Functional & Performance Testing Template
Model Performance Test

Date	27 June 2025
Team ID	LTVIP2025TMID31381
Project Name	HealthAI
Marks	

5.1. GenAI Functional Test

Functional testing ensures that each component of the HealthAI application operates as intended, with a particular focus on the accuracy, relevance, and safety of the Generative AI outputs and their integration with the database.
Test Environment:

- Platform: Local Development Environment (e.g., VS Code with Python installed), capable of running Streamlit applications. GPU acceleration configured if available and desired for LLM performance.
- Python Version: Consistent with project requirements -(Python 3.x).
- Library Versions: All required libraries (transformers, torch, streamlit, pymongo, hashlib) are installed and compatible.
- MongoDB: An active MongoDB Atlas cluster (recommended for ease of setup and accessibility) or a locally running MongoDB instance that the application can connect to using the provided environment variables.
  Test Cases:

1. User Authentication:

➢ Sign Up - Success:

■ Action: In the sidebar, select "Sign Up", enter a new, unique username and a password. Click "Sign Up" button.

■ Expected: "Registration successful. Please log in." message is displayed. Verify a new user document (containing the username and hashed password) exists in the MongoDB users collection.

➢ Sign Up - Duplicate Username:

■ Action: Attempt to sign up with a username that is already registered.

■ Expected: "Username already exists." error message is displayed. No new user document should be created in MongoDB.

- Login - Success:

■ Action: Select "Login", enter the valid username and password of a registered user. Click "Login" button.

■ Expected: "Login successful." message is displayed in the sidebar. The main content area should change to display the application's functional tabs, and st.session_state should show logged_in=True, username, and user_id.

- Login - Invalid Credentials:

■ Action: Attempt to log in with an incorrect username or password.

■ Expected: "Invalid username or password." error message is displayed. The UI should remain on the login/sign-up screen, and st.session_state should remain logged_in=False.

- Logout:

■ Action: After logging in, click the "Logout" button in the sidebar.

■ Expected: The st.session_state variables related to login (logged_in, username, user_id) are cleared, and the UI reverts to the login/sign-up screen.

2. Patient Management:

- Pre-condition: User is logged in.
- Add Patient - Success:

■ Action: In the "Patient Management" section, enter valid Name, Age, and select Gender. Click "Add Patient".

■ Expected: "Patient added successfully." message is displayed. A new Patient ID (a 24-character hex string, e.g., 60c72b2f9f1b2c3d4e5f6a7b) is displayed in the UI and also pre-fills the "Enter Patient ID" field. Verify a new

patient document exists in the MongoDB patients collection, correctly linked to the logged-in user_id.

- Add Patient - Missing Name:

  ■ Action: Leave the "Patient Name" field blank. Click "Add Patient".

  ■ Expected: "Please enter a patient name." warning message is displayed. No patient document should be added to MongoDB.

- Retrieve Patient Data - Valid ID (Owned by User):

  ■ Action: Enter a valid Patient ID that was previously added by the current logged-in user into the "Enter Patient ID" field. Click "Get Patient Data".

  ■ Expected: All stored patient details (ID, Name, Age, Gender, Created At timestamp) are displayed accurately in the "Patient Details" text box.

- Retrieve Patient Data - Invalid ID Format:

  ■ Action: Enter a string that is not a valid 24-character hexadecimal MongoDB ObjectId (e.g., "123", "abc", too short/long hex string). Click "Get Patient Data".

  ■ Expected: Error message: "'...' is not a valid MongoDB ObjectId. Please provide a 24-character hex string."

- Retrieve Patient Data - Non-existent ID (Valid Format):

  ■ Action: Enter a valid-format Patient ID (24-character hex string) that does not exist in the database. Click "Get Patient Data".

  ■ Expected: Error message: "Error: Patient not found or None." ○ Retrieve Patient Data - ID Not Owned by User (Conceptual):

  ■ Action: (This test would require two distinct user accounts). Logged in as User A, try to retrieve a Patient ID that was created by User B.

  ■ Expected: Error message: "Error: Patient not found or None." (due to the user_id filter applied in get_patient_data_mongodb, ensuring data isolation).

Functional & Performance Testing Template
Model Performance Test

3. Symptoms Identifier (Scenario 1):

- Pre-condition: User is logged in.
- Test Case 1: Common Symptoms, Existing Patient:

  ■ Action: Ensure a valid Patient ID is entered. Input descriptive symptoms: "persistent headache, fatigue, mild fever". Click "Predict Disease".

  ■ Expected: AI output (from IBM Granite model) includes 2-3 plausible potential conditions (e.g., Common Cold, Flu, Migraine). A likelihood assessment (e.g., "medium probability") is provided for each. Clear "recommended next steps" (e.g., "consult a doctor if symptoms worsen, rest and hydrate") are included. The standard medical disclaimer is present. Verify that the symptom description and the AI's full prediction are stored as a new symptoms_prediction record in MongoDB's health_records collection, correctly linked to the specified patient.

- Test Case 2: Vague Symptoms:

  ■ Action: Enter a valid Patient ID. Symptoms: "I feel generally unwell." Click "Predict Disease".

  ■ Expected: AI provides more general potential predictions (e.g., viral infection, stress), still with next steps and the disclaimer.

- Test Case 3: Missing Patient ID:

  ■ Action: Leave "Your Patient ID" blank. Input symptoms. Click "Predict Disease".

  ■ Expected: "Please enter your Patient ID." error message.

- Test Case 4: Invalid Patient ID Format:

  ■ Action: Enter an invalid Patient ID format (e.g., "not_an_id"). Click "Predict Disease".

  ■ Expected: "Error: Invalid Patient ID format: 'not_an_id'. Please provide a 24-character hex string MongoDB ID." error message.

- Test Case 5: Patient ID Not Found (Valid Format):

■ Action: Enter a valid-format but non-existent Patient ID. Click "Predict Disease".

■ Expected: "Patient ID ... not found or None. Proceeding with generic profile." warning, followed by an AI response based on generic context. (Record will not be saved for this patient).

- Test Case 6: No Symptoms Input:

■ Action: Enter valid Patient ID. Leave "Describe your symptoms" blank. Click "Predict Disease".

■ Expected: "Please describe your symptoms." warning message.

4. Home Remedies:

- Pre-condition: User is logged in.
- Test Case 1: Common Ailment:

■ Action: In the "Home Remedies" section, enter disease: "sore throat". Click "Get Home Remedies".

■ Expected: AI (IBM Granite model) generates a list of relevant, common home remedies (e.g., salt water gargle, honey and lemon tea, ginger tea) with brief explanations of their usage. The standard medical disclaimer is included.

- Test Case 2: Serious Condition:

■ Action: Enter disease: "heart attack". Click "Get Home Remedies".

■ Expected: AI should strongly advise seeking immediate professional medical attention and provide very minimal or no specific home remedies, emphasizing the severity of the condition and reinforcing the disclaimer.

- Test Case 3: No Disease Input:

■ Action: Leave the disease input blank. Click "Get Home Remedies".

■ Expected: "Please enter a disease name." warning message.

5. Personalized Treatment Plans (Scenario 2):

Functional & Performance Testing Template
Model Performance Test

- Pre-condition: User is logged in.
- Test Case 1: Diagnosed Condition, Existing Patient:

  ■ Action: In the "Treatment Plans" section, enter a valid Patient ID and a common diagnosed condition: "Type 2 Diabetes". Click "Generate Treatment Plan".

  ■ Expected: AI (IBM Granite model) provides a comprehensive general plan. This plan should include sections for medication types (e.g., "oral hypoglycemics," "insulin"), lifestyle modifications (e.g., specific dietary advice, exercise recommendations, stress management techniques), and suggested follow-up testing (e.g., "HbA1c tests"). The response explicitly emphasizes that this is a general plan and requires a doctor's consultation for specific prescriptions and tailored advice, along with a strong medical disclaimer. Verify that the input condition and the AI's generated plan are saved as a treatment_plan record in MongoDB.

- Test Case 2: Invalid Patient ID:

  ■ Action: Enter an invalid Patient ID format. Click "Generate Treatment Plan".

  ■ Expected: "Error: Invalid Patient ID format..." error message.

- Test Case3: Patient ID Not Found:

  ■ Action: Enter a valid-format but non-existent Patient ID. Click "Generate Treatment Plan".

  ■ Expected: "Patient ID ... not found or None. A personalized treatment plan requires valid patient data." error message.

- Test Case 4: No Condition Input:

  ■ Action: Enter a valid Patient ID. Leave the "diagnosed condition" blank. Click "Generate Treatment Plan".

  ■ Expected: "Please enter a diagnosed condition." warning message.

6. Health Analytics Dashboard (Scenario 3):

Functional & Performance Testing Template
Model Performance Test

- Pre-condition: User is logged in.
- Test Case 1: Descriptive Vitals with Trends, Existing Patient:

  ■ Action: In the "Health Analytics" section, enter a valid Patient ID. Input vital signs description: "My blood pressure has been consistently around 140/90 mmHg for the past month. Heart rate averages 85 bpm and I feel tired and lethargic." Click "Analyze Health Trends".

  ■ Expected: AI (IBM Granite model) identifies the elevated BP and HR, comments on reported fatigue/lethargy, suggests potential health concerns associated with these trends, and offers general recommendations for improvement (e.g., "consult a doctor for high BP management," "consider sleep hygiene and stress reduction"). Includes the medical disclaimer. Verify that the vital signs description and the AI's analysis are stored as a vitals_analysis record in MongoDB.

- Test Case 2: Invalid Patient ID:

  ■ Action: Enter an invalid Patient ID format. Click "Analyze Health Trends".

  ■ Expected: "Error: Invalid Patient ID format..." error message.

- Test Case 3: Patient ID Not Found:

  ■ Action: Enter a valid-format but non-existent Patient ID. Click "Analyze Health Trends".

  ■ Expected: "Patient ID ... not found or None. Proceeding with generic profile." warning, followed by an AI response based on generic context. (Record will not be saved for this patient).

- Test Case 4: No Vitals Input:

  ■ Action: Enter a valid Patient ID. Leave the "Describe your vital signs" text area blank. Click "Analyze Health Trends".

  ■ Expected: "Please describe your vital signs and health trends." warning message.

7. Patient Chat Interface (Scenario 4):

Functional & Performance Testing Template
Model Performance Test

- Pre-condition: User is logged in.
- Test Case 1: General Health Question:

  ■ Action: In the "Patient Chat" section, type "What are the common symptoms of flu?" in the chat input. Press Enter.

  ■ Expected: AI (IBM Granite model) provides factual information about common flu symptoms. The response is empathetic and includes the standard medical disclaimer. The conversation is added to the chat history.

- Test Case 2: Request for Diagnosis:

  ■ Action: Type "Do I have strep throat?" in the chat.

  ■ Expected: AI explicitly states it cannot diagnose medical conditions, strongly advises seeking professional medical help, and may provide general information about strep throat symptoms.

- Test Case 3: Follow-up Question (Context Retention):

  ■ Action: After Test Case 1 (asking about flu symptoms), type "What about prevention?"

  ■ Expected: AI provides prevention tips specifically for flu, demonstrating that it retained context from the previous turn in the conversation.

- Test Case 4: Non-Medical Query:

  ■ Action: Type "What's the best movie to watch tonight?" in the chat.

  ■ Expected: AI gracefully deflects the query, stating that it's outside its scope or redirects the conversation back to health-related topics.

- Test Case 5: Empty Input:

  ■ Action: Submit an empty chat input.

  ■ Expected: No new message should be added to the chat history, and no error should occur. These functional test cases comprehensively cover the primary user interactions and backend logic, including user authentication, database persistence (for patient and health records), LLM responsiveness, and robust error handling. For a production-grade application, these would

Functional & Performance Testing Template
Model Performance Test

be supplemented with automated unit tests, more extensive integration tests, and formal performance benchmarks.

Test Scenarios & Results

Test Case ID	Scenario (What to test)	Test Steps (How to test)	Expected Result	Actual Result	Pass/Fail
FT-01	Text Input Validation (e.g., topic, job title)	Enter valid and invalid text in input fields	Valid inputs accepted, errors for invalid inputs	Valid inputs are accepted, Errors are invalid	pass
FT-02	Number Input Validation (e.g., word count, size, rooms)	Enter numbers within and outside the valid range	Accepts valid values, shows error for out-of-range	Accepts valid values, shows error for out-of-range	pass
FT-03	Content Generation (e.g., blog)	Provide complete inputs and click "Generate"	Correct content is generated based on input	Yes content is generated	pass
FT-04	API Connection Check	Check if API key is correct and model responds	API responds successfully	Api is Successfully responds	pass

Functional & Performance Testing Template
Model Performance Test

PT-01	Response Time Test	Use a timer to check content generation time	Should be under 3 seconds	2 sec	Pass
PT-02	API Speed Test	Send multiple API calls at the same time	API should not slow down	Not slowed	pass