# Introduction

<< This section gives introductory information regarding the project, the system to be tested, and the testing approach.>>

## Purpose

<< Identify the project and stipulate the test plan purpose by indicating what the document contains (e.g., organizational responsibilities, test approach, test schedule. There are generally four different types of test plans: project test plan that describes the overall strategy for testing; the system test plan that describes the system from the customer’s point of view; integration test plan that describes integration of units and subsystems; unit test plan that describes modules or classes. This section needs to identify which of these this document is.>>

The purpose of this test is to allow more practice in black box testing.

## Scope

<<Specify the project software releases/versions encompassed by the plan. >>

The scope is supposed to test a NxN table sorter that we do not have access too. The method we will use to test this is black box testing.

## System Overview

<<Describe the system to be exercised by the testing approach specified in the plan. This overview serves to identify aspects of the system operation that will be the focus of the plan’s testing approach. This should align with the systems overview of other documents in the project.>>

Testing without source code is tricky but not impossible. I would like to test the bounds of the matrix since it is an NxN matrix. The next thing I would investigate is the expected output with any input which would be a sorted array. The next test would be on checking out of bounds indexes.

## Suspension and Exit Criteria

<< “suspension criteria” describes when we suspend testing, to be resumed at a later time. For example, if 40% of the test cases fail, or if any of the critical test cases fail. If there are no suspension criteria, indicate that all tests cases will be executed. “Exit criteria” indicates when testing stops. This could be based on run rate (number of test cases run divided by number of test cases specified) or pass rate (number of test cases passed divided by number of test cases run, or test cases passed divided by number of test cases specified). Nominally, we expect to run all of the specified tests. We want the pass rate to be high. We might specify that all critical tests must pass, and 90% of the non-critical must pass. In general, we want this to be high. >>

Suspension and exit criteria would happen once 100% of these test cases pass.

# Test Items and Features

<< This section describes the test items (e.g., components, classes, functions or methods) and the features to be tested. It may also list features not to be tested. A class diagram is useful. A table of features is useful. >>

The features we are going to be testing is the Tables-sorter algorithm which we don’t know how the source code looks like.

# Testing Approach

<<Describe the approach to be used to the test the system. This description includes specifying the types of tests to be performed, e.g., tests designed to exercise system functions one by one; tests designed to exercise sequences of functions that approximate operational use of the system; tests designed to stress the system to its design and requirements limits. The description lists the specific tests to be performed, but does not give the test steps. For each of these tests, give it a name and specify its objective. Label the criticality of the test cases. >>

Table 1: Test Plan

|  |  |  |
| --- | --- | --- |
| **TEST SUITE <MatrixbbTest>** | | |
| **Description of Test Suite** | **Focus on bounds, expected output , and index comparisons** | |
| **Test Case Identifier** | **Objective** | **Criticality** |
| testPostition | **Test to see if you can set and get elements of matrix** | **high** |
| testSorted | **Test to see if the sort algorithm is correct** | **high** |
| testBound | **Test to see if indexes are out of bounds of the matrix** | **high** |
| testLeftandRight | **Test left and right indexes to see if they are in order** | **high** |