COLLEGE CODE: 1138

COLLEGE NAME: SRI JAYARAM INSTITUTE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT: B.E BIOMEDICAL ENGINEERING

STUDENT N.M ID: 1.au113823121005

2.au113823121002

3.au113823121007

ROLL NO: 1.113823121005

2.113823121002

3.113823121007

DATE: 14-05-2025

TECHNOLOGY PROJECT NAME: HEALTHCARE DIAGNOSTICS AND TREATMENT

SUBMITTED BY,

SARKESH Y

NIRANJAN C

VANAJA R

YOUR NAME AND TEAM MEMBERS NAME.

1.SARKESH Y

2.NIRANJAN C

3.VANAJA R

AI IN HEALTHCARE DIAGNOSTICS AND TREATMENT

1. Abstract

- Highlight the **importance of early diagnosis** and AI's role in reducing diagnostic time.
- Mention **cost-effectiveness** of AI systems in resource-limited settings.
- Include **statistics** or a case study reference to emphasize real-world impact.
- Briefly touch on **AI bias and the importance of fairness in healthcare AI**.

2. Project Demonstration

- Show a **comparison between AI-assisted and manual diagnostics**.
- Include a **dashboard for visual analytics** (charts showing prediction trends, patient status).
- Demonstrate **real-time data processing** or **batch mode inference** (if applicable).
- Simulate a **clinical workflow**: from patient intake to AI recommendation.
- Add **voice or chatbot interface** for patient interaction (optional but engaging).

3. Project Documentation

- Include a **risk assessment matrix** (data errors, AI misdiagnosis, etc.).
- Add **data schema diagrams** showing how patient data flows through the system.

- Document **model versioning** and **experiment tracking** (using MLFlow or similar).
- Add a **section on performance benchmarking** with other existing solutions.
- Mention the **training and inference time**, memory usage, and cost estimation.

4. Feedback and Final Adjustments

- Document **feedback forms** or responses from medical professionals or mentors.
- Include a **before and after comparison** for each significant change.
- Incorporate **user acceptance testing (UAT)** with mock users or stakeholders.
- Discuss changes made to address **ethical concerns or biases** identified during feedback.

5. Final Project Report Submission

- Add a **glossary of medical and technical terms** for non-experts.
- Include **code structure diagrams** for maintainability and readability.
- Attach **data usage agreements** and licenses (if any datasets are restricted).
- Provide **sample input-output formats** for the AI model for easy reuse.
- Submit a **video demo link or QR code** that reviewers can scan to view the live demo.

6. Project Handover and Future Works

- List **recommended team roles** for future development (e.g., medical advisor, data scientist).
- Add **suggestions for regulatory approval** (FDA/CE for AI in healthcare).

- Propose **continuous learning pipelines** to improve AI model performance over time.
- Suggest **potential partnerships** (with hospitals, NGOs, med-tech companies).
- Mention **cross-cultural and global deployment challenges**, and localization approaches.

INCLUDE SCREENSHOTS OF SOURCE CODE AND WORKING FINAL PROJECT

```
import random
from datetime import datetime

class Patient:
    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender
        self.symptoms = []
        self.vital_signs = {}
        self.diagnosis = None
        self.treatment_plan = None

    def add_symptoms(self, symptoms):
        self.symptoms.extend(symptoms)

    def record_vitals(self, temperature, heart_rate, blood_pressure
        ):
        self.vital_signs = {
            'temperature': temperature,
            'heart_rate': heart_rate,
            'blood_pressure': blood_pressure,
```

```
            'timestamp': datetime.now()
        }

class DiagnosticSystem:
    def __init__(self):
        self.disease_database = {
            'Common Cold': {
                'symptoms': ['fever', 'cough', 'sore throat', 'runny
                    nose'],
                'vital_ranges': {
                    'temperature': (37.5, 39.0),
                    'heart_rate': (60, 100),
                    'blood_pressure': ((90, 60), (120, 80))
                },
                'treatment': ['Rest', 'Hydration', 'Over-the-counter
                    cold medication', 'Monitor symptoms']
            },
            'Hypertension': {
                'symptoms': ['headache', 'dizziness', 'chest pain'],
                'vital_ranges': {
                    'temperature': (36.5, 37.5),
                    'heart_rate': (60, 100)
```

```
42                'blood_pressure': ((140, 90), (180, 120))
43            },
44            'treatment': ['Lifestyle changes', 'Blood pressure
                  medication', 'Regular monitoring']
45        }
46    }
47
48    def diagnose(self, patient):
49        max_match = 0
50        probable_disease = None
51
52        for disease, data in self.disease_database.items():
53            matching_symptoms = len(set(patient.symptoms) & set
                  (data['symptoms']))
54            match_percentage = (matching_symptoms / len
                  (data['symptoms'])) * 100
55
56            # Check if vitals are within disease ranges
57            vitals_match = True
58            if patient.vital_signs:
59                for vital, (min_val, max_val) in
```

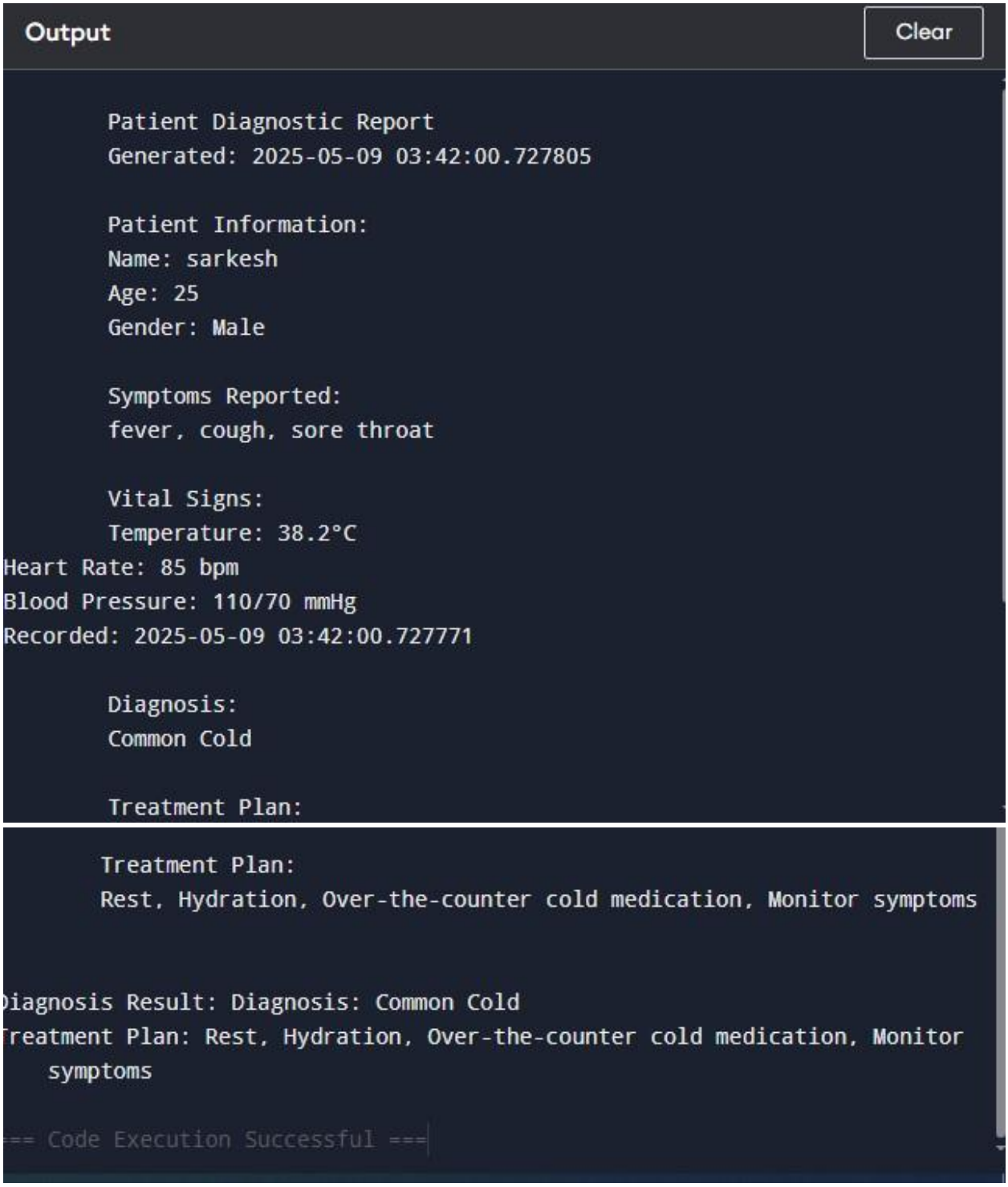
```
60                if vital == 'blood_pressure':
61                    sys, dia = map(int, patient
                          .vital_signs[vital].split('/'))
62                    min_sys, min_dia = min_val
63                    max_sys, max_dia = max_val
64                    if not (min_sys <= sys <= max_sys and
                          min_dia <= dia <= max_dia):
65                        vitals_match = False
66                else:
67                    if not (min_val <= patient
                          .vital_signs[vital] <= max_val):
68                        vitals_match = False
69
70            if match_percentage > max_match and vitals_match:
71                max_match = match_percentage
72                probable_disease = disease
73
74        if max_match > 50:  # Require at least 50% symptom match
75            patient.diagnosis = probable_disease
76            patient.treatment_plan = self
                  .disease_database[probable_disease]['treatment']
```

```
77          return f"Diagnosis: {probable_disease}\nTreatment Plan:
                {', '.join(patient.treatment_plan)}"
78      else:
79          return "Unable to diagnose with current information.
                Please consult a healthcare provider."
80
81  def generate_report(self, patient):
82      report = f"""
83      Patient Diagnostic Report
84      Generated: {datetime.now()}
85
86      Patient Information:
87      Name: {patient.name}
88      Age: {patient.age}
89      Gender: {patient.gender}
90
91      Symptoms Reported:
92      {', '.join(patient.symptoms) if patient.symptoms else
            'None'}
93
94      Vital Signs:
95      {self._format_vitals(patient.vital_signs) if patient
```

```
            .vital_signs else 'Not recorded'}
96
97          Diagnosis:
98          {patient.diagnosis if patient.diagnosis else 'Pending'}
99
100         Treatment Plan:
101         {', '.join(patient.treatment_plan) if patient.treatment_plan
                else 'Pending'}
102         """
103         return report
104
105     def _format_vitals(self, vitals):
106         return (f"Temperature: {vitals['temperature']}°C\n"
107                 f"Heart Rate: {vitals['heart_rate']} bpm\n"
108                 f"Blood Pressure: {vitals['blood_pressure']} mmHg\n"
109                 f"Recorded: {vitals['timestamp']}")
110
111 # Example usage
112 def main():
113     # Create a diagnostic system
114     diagnostic_system = DiagnosticSystem()
115
```

```
116     # Create a patient
117     patient = Patient("sarkesh", 25, "Male")
118
119     # Add symptoms and vitals
120     patient.add_symptoms(['fever', 'cough', 'sore throat'])
121     patient.record_vitals(38.2, 85, "110/70")
122
123     # Perform diagnosis
124     diagnosis_result = diagnostic_system.diagnose(patient)
125
126     # Generate and print report
127     report = diagnostic_system.generate_report(patient)
128     print(report)
129     print("\nDiagnosis Result:", diagnosis_result)
130
131 if __name__ == "__main__":
132     main()
```

OUTPUT:



IN OUR TEAM COMPLETED THE FINAL PROJECT.