# 

# Test Plan

Timeline-Manager



*Group 7:*

*Amelie Löwe*

*Aya Kathem*

*Caroline Nilsson*

*Indré Kvedaraite*

*Johan Eriksson*

*Pranav Patel*

*Stefanos Bampovits*

Test Plan 1

Timeline-Manager 1

1. Introduction 3

2. Objectives 3

3. Scope 3

4. Strategy 3

5. Test-Log 4

6. Requirements 4

7. Risk 4

8. Tabular (Test-Cases) 5

8.1. Test-Case: Add Timeline 5

8.2. Test-Case: Add Event 6

8.3. Test-Case: Add Duration Event 7

8.4. Test-Case: Edit Event 8

8.5. Test-Case: Edit Duration Event 9

8.6. Test-Case: Delete Timeline 10

8.7. Test-Case: Delete Event 12

8.8. Test-Case: Save and Load Timeline 13

# Introduction

The Test Plan contains of five test phases that are located within the iterations, meaning that each iteration also has a test phase where new tests are created for the implementations for that specific iteration. But aside from the new tests all the previous tests are also performed to ensure that changes within the implementation has not affected the functionality of previous implementations.

The main goal with the tests are to ensure functionality to all requirements given by the stakeholder but also to find flaws or other errors that might need to be fixed by the developers.

# Objectives

The purpose of the Test Plan is to clarify the Test-Cases and give structure to the Test-Phases. Another important purpose of the Test Plan is to have an overview of passed and failed tests and to clarify what the goal of the tests are.

# Scope

The tests performed on the Timeline-Manager is both JUnit testing (automated-testing) and Manual-Testing (done by interacting with the application). The JUnit tests are created for logics and models while Manual-Testing is used for the User-Interface (buttons, popup windows, etc.). Each iteration a member of the group will be responsible for the tests and in each test-phase the following will be performed by the Tester:

* Test-Cases (New implementations)
* Overview previous Test-Cases
* Implement JUnit tests
* Perform Manual-Tests
* Perform Manual-tests and JUnit tests
* Update Test-Log

|  |  |  |
| --- | --- | --- |
| Test Class | Test-Case | Step |
| ApplicationControlTest | - | - |
| ApplicationTest | 001  002  003  (Set/Get) no Test-Case | 6, 7  6  7 |
| TimelineControlTest | 001  006 | 1, 2, 3, 4, 5, 6, 7  1, 2 |
| TimelineTest | (Set/Get) no Test-Case |  |
| EventControlTest | 002  003  004  005  007 | 1, 2, 3, 4, 5, 6  1, 2, 3, 4, 5, 6, 7  1, 2, 3, 4, 5, 6  1, 2, 3, 4, 5, 6, 7, 8  1 |
| EventTest | (Set/Get) no Test-Case |  |
| FileHandlerTest | 008 | 1, 2 |
| Manual-Tests | 006  007  008 | 3, 4, 5, 6  2, 3, 4, 5  3, 4, 5 |

# Strategy

The approach in testing the requirements functionality will mainly be dynamic but in addition to the dynamic tests the group has online code-review meetings. At these meetings the group members perform static tests and feedback regarding the implementation is given to the developer. Each developer also performs static analysis on their own implementation to find errors and mistakes along with raising issues for discussion within the group. The dynamic tests will be created during the specific iteration for intended implementation, but in order to prevent changes that flaws the functionality all tests will be performed again after each finished iteration.

# Test-Log

After each finished iteration the developer responsible for the tests run the tests for all implemented code and update the pass/fail status. Failed tests will be stated in the tabular below for easy access in next iteration.

|  |  |  |  |
| --- | --- | --- | --- |
| Tester | Test-Cases ID | Failed Tests | Test-Phase |
| Pranav Patel | 001 | N/A | #1 |
| Johan Eriksson | 002, 003 | N/A | #1 |
| Indre Kvedaraite | 004, 005 | N/A | #2 |
| Caroline Nilsson | 006, 007, 008 | - | #3 |

# Requirements

* Access to GitHub repository: Group7-1DV508/1DV508-group7
* Access to GitHub repository: Group7-1DV508/1DV508-group7-docs
* Experience/knowledge of working with JUnit

# Risk

|  |  |  |
| --- | --- | --- |
| Risk: | Impact on Project: | Mitigation: |
| Misunderstandings regarding tasks or implementation within the group | Implementation delay | Have an open dialog within the group during the development process. |
| Changes made to the already tested implementations | Functionality flaws and errors | All tests are performed at the end of each iteration. |
| Insufficient tests | Hidden errors within the application. Requirements not fulfilled | Group code-review meetings where implementations, JUnit tests and Test-Cases are reviewed by the group and feedback is given. |

In order to prevent negative impacts on the project following risks have been evaluated and actions have been taken to mitigate project impacts. In the tabular below the risks and how these risks might affect the project along with group efforts to minimize the negative impacts on the project.

# Tabular (Test-Cases)

* 1. Test-Case: Add Timeline

|  |  |
| --- | --- |
| Add Timeline | |
| Test Case ID: 001 | Test Designed by: Pranav Patel |
| Test Priority(High/Med/Low): High | Test Designed Date: 2017/04/19 |
| Test Title: Add Timeline | Test Executed by: Pranav Patel |
| Description: Add a Timeline to the application | Test Executed Date: 2017/04/19 |

|  |
| --- |
| Pre-conditions: App object initialized (application running) |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status(Pass/Fail)** | **Notes** |
| 1 | Add timeline with correct name, start and end dates | Timeline object, string, LocalDateTime variable, LocalDateTime variable | Method returns true, timeline added | Method returned true, timeline added to timeline list | **Pass** | JUnit-Test |
| 2 | Add timeline with incorrect input for name, start and end dates | Timeline object, string, LocalDateTime variable, LocalDateTime variable | Method returns false, timeline not added | Method returned false, timeline was not added to timeline list | **Pass** | JUnit-Test |
| 3 | Check if added timeline has correct name | Timeline object, string | Method returns correct name | Method returned correct name | **Pass** | JUnit-Test |
| 4 | Check if added timeline has correct start date | Timeline object, LocalDateTime variable | Method returns correct start date | Method returned correct start date | **Pass** | JUnit-Test |
| 5 | Check if added timeline has correct end date | Timeline object, LocalDateTime variable | Method returns correct end date | Method returned correct end date | **Pass** | JUnit-Test |
| 6 | Check if added timeline is ”current” | Timeline object, String, LocalDateTime, LocalDateTime | Method returns name, start date and end date of newly added Timeline | Method returned name, start and end date of newly added timeline | **Pass** | JUnit-Test |
| 7 | Check if the ChangeListener is called when timeline is added | Timeline object, String, LocalDateTime, LocalDateTime | Variable = true | Variable was true | **Pass** | JUnit-Test |

* 1. Test-Case: Add Event

|  |  |
| --- | --- |
| Add Event | |
| Test Case ID: 002 | Test Designed by: Johan Eriksson |
| Test Priority(High/Med/Low): High | Test Designed Date: 2017/04/15 |
| Test Title: Add Event | Test Executed by: Johan Eriksson |
| Description: Add Event with name, description and start date. | Test Executed Date: 2017/04/15 |

|  |
| --- |
| Pre-conditions: Timeline is created |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status(Pass/Fail)** | **Notes** |
| 1 | Add event with correct input for name, description and date | Event object, string, string, LocalDateTime variable | Method returns true, event added to timeline’s | Method returned true, event was added to event list | **Pass** | JUnit-Test |
| 2 | Add event with incorrect input for name, description and date | Event object, string, string, LocalDateTime variable | Method returns false, event not added | Method returned false, event was not added to event list | **Pass** | JUnit-Test |
| 3 | Check if added event has correct name | Event object, string | Method returns correct name | Method returned correct name | **Pass** | JUnit-Test |
| 4 | Check if added event has correct description | Event object, string | Method returns correct description | Method returned correct description | **Pass** | JUnit-Test |
| 5 | Check if added event has correct date | Event object, LocalDateTime variable | Method returns correct date | Method returned correct date | **Pass** | JUnit-Test |
| 6 | Check if ChangeListener is called when Event is added | Event object, string, string, LocalDateTime variable | Variable = true | Variable was true | **Pass** | JUnit-Test |

* 1. Test-Case: Add Duration Event

|  |  |
| --- | --- |
| Add Duration Event | |
| Test Case ID: 003 | Test Designed by: Johan Eriksson |
| Test Priority(High/Med/Low): High | Test Designed Date: 2017/04/15 |
| Test Title: Add Duration Event | Test Executed by: Johan Eriksson |
| Description: Add event with name, description, start and end date | Test Executed Date: 2017/04/15 |

|  |
| --- |
| Pre-conditions: Timeline is created |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status(Pass/Fail)** | **Notes** |
| 1 | Add event with correct input for name, description, start and end dates | Event object, string, string, LocalDateTime variable, LocalDateTime variable | Method returns true, event added to timeline’s | Method returned true, event was added to event list | **Pass** | JUnit-Test |
| 2 | Add event with incorrect input for name, description, start and end dates | Event object, string, string, LocalDateTime variable, LocalDateTime variable | Method returns false, event not added | Method returned false, event was not added to event list | **Pass** | JUnit-Test |
| 3 | Check if added event has correct name | Event object, string | Method returns correct name | Method returned correct name | **Pass** | JUnit-Test |
| 4 | Check if added event has correct description | Event object, string | Method returns correct description | Method returned correct description | **Pass** | JUnit-Test |
| 5 | Check if added event has correct start date | Event object, LocalDateTime variable | Method returns correct start date | Method returned correct start date | **Pass** | JUnit-Test |
| 6 | Check if added event has correct end date | Event object, LocalDateTime variable | Method returns correct end date | Method returned correct end date | **Pass** | JUnit-Test |
| 7 | Check if ChangeListener is called when Event is added | Event object, string, string, LocalDateTime variable | Variable = true | Variable was true | **Pass** | JUnit-Test |

* 1. Test-Case: Edit Event

|  |  |
| --- | --- |
| Edit Event | |
| Test Case ID: 004 | Test Designed by: Indre Kvedaraite |
| Test Priority(High/Med/Low): High | Test Designed Date: 2017/04/22 |
| Test Title: Edit Event | Test Executed by: Indre Kvedaraite |
| Description: Editing existing event, changing its name, date and description | Test Executed Date: 2017/04/22 |

|  |
| --- |
| Pre-conditions: Timeline and Event is created |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status(Pass/Fail)** | **Notes** |
| 1 | Edit event’s name with correct input | Event object, string | Method returns true, event’s name is changed | Method returned true, event’s name changed | **Pass** | JUnit-Test |
| 2 | Edit event’s description with correct input | Event object, string | Method returns true, event’s description changed | Method returned true, event’s description changed | **Pass** | JUnit-Test |
| 3 | Edit event’s date with correct input | Event object, LocalDateTime variable | Method returns true, event’s date changed | Method return true, event’s date changed | **Pass** | JUnit-Test |
| 4 | Edit event’s name with empty string | Event object, empty or null string | Method returns false, event’s name not changed | Method returned false, event’s name was not changed | **Pass** | JUnit-Test |
| 5 | Edit event’s description with empty string | Event object, empty or null string | Method returns false, event’s description not changed | Method returned false, event’s description was not changed | **Pass** | JUnit-Test |
| 6 | Edit event’s date with empty LocalDateTime variable | Event object, null LocalDateTime variable | Method returns false, event’s date not changed | Method returned false, event’s date was not changed | **Pass** | JUnit-Test |

* 1. Test-Case: Edit Duration Event

|  |  |
| --- | --- |
| Edit Duration Event | |
| Test Case ID: 005 | Test Designed by: Indre Kvedaraite |
| Test Priority(High/Med/Low): High | Test Designed Date: 2017/04/22 |
| Test Title: Edit Duration Event | Test Executed by: Indre Kvedaraite |
| Description: Editing existing event, changing its name, start and end dates and description | Test Executed Date: 2017/04/22 |

|  |
| --- |
| Pre-conditions: Timeline and Event is created |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status(Pass/Fail)** | **Notes** |
| 1 | Edit event’s name with correct input | Event object, string | Method returns true, event’s name is changed | Method returned true, event’s name changed | **Pass** | JUnit-Test |
| 2 | Edit event’s description with correct input | Event object, string | Method returns true, event’s description changed | Method returned true, event’s description changed | **Pass** | JUnit-Test |
| 3 | Edit event’s start date with correct input | Event object, LocalDateTime variable | Method returns true, event’s start date changed | Method returned true, event’s start date changed | **Pass** | JUnit-Test |
| 4 | Edit event’s end date with correct input | Event object, LocalDateTime variable | Method returns true, event’s end date changed | Method returned true, event’s end date changed | **Pass** | JUnit-Test |
| 5 | Edit event’s name with empty string | Event object, null or empty string | Method returns false, event’s name not changed | Method returned false, event’s name didn’t change | **Pass** | JUnit-Test |
| 6 | Edit event’s description with empty string | Event object, null or empty string | Method returns false, event’s description not changed | Method returned false, event’s description didn’t change | **Pass** | JUnit-Test |
| 7 | Edit event’s start date with empty LocalDateTime variable | Event object, null LocalDateTime variable | Method returns false, event’s start date not changed | Method returned false, event’s start date didn’t change | **Pass** | JUnit-Test |
| 8 | Edit event’s end date with empty LocalDateTime variable | Event object, null LocalDateTime variable | Method returns false, event’s end date not changed | Method returned false, event’s end date didn’t change | **Pass** | JUnit-Test |

* 1. Test-Case: Delete Timeline

|  |  |
| --- | --- |
| Delete Timeline | |
| Test Case ID: 006 | Test Designed by: Caroline Nilsson |
| Test Priority(High/Med/Low): Medium | Test Designed Date: 2017/05/01 |
| Test Title: Delete Timeline | Test Executed by: Caroline Nilsson |
| Description: Delete an existing timeline from both the Timeline Manager and the XML-File if the Timeline has been saved. | Test Executed Date: 2017/05/02 |

|  |
| --- |
| Pre-conditions: Timeline and Event is created |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status(Pass/Fail)** | **Notes** |
| 1 | Delete saved Timeline | Timeline | Timeline is removed from ArrayList of open timelines, XML-File is deleted | N/A | **N/A** | JUnit-Test |
| 2 | Delete non-saved Timeline | Timeline | Timeline is removed from ArrayList with open timelines | Timeline was removed from ArrayList | **Pass** | JUnit-Test |
| 3 | Delete Timeline: confirmation window | “Delete Timeline” Button | When clicked a confirmation popup window show | Conformation window shows | **Pass** | Manual-Test |
| 4 | Conformation window (Timeline),  Buttons | Conformation window:  “Ok” Button | When “Ok” is clicked the timeline is deleted and does not show to the user. | “Ok” = timeline deleted | **Pass** | Manual-Test |
| 5 | Conformation window (Timeline),  Cancel Button | Conformation window:  “Cancel” Button | When “Cancel” is clicked timeline resumes before delete button was clicked | “Cancel” = resumes at timeline | **Pass** | Manual-Test |
| 6 | Delete Timeline (View Update) |  | When a timeline is deleted the View shall not contain the timeline | Timeline is removed from the View (empty if no timelines are open otherwise the first one shows) | **Pass** | Manual-Test |

* 1. Test-Case: Delete Event

|  |  |
| --- | --- |
| Delete Event | |
| Test Case ID: 007 | Test Designed by: Caroline Nilsson |
| Test Priority(High/Med/Low): Medium | Test Designed Date: 2017/05/01 |
| Test Title: Delete Event | Test Executed by: Caroline Nilsson |
| Description: Delete an Event from Timeline. | Test Executed Date: 2017/05/02 |

|  |
| --- |
| Pre-conditions: Timeline and Event is created |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status(Pass/Fail)** | **Notes** |
| 1 | Delete Event from timeline | Event | Event is removed from timeline ArrayList of events | Event was removed from ArrayList | **Pass** | JUnit-Test |
| 2 | Delete Event confirmation window | Delete event button | When delete event is clicked, a confirmation window shows to the user | Confirmation window shows to the user | **Pass** | Manual-Test |
| 3 | Confirmation Window (Event)  Buttons | Confirmation Window:  “Ok” Button | When “Ok” is clicked the event is deleted, | “Ok” = event deleted | **Pass** | Manual-Test |
| 4 | Confirmation Window (Event)  Buttons | Confirmation Window:  “Cancel” Button | When “Cancel” is clicked resumes at event information window | “Cancel” = resume at event information | **Pass** | Manual-Test |
| 5 | Event Deleted (View Update) |  | When event has been deleted, it no longer shows to the user and remaining events are aligned properly | Event is no longer visible.  Remaining events are aligned correctly | **Pass** | Manual-Test |

* 1. Test-Case: Save and Load Timeline

|  |  |
| --- | --- |
| Save & Load Timeline | |
| Test Case ID: 008 | Test Designed by: Caroline Nilsson |
| Test Priority(High/Med/Low): Medium | Test Designed Date: 2017/05/01 |
| Test Title: Save and Load Timeline | Test Executed by: Caroline Nilsson |
| Description: Saving a Timeline and the Events belonging to the Timeline. Load Timeline from XML-file | Test Executed Date: 2017/05/01 |

|  |
| --- |
| Pre-conditions: Timeline and Event is created |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status(Pass/Fail)** | **Notes** |
| 1 | Save timeline to an XML-file | Timeline, File | XML-File contains Timeline and Events belonging to the Timeline | XML-File contains the saved Timeline and Events | **Pass** | JUnit-Test |
| 2 | Load timeline from an XML-file | File | Timeline with correct input is created | Timeline has been created from XML-File with correct input for Timeline and Events | **Pass** | JUnit-Test |
| 3 | Save timeline, interrupted by pressing cancel in FileChooser | Timeline, File = null | Exception  Alert Window. |  | N/A | Manual-Test |
| 4 | Load timeline with incorrect File path | File = non XML-File | Exception  Alert Window |  | N/A | Manual-Test |
| 5 | Load timeline with correct File path | File | Timeline and Events shows to the user |  | N/A | Manual-Test |