|  |
| --- |
|  |

|  |
| --- |
| Imam Abdulrahman Bin Faisal University |
| College of Computer Science & Information Technology  Department of Computer Science |

**CS 411 – Software Engineering**

**Term 1 – 2018/2019**

Software Test Plans

For

Railway.Manage();



Version 0.1

CS Year 4, G1

Ms.Wadha Almattar

*December 1,2018*

Software Test Plan (STP) was prepared and provided as a deliverable for Software Engineering, CS 411, Term 1, and it will be used by all developers and stakeholders.

This document is based in part on the IEEE Recommended Practice for STP Descriptions.

Team Members:

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | ID | Role |
| 1 | Muneera abdullah alhajri | 2160007230 | Leader |
| 2 | Reema Ibrahim Alyousef | 2160003133 | Member |
| 3 | Rahaf Saleh Alzahrani | 2160006662 | Member |
| 4 | Tasneem Hamdy dosoqi | 2160007430 | Member |

**Table of Content**

[Revision History 3](#_Toc531431499)

[Table of Tables 4](#_Toc531431500)

[1. Introduction 5](#_Toc531431501)

[1.1. Objectives 5](#_Toc531431502)

[2. Test items 7](#_Toc531431503)

[3. Features to be tested 7](#_Toc531431504)

[4. Approach 7](#_Toc531431505)

[4.1 Data and Database Integrity Testing 7](#_Toc531431506)

[4.1.1 Test Cases 7](#_Toc531431507)

[4.2 Component Testing 8](#_Toc531431508)

[4.2.1 Test Cases 9](#_Toc531431509)

[4.3 Integration Testing 13](#_Toc531431510)

[4.3.1 Test case 13](#_Toc531431511)

[4.4 Job Stream Testing 15](#_Toc531431512)

[4.5 Interface Testing 15](#_Toc531431513)

[4.6 Security Testing 15](#_Toc531431514)

[4.7 Recovery Testing 15](#_Toc531431515)

[4.8 Performance Testing 16](#_Toc531431516)

[4.9 Regression Testing 16](#_Toc531431517)

[4.10 Beta Testing 16](#_Toc531431518)

[5. Pass / fail criteria 16](#_Toc531431519)

[6. Testing process 17](#_Toc531431520)

[7. Environmental requirements 18](#_Toc531431521)

[8. Change management procedures 19](#_Toc531431522)

[9. Plan approvals 19](#_Toc531431523)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| All members | Dec 1, 2018 | Prepared initial version | 0.1 |

# Table of Tables

[Table 1 Definitions 6](#_Toc531430033)

[Table 2 Acronyms 6](#_Toc531430034)

[Table 3 Data and Database Integrity Test Cases 7](#_Toc531430035)

[Table 4 "view Ticket " Test case 9](#_Toc531430036)

[Table 5 "Login" Test case 10](#_Toc531430037)

[Table 6 "Add" Test case 10](#_Toc531430038)

[Table 7 "Delete" Test case 11](#_Toc531430039)

[Table 8 "Edit" Test case 11](#_Toc531430040)

[Table 9 "passenger information" Test case 12](#_Toc531430041)

[Table 10 "Passenger booking" Test case 13](#_Toc531430042)

[Table 11 "view ticket" Test case 14](#_Toc531430043)

[Table 12 "Operator home page " Test case 14](#_Toc531430044)

[Table 13 "passenger home page" Test case 15](#_Toc531430045)

[Table 14 software testing resources 17](#_Toc531430046)

[Table 15 Testing Task's schedule 18](#_Toc531430047)

[Table 16 Plan Approvals 19](#_Toc531430048)

# Introduction

The Software Test Plan (STP) is intended to describe how the testing phase will be approached during the development of the Railway.Manage(); project. This document is designed to specify the scope, resources, and schedule of the test plan.   
  
This plan is required to identify the features to be tested, alongside specifying the types of tests to be performed and the members responsible for carrying out these tests.

# Objectives

This document is expected to specify the testing plan for the Railway.Manage(); project. This will be achieved by outlining the plan’s scope, resources and the allocated schedule for completing each task of the testing phase.

STP will also outline how the testing will be approached by specifying the types of tests to be conducted, their deliverables, and allocation of the necessary resources to perform these tests.  
This plan must be accurate and complete in order for all the functions to be tested in a suitable and organized manner to assure that they work as intended by both the team and the client alike.

* 1. **Testing Strategy**

Each function of this project must pass several testing stages to evaluate whether it satisfies the client’s requirements listed in the SRS document, and to compare it to the expected outputs outlined by the testing team.

For each task of the testing phase, a test plan is defined alongside the deliverables, constraints and assumptions of these tasks. It will also go through the appropriate test type in order to judge whether it fails or passes the performance criteria.

Testing will be performed at several stages in the life cycle of the project’s development. Since testing is a very dependent operation, planning and conducting tests will be a continuous process throughout the development phase of the Railway.Manage(); project. No new components or features are added unless they pass the criteria made by the testing team.

Note that any major issues discovered by the test plan must be clearly informed to the project manager, development team and the rest of the testing personnel.  
(A standard test approach is clearly defined in great detail in Section 5 of this document. Please refer to it if needed)

* 1. **Scope**

As previously stated, the testing phase requires continuous rework and updates for it to be successfully conducted. The testing team will provide upgrades to this document regularly to state whether any changes occurred during the testing stage of this project.  
  
**Any updates incorporated into this document must be informed earlier to the development team and the project manager alike.**

* 1. **Reference Material**

**Listed below are the references used in this document:**

[1] IEEE Standard for Software Project. [Online] 1998. IEEE Std 1058-1998.

* 1. **Definitions and Acronyms**

Any technical terminology mentioned in this document is specified and explained in Table 1.

Table 1 Definitions

|  |  |
| --- | --- |
| **Terminology** | **Definition** |
| **IEE Standard** | A universal standard followed by software developers. |
| **STP** | A test plan that coordinates any tests made in software development, to make sure that the project fulfils its preconditions. |
| **SRS** | A document that specifies all the requirements that needs to be completed before the completion of the project. |
| **Bugs** | Errors and issues that can be encountered during the run time. |
| **Interface** | The view provided to the user, considered the connection between the user and the system. |
| **Errors** | Deficiencies and faults during the run time of the system. |

Table 2 below lists all the acronyms used in this document.

Table 2 Acronyms

|  |  |
| --- | --- |
| **Acronym** | **Definition** |
| **STP** | Software Test Plan. |
| **SRS** | Software Requirement Specifications. |
| **IEE** | Institute of Electrical and Electronics Engineers. |
| **GUI** | Graphical User Interface. |

# Test items

As the requirements are approved in the Software Requirements Specification (SRS), the system will be tested as a whole to ensure it meets its requirements. And as what has been approved on Software Design Specification (SDS) the system will be tested to ensure that the design and how the developed software system works matches the pre-listed information on it. Also, the interface will be tasted as well to ensure that it meets the GUI standards. The data with the database integrity will be tested alike.

# Features to be tested

Railway.Manage(); application will adhere to the requirements specified in the SRS document, section 3.2; Functional Requirements. Also it will follow System Functionality identified in the SDS document, section 2.1.

# Approach

## 4.1 Data and Database Integrity Testing

The database should be tested as a separate component, and without integration with the Railway.manag(); it will be tested. This guarantees that database access methods and processes function properly without data discrepancy.

The technique of this test is access method to the database and figure out valid and invalid process. It will check the database to certify that the data is populated as intended and review the returned data to emphasize that the correct data was retrieved.

**accomplishment criteria:** All database access methods and processes operate as expected without any data perversion.

**Special consideration:** Small sized database with limited number of records should be used to increase the visibility of unexpected results.

### 4.1.1 Test Cases

In this section, test cases of testing the database follow ORACLE guidelines [5]. The test cases are not limited to the following tests in Table 3.

Table 3 Data and Database Integrity Test Cases

|  |  |  |  |
| --- | --- | --- | --- |
| Test ID | Test Description | Expected Result | Verified  (Yes/No) |
| 1 | Retrieving records of each table in the database. | All rows are retrieved | Yes |
| 2 | Entering duplicated primary key in each table in the database. | unaccepted procedure. | Yes |
| 3 | Updating a row in each table. | The rows are updated. | Yes |
| 4 | Inserting NULL values to columns that do not accept NULL values. | unaccepted procedure. | Yes |
| 5 | Inserting values to column with different data type. | unaccepted procedure. | Yes |
| 6 | Inserting a row with a column that exceeds its data length. | unaccepted procedure. | Yes |
| 7 | Inserting a row with inconsistent data in the foreign key field. | unaccepted procedure. | Yes |
| 8 | Delete a row in each table. | The row is deleted with the following considerations:  🡪 If the row has a primary key that is a foreign key in another table with a “Cascade on Delete” property, then the corresponding records will automatically be deleted from the latter table.  🡪If the row has a primary key that is a foreign key in another table with “Set NULL on Delete” property, then the corresponding records in the latter table will have the foreign keys set to null. | Yes |

## 4.2 Component Testing

The key goal of component testing is to take each testable component of the software in the application, isolate it from other components, and determine whether it functions as expected or not. Component testing has proven its value in that a large percentage of defects are identified during its use.

**Technique:** Each component is tested separately before integrating them. Valid and invalid data is used to verify functions listed in the SDS.

1. The expected results occur when valid data is used.
2. The appropriate error/warning messages are displayed when invalid data is used.
3. All rules are properly applied.

**Completion criteria:** Each component functions as required without any defects.   
**Special consideration:** Divide the components into smaller tasks to increase visibility of unexpected behavior.

### 4.2.1 Test Cases

is section providing test cases for each component of the application. Test cases verify each use case in the use case diagram.

#### 4.2.1.1 Common Functions

##### *4.2.1.1.1 Ticket Viewing*

Table 4 shows test cases for the “Login” function. Test cases are not limited to the following items in the table.

Table 4 "view Ticket " Test case

|  |  |
| --- | --- |
| Test ID | View Ticket |
| Prerequisite | User own a ticket Id, which is already registered in DB. |
| Test procedure | Tap on “View Ticket “button and enter a Ticket Id with each of the following:   1. Correct ticket Id. 2. Incorrect ticket Id. |
| Expected Result | Tap on “View Ticket “button:   1. Successful ticket Id and show full information about user trip. 2. Error message is displayed: “incorrect ticket Id”. |
| Actual Result | Same as expected results. |
| Verified (Yes/No) | Yes |

#### 4.2.1.2 User1: Operator

##### 4.2.1.2.1 Login

Table 5 shows test cases for the “Login” function. Test cases are not limited to the following items in the table.

Table 5 "Login" Test case

|  |  |
| --- | --- |
| Test ID | Login |
| Prerequisite | Own an account. User is already registered in DB. |
| Test procedure | Tap on “Login “button with each of the following:   1. Correct username and password. 2. Correct username and incorrect password. 3. Incorrect username and correct password. 4. Incorrect username and password. |
| Expected Result | Tap on “Login “button:   1. Successful login and redirects an operator to his/her homepage. 2. Error message is displayed: “incorrect username and password”. 3. Contact administration if you lost your password. |
| Actual Result | Same as expected results. |
| Verified  (Yes/No) | Yes |

##### 4.2.1.2.2 Add

Table 6 shows test cases for the “Adding” function. Test cases are not limited to the following items in the table.

Table 6 "Add" Test case

|  |  |
| --- | --- |
| Test ID | Add |
| Prerequisite | Own an account. User is already registered in DB. |
| Test procedure | full each of the following information:   1. Trip Id. 2. Trip State. 3. Departure city 4. Arrival city. 5. Departure time. 6. Arrival time.   Tap on “Save “button to insert new trip on the schedule. |
| Expected Result | Tap on “Save “button:   1. Successful adding trip. 2. Error message is displayed: “Trip already exist”. |
| Actual Result | Same as expected results. |
| Verified  (Yes/No) | Yes |

##### 4.2.1.2.3 Delete

Table 7 shows test cases for the “Deleting” function. Test cases are not limited to the following items in the table.

Table 7 "Delete" Test case

|  |  |
| --- | --- |
| Test ID | Delete |
| Prerequisite | Own an account. User is already registered in DB. |
| Test procedure | full each of the following information:   1. Trip Id. 2. Trip State. 3. Departure city 4. Arrival city. 5. Departure time. 6. Arrival time.   Tap on “Delete “button to delete existing trip on the schedule. |
| Expected Result | Tap on “Delete “button:   1. Successful delete trip. 2. Error message is displayed: “Trip does not exist”. |
| Actual Result | Same as expected results. |
| Verified  (Yes/No) | Yes |

##### 4.2.1.2.4 Edit

Table 8 shows test cases for the “Editing” function. Test cases are not limited to the following items in the table.

Table 8 "Edit" Test case

|  |  |
| --- | --- |
| Test ID | Edit |
| Prerequisite | Own an account. User is already registered in DB. |
| Test procedure | full each of the following information:   1. Trip Id. 2. Trip State. 3. Departure city 4. Arrival city. 5. Departure time. 6. Arrival time.   Tap on “Update “button to update existing trip on the schedule. |
| Expected Result | Tap on “Update” button:   1. Successful updating trip. 2. Error message is displayed: “Trip does not exist”. |
| Actual Result | Same as expected results. |
| Verified  (Yes/No) | Yes |

#### 4.2.1.3 User1: Passenger

##### 4.2.1.3.1 Passenger information

Table 9 shows test cases for the “Passenger information” function. Test cases are not limited to the following items in the table.

Table 9 "passenger information" Test case

|  |  |
| --- | --- |
| Test ID | Passenger information |
| Prerequisite | The user is a Passenger. |
| Test procedure | full each of the following information:   1. First name. 2. Last name 3. National Id. 4. Email. 5. Phone number. 6. Arrival time.   Tap on “Next “button to get a Ticket id. |
| Expected Result | Tap on “Next” button:   1. information entered is correct. 2. Error message is displayed: “please enter at least first name, last name, national id”. |
| Actual Result | Same as expected results. |
| Verified  (Yes/No) | Yes |

##### 4.2.1.3.2 Passenger Booking

Table 10 shows test cases for the “Passenger booking” function. Test cases are not limited to the following items in the table.

Table 10 "Passenger booking" Test case

|  |  |
| --- | --- |
| Test ID | Passenger Booking |
| Prerequisite | The user is a Passenger. |
| Test procedure | full each of the following information:   1. One way or round trip. 2. Departure station 3. Arrival station. 4. Departure date. 5. Return date. 6. Number of adults and children.   Tap on “Next “button to go to information stage. |
| Expected Result | Tap on “Next” button:   1. information entered is correct. 2. Error message is displayed: “please select either one way or round trip”. 3. Error message is displayed: “please select source and destination”. 4. Error message is displayed: “please select departure and return date”. 5. Error message is displayed: “please select departure date”. 6. Error message is displayed: “please select at least one adult”. |
| Actual Result | Same as expected results. |
| Verified  (Yes/No) | Yes |

## Integration Testing

Integration testing is implemented after component testing to make sure the components interact with each other and refers to connecting all the components or units together, resulting in the complete system.

**Technique:** Test the functions of the combined components in an incremental manner by adding one component at a time. This will help to find an error easily and reduces execution time. That will increase the visibility of the bugs.

**Completion criteria:** Combining all components to reach the end of the system that should work as intended predicted.  
**Special consideration:** Go back to component testing if the integration of a new component incorporates errors.

### 4.3.1 Test case

#### 4.3.1.1 View Ticket

Table 11 shows test cases of the integration between components in “View ticket page”. Test cases are not limited to the following items in the table.

Table 11 "view ticket" Test case

|  |  |
| --- | --- |
| Test ID | View Ticket |
| Prerequisite | User own a ticket Id, which is already registered in DB. |
| Test procedure | Try the following procedure:   1. Access the user’s “Home page” and do component test case in ( section *4.2.1.1.1*) |
| Expected Result | The results of the previous separated procedure are the same as the components result in the previous section with the following additional results:   * Full information about entered ticket . |
| Actual Result | Same as expected results. |
| Verified (Yes/No) | Yes |

#### 4.3.1.2 Operator home page

Table 12 shows test cases of the integration between components in “Operator home page”. Test cases are not limited to the following items in the table.

Table 12 "Operator home page " Test case

|  |  |
| --- | --- |
| Test ID | Operator home page |
| Prerequisite | The user is logged into system. |
| Test procedure | Try the following procedure:   1. Access the operator’s “Home page” and do component test case in ( section 4.2.1.2) |
| Expected Result | The results of the previous separated procedure are the same as the components result in the previous section with the following additional results:   1. The operator’s update reflected on the database. 2. All confirmation and error message are displayed. |
| Actual Result | Same as expected results. |
| Verified  (Yes/No) | Yes |

#### 4.3.1.3 Passenger home page

Table 13 shows test cases of the integration between components in “Passenger home page”. Test cases are not limited to the following items in the table.

Table 13 "passenger home page" Test case

|  |  |
| --- | --- |
| Test ID | Passenger home page |
| Prerequisite | The user is passenger. |
| Test procedure | Try the following procedure:   1. Access the passenger’s “Information page” and do component test case in (section 4.2.1.3.1). 2. Access the passenger to “Booking page “and do component test case in (section 4.2.1.3.2). |
| Expected Result | The results of the previous separated procedure are the same as the components result in the previous section with the following additional results:   1. The passenger registration reflected on the database. 2. All confirmation and error message are displayed. |
| Actual Result | Same as expected results. |
| Verified  (Yes/No) | Yes |

## 4.4 Job Stream Testing

Job Stream Testing is to ensure that the Railway.Manage(); system operates in the production environment. It makes sure that consists of a sequence of jobs flow is running correctly together with times, priorities, and other dependencies that determine the order of processing the requirement.

## 4.5 Interface Testing

Interface testing is executed to guarantee that the interactions between the whole system and components’ passage of data and control are working properly and errors are handled perfectly.

This section emphasizes on the connection between the application and the external component.

## 4.6 Security Testing

To guarantee that all authorization is in deeded the Security Testing will be generated , it ensures that the systems control and audit ability features of the application are functional. And to prevent unwanted access from unauthorized users to change or even view any critical information that is not prohibited. For example, the passenger is not allowed to modify any information in the train schedule, only operators are able to.

**Technique:** Accessing the system with incorrect usernames and passwords should restrict the user from entering the system.

**Completion criteria**: Security testing verifies that only registered users can access their accounts.

## 4.7 Recovery Testing

The recovery testing is a non-functional testing that is necessary for the system. It checks how the software will recover after any crash or hardware failure. Recovery testing will force the system to fail in different ways then verify that the recovery is performed properly. In Railway.Manage(); the system is designed to recover automatically by itself by implementing check pointing strategy where a fault tolerance is added to the software and whenever there is failure, it will save a snapshot of the current software state then restart from that point. For data failure, there will be a backup for all the data in the database regularly.

## 4.8 Performance Testing

The performance test of the Railway.Manage(); will be conducted via measuring the response time, sturdiness under heavy workload, and consistency of the system’s various test runs.

This type of testing is especially critical to assure and validate whether the performance requirements are satisfied, therefore it is essential to conduct this test several time with different scenarios and inputs at each try to investigate how deficiencies can be generated from various events.

To consider a response time acceptable, it should not exceed 3 seconds during the retrieval of any necessary data from the database, initial load up cannot exceed 20 seconds and transitioning from one interface to another should happen within the span of 0.5 seconds. GUI should be coherent and consistent across several devices; the testing team will run the applications on various devices to check for any inconsistencies in the application’s look-and-feel.

**Technique**: Run the different functions of the system under various scenarios and test-cases. **Completion criteria:** Valid output of the functions, acceptable runs time and execution as determined by the testing team.

## 4.9 Regression Testing

When adding any new feature or modification to the system, the system must be refactored and tested several times in order to assure that the modifications are not affecting the previously added functions.

This test is critical to carry out, whether the modifications seem insignificant such as a new line in the code, or adding a whole new component to the system.

**Technique**: After integrating any addition, sub-components alongside the whole system should be tested to assure that they work as expected.

**Completion criteria:** Successful integration, no new errors should be generated from the new addition.

## 4.10 Beta Testing

Beta testing is a pre-release version of Railway.Manage(); software and it is important to make sure that the software meets the required functional requirements and to detect any faults or defects that may be found in the software. This test will be conducted by some selected customers.

This test is done in an environment that cannot be controlled by the developers. During the test, the customers will record every problem that may arise and then report these issues to the developers to modify it.

# Pass / fail criteria

Railway.Manage(); must pass this criteria in order for it to be considered successful:

Functions related to each of the primary users shall work as specified in the SRS document.

1. All the data recorded in the database must be accurate and up-to-date.
2. Queries and retrievals must be done successfully.
3. If any bugs were discovered, they must be eliminated, and the affected feature should pass all the testing cases again.
4. All invalid user inputs should be considered and dealt with appropriately to avoid run-time errors.
5. GUI should be consistent and coherent in multiple devices.
6. Inserting the booking and all related data should be recorded in the database successfully.
7. Retrieval of information should not take more than 1.5 seconds.
8. Login function must be secure and impenetrable.
9. All functions should comply with the requirements set in the SRS document.

If any of these conditions are not met, they must be fixed immediately before commencing the implementation on any other function.

# Testing process

This chapter defines test deliverables, testing tasks, responsibilities, resources and schedule.

* 1. **Test Deliverables**

While working on the test stage, the software test plan document (STP document) will be produced. After finishing the testing stage, the final software project will be delivered and presented.

* 1. **Testing Tasks**

There are several testing tasks during the test stage. These tasks are listed below:

* Prepare the SRS and SDS documents to make sure the software meets the requirements and is functioning as expected.
* Produce the STP while working during test stage.
* Prepare the hardware and software test environment.
* Perform all test activities to the software using different methods.
* Handle errors that may occur in the system during the test stage.
* Maintain the software when any change occurs.
  1. **Responsibilities**

During the test stage for the software, all the team members will be responsible for testing the software. Also, all the test tasks must be integrated, and each component specification followed. Moreover, the errors that may occur in the software during the test stage must be handled.

* 1. **Resources**

Table 3 below shows the resources that will be needed to complete the software testing phase.

Table 14 software testing resources

|  |  |
| --- | --- |
| Resource | Description |
| Hardware | Personal computers |
| software | * Net Beans 8.1 IDE * MySQL Workbench |
| human | Skilled team members, these skills are mentioned previously in the SPMP document, section 3.1.3, project staff training. |

* 1. **Schedule**

Table 4 Below shows the schedule of the testing tasks.

Table 15 Testing Task's schedule

|  |  |
| --- | --- |
| Task | Date |
| Software Requirement Specification (SRS) | October 29, 2018 |
| Software Design Specification (SDS) | November 9, 2018 |
| Develop test cases | November 20, 2018 |
| Perform all test activities using different methods. | November 26, 2018 |
| Handling errors occurred during test stage | November 29, 2018 |
| Modify the software | November 30, 2018 |
| Develop STP document | December 1, 2018 |

# Environmental requirements

This chapter define the environmental requirements: hardware, software, security, server, publications*,* and incorporated risks and assumptions.

* 1. **Hardware**

The necessary hardware needed to apply the testing tasks is a regular functioning desktop computer.

* 1. **Software**

The required software needed to execute the testing tasks are:

* Net Beans 8.1 IDE
* MySQL Workbench
  1. **Security**

To produce high secure software, the system is developed using Java programming language which provides several features that are designed to improve the security of the developed software. Also, the software provides a log in strategy to the system and database so making changes in the system or database is permitted only for the registered operators.

* 1. **Server**

MySQL Server will be required to employ the testing tasks.

* 1. **Publications**

There are two required documents along with the STP document that are produced during the testing stage. These documents are: Software Requirement Specification (SRS) document and Software Design Specification (SDS) document to make sure that the software is meeting the requirements and is functioning as expected.

* 1. **Risks and Assumptions**

To produce a high-quality software, the system must function as expected. Therefore, Railway.Manage(); will go through the Software Development Life Cycle (SDLC) stages. The aim of the SDLC is to produce a high-quality software that exceeds the expectations. The rapid development of the software may cause changes in the code which leads to changes in the plan. Listed below are the potential risks and their respective plans of action.

* + 1. **Test Item Availability**

During testing unit, an inaccessibility of an item may interrupt the testing of that unit which in turn delays the whole testing process. The contingency plan for this issue is to test other items that are ready and waiting to be tested in that unit until the incomplete item is ready and available.

* + 1. **Test Resources Availability**

During testing units, if the resources that need to be tested is unavailable, it may defer the test plan for the completed units. For example, if there was a heavy load or any technical issues on the server, it may cause a shutdown which may harm the system. The contingency plan for this issue is to have a backup server so if there is any issue in the original server, the system will shift to this backup server.

* + 1. **Time Constraints**

Any changes in the software like adding new or deleting features to the software, or any unexpected delays will cause the delivered time to be delayed in return. The contingency plan for this issue is to make sure all the functions in the software are working properly as planned. Also, the personnel may expect an increase in the regular working hours to make sure the estimated delivery time is met.

# Change management procedures

As stated previously in section 1, updates are expected to occur during the testing phase of this project. All of these changes must be clearly documented after acquiring the approval of the project manager and the client.

After proposing a necessary update, a meeting will be held in order to discuss and review the consequences of the proposed change.  
Note that no changes will be done unless they are reviewed and approved by the individuals stated in section 9.

# Plan approvals

Table 5 below outlines the individuals responsible for approving any modifications to the plan.

Table 16 Plan Approvals

|  |  |  |
| --- | --- | --- |
| Name | Signature | Date of Approval |
| Project Supervisor: Wadha Almattar |  |  |
| Project Manager: Muneera Alhajri |  |  |
| Testing Team Leader: Tasneem Dosoqi |  |  |
| Development Team Leader: Reema Alyousef |  |  |
| Research Unit Leader: Rahaf Alzahrani |  |  |