Project Proposal: NeuraCareAI

Project Objective

NeuraCareAI is a unified AI health assistant that combines **custom-trained deep learning models** and **LLM-powered reasoning** to:

- Detect **brain tumors** from MRI scans using trained models.
- Offer **emotion-based mental health support** via webcam.
- Analyze **medical reports and images** to provide **next-step guidance**.
- Generate **personalized treatment plans** using user data.
- Predict **health risks** and suggest preventive actions.
- Enable **telehealth** interaction via an intelligent chatbot.

This project supports the **NeuraViaHacks** pillars:

- **Detect** → Brain tumor detection, risk prediction
- **Connect** → Chatbot, emotion detection, telehealth
- **Personalize** → Treatment plans, mental health nudges, user profiling

---

Features Overview

#	Feature	Tech Stack	Purpose
1	Brain Tumor Detection	Custom Trained CNN (PyTorch/TensorFlow)	Early detection of brain tumors from MRI
2	Medical Report Analyzer	LangChain + Ollama	Interpret MRI/X-ray findings and suggest next actions
3	Health Assistant Chatbot	LangChain + Ollama	Talk with users, answer medical queries
4	Health Risk Prediction	LLM + prompt logic	Predict user risks (stroke, diabetes, etc.)
5	Personalized Treatment Plan	LangChain + LLM	AI-generated wellness plan per user profile
6	Emotion Detection via Webcam	DeepFace + LLM	Detect emotion → Recommend self-care or talk to bot

#	Feature	Tech Stack	Purpose
7	Streamlit Interface	Streamlit	Clean UI for user interaction
8	Workflow Orchestration	LangGraph	Multi-step task chaining
9	Backend API	FastAPI	Interface between frontend, models, and DB
10	Database	SQL (PostgreSQL/SQLite)	Store all user data, scans, interactions

---

Technologies Used

Category	Tools
AI/ML	PyTorch/TensorFlow (Tumor Detection), DeepFace (Emotion)
LLMs	Ollama (Mistral/LLaMA3), LangChain
Workflow	LangGraph
Backend	FastAPI
Frontend	Streamlit
OCR	Tesseract (for report parsing)
Database	PostgreSQL or SQLite
Containerization	Docker (optional for deployment)

Feature Details

1. Brain Tumor Detection

- **Custom-trained model** (CNN on BRATS2020 or similar dataset)
- **Input:** MRI scan (DICOM/PNG/JPG)
- **Output:**

```
{
  "tumor_detected": true,
  "tumor_type": "Glioblastoma",
  "confidence": "94.2%"
}
```

- **Tech:** PyTorch/TensorFlow, FastAPI, Streamlit

---

2. Medical Report Analyzer

- **Input:** MRI summary, lab reports (text or OCR)
- **Process:** Extracts diagnosis, uses LLM for advice
- **Output:**

```
  {
    "diagnosis": "Left frontal meningioma",
    "recommendation": "Consult neurosurgeon. Avoid strenuous activity.'
  }
```

- **Tech:** LangChain + Ollama (LLM), Tesseract OCR

---

3. Health Assistant Chatbot

- LLM-based assistant trained on health prompts
- Memory-based, context-aware chat
- Tools: symptom lookup, health Q&A

---

4. Health Risk Prediction

- **Input:** Age, gender, lifestyle, history
- **LLM Prompt:** Predict risk of major diseases
- **Output:**

```
  {
    "stroke": "High Risk",
    "diabetes": "Moderate Risk"
  }
```

---

5. Personalized Treatment Plan

- LLM-generated plan using:
    - Age, gender, BMI
    - Diagnoses
    - Mental/emotional health
- Output:

```
{
  "Day 1": "Start Medication X, 30-minute walk",
  "Day 2": "MRI checkup, 2L hydration"
}
```

---

6. Emotion Detection via Webcam

- Real-time facial emotion recognition
- Output:

```
{
  "emotion": "sad",
  "confidence": 91.4%,
  "suggestion": "You seem down. Take a break, go for a walk or journa
}
```

- Tools: OpenCV + DeepFace + LangChain

---

7. LangGraph Workflow Example

```
graph TD
A[User Uploads MRI] --> B[Grok Model Predicts Tumor]
B --> C[Diagnosis + Confidence]
C --> D[LLM Suggests Next Steps]
D --> E[Treatment Plan Generated]
E --> F[Chatbot Follows Up with Support]
```

File Structure

```
neura-care-ai/
├── backend/
│   ├── main.py  ← FastAPI entry
│   ├── routes/
│   │   ├── scan.py
│   │   ├── report.py
│   │   ├── emotion.py
│   │   ├── chat.py
│   ├── services/
│   │   ├── tumor_model.py
│   │   ├── emotion_detector.py
│   │   ├── llm_tools.py
│   │   ├── report_parser.py
│   │   ├── treatment_planner.py
│   └── db/
│       ├── models.py
│       └── database.py
├── workflows/
│   └── pipeline.py  ← LangGraph orchestration
├── models/
│   └── trained_model.pth  ← Trained CNN
├── frontend/
│   ├── app.py  ← Streamlit app
│   └── components/
│       ├── uploader.py
│       ├── chatbot_ui.py
│       ├── emotion_webcam.py
│       └── report_input.py
├── requirements.txt
├── Dockerfile
└── README.md
```

Database Tables

```
users(id, name, age, email, gender, history)
scans(id, user_id, image_path, result, tumor_type, confidence)
reports(id, user_id, content, extracted_diagnosis, recommendation)
```

```
treatments(id, user_id, plan_json, created_at)
interactions(id, user_id, query, response)
emotions(id, user_id, emotion, confidence, timestamp)
```

Deployment Strategy

- **Dockerized App** for easy local/cloud deployment
- Optional GPU instance (AWS/GCP) for model inference
- Ollama model hosted locally or via LangChain agent
- Frontend served via **Streamlit Sharing**, or internal server

---

Security & Privacy

- JWT authentication
- Role-based access (Patient, Doctor, Admin)
- HIPAA-style handling of PHI
- Local inference ensures data doesn't leave system

Project Timeline

Week	Goal
1	FastAPI setup, user login, Streamlit UI
2	Train and integrate brain tumor model
3	Add LLM chatbot + LangChain memory
4	Report analysis & treatment planning
5	Emotion detection & recommendation
6	Full integration, testing, polish, deploy

Summary

NeuraCareAI is an innovative health-tech platform that:

- **Detects** critical health issues like brain tumors.

- **Connects** patients to support via chatbot & emotional AI.
- **Personalizes** care using language models and patient data.

It uses **real AI** — not just APIs — by training its own tumor detection model and combining it with the power of **LLMs, LangGraph, and LangChain**