TIMELINE MANAGER

TEST PLAN

Version 1.2

21/04/2017

VERSION HISTORY

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	Mironov Georgiana	17/04/2017	Oskar Mendel	19/04/2017	Test Plan draft
1.2	Mironov Georgiana	21/04/2017			Test Plan update

UP Template Version: 12/31/07

TABLE OF CONTENTS

1 INTRODUCTION 4
1.1 Purpose of the Test Plan document 4
1.2 Objectives 4
1.3 Scope 5
1.4 Risks and constraints 5
2 TEST STRATEGY 6
2.1 Unit testing 6
2.1.1 Testing Functional Requirements 6
2.1.2 Items to be Tested / Not Tested 7
2.2 Integration testing 7
2.2.1 Items to be Tested / Not Tested 7
2.3 System testing 8
2.3.1 Testing of Non-Functional Requirements 8
2.4 Operational acceptance testing 9
3 ENVIRONMENTAL REQUIREMENTS 10
4 TEST SCHEDULE 10
5 DELIVERABLES 10
6 DEPENDENCIES 11
6.1 Personnel Dependencies 11
6.2 Software Dependencies 11
7 DOCUMENTATION 11
APPENDIX A: REFERENCES 12
APPENDIX B: KEY TERMS 13

1 INTRODUCTION

This section describes what the purpose of this document is, what the objectives of this document are, as well as the scope of the tests possible constraints. The aim is to provide a framework that can be used, in this case, by the project leader and the testers to plan and execute the necessary tests in a timely and cost-effective manner.

1.1 PURPOSE OF THE TEST PLAN DOCUMENT

The Test Plan document documents and tracks the necessary information required to effectively define the approach to be used in the testing of the project's product, the Timeline Manager. The Test Plan document is created during the Planning Phase of the project. Its intended audience is the project manager, project team, and testing team. Some portions of this document may on occasion be shared with the client/user whose input into the testing process is needed.

1.2 OBJECTIVES

The primary objective of the test series is to ensure the validation of the requirements, and to clear any errors that will occur during the development and possibly affect the Timeline Manager during runtime. All tests must be passed in order for the software to be released. The Test Plan also supports the following objectives:

- Defining the activities required to prepare for and conduct Unit, Integration, System, and Operational Acceptance (namely, User Acceptance) Testing;
- Communicating to all responsible parties the test strategies;
- Defining deliverables and responsible parties;
- Communicating to all responsible parties any dependencies or risks.

1.3 SCOPE

The tests should validate the use cases, the requirements (both functional and non-functional), system architecture and object design. The requirements as well as the system architecture can be found in the documents referenced at the end of this document.

1.4 RISKS AND CONSTRAINTS

Among the factors that could hinder the steady development of the software are included time constraints, miscommunication or misunderstanding between team members. Furthermore, the following risks have been identified as well as the correct action to mitigate their impact on the project. The severity of the risk is based on how the project would be affected if the risk was triggered. The trigger is what event would cause the risk to become a problem to be handled.

#	Risk	Impact	Trigger	Mitigation Plan
1	Scope Creep – as testers become more familiar with the tool, they will want more functionality	High	Delays in implementation date	Each iteration, functionality will be closely monitored. Priorities will be set and discussed by those responsible.
2	Changes to the functionality may negate the tests already written and test cases already written may be lost	High (affects delivery date)	Loss of all test cases	Export data prior to any upgrade, massage as necessary and re-import after upgrade.
3	Aggressive schedule	High (affects delivery date)	Slips in a scheduled phase (delaying an iteration)	No iteration will be postponed or delayed.

2 TEST STRATEGY

The project is using an agile approach, with weekly iterations. At the end of each week the requirements identified for that iteration will be delivered to the team and will be tested. The test strategy consists of a series of different tests that will fully exercise the system of the Timeline Manager application. The primary purpose of these tests is to uncover the limitations and measure the full capabilities of the software. A list of the various planned tests and a brief explanation follows below.

2.1 UNIT TESTING

Unit testing refers to verifying the functionality of a specific section of code, usually at function level. In our object-oriented environment, this will be done at class level, and at method level. In the next subsection, the list of functional requirements to be tested can be found.

2.1.1 Testing Functional Requirements

List of functional requirements to be tested:

a) Add new timeline with start- and end time

b) Add multiple timelines with separate start- and end time

c) Add any number of events to timeline

d) Add non-duration event to timeline

e) Add event with duration to timeline

f) Modify title, description, or time(s) for an event

g) Modify complete timeline

h) Delete event on a timeline

i) Delete complete timeline

j) Add image to event

k) Save and load timelines

l) Store data in a text file

m) Place events at the correct time in the timeline

n) Show title for all events

o) For events with duration, show the time span with a bar

p) Click on an event to open a popup window showing title, descriptive text, and time(s) for the event

q) Scroll horizontally and/or vertically if all timelines cannot fit in the display window

r) Scroll in and/or out to zoom in or out of a timeline's time span

2.1.2 Items to be Tested / Not Tested

Item to Test	Test Description	Test Date	Responsibility
<Insert method name> Will be updated with each method created;	*<Insert requirement which it refers to>*		

2.2 INTEGRATION TESTING

The following testing phase is called integration testing, in which the individual software modules are combined and tested as a group.

2.2.1 Items to be Tested / Not Tested

Item to Test	Test Description	Test Date	Responsibility
Event.java	*Check that events are created correctly and are communicating with timelines appropriately*		
Timeline.java	*Check that timelines are created correctly and are communicating with events appropriately*		
TimelineModel.java	*Check that a correct timeline model is created*		
FileManager.java	*Check that timeline files are are handled correctly (save, load)*		
TimelineConverter.java	*Check that timelines are correctly properly converted into text and*		

	vice-versa		
NavigationController.java	*Check that controller appropriately handles the actions for UI events related to the navigation area of the application*		
TimelineController.java	*Check that controller handles the actions for the UI events related to the timeline area of the application in the intended way*		
TimelineListingController.java	*Check that the controller manages the listing area of the application correctly*		

2.3 SYSTEM TESTING

The System tests will focus on the behavior of the Timeline Manager system. User scenarios will be executed against the system as well as error message testing. Overall, the system tests will test the integrated system and verify that it meets all the requirements defined in the requirements document. In the next subsection, a list on non-functional requirements to be tested can be found.

2.3.1 Testing of Non-Functional Requirements

List of non-functional requirements to be tested:

a) System interface is user-friendly
b) Workload and response time are reasonable (program does not freeze/ lag)
c) Time- and date-related data is accurate according to current time
d) Program is portable, meaning it can be run on any system as long as it fits the system requirements (namely it must contain the Java Runtime Environment)
e) Program is easy to operate, scoring high on the system usability scale
f) Program does not exceed over 500MB in size (excluding user input)
g) System does not interfere with/affect any other running applications.

2.4 OPERATIONAL ACCEPTANCE TESTING

Once the Timeline Manager application is ready for implementation, tests will be performed following the guidelines of User Acceptance Testing. The purpose of these tests is to confirm that the system is developed according to the specified user requirements and is ready for operational use.

3 ENVIRONMENTAL REQUIREMENTS

In order to facilitate the testing process, the following requirements must be fulfilled:

- Software: Java SE Development Kit, namely the JRE (Java Runtime Environment) to run the Timeline Manager application and to access its source code and perform manual testing, and the Junit testing framework for Java to run automated testing.

4 TEST SCHEDULE

Task Name	Start	Finish	Effort	Comments
Test Planning				
Testing of iteration 1 + results				
Testing of iteration 2 + results				
Testing of iteration 3 + results				
Testing of iteration 4 + results				
Testing of iteration 5 + results				
Final results				

5 DELIVERABLES

Deliverable	For	Date / Milestone
Test Plan	Project Manager; Testers;	19/04/2017
Develop Test Cases	Testers	
Develop Automated Test Suites	Testers	
Produce Traceability Matrix Document	Project Manager	
Produce Test Cases Document	Project Manager	
Execute manual and automated tests	Testers	
Test Results	Project Manager	

6 DEPENDENCIES

6.1 PERSONNEL DEPENDENCIES

Due to the lack of experienced testers in the test team, the testers available are required to be more careful and correctly revise their operations before developing, performing and validating tests.

6.2 SOFTWARE DEPENDENCIES

The source code must be unit tested and provided within the scheduled time outlined in the Project Schedule.

7 DOCUMENTATION

The following documentation will be available at the end of the test phase:

- Test Plan
- Test Cases
- Requirements Validation Matrix
- Final Test Summary Report

Appendix A: References

The following table summarizes the documents referenced in this document.

Document Name and Version	Description	Location
Analysis Document	List of requirements	Analysis_Document.PDF
Timeline Manager – High-Level Design ver_1	System Architecture	Timeline Manager - High-Level-Design_v1.pdf

Appendix B: Key Terms

The following table provides definitions for terms relevant to this document.

Term	**Definition**
Scope creep	Which is also called requirement creep, function creep, feature creep, or kitchen sink syndrome, in project management refers to changes, continuous or uncontrolled growth in a project's scope, at any point after the project begins.
Agile	Relating to or denoting a method of project management, used especially for software development, that is characterized by the division of tasks into short phases of work and frequent reassessment and adaptation of plans.