Problem Definition and Design Thinking

Title: AI-Powered Early Disease Detection System

Problem Statement:

Many life-threatening diseases such as cancer, diabetes, and cardiovascular disorders often remain undetected until their advanced stages. There is a critical need for an AI-powered system that can analyze medical data (such as imaging, lab results, or genetic information) to detect early signs of diseases, enabling timely diagnosis and intervention.

Target Audience:

- General physicians and diagnostic centers.
- Patients at risk of chronic or genetic diseases.
- Public health organizations in resource-limited settings.

Objectives:

- To develop an AI-based tool that detects early signs of diseases like cancer, diabetes, and heart conditions.
- To assist healthcare professionals in identifying at-risk individuals before symptoms occur.
- To promote preventive healthcare and timely intervention.
- To make healthcare diagnostics more accessible and efficient.

Design Thinking Approach:

Empathize:

Doctors need faster, more accurate tools to screen large volumes of patients. Patients often don't undergo regular checkups due to cost or unawareness. Diagnostic delays can lead to late-stage disease detection and reduced survival rates.

Key User Concerns:

- Accuracy and reliability of predictions.
- Clear and understandable reports for both doctors and patients.
- Data privacy and protection.
- Compatibility with existing health systems.

- Cost-effectiveness in low-resource settings.

Define:

Many critical illnesses go undetected in their early stages due to lack of visible symptoms or insufficient screening resources. An AI-powered system can help detect early warning signs using available health data, enabling earlier diagnosis and treatment.

Key Features Required:

- User interface for uploading patient data (e.g., scans, reports)
- AI models trained for disease-specific predictions.
- Risk scoring and early warning alerts.
- Visualization of findings (charts, heatmaps, etc.).
- Report generation with explanations.
- Role-based access and data security.

Ideate:

- Use deep learning (CNNs) for detecting cancerous patterns in X-rays or MRIs.
- Apply machine learning models on blood test results to predict diabetes or CKD risks.
- Design a web-based interface for ease of use in hospitals.
- Implement explainable AI to build trust among doctors.
- Use synthetic datasets if real data access is limited during development.

Brainstorming Results:

- Focus on two major disease types in the prototype: Breast Cancer (via mammogram images) and Diabetes (via lab data).
- Use open-source datasets from Kaggle/NIH for model training.
- Implement a dashboard with individual risk scores and visual heatmaps for interpretability.
- Prioritize privacy compliance and minimal hardware requirements.

Prototype:

We aim to build a functional web-based prototype that demonstrates the core functionalities of an AI-powered early disease detection system. The prototype will allow users (e.g., doctors or technicians) to upload medical data and receive risk predictions, visual interpretations, and a downloadable report. It focuses on two

primary diseases—Breast Cancer (using mammogram images) and Diabetes (using lab test data).

Key Components of the Prototype:

- **Frontend Dashboard:** Upload files, view results, patient history.
- **Backend AI Engine:** Models to analyze inputs and return risk predictions.
- **Database:** Store anonymized patient data for testing.
- **Visualization Tools:** Show heatmaps for imaging and charts for lab data.
- **Report Generator:** Summarizes predictions with actionable insights.

Test:

The prototype will undergo initial testing in a simulated environment using open-source datasets. The focus will be on validating model predictions, assessing the usability of the dashboard, and ensuring the overall performance of the system across various test cases.

Testing Goals:

- Evaluate model accuracy using standard metrics (precision, recall, F1-score).
- Ensure ease of use for medical practitioners through user feedback.
- Check system response time for real-time usability.
- Confirm data privacy compliance and security protocols.
- Identify false positives/negatives and refine the model.