Project Design Phase
Proposed Solution Template

Date	27 June 2025
Team ID	LTVIP2025TMID31381
Project Name	HealthAI
Marks	2 marks

4.2. Proposed Solution

The HealthAI application provides a user-friendly and intelligent platform for preliminary health insights, designed with Python, Streamlit, MongoDB, and the IBM Granite model, enabling personalized and secure interaction. Detailed Implementation of Core Features:

1. User Authentication (Sign-in/Sign-up):

- Implementation: User data, including hashed passwords (using hashlib.sha256 for basic security), is stored in a users collection in MongoDB.
- Streamlit UI: A dedicated login/sign-up section is presented in the Streamlit sidebar, controlled by st.sidebar.radio.
- Flow:

  ■ Sign Up: Users provide a new username and password. The register_user function hashes the password and inserts a new user document into the users collection. Built-in checks prevent duplicate usernames.

  ■ Login: Users provide credentials. The login_user function retrieves the user from MongoDB and verifies the input password against the stored hashed password.

  ■ Session Management: Streamlit's st.session_state is critically utilized to store the logged_in status, username, and the user_id (the MongoDB _id of the user document). This state persists across reruns within a single user session, enabling access control to

personalized features. A "Logout" button is provided to clear this session state.

2. Patient Management:

- Streamlit UI: A dedicated section (st.subheader) provides input fields for patient's name, age, and gender.
- Data Linking: When a new patient is added via add_patient_mongodb, their record is stored in the patients collection. Crucially, this patient document is explicitly linked to the user_id of the currently logged-in user, ensuring data ownership and privacy.
- ID Handling: The unique MongoDB ObjectId generated for each new patient is displayed to the user and conveniently stored in st.session_state['new_patient_id'] for easy pre-filling in other forms. Users are explicitly instructed to copy this 24-character hex string for linking records.
- Retrieval & Access Control: The get_patient_data_mongodb function retrieves patient documents, but critically includes a filter to ensure that only patients linked to the current user_id can be accessed, reinforcing the privacy model. Robust ObjectId.is_valid() checks are integrated for all patient ID inputs to prevent format errors.

3. Symptoms Identifier (Scenario 1):

- User Input: A st.text_area allows users to describe their symptoms, and a st.text_input is provided for the patient_id (pre-filled if a new patient was just added).
- Personalization: The backend symptoms_identifier_ui function fetches the patient's demographics (get_patient_data_mongodb) using the provided patient_id. This patient information (name, age, gender) is then seamlessly integrated into the LLM prompt, enabling the IBM Granite model to provide more contextually relevant potential conditions, likelihood assessments, and actionable next steps.

- LLM Integration: The generate_response function leverages the transformers.pipeline to efficiently take the constructed prompt and generate the AI's prediction.
- Persistence: The user's original symptom description and the AI's full, generated prediction are stored as a symptoms_prediction document in the health_records collection, precisely linked to the patient's ObjectId.

4. Home Remedies:

- User Input: A straightforward st.text_input allows the user to enter a disease name (e.g., "sore throat").
- LLM Role: The IBM Granite model is specifically prompted to act as an "AI assistant specialized in natural home remedies," generating a list of safe and common natural home remedies along with concise usage instructions.
- Disclaimer: A prominent medical disclaimer is consistently appended to every AI response, clearly stating that the remedies are traditional and should not replace professional medical advice.

5. Personalized Treatment Plans (Scenario 2):

- User Input: Text inputs for patient_id and the diagnosed condition.
- Personalization & LLM Role: Similar to the Symptoms Identifier, the generate_treatment_plan_ui function retrieves patient demographic data, which is then used to personalize the LLM prompt. The IBM Granite model is instructed to act as an "AI medical expert" and generate a comprehensive, evidence-based (but general) treatment plan. This plan specifically covers medication types (avoiding specific drug recommendations), broad lifestyle modifications (e.g., diet, exercise, stress management), and general suggestions for follow-up testing.
- Data Validation: The system strictly requires a valid patient_id that belongs to the logged-in user, ensuring the foundation for personalization is present.

- Persistence: The input diagnosed condition and the AI's complete generated treatment plan are securely saved as a treatment_plan record in the health_records collection.

6. Health Analytics Dashboard (Scenario 3):

- User Input: Text inputs for patient_id and a detailed textual description of vital signs or health trends over a specified period (e.g., "My blood pressure has been consistently around 140/90 mmHg for the past month. Heart rate averages 85 bpm and I feel tired.").
- LLM Role: The IBM Granite model acts as an "AI health analyst," interpreting the nuances of the textual description. It then provides insights into potential health concerns based on the described trends, highlights any identified patterns, and offers general recommendations for improvement.
- Future Enhancement: While the current implementation focuses on LLM-based textual analysis, a future enhancement could integrate Python libraries like numpy and pandas for processing actual numerical vital sign data, combined with charting libraries (e.g., Plotly, Matplotlib) to create a true data visualization dashboard.
- Persistence: The user's detailed description of vital signs and the AI's subsequent analysis are stored as a vitals_analysis record within the health_records collection.

7. Patient Chat Interface (Scenario 4):

- Streamlit Component: This feature elegantly uses Streamlit's native st.chat_input and st.chat_message components to provide a natural, conversational user interface.
- LLM Role & Context: The IBM Granite model serves as the conversational AI. The entire chat history (persisted within st.session_state.messages for the duration of the session) is passed to the LLM as context, enabling it to generate coherent and relevant responses that build upon previous turns in the conversation.

Project Design Phase
Proposed Solution Template

- Ethical AI: Each AI response is meticulously crafted to be empathetic, factual, and always includes the crucial medical disclaimer. This guides users responsibly, emphasizing that AI advice is not a substitute for professional medical consultation and encouraging them to seek qualified healthcare when necessary.
- Loading Optimization: The use of @st.cache_resource for the MongoDB client connection (get_mongodb_client) and the LLM loading (load_llm_model) is a key optimization. This ensures that these computationally expensive operations are performed only once when the Streamlit application starts or is first accessed, rather significantly improving perceived performance and resource efficiency.
- **Proposed Solution Template:**
- Project team shall fill the following information in the proposed solution template.

• **S.No.**	• **Parameter**	• **Description**
1	• Problem Statement (Problem to be solved)	unlimited access to users, personalized healthcare solution.
2	• Idea / Solution description	• HealthAI is a Python-powered generative AI assistant that delivers personalized symptom analysis, natural remedies, treatment plans, and health insights
3	• Novelty / Uniqueness	• HealthAI uniquely combines generative AI with

Project Design Phase
Proposed Solution Template

		real-time patient profiling and health data analysis to deliver personalized, conversational healthcare support in a fully Python-based environment.
4	• Social Impact / Customer Satisfaction	• HealthAI empowers individuals with instant, AI-driven healthcare guidance, enhancing accessibility, reducing uncertainty, and fostering user confidence in managing personal health.
5	• Business Model (Revenue Model)	• By adding subscription
6	• Scalability of the Solution	• HealthAI is highly scalable, with its modular Python architecture, Hugging Face integration, and database support enabling seamless

Project Design Phase
Proposed Solution Template

		expansion across users, features, and deployment platforms.