**Proof of Concept (PoC) definition** for a **Symptom Checker** application:

**1. Objective**

To build a PoC that allows users to input symptoms and receive basic health advice. This PoC will serve as a demonstration of the feasibility of delivering symptom-based suggestions, using a rule-based or data-driven approach, without implementing a full-scale production system.

**2. Core Features**

**2.1 User Symptom Input**

* Users can enter symptoms as free text or select from a predefined list.
* Input example:  
  "fever, sore throat, headache"

**2.2 Symptom Matching**

* The system uses a simple rule-based or pre-trained model to match symptoms with predefined conditions.
* Example:
  + If the input includes "fever" and "body ache", match it with "Influenza (Flu)".

**2.3 Health Advice**

* Provide general health advice based on the matched condition.
* Example output for "Flu":  
  "Rest, stay hydrated, and consult a healthcare provider if symptoms worsen."

**2.4 Optional Feedback Mechanism**

* Users can indicate if the advice was helpful for continuous improvement in later stages.

**3. Scope**

The PoC will demonstrate:

* User interaction through a simple, responsive frontend.
* Basic symptom-to-condition matching logic in the backend.
* Retrieval of predefined health advice linked to conditions.

**4. Assumptions**

* The PoC uses a small, static dataset of symptoms and conditions.
* Matching is rule-based or utilizes basic classification (without AI-based diagnosis refinement).
* Medical accuracy is not guaranteed, and the system provides general suggestions only.

**5. Non-Functional Requirements**

* **Usability**: Simple and intuitive UI.
* **Performance**: Handle multiple symptom inputs within minimal response time.
* **Security**: Basic input validation to avoid script injection.

**6. Architecture Overview**

| **Component** | **Description** |
| --- | --- |
| **Frontend** | Provides the user interface for symptom input and displays health advice results. |
| **Backend** | Receives symptoms, processes the input, and returns advice. |
| **Data Store** | A simple database or JSON file with symptom-condition-advice mappings. |

**Frontend**

* Technology: **React.js**, **HTML/CSS/JS**, or **Flutter** for mobile.
* Features:
  + **Input Field**: Free-text for symptoms.
  + **Submit Button**: Triggers symptom analysis.
  + **Results Section**: Displays matched conditions and corresponding advice.

**Backend**

* Technology: **Flask** or **FastAPI** (Python) / **Node.js (Express)**.
* Endpoints:
  + POST /check-symptoms: Accepts symptoms and returns health advice.

**Example Backend Flow**:

1. Receive a POST request with symptoms.
2. Tokenize and normalize input.
3. Match symptoms to conditions in a static dataset.
4. Return condition name and health advice.

**Example Static Data Structure**

json

CopyEdit

[ { "condition": "Common Cold", "symptoms": ["cough", "sore throat", "runny nose"], "advice": "Drink plenty of fluids, rest, and use over-the-counter cold remedies." }, { "condition": "Flu", "symptoms": ["fever", "chills", "body aches"], "advice": "Rest, stay hydrated, and consult a doctor if severe symptoms develop." } ]

**Development Steps**

**1. Frontend**

* Create a symptom input field.
* Add a button to submit symptoms.
* Display matched conditions and advice.

**2. Backend**

* Develop a rule-based symptom matching system.
* Create a response format for conditions and advice.

**3. Testing**

* Validate correct condition matching for different inputs.
* Handle cases where no condition is found.

**Success Criteria**

1. Users can input symptoms and receive related conditions with advice.
2. Responses are generated within a reasonable time (under 1 second).
3. Basic error handling for invalid or empty inputs.

**Limitations of the PoC**

* Limited dataset: Only a few conditions and symptoms.
* No real-time learning or AI diagnosis.
* General, non-personalized health advice.

**Example Interaction**

1. **Input**: "fever, cough, fatigue"
2. **Output**:
   * Conditions: "Flu"
   * Advice: "Rest, drink fluids, and consult a doctor if symptoms persist."

**Enhancements for Future Versions**

* **Dynamic Learning Model**: Integrate ML for better symptom classification.
* **Medical Database Integration**: Connect with third-party APIs like Infermedica.
* **Localization**: Translate advice into multiple languages.