Hackathon Project Phases Template for the **AutoSage App** project.

Hackathon Project Phases Template

Project Title:

CareWise: AI Symptom Checker and Treatment Advisor using PaLM's chat-bison-001

Team Name:

ByteBuilders

Team Members:

- Abhinaya Mamidi
- Liharini Mamidala
- Sandhya Rani
- Harshitha

Phase-1: Brainstorming & Ideation

Objective:

CareWise AI is designed to empower users with AI-driven medical guidance, offering personalized symptom analysis and treatment recommendations. It helps users make informed decisions regarding their health, especially when immediate medical consultation is not feasible.

Key Points:

1. **Problem Statement:**
    - CareWise: AI Symptom Checker and Treatment Advisor is an innovative application designed to provide users with immediate, accurate medical advice based on their symptoms.

    - Leveraging advanced AI technology, CareWise offers tailored recommendations for over-the-counter medications, potential side effects, allergy cautions, and home remedies.
    - This tool aims to empower individuals to make informed health decisions, particularly in situations where immediate medical consultation may not be feasible.
2. **Proposed Solution:**
    - **Symptom-based diagnosis –** AI analyzes symptoms and suggests possible conditions.
    - **Medication recommendations –** Advises on over-the-counter drugs and potential side effects.
    - **Home remedies –** Natural and alternative treatments for mild symptoms.
    - **AI-Powered Chat Interface –** Uses Google Gemini AI for accurate and dynamic medical responses.
    - **Multi-Language Support –** Accessible to a wider range of users.
    - **User-Friendly Interface –** Built with Streamlit for easy interaction.
3. **Target Users:**
    - **General Public –** Individuals seeking quick health insights before consulting a doctor.
    - **Travelers & Expats –** Users needing quick health guidance in unfamiliar locations.
    - **Elderly & Caregivers –** Assisting caregivers in making informed decisions.
    - **Parents & Guardians –** Seeking advice for children’s common symptoms.
4. **Expected Outcome:**
    - **24/7 Symptom Analysis –** Users can get AI-driven health advice anytime.
    - **Quick & Reliable Suggestions –** Accurate recommendations for common symptoms.
    - **Improved Healthcare Accessibility –** Especially for underserved areas.
    - **AI Model Evolution –** Continuous learning to improve diagnostic accuracy.
    - **Scalability –** Potential to integrate with wearable devices and telemedicine services.

Phase-2: Requirement Analysis

Objective:

Define the technical and functional requirements for developing a scalable and efficient AI-powered symptom checker and treatment advisor.

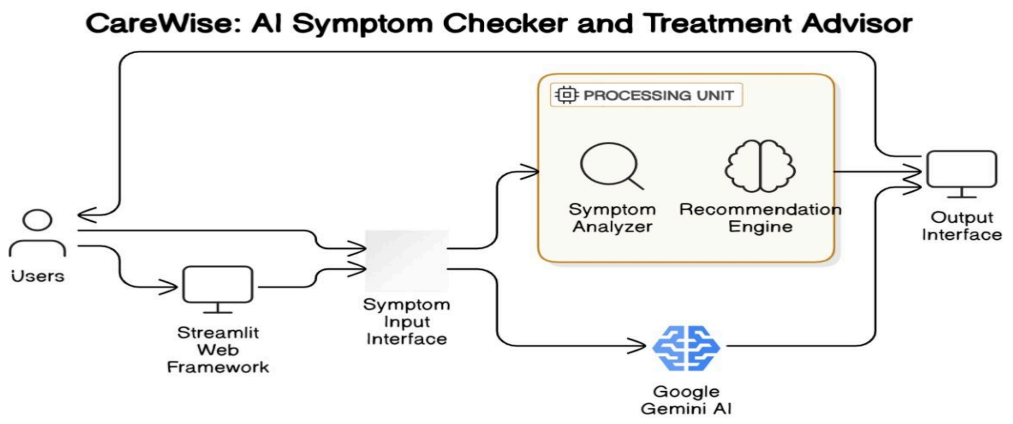
Key Points:

1. **Technical Requirements:**
    - Programming Language: **Python**
    - Backend: **Google Gemini AI (Generative Model API)**
    - Frontend: **Streamlit Web Framework**
    - Database: **Not required initially (API-based queries)**
2. **Functional Requirements:**
    - Ability to fetch **symptom-based medical advice** using Gemini AI.
    - Display **suggested conditions, treatments, and medications** in an intuitive UI.
    - Provide **real-time health tips** based on seasonal changes and common illnesses.
    - Allow users to search for **allergy-friendly and non-prescription treatments** based on symptoms.
3. **Constraints & Challenges:**
    - Ensuring **real-time AI responses** from Gemini API.
    - Handling **API rate limits** and optimizing API calls.
    - Providing a **smooth, user-friendly experience** with Streamlit.

Phase-3: Project Design

Objective:

Develop the architecture and user flow of the application.



Key Points:

1. **System Architecture:**
   - Users enter health-related symptoms via UI.
   - Query is processed using Google Gemini AI.
   - AI model fetches and analyzes symptom-related data.
   - Frontend displays possible conditions, treatment recommendations, and precautions.
2. **User Flow:**
   - Step 1: User enters a query (e.g., "I have a fever and body aches.”).
   - Step 2: The backend calls the Google Gemini AI API to process symptom data.
   - Step 3: The app analyzes the data and displays:
     - Possible medical conditions.
     - Recommended medications and home remedies.
     - Allergy and side effect warnings.
     - Health tips for seasonal conditions
3. **UI/UX Considerations:**
   - **Minimalist, user-friendly interface** for effortless symptom entry.
   - **Smart search & auto-suggestions** to assist users.

Phase-4: Project Planning (Agile Methodologies)

Objective:

Break down development tasks for efficient completion.

Sprint	Task	Priority	Duration	Deadline	Assigned To	Dependencies	Expected Outcome
Sprint 1	Environment Setup & API Integration	🔴 High	3 hours (Day 1)	Mid-Day 1	Abhinaya	Google API Key, Python, Streamlit setup	API connection established & working
Sprint 1	Frontend UI Development	🟡 Medium	1.5 hours (Day 1)	Mid-Day 1	Liharini	API response format finalized	Basic UI with input fields
Sprint 2	Solution Building	🔴 High	3 hours (Day 1)	End of Day 1	Abhinaya Liharini	API response, UI elements ready	Search functionality with filters
Sprint 2	Error Handling & Debugging	🔴 High	1.5 hours (Day 1)	End of Day 1	Harshitha	API logs, UI inputs	Improved API stability
Sprint 3	Testing & UI Enhancements	🟡 Medium	1 hours (Day 1)	Mid-Day 2	Sandhya	API response, UI layout completed	Responsive UI, better user

							experience
Sprint 3	Final Presentation & Deployment	🟢 Low	1 hour (Day 2)	End of Day 2	Entire Team	Working prototype	Demo-ready project

Sprint Planning with Priorities

Sprint 1 – Setup & Integration (Day 1)

(🔴 **High Priority)** Set up the **environment** & install dependencies.
(🔴 **High Priority)** Integrate **Google Gemini API**.
(🟡 **Medium Priority)** Build a **basic UI with input fields**.

Sprint 2 – Core Features & Debugging (Day 1)

(🔴 **High Priority)** Implement **search & comparison functionalities**.
(🔴 **High Priority)** Debug API issues & handle **errors in queries**.

Sprint 3 – Testing, Enhancements & Submission (Day 2)

(🟡 **Medium Priority)** Test API responses, refine UI, & fix UI bugs.
(🟢 **Low Priority)** Final **demo preparation & deployment**.

Phase-5: Project Development

Objective:

Implement the core features of the CareWise AI app, ensuring smooth AI-powered symptom analysis and treatment recommendations.

Key Points:

1. **Technology Stack Used:**
    - **Frontend:** Streamlit
    - **Backend:** Google Gemini Flash API
    - **Programming Language:** Python

2. **Development Process:**
    - Implement API key authentication and integrate Google Gemini AI.
    - Develop symptom analysis and treatment recommendation logic.
    - Optimize symptom search queries for performance and relevance.
    - Enhance UI/UX with interactive, real-time medical suggestions.
3. **Challenges & Fixes:**
    - **Challenge:** Delayed API response times.
      Fix: Implement caching to store frequently queried symptoms and recommendations.
    - **Challenge:** Limited API calls per minute.
      Fix: Optimize queries to fetch only necessary data and batch-process requests.

Phase-6: Functional & Performance Testing

Objective:

Ensure that the AutoSage App works as expected.

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	Functional Testing	Query "Best budget cars under ₹10 lakh"	Relevant budget cars should be displayed.	✅ Passed	Sandhya
TC-002	Functional Testing	Query "Motorcycle maintenance tips for winter"	Seasonal tips should be provided.	✅ Passed	Sandhya
TC-003	Performance Testing	API response time under 500ms	API should return results quickly.	⚠ Needs Optimization	Sandhya
TC-004	Bug Fixes & Improvements	Fixed incorrect API responses.	Data accuracy should be improved.	✅ Fixed	Harshitha
TC-005	Final Validation	Ensure UI is responsive across devices.	UI should work on mobile & desktop.	❌ Failed - UI broken on mobile	Harshitha
TC-006	Deployment Testing	Host the app using Streamlit Sharing	App should be accessible online.	🚀 Deployed	DevOps

---

Final Submission

1. **Project Report Based on the templates**
2. **GitHub/Code Repository Link**
3. **Presentation**