**SE 4485: Software Engineering Projects**

Spring 2024

Test Plan

| Group Number | **8** |
| --- | --- |
| Project Title | **Crop Yield Prediction Model** |
| Sponsoring Company |  |
| Sponsor(s) | **Marc J. Perna (*Main Sponsor*)** |
| Students | **1. Ryan Havens (*Team Leader*)**  **2. Cameron Sutton**  **3. Melvin Sajeev**  **4. Ibrahim Barney**  **5. Nisai Sun** |

# Abstract

This document is an overview of test cases for the predictive crop-yield model. The document provides a formal explanation for tests performed and reasons each test was performed to ensure complete functional requirements.

# Table of Contents

[**Abstract 1**](#_heading=h.u9n5317oy4uf)

[**Table of Contents 2**](#_heading=h.hqg7w9c5bp5v)

[**List of Figures 3**](#_heading=h.5sqpqr948yfm)

[**List of Tables 4**](#_heading=h.q73n9rszhrlh)

[**Introduction 5**](#_heading=h.kt5eyrhm588d)

[**Requirements/Specifications-Based System Level Test Cases 6**](#_heading=h.ewjlwcqvcach)

[**Techniques for Test Generation 9**](#_heading=h.z6du08dk1uzk)

[**Traceability of Test Cases to Use Cases 10**](#_heading=h.kglqgim8fwy6)

[**Evidence Test Plan Under Configuration Management 11**](#_heading=h.sk82karohqx8)

[**Configuration Management Table 12**](#_heading=h.3md9zrnuylqm)

[**Engineering Standards And Constraints 14**](#_heading=h.ui6h1mhqcnk3)

[**Additional References 15**](#_heading=h.kzylk2b0kade)

# List of Figures

*N/A*

# List of Tables

* [Techniques For Test Generation (Test Cases)](#_heading=h.z6du08dk1uzk)
* [Requirements/Specification-Based System Level Test Cases](#_heading=h.ewjlwcqvcach)
* [Traceability of Test Cases to Use Cases](#_heading=h.kglqgim8fwy6)
* [Configuration Management Table](#_heading=h.3md9zrnuylqm)

# Introduction

The purpose of testing a system is to show the stakeholders that the deliverables have been met and that the system is reliable and efficient. This document is a formal cover of the test cases for testing the predictive crop yield application. The purpose of this document is to clearly describe how the test cases ensure functional requirements for the project are being met. This paper will start by covering requirements /specifications for the test cases. Then move into the test cases themselves. And finish by mapping the test cases to established requirements and use cases to show that they cover all deliverables of the project.

# Requirements/Specifications-Based System Level Test Cases

| **FRs** | **Test Case** | **How Test Case(s) cover the FR** |
| --- | --- | --- |
| FR 1 | 3  5  6 | This functional requirement is part of the pre-processing data portion for this we need the normal system coverage tests (5 and 6) as well as the pre-process functional test (3). |
| FR 2 | 3  5  6 | This functional requirement is part of the pre processing data portion for this we need the normal system coverage tests (5 and 6) as well as the pre process functional test (3) |
| FR 3 | 3  5  6 | This functional requirement is part of the pre processing data portion for this we need the normal system coverage tests (5 and 6) as well as the pre process functional test (3) |
| FR 4 | 3  5  6 | This functional requirement is part of the pre processing data portion for this we need the normal system coverage tests (5 and 6) as well as the pre process functional test (3) |
| FR 5 | 3  5  6 | This functional requirement is part of the pre processing data portion for this we need the normal system coverage tests (5 and 6) as well as the pre process functional test (3) |
| FR 6 | 3  5  6 | This functional requirement is part of the pre processing data portion for this we need the normal system coverage tests (5 and 6) as well as the pre process functional test (3) |
| FR 8 | 3  5  6 | This functional requirement is part of the pre processing data portion for this we need the normal system coverage tests (5 and 6) as well as the pre process functional test (3) |
| FR 9 | 1  2  4  5  6  7 | This FR is part of the User interface portion,(as well as the output) so it needs tests for functional correctness (1 and 4) completeness(2) performance capacity(7) and then the system coverage tests (5 and 6) |
| FR 10 | 1  2  4  5  6  7 | This FR is part of the User interface portion,(as well as the output) so it needs tests for functional correctness (1 and 4) completeness(2) performance capacity(7) and then the system coverage tests (5 and 6) |
| FR 11 | 1  2  4  5  6  7 | This FR is part of the User interface portion,(as well as the output) so it needs tests for functional correctness (1 and 4) completeness(2) performance capacity(7) and then the system coverage tests (5 and 6) |
| FR 12 | 1  2  4  5  6  7 | This FR is part of the User interface portion,(as well as the output) so it needs tests for functional correctness (1 and 4) completeness(2) performance capacity(7) and then the system coverage tests (5 and 6) |
| FR 13 | 1  2  4  5  6  7 | This FR is part of the User interface portion,(as well as the output) so it needs tests for functional correctness (1 and 4) completeness(2) performance capacity(7) and then the system coverage tests (5 and 6) |
| FR 14 | 1  2  4  5  6 | This FR is about the model making predictions based on Machine Learning algorithms. For this we need to make sure the data is accurate (Functional correctness) (1 and 4) also requires functional completeness (2). and then the system coverage tests (5 and 6) |
| FR 15 | 1  2  4  5  6 | This FR is about the system using cleaned data sets.. For this we need to make sure the data is accurate (Functional correctness) (1 and 4) also requires functional completeness (2). and then the system coverage tests (5 and 6) |
| FR 16 | 1  2  4  5  6 | This FR is to make sure the data is accurate so we use functional correctness tests (1 and 4) functional completeness (2) and the system coverage tests (5 and 6) |

\*FR 7 was removed due to a change in our project

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Techniques for Test Generation

| **Test Case** | **Technique** | **Description** | **Rationale** | **Quality Characteristic**  **[1]** | **Black Box/White Box\*** |
| --- | --- | --- | --- | --- | --- |
| 1 | Functional Test | *Input Validation*: Provide system with invalid state and county names | Ensure system can handle wrong input gracefully | Functional Correctness | White Box |
| 2 | Functional Test | *Boundary Testing*: provide extreme values for input | Ensure system behaves correctly within limits | Functional completeness | White Box |
| 3 | Functional Test | *Input Validation*: Give a test csv file with sample data so we control the expected output | Verifies the data preprocessing produces the expected output. | Functional Correctness | White Box |
| 4 | Accuracy Test | Prediction Accuracy | Verify Accuracy of prediction against known data | Functional Correctness | White Box |
| 5 | Structural Test | *Code Coverage*: Exercise all branches and paths in backend code | Ensure comprehensive testing of backend logic | Functional completeness | Black Box and White Box  (Full System test but goes through each individual component) |
| 6 | System Integration Test | Tests end to end flow of the entire system | Goes through each component thoroughly to make sure the system performs as expected | Functional completeness | Black Box and White Box  (Full System test but goes through each individual component) |
| 7 | Performance Test | *Stress Testing*: Simulate high concurrent user requests | Assess system performance under heavy load | Capacity | White Box |

# 

# Traceability of Test Cases to Use Cases

| **Test Case** | **FRs** | **Use Case** | **Use Case Description** |
| --- | --- | --- | --- |
| 3  5  6 | FR 1 - FR 7 | Data Preprocessing | Processing the data set so it’s ready to be inputted into the model. |
| 1  2  4  5  6 | FR 14 / 15 | Machine Learning | The model is trained on the data we have |
| 1  4  5  6 | FR 12 / 13 | Output Formatting | The data is taken from the model and is formatted for additional use |
| 1  2  4  5  6 | FR 16 | Accuracy and Performance | Make sure the model is able to accurately predict the data |
| 1  2  4  5  6  7 | FR 9 - FR 13 | User Interface | The user will get the output based on the user input |

# 

# Evidence Test Plan Under Configuration Management

1. **Name of the CM tool used by your team**: Google Docs

1. **Version number of each document after it is checked in**: Google Docs does not have version numbers, but has versions based on timestamps for each session. We can view these in the version history in *File > Version History*.

1. **Version number of each document before it is checked out**: Google Docs does not have check out, or check in, the team works on the latest time stamped version of the document.

1. **Difference between two consecutive versions**: Users can view the revision history through Google docs to see the different versions of the document.

1. **Review of each change**:
   1. After the document is finalized, we have the team double check it to make sure all the input is correct.

1. **Other information that helps the understanding of each change**:
   1. Different team members are assigned different tasks and put this information in the table below.
   2. Every team member is added with their gmail accounts so we know who made each change.

# Configuration Management Table

| **Section Updated** | **Update Description** | **Submitted by** | **Date Updated** |
| --- | --- | --- | --- |
| Entire Document | Creation and added members | Ibrahim | 04/10/2024 |
| Engineering Standards and constraints | Completed | Ibrahim | 04/13/2024 |
| Entire Document | Initial Set Up and Formatting | Ryan | 04/16/2024 |
| Configuration Management | Completed | Ryan | 04/18/2024 |
| Techniques For Test Generation | Completed Table | Ryan | 04/18/2024 |
| Introduction | Completed | Ryan | 04/18/2024 |
| List of Tables | Completed | Ryan | 04/18/2024 |
| Abstract | Completed | Ryan | 04/18/2024 |
| Table of Contents | Completed | Ryan | 04/18/2024 |
| Techniques For Test Generation | Added to Table | Ibrahim | 04/19/2024 |
| Traceability Table | Completed | Ibrahim | 04/19/2024 |
| Introduction | Tweaked the Introduction [Completed] | Ibrahim | 04/19/2024 |
| Requirements/Specifications-Based System Level Testing | Completed | Ibrahim and Melvin | 04/19/2024 |
| Updated and fixed formatting | Completed | Nisai | 04/19/2024 |
| Techniques for test generation | Made the quality attributes column match the iso table | Ibrahim | 04/19/2024 |

# 

# Engineering Standards And Constraints

* IEEE Std 829-1983: Software Testing [[pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=573169)]
* ISO/IEC/IEEE Std 29119-1-(Revision-2022): Part 1 - Software Testing General Concepts [[pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9698145)]
* ISO/IEC/IEEE Std 29119-2-(Revision-2021): Part 2 - Test Process [[pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9591508)]
* ISO/IEC/IEEE Std 29119-3-(Revision-2021): Part 3 - Test Documentation [[pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9591577)]
* ISO/IEC/IEEE Std 29119-4-(Revision-2021): Part 4 - Test Techniques [[pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9591574)]

## 

# Additional References

* [1] "IEEE/ISO/IEC International Standard - Software and systems engineering--Software testing--Part 4: Test techniques," in ISO/IEC/IEEE 29119-4:2021(E) , vol., no., pp.1-148, 28 Oct. 2021, doi: 10.1109/IEEESTD.2021.9591574. keywords: {IEEE Standards;ISO Standards;IEC Standards;Software engineering;Software testing;System analysis and design}.
* [2] Jorgensen, P.C., 2013. Software Testing: A Craftsman's Approach. Auerbach Publications
* [3] Mathur, A.P., 2013. Foundations of Software Testing, 2/e. Pearson Education