Title: Diabetes Detection System - System Test Plan

1. Introduction
   1. Purpose

The purpose of this document is to outline the test approach and plan for the Diabetes Detection System. The testing process aims to verify that the system functions as intended and accurately predicts the likelihood of an individual having diabetes based on their medical information.

* 1. References

The System Requirements Specification (SRS) document will be used as a reference for this test plan.

1. Test Plan Identifier

TP001-DiabetesDetection

1. Test Items

Machine learning algorithms and test accuracy

1. Features to be Tested.

All functionalities of the Diabetes Detection System will be tested, including:

* Data collection and preprocessing
* Naive Bayes algorithm implementation
* Decision Tree algorithm implementation
* Model training and evaluation
* Integration of models into the software application
* User interface for inputting medical information and receiving diabetes predictions

1. Features not to be Tested.

Visual design elements such as colour schemes, font sizes, and other UI/UX features unrelated to the core functionalities of the Diabetes Detection System will not be part of the testing process.

1. Test Approach

The chosen approach for testing the Diabetes Detection System is of manual testing.

1. Pass/Fail Criteria

Each test case will have specific pass/fail criteria defined. The criteria will be based on the expected system behaviour and accuracy of the diabetes predictions.

1. Suspension Criteria and Resumption Requirements

If critical defects are discovered during testing, the testing process may be suspended until the issues are resolved. Before resuming the testing process, it is mandatory to retest the affected areas and ensure that the identified defects have been resolved.

1. Test Deliverables

The test deliverables include:

* Test plan document
* Test cases and test scripts
* Test execution logs and results
* Defect reports
* Final test summary report

10. Environmental Needs

The Diabetes Detection System should be tested in an environment that meets the following requirements:

* Operating System: [Specify the targeted operating system(s) and version(s)]
* Programming Language: [Specify the programming language(s) and version(s)] - Libraries and Frameworks: [Specify any specific libraries or frameworks required for the project]

11. Staffing and Training Needs

A testing team consisting of at least one tester is required. The tester should be trained on the functionalities and usage of the Diabetes Detection System.

12. Responsibilities

The testing team is responsible for:

* Performing test cases and test scripts
* Recording and reporting defects following the defined process
* Collaborating with the development team to address and resolve identified issues

1. Schedule

The testing schedule will be determined based on the project timeline and milestones. Start and end dates for each testing phase will be defined in coordination with the development team.

1. Planning Risks and Contingencies

Possible risks, such as defects or delays, may arise during the testing process. It is important to proactively identify and plan for such risks, including adjusting the testing schedule or allocating additional resources if necessary, to mitigate any potential impact on the project.

1. Approvals

The System Test Plan requires approval from the relevant stakeholders, including project managers, developers, and testers.

1. Glossary

* SRS: System Requirements Specification
* UI: User Interface
* UX: User Experience

17.Approvals: CVSN Reddy Sir

18. Test cases

Sure! Here are some system test cases you can use to evaluate the program:

1. Test Case 1: Check DataFrame Load\*\*: Verify that the DataFrame is loaded correctly from the "diabetes.csv.xls" file.

- Expected Result: The DataFrame is loaded without any errors and contains the expected columns and data.

2. Test Case 2: Check Train-Test Split\*\*: Verify that the train-test split is performed correctly with a test size of 0.2 and a random state of 42.

- Expected Result: The train and test datasets are generated with the correct sizes, and the random state ensures reproducibility.

3. Test Case 3: Check Model Training\*\*: Verify that the model is trained using the DecisionTreeClassifier with the entropy criterion and a random state of 0.

- Expected Result: The model is trained without any errors, and the training process completes successfully.

4. Test Case 4: Check Prediction\*\*: Verify that the model predicts the target variable (diabetes) correctly on the test dataset.

- Expected Result: The model predicts the target variable for all instances in the test dataset without any errors.

5. Test Case 5: Check Accuracy Calculation\*\*: Verify that the accuracy score is calculated correctly for the model's predictions on the test dataset.

- Expected Result: The accuracy score is calculated and returned without any errors, providing a measure of the model's performance.

6. Test Case 6: Check Visualization\*\*: Verify that the accuracy score is visualized correctly using Matplotlib and Seaborn.

- Expected Result: A plot is generated, representing the accuracy score, without any errors.

These test cases cover various aspects of the program, including data loading, data splitting, model training, prediction, accuracy calculation, and visualization. By running these test cases, you can ensure that the program functions as intended and produces the expected results.