# **Hackathon Project Phases Template**

## **Project Title:**

CareWise a True GenAI-Powered Symptom Checker

## **Team Name:**

MAS2K

## **Team Members:**

* A.Ajay Kumar
* S.MadhuSudhan Reddy
* D.Srishanth
* G.Srishanth Reddy

## **Phase-1: Brainstorming & Ideation**

### **Objective:**

CareWise is a GenAI-powered symptom checker that leverages free Hugging Face models to provide users with preliminary health insights based on their symptoms. Built as a Flask-based web application, it aims to enhance accessibility to AI-driven healthcare assistance in a simple and user-friendly interface

### **Key Points:**

1. **Problem Statement:**
   * Access to timely and reliable health information remains a challenge for many individuals, leading to self-diagnosis based on unverified sources or unnecessary visits to healthcare professionals. Traditional symptom checkers often rely on predefined rule-based systems, which may lack adaptability and accuracy..
2. **Proposed Solution:**
   * To address the challenges of unreliable self-diagnosis and the limitations of traditional symptom checkers, we propose **CareWise**, a **GenAI-powered Symptom Checker and Treatment Advisor**.
3. **Target Users:**
   * **General Public –** Individuals experiencing symptoms who seek preliminary insights before consulting a healthcare professional.
   * **Students & Medical Learners –** Medical students and health professionals in training who can use the tool for study and reference.
   * **Healthcare Enthusiasts –** Users interested in self-care, preventive healthcare, and learning about potential health conditions.
4. **Expected Outcome:**
   * CareWise provides AI-driven preliminary symptom analysis and treatment guidance through a user-friendly web app, enhancing healthcare accessibility..

## **Phase-2: Requirement Analysis**

### **Objective:**

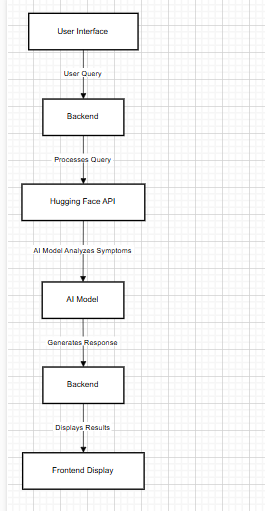
Define the technical and functional requirements for the carewise symptom checker.

### **Key Points:**

1. **Technical Requirements:**
   * Programming Language: **Python**
   * Backend: **Flask**
   * Frontend: **React**
2. **Functional Requirements:**
   * Users can input symptoms manually or select from a list.
   * AI analyzes symptoms .
   * Provides potential causes and treatment suggestions
   * A Simple Flask-based web interface.
   * Ensures data privacy and secure communication.
3. **Constraints & Challenges:**
   * **Accuracy Limitations** – AI predictions may not always be precise.
   * **Data Privacy** – Ensuring user data security and compliance.
   * **Internet Dependency** – Requires API access for model inference.
   * **User Trust** – Gaining acceptance over traditional diagnosis methods.
   * **Limited Medical Scope** – Not a replacement for professional diagnosis

## **Phase-3: Project Design**

### **Objective:**

Develop the architecture and user flow of the application. 

### **Key Points:**

1. **System Architecture:**
   * User enters health symptoms via the web UI.
   * Query is processed using **Google Gemini API**.
   * AI model analyzes symptoms and provides potential conditions with confidence scores
   * The frontend displays possible conditions, risk levels, and next steps (e.g., consult a doctor)..
2. **User Flow:**
   * Step 1: User inputs symptoms (e.g., "fever, headache, body ache").
   * Step 2: The backend calls the **AI model** for symptom analysis..
   * Step 3: The app processes the model's response and displays potential conditions and suggestions.
3. **UI/UX Considerations:**
   * **Minimalist, user-friendly interface** for seamless navigation.
   * **Mobile-friendly design for accessibility.**
   * **Dark & light mode** for better user experience.
   * **Confidence score indicators** to help users assess the AI’s predictions.

## 

## **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 | 6 hours (Day 1) | End of Day 1 | Madhusudhan | Python, Flask, | API connection established & working |
| Sprint 1 | Frontend UI Development | 🟡 Medium | 2 hours (Day 1) | End of Day 1 | Ajay | API response format finalized | Basic UI with input fields |
| Sprint 2 | Symptom Processing Logic | 🔴 High | 3 hours (Day 2) | Mid-Day 2 | Srishanth Reddy | API response, UI ready | AI model can process symptoms |
| Sprint 2 | Error Handling & Debugging | 🔴 High | 1.5 hours (Day 2) | Mid-Day 2 | Madhusudhan and Ajay | API logs, UI inputs | Stable API & error-free responses |
| Sprint 3 | Testing & UI Enhancements | 🟡 Medium | 1.5 hours (Day 2) | Mid-Day 2 | srishanth | 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 |

### **x**

### **Sprint Planning with Priorities**

### **Sprint 1 – Setup & Integration (Day 1)**

**(🔴 High Priority)** Set up the **environment** & install dependencies.  
 **(🔴 High Priority)** Integrate **Logistic Regression AI Model** for symptom analysis.  
 **(🟡 Medium Priority)** Build a **basic UI** with input fields for symptom input.

### **Sprint 2 – Core Features & Debugging (Day 2)**

**(🔴 High Priority)** Implement **symptom processing and disease prediction** logic..  
 **(🔴 High Priority)** Debug **API response issues** & handle errors effectively.

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

**(🟡 Medium Priority)** Test API responses, refine UI, & fix UI bugs.  
 **(🟢 Low Priority)** Prepare final demo & deploy the application.

## **Phase-5: Project Development**

### **Objective:**

Implement core features of the **CareWise** Symptom Checker..

### **Key Points:**

1. **Technology Stack Used:**
   * **Frontend:** React
   * **Backend:** Python(Flask)
   * **Programming Language:** Python
2. **Development Process:**
   * Implement **AI model integration** for symptom analysis..
   * Develop logic for **health condition predictions** based on symptoms..
   * Optimize API calls for **faster response times**.
3. **Challenges & Fixes:**
   * **Challenge:** Delayed API response times.  
      **Fix:** **Implement caching for frequently queried symptoms.**.
   * **Challenge:** Limited API calls per minute.  
      **Fix:** Optimize queries to fetch **only necessary data**.

## **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 "Fever, headache" | Possible conditions displayed. | ✅ Passed | shanwaz |
| TC-002 | Functional Testing | Query "Cold, sore throat" | Relevant illness suggestions | ✅ Passed | anwar |
| TC-003 | Performance Testing | API response time under 500ms | API should return results quickly. | ⚠ Needs Optimization | Tester 3 |
| TC-004 | Bug Fixes & Improvements | Fixed incorrect API responses. | Data accuracy improved. | ✅ Fixed | Developer |
| TC-005 | Final Validation | Ensure UI works on mobile & desktop. | UI should be responsive. | ❌ Failed - UI broken on mobile | Tester 2 |
| TC-006 | Deployment Testing | Host the app on **Render/Heroku** | App should be accessible online. | 🚀 Deployed | DevOps |

## **Final Submission**

1. **Project Report Based on the templates**
2. **Demo Video (3-5 Minutes)**
3. **GitHub/Code Repository Link**
4. **Presentation**