Project Proposal: Medical Chatbot “Nebras”

1. Project Description

In modern healthcare systems, patients often struggle to access timely medical guidance due to limited staff, long waiting times, and inconsistent availability of healthcare professionals. To address these challenges, the project “Nebras” aims to develop a personalized, AI-powered medical chatbot capable of providing reliable, context-aware responses to health-related queries, symptom checks, and general wellness information. The chatbot leverages Natural Language Processing (NLP) and Generative AI to simulate human-like conversations, helping users receive accurate information instantly. It is not a substitute for doctors but rather a smart assistant to support patients and healthcare staff.

2. Team Members & Roles

Member Name	Role
Nader Mohamed Salama	Team Leader / Lead ML Engineer
Ahmed Hesham Mohamed	Data Engineer & Backend Developer
Hamdy Waleed Hamdy	Frontend & API Integration
Nada Ahmed Hamoda	NLP Engineer / Model Training
Nooran Abdulbasit Ali	Documentation & Quality Assurance
Israa Ahmed Abdulmohsen	Data Preprocessing & Evaluation

3. Objectives

1. Build an AI-driven chatbot capable of answering medical inquiries based on curated healthcare data.
2. Ensure accurate, reliable, and empathetic responses using fine-tuned medical language models.
3. Automate FAQ and symptom-based triage to assist patients 24/7.
4. Integrate medical data securely from JSON datasets.
5. Continuously improve chatbot performance through feedback and retraining loops.

4. Tools & Technologies

- Programming Languages: Python
- Frameworks & Libraries: PyTorch, Hugging Face Transformers, Flask / FastAPI
- NLP Tools: spaCy, NLTK
- Data Management: JSON, MySQL / MongoDB
- MLOps & Deployment: MLflow, Docker, GitHub Actions (CI/CD)
- Cloud Platforms: Google Colab, AWS / GCP
- Evaluation Metrics: BLEU, ROUGE, Perplexity, and Human Evaluation

5. Milestones & Timeline (3-Month Plan)

Phase	Weeks	Official Milestone	Key Deliverables
1. Foundation	1–3	M1: Data Collection & Preprocessing	Cleaned medical dataset, preprocessing documentation
2. Core Development	4–8	M2: Chatbot Development & Training	Fine-tuned medical chatbot, evaluation report
3. Enhancement	9–10	M3: Advanced Techniques & Integration	Enhanced model with attention, feedback-based learning pipeline
4. Productionizing	11	M4: MLOps & Deployment	Deployed chatbot API, CI/CD pipeline, MLflow tracking

Phase	Weeks	Official Milestone	Key Deliverables
5. Delivery	12	M5: Final Presentation & Demo	Final report, live demo, and documentation

6. Key Performance Indicators (KPIs)

KPI	Description	Metric / Evaluation
Data Preparation Quality	Measures dataset completeness, cleanliness, and balance.	% of valid entries after preprocessing
Model Performance & Accuracy	Evaluates chatbot accuracy and fluency in medical responses.	BLEU, ROUGE, F1-score
Pipeline Integration & Automation	Assesses automation of preprocessing, training, and inference.	Number of integrated steps in pipeline
MLOps & Deployment Readiness	Evaluates CI/CD, model tracking, and scalability.	Successful pipeline runs in MLflow & GitHub Actions
Output Quality & Usability	Assesses human evaluation of chatbot's medical responses.	User satisfaction score (1–5 scale)
Documentation & Presentation Quality	Checks completeness and clarity of documentation and presentation.	Mentor evaluation & presentation score

7. Expected Outcomes

- A functional medical chatbot capable of handling user queries on symptoms, medications, and health tips.
- Integration with structured JSON-based medical data for accuracy.
- Improved user experience through dynamic and context-aware responses.
- A scalable, deployable AI system following MLOps principles.
- Clear analytical dashboard tracking accuracy, latency, and user satisfaction.

8. Risk Management

Risk	Mitigation Strategy
Model training too slow	Use smaller medical models (e.g., BioGPT or DistilGPT2). Leverage Colab Pro GPUs.
Low response quality	Emphasize data cleaning and add response ranking/filtering. Conduct early human evaluation.
Complex integration	Build a simplified retraining pipeline using cron jobs if Airflow/Kubeflow is too heavy.
Deployment cost / access	Use free cloud tiers or containerize via Docker for easy portability.