json(

# Train a model for personalized treatment recommendations
X = patient_data[['age', 'real_time_data']]
y = patient_data['history']

X_train, X_test, y_train, y_test = train_test_split(X, y, t
    =0.2, random_state=42)
model = RandomForestClassifier(n_estimators=100, random_sta
model.fit(X_train, y_train)

# Save trained model for future use
joblib.dump(model, 'treatment_model.pkl')

# Load the trained model and make predictions
def recommend_treatment(patient_data):
    model = joblib.load('treatment_model.pkl')
    features = np.array(patient_data).reshape(1, -1)
    treatment = model.predict(features)
    return treatment[0]
```