Project Initialization and Planning Phase

Date	14 June 2025
Team ID	SWTID1749641473
Project Title	Early Prediction for Chronic Kidney Disease Detection: A Progressive Approach to Health Management
Maximum Marks	3 Marks

Project Proposal (Proposed Solution) template

This project proposal outlines a solution to address a specific problem. With a clear objective, defined scope, and a concise problem statement, the proposed solution details the approach, key features, and resource requirements, including hardware, software, and personnel.

Project Overview	
Objective	Utilize conventional clinical and laboratory data to develop a streamlined machine-learning system capable of predicting the likelihood of a patient having Chronic Kidney Disease (CKD). This will facilitate prompt action in areas with constrained resources.
Scope	The project includes collecting data, cleaning it up, training a model with XGBoost, testing how well it works, and making it available through a basic web interface. It doesn't mean connecting real-time data to hospitals or building apps for mobile devices.
Problem Statement	
Description	Identifying chronic kidney disease in its early stages may be challenging, particularly in clinics with limited resources. Manually verifying symptoms and laboratory data is time-consuming and prone to errors, potentially resulting in delayed or missed diagnoses.
Impact	Automating the detection of chronic kidney disease (CKD) might reduce diagnostic time, enhance accuracy, and facilitate timely treatment by healthcare practitioners, thereby saving lives and decreasing long-term care costs.
Proposed Solution	
Approach	1. Use the UCI CKD dataset (400 records, 24 clinical features). 2. Perform data cleaning and impute missing values.

	3. Encode categorical variables and scale numerical ones. 4. Train an XGBoost classification model. 5. Evaluate model performance using accuracy, precision, recall, and F1-score. 6. Build a Flask-based web application to take user input and display prediction results. 7. Host the application locally or on a cloud platform.
Key Features	• High accuracy using XGBoost classifier • Simple and user-friendly web interface • Fast prediction output • Lightweight and deployable on standard systems • Easy to update and retrain with new data

Resource Requirements

Resource Type	Description	Specification/Allocation
Hardware		
Computing Resources	Required to train and run the XGBoost model efficiently.	1 × 4-core CPU (e.g., Intel i5 or AWS t3.medium); No GPU required
Memory	For handling data loading, model training, and runtime execution.	8 GB RAM
Storage	For storing dataset, model files, logs, and dependencies.	10 GB SSD
Software		
Frameworks	To build the web interface and handle model deployment.	Flask (v2.x), Python (v3.11)
Libraries	For machine learning, data handling, and preprocessing.	XGBoost, scikit-learn, pandas, numpy, joblib
Development Environment	For coding, testing, and version control.	Jupyter Notebook / VS Code, Git + GitHub
Data		
Data	Public dataset used for model training and testing. Number	UCI CKD Dataset (Kaggle mirror), 400 patient records,

	of samples and file type.	~15 KB, CSV format