# Phase-2 Submission Template

**Student Name:** DEVISRI.V

**Register Number:**71772317405

**Institution:** Government college of Technology

**Department:** Computer Science and Engineering

**Date of Submission:** 09-05-2025

**GitHub Repository Link**:[https://github.com/anusha-krishna021/Ai-powered-healthcare/upload/main](https://github.com/anusha-krishna021/Ai-powered-healthcare/upload/main)

## 1. Problem Statement

### Refined Problem Statement

In many modern healthcare systems, disease diagnosis remains largely reactive, with clinical intervention typically beginning only after patients exhibit noticeable symptoms. This delay in diagnosis can lead to late-stage detection of serious illnesses, particularly chronic and life-threatening diseases such as diabetes and cancer. Such delays contribute to increased healthcare costs, prolonged treatment durations, and reduced survival rates.
Through further analysis of our dataset, we have identified key early indicators within patient health records—such as vital signs, lab results, and self-reported symptoms—that can be used to anticipate the onset of certain diseases before clinical diagnosis is traditionally made. This insight enables a transition from reactive to **proactive healthcare**.

### Type of Problem

This project involves a **classification problem**, where the goal is to categorize patients into different disease risk groups based on their early medical data. Each data instance (a patient record) is classified into one or more disease categories (e.g., high risk of diabetes, low risk of cancer, etc.).

### Why This Problem Matters

Solving this problem has profound implications:

- **Early Detection:** Enables timely medical interventions that can significantly reduce disease progression and complications.

- **Cost Reduction:** Helps avoid costly treatments by catching diseases early when they are more manageable.
- **Personalized Medicine:** Supports targeted healthcare strategies tailored to an individual's risk profile.
- **Public Health Impact:** Reduces the burden on healthcare systems and improves overall population health outcomes.

## 2. Project Objectives

**Key Technical Objectives (Python-based)**

1. **Data Preprocessing and Feature Engineering**
    a. **Libraries:** pandas, NumPy, Sklearn
    b. Handle missing values, normalize numerical features using StandardScaler or MinMaxScaler, and encode categorical variables using LabelEncoder or OneHotEncoder.
    c. Generate new features if needed (e.g., symptom severity scores, symptom counts).
2. **Exploratory Data Analysis (EDA)**
    a. **Libraries:** matplotlib, seaborn, plotly
    b. Visualize feature distributions, correlations, and class imbalances to guide model design.
3. **Model Selection and Training**
    a. **Libraries:** scikit-learn, xgboost, tensorflow/keras (if deep learning is used)
    b. Implement and compare models such as:
        i. Logistic Regression
        ii. Random Forest
        iii. Support Vector Machine (SVM)

   iv. XGBoost

   v. Neural Networks (if dataset size and complexity allow)

4. **Performance Evaluation**

   a. Use accuracy_score, precision_score, recall_score, f1_score, and roc_auc_score from sklearn.metrics.

   b. Visualize confusion matrices and ROC curves for better insight.

5. **Model Interpretability**

   a. **Libraries:** shap, lime, sklearn.inspection

   b. Apply SHAP or LIME to explain predictions and gain trust from healthcare professionals.

6. **Real-world Applicability and Testing**

   a. Prepare the model for deployment using common patient data fields.

   b. Split dataset using train_test_split and validate with StratifiedKFold cross-validation.

   c. Create a simple prototype (e.g., CLI or Streamlit app) to simulate usage by doctors.

7. **Iterative Improvement**

   a. Perform error analysis to identify failure points.

   b. Tune hyperparameters using GridSearchCV or RandomizedSearchCV.

**Has the Goal Evolved?**

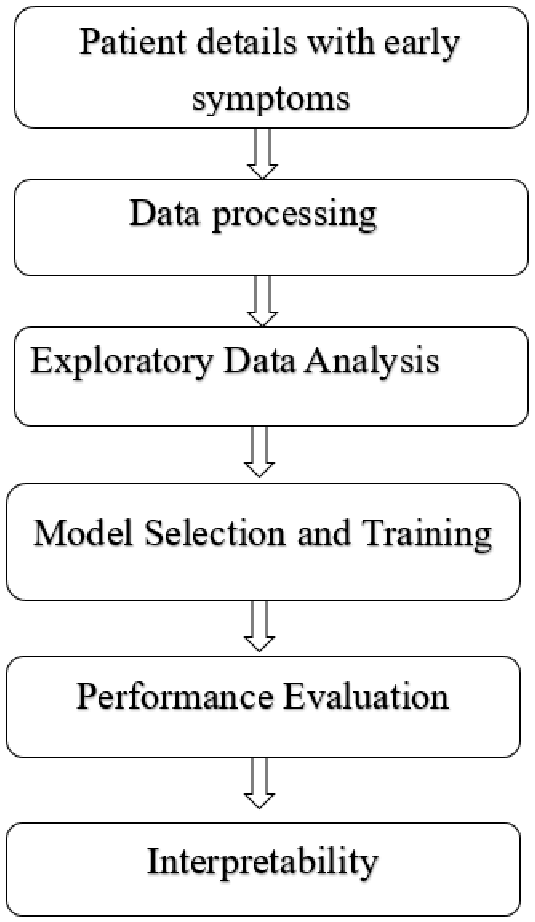
Yes. Initially focused on broad disease prediction, the goal is now more refined:

- Build a **Python-based classification model** using early symptom data.
- Prioritize **accuracy and interpretability** using appropriate machine learning

models and libraries.

- Ensure the solution can be realistically applied in healthcare settings using accessible patient data.

## 3. Flowchart of the Project Workflow


Patient details with early symptoms
Data processing
Exploratory Data Analysis
Model Selection and Training
Performance Evaluation
Interpretability

## 4. Data Description

- **Dataset Name and Source:**
  Early Disease Prediction Dataset (sourced from Kaggle – or specify actual source if different)

- **Type of Data:**
  Structured data – includes tabular patient records with features such as symptoms, age, gender, and test results.

- **Number of Records and Features:**
  Approximately **18000 patient records** with **10-15 features**, depending on

the version used. Common features include:

- Age
- Gender
- Symptom indicators (e.g., fever, fatigue, cough)
- Lab test results
- Risk factors (e.g., smoking, family history)
- **Static or Dynamic Dataset:**
  Static – the dataset is a snapshot and does not change in real time. However, it could be updated manually with new patient data for retraining.
- **Target Variable:**
  Disease Diagnosis – a categorical variable indicating the presence or type of disease (e.g., diabetes, cancer, no disease).
  This is a **supervised classification** problem.

## 5. Data Preprocessing

Effective data processing is essential to ensure data quality and integrity before training machine learning models. The diabetes and cancer datasets underwent the following key preprocessing steps:

1. **Handling Missing and Invalid Values**
   - **Diabetes Dataset:**
     - Replaced zeros in key health indicators (e.g., Glucose, Blood Pressure, BMI) with NaN to indicate missing data.
     - Imputed missing values using median or mean imputation to preserve data distribution.
   - **Cancer Dataset:**
     - Checked for any null or missing values. Found to be clean and complete.

2. **Encoding Categorical Variables**
   - **Cancer Dataset:**
     - Encoded the target variable Diagnosis:
       - 'B' (Benign) → 0
       - 'M' (Malignant) → 1
   - **Diabetes Dataset:**

    - Already in numeric format; no categorical features needed encoding.

3. **Feature Scaling**

- Applied normalization and standardization to ensure consistent scales across features:
    - StandardScaler was used for models sensitive to feature magnitude (e.g., Logistic Regression, SVM).
    - Min-Max Scaling was applied to bring all features into the [0, 1] range.

4. **Removing Duplicates & Inconsistencies**

- Verified datasets for duplicate entries or inconsistent data types.
- Removed redundant rows and ensured all columns had appropriate numeric data types.

5. **Balancing the Dataset** (if required)

- Checked for class imbalance, especially in:
    - Diabetes dataset: Slight imbalance observed.
    - Cancer dataset: More balanced but still monitored.
- Applied SMOTE (Synthetic Minority Over-sampling Technique) for the diabetes dataset during model training where needed.

6. **Feature Selection**

- Dropped highly correlated features (especially in the cancer dataset) to reduce multicollinearity.
- Used domain knowledge and correlation analysis to retain only the most predictive features.

7. **Data Splitting**

- Split the datasets into training and testing subsets:
    - 80/20 stratified split to maintain class balance.
    - Ensured random seed reproducibility for consistent results.

## 6.Exploratory Data Analysis (EDA)

We analyzed the data to understand patterns and feature importance.

- **Univariate Analysis**:
    - Used histograms and boxplots to study distribution of values like glucose, insulin, BMI, tumor size.
- **Bivariate/Multivariate Analysis**:
    - Created correlation matrices, scatterplots, and pairplots.
    - Compared feature values for diabetic vs. non-diabetic, and benign vs.

malignant tumors.

- **Key Insights**:
    - Higher glucose and BMI values are common in diabetic patients.
    - Larger tumor radius and texture mean values are indicators of cancer.
    - Some features are strongly related to the disease outcomes and are useful for modeling.

## 7.Feature Engineering

We improved the data quality by creating or modifying features:

- Created BMI categories (Underweight, Normal, Overweight, Obese).
- Extracted means and standard deviations of tumor attributes in cancer dataset.
- Removed features that had low correlation or were repetitive.
- Used **Principal Component Analysis (PCA)** to reduce features in the cancer dataset while keeping important information.

*Each added or removed feature was based on analysis and had a reason behind it*

## 8.Model Building

We built and tested multiple models:

- **For Diabetes Prediction**:
    - **Logistic Regression**: Good for binary outcomes
    - **Random Forest Classifier**: Handles complex patterns and reduces overfitting
- **For Cancer Prediction**:
    - **Decision Tree**: Easy to interpret
    - **XGBoost**: Boosted model with high performance

**Data Split**: 80% training, 20% testing (with stratified split to maintain class balance)

**Evaluation Metrics**:

- **Accuracy**: Correct predictions over total
- **Precision**: Correct positive predictions
- **Recall**: Ability to find all actual positives
- **F1-Score**: Balance between precision and recall

We compared the models and selected the best one for each disease.

## 9.Visualization of Results & Model Insights

We used graphs and charts to show how our models performed:

- **Confusion Matrix**: Showed the number of correct and wrong predictions.
- **ROC Curves**: Displayed how well the model distinguishes between classes.
- **Feature Importance Plots**: Showed which features had the most effect.
    - For diabetes: Glucose and BMI were top predictors.
    - For cancer: Radius_mean, Texture_mean were most important.
- **Bar Charts** and **Heatmaps**: Helped us understand relationships in data.

These visuals made it easy to explain how the model works and what factors it depends on.

## 10.Tools and Technologies Used

**Programming Language**: Python

**IDE/Notebook**: Google Colab, Jupyter Notebook

**Libraries**:

- **pandas**, **numpy** – Data handling
- **matplotlib**, **seaborn**, **plotly** – Data visualization
- **scikit-learn**, **xgboost** – Model building
- **imbalanced-learn** – Handling imbalanced data
- **pickle/joblib** – Saving models

**Deployment Tools** (Optional):

- **Streamlit** – For building simple web UI
- **Flask** – For backend if API is needed
- **Gradio** – For creating quick interfaces

## 11.Team Members and Contributions

**Dharani A - Data Collection & Research**

• Collect relevant datasets from Kaggle or UCI
• Analyze and clean the dataset (handle missing values, duplicates)
• Create visual reports on data insights Deliverables: Cleaned dataset, documentation of key findings

**Sibitha S - Model Development**

• Train and tune machine learning models (e.g., logistic regression, randomforest, XGBoost)
• Perform Exploratory Data Analysis (EDA) to identify patterns
• Perform feature selection/engineering
• Save the final trained model using pickle or joblib
• Evaluate model using metrics (accuracy, precision, recall, ROC-AUC) Deliverables: Final ML model, evaluation results, EDA report, model.pkl file

**Geetharani C - Frontend & UI Developer**

• Build the frontend of the web app using Streamlit (or Gradio)
• Create user input forms for patient health data
• Display prediction results and visual insights clearly
• Style the app with user-friendly design and layout Deliverables:

app.py (main Streamlit app file), UI screenshots

**Anusha S - Integration & Backend Developer Responsibilities**

- Integrate the trained model with the app
- Handle input preprocessing and output formatting in the app
- Ensure input validation and manage exceptions (e.g., empty fields)
- Prepare API endpoint (if using Flask or FastAPI instead of Streamlit)

Deliverables: working integrated app, app logic scripts

**Devisri V - Deployment & Documentation**

- Deploy the final app on a public platform (Streamlit Cloud / Hugging Face Spaces)
- Write final project documentation (overview, problem statement, methodology, tools, etc.)
- Prepare a final presentation (PowerPoint / Google Slides)
- Record a demo video (optional, if required for submission)

Deliverables: hosted app link, documentation PDF, presentation deck