**Test Cases for**

**<Project-Name>**

**<Revision>**

**<Date>**

**<Author Name>**

**<Group Name>**

Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev** | **Date** | **Author** | **Change Description** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

*This document is originating from Neil Bitzenhofer and DataCard Corporation.*

Table of Contents

*1.* Test Requirements [1](#__RefHeading___Toc336944966)

1.1 Objective [1](#__RefHeading___Toc336944967)

1.2 Definitions and Acronyms [1](#__RefHeading___Toc336944968)

1.3 Traceability Matrix [1](#__RefHeading___Toc336944969)

2. Test Cases [2](#__RefHeading___Toc336944970)

2.1 Description [2](#__RefHeading___Toc336944971)

2.2 Test Case Template [2](#__RefHeading___Toc336944972)

2.3 Test Case Example [2](#__RefHeading___Toc336944973)

# Test Requirements

## Objective

The purpose of the Test Requirements section is to list ALL hardware and software test requirements, whether explicitly determined from any relevant documents or implicitly determined from experience and product knowledge. For most projects, the documents referred to may be the Product Definition Document, Software/Hardware Requirements Specification and perhaps the Software/Hardware Design Specification. A Test Case Matrix is provided that simply lists all the test cases by title or description, and includes a method of tracking when the test case was run and whether it passed or not.

## Definitions and Acronyms

List any technical terms or acronyms used in the document, along with their meanings.

Examples for this document:

SRS Software Requirements Specification

TM Traceability Matrix

## Traceability Matrix

The purpose of a Traceability Matrix is to show which test cases verify which requirements. A possible format for a Traceability Matrix is as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Requirement  \ Test Case | Test Case ID 1 | Test Case ID 2 | Test Case ID 3 | Test Case ID 4 |
| Requirement 1 | X |  |  |  |
| Requirement 2 |  | X | X |  |
| Requirement 3 |  |  |  | X |

# Test Cases

## Description

For convenience, test cases may be logically grouped together in what are called Test Modules, based on areas of testing. The use of Test Modules helps make the Traceability Matrices and archiving Test Cases more manageable.

## Test Case Template

**Test Case ID**  **Test Case Title**

*The underlined Test Case ID may be any convenient identifier, as decided upon by the tester. Identifiers should follow a consistent pattern within Test Cases, and a similar consistency should apply across Test Modules written for the same project.*

|  |  |
| --- | --- |
| **Description**: | The purpose of the Test Case, usually to verify a specific requirement. This is a high-level description of the test. |
| **Test Inputs**: | Any required data input for the Test Case. |
| **Expected Results**: | Describe the expected results and outputs from this Test Case. This may be any possible output including an exception or error. |
| **Dependencies**: | If correct execution of this Test Case depends on being preceded by any other Test Cases, that fact should be mentioned here. Similarly, any dependency on factors outside the immediate test environment should also be mentioned. |
| **Initialization**: | If the system software or hardware has to be initialized in a particular manner in order for this Test Case to succeed, such initialization should be mentioned here. |
| **Test Steps**: | Describe what will take place during the Test Case. This is the Test Procedure, which can be specified by test case steps (an enumerated list of steps). |
| **Owner**: | The person(s) or department responsible for keeping the Test Cases accurate. This may be covered in the overall description if a single individual/team owns all tests. |

*At this point, other relevant data tables or information can be included that will help in executing the Test Case successfully. This may include test tools.*

## Test Case Example

**E.1.1.1 Get Non-empty Form List (Form Storage System)**

|  |  |
| --- | --- |
| **Description**: | Checks that the saved form has a single form entry. |
| **Test Inputs**: | None |
| **Expected Results**: | A list of forms is returned. There is exactly one entry in the form list. No exception is thrown. |
| **Dependencies**: | None |
| **Initialization**: | The user name is already set in the system. |
| **Test Steps**: | 1. Save an empty form with description “Greg’s data”. No other data in the form is filled in. 2. Retrieve (loads) the form. 3. Verify that the retrieved form only contains one entry. 4. Verify that the entry is an element with an id number and a description “Greg’s data”. |

**E.1.1.1 Get List of Students for valid GPC in GRADS**

|  |  |
| --- | --- |
| **Description**: | Ensures that a valid GPC can access a list of their students |
| **Test Inputs**: | StudentRecord database initialized with 29 CSCE students and 1 MATH student. User database initialized with CSCE GPC “ggay”. |
| **Expected Results**: | A list of 29 CSCE students is returned. No exception is thrown. |
| **Dependencies**: | None |
| **Initialization**: | All databased are loaded, and the user name “ggay” is already set in the system. |
| **Test Steps**: | 1. The list of students is requested from GRADS. 2. Verify that the list contains exactly 29 students. 3. Verify that all students have Department “CSCE” |

**E.1.1.1 GPC can get Student Record for a valid student in GRADS**

|  |  |
| --- | --- |
| **Description**: | Ensures that a valid GPC can access the student record for a valid CSCE student. |
| **Test Inputs**: | StudentRecord database initialized with CSCE student “rbob”. Record for “rbob” contains the following fields:  ID: rbob  first name: Robert  last name: Bob  department: CSCE  term began: Fall, 2013  degree sought: PHD, Spring, 2018  previous degrees: BS, Spring, 2008  advisor: Manton Matthews, CSCE  committee: Duncan Buell, CSCE; Jason Bakos, CSCE; Richard McGehee, MATH  courses taken:   * nap734, Naptime, 5 credits, Spring, 2014, B * csce513, Computer Architecture, 3 credits, Fall 2013, A * csce531, Compiler Construction, 3 credits, Spring 2014, A   milestones:   * dissertation advisor selected * dissertation committee formed   User database initialized with CSCE GPC “ggay”. |
| **Expected Results**: | The student record for “rbob” is returned with the expected values given in the input. No exception is thrown |
| **Dependencies**: | None |
| **Initialization**: | All databased are loaded, and the user name “ggay” is already set in the system. |
| **Test Steps**: | 1. The student record for “rbob” is requested from GRADS. 2. For each field in the returned record, check whether the correct information is returned. |