POLITECNICO
MILANO 1863

Department of Computer Science and Engineering

Integration Test Plan Document

TrackMe

- v1.0 -

Authors:

Avila, Diego 903988

Schiatti, Laura 904738

Virdi, Sukhpreet 904204

January 13th , 2019

Contents

1 Introduction **1**

1.1 Context 1

1.2 Purpose and Scope 1

1.3 Definitions, Acronyms, Abbreviations 2

1.3.1 Definitions 2

1.3.2 Acronyms 2

1.3.3 Abbreviations 3

1.4 Revision history 3

2 Integration Strategy **4**

2.1 Entry Criteria 4

2.2 Elements to be integrated 4

2.3 Integration testing strategy 5

2.4 Sequence of Component/Function Integration 5

2.4.1 Software Integration Sequence 5

2.4.2 Subsystem Integration Sequence 5

3 Individual Steps and Test Description **6**

4 Performance Analysis **7**

5 Tools and Testing equipment **8**

5.1 Tools 8

5.2 Test Equipment 8

6 Required Program Stubs and Test Data **9**

7 Effort spent **10**

8 References **11**

List of Figures

List of Tables

1.1 Revision history timeline 3

7.1 Time spent by all team members 10
7.2 Time spent by each team member 10

Chapter 1

Introduction

1.1 Context

TrackMe develops health-monitoring devices devoted to measure and record different parameters related to the health status of a person (i.e. body temperature, blood pressure, heart pulse rate and percentage of O_2 in the blood) and also their location. TrackMe health smartwatches are synchronized with an app that gives users access to their data and stats. Also, TrackMe is offering new services to their customers, so as to exploit the data collected from those devices.

1.2 Purpose and Scope

This document represents the Integration Testing Plan Document for TrackMe Service.

Integration testing is a key activity to guarantee that all the different subsystems composing Data4Help and AutomatedSOS interoperate consistently with the requirements they are supposed to fulfil and without exhibiting unexpected behaviours. The purpose of this document is to outline, in a clear and comprehensive way, the main aspects concerning the organization of the integration testing activity for all the components that make up the system.
More precisely, the document presents:

- A list of the subsystems and their subcomponents involved in the integration activity that will have to be tested
- The criteria that must be met by the project status before integration testing of the outlined elements may begin
- A description of the integration testing approach and the rationale behind it
- The sequence in which components and subsystems will be integrated
- A description of the planned testing activities for each integration step, including their input data and the expected output

- Some performance measures that should be performed on the components to check they are fulfilling the requirements
- A list of all the tools that will have to be employed during the testing activities, together with a description of the operational environment in which the tests will be executed

.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- **Health status**: Collection of the last measured overall physical health parameters of a user or a group of users.
- **Running circuit**: Path defined by the organizer for the run, using the set of nodes.
- **Anonymize**: The action of anonymize means that an individual's identity cannot be inferred using the available data.
- **Parameter out of its normal range**: Meaning that the parameter is under or above a defined threshold.

1.3.2 Acronyms

- DD: Design Document
- RASD: Requirement Analysis and Specification Document
- D4H: Data4Help
- ASOS: AutomatedSOS
- T4R: Track4Run
- GUI: Graphical User Interface
- MVC: Model View Controller is a design pattern used for GUIs
- JMS: Java Message Service
- DSL: Digital Subscriber Line

1.3.3 Abbreviations

- $[Rn]$: n-requirement.

1.4 Revision history

It is important to keep track of the revisions made to this document:

Version	Last modified date
1.0	13^{th} January, 2019

Table 1.1: Revision history timeline

Chapter 2

Integration Strategy

2.1 Entry Criteria

In order for the integration testing to be possible and to produce meaningful results, there are a number of conditions on the progress of the project that have to be met.
First of all, the **Requirements Analysis and Specification Document** and the **Design Document** must have been fully written. This is a required step in order to have a complete picture of the interactions between the different components of the system and of the functionalities they offer.
Secondly, the integration process should start only when the estimated percentage of completion of every component with respect to its functionalities is:

- **100%** for the **Data4HelpWebService** component
- At least **90%** for the **LoginService and RegisterService** subsystem
- At least **70%** for the **SearchManager and RequestService** subsystem
- At least **50%** for the **ASOSService** applications

It should be noted that these percentages refer to the status of the project at the beginning of the integration testing phase and they do not represent the minimum completion percentage necessary to consider a component for integration, which must be at least **90%**. The choice of having different completion percentages for the different components has been made to reflect their order of integration and to take into account the required time to fully perform integration testing.

2.2 Elements to be integrated

In the following paragraph we're going to provide a list of all the components that need to be integrated together.
As specified in TrackMe Design Document, the system is built upon the interactions of many high-level components, each one implementing a specific set of functionalities. For the sake of modularity, each subsystem is further obtained by the combination of several lower-level components. Because of this software architecture, the integration phase will

involve the integration of components at two different levels of abstraction.

cntd....

2.3 Integration testing strategy

2.4 Sequence of Component/Function Integration

2.4.1 Software Integration Sequence

2.4.2 Subsystem Integration Sequence

Chapter 3

Individual Steps and Test Description

Chapter 4

Performance Analysis

While a full fledged performance analysis of the entire TrackMe infrastructure will be executed only in the system integration phase, it is still useful to perform some preliminary measures on components whose performances can be tested in isolation

Chapter 5

Tools and Testing equipment

5.1 Tools

In order to test the various components of TrackMe more effectively, we are going to make usage of a number of automated testing tools.
For what concerns the business logic components running in the Java Enterprise Edition runtime environment, we are going to take advantage of two tools.
The fi
rst one is the **Arquillian integration testing framework**. This tool enables us to execute tests against a Java container in order to check that the interaction between a component and its surrounding execution environment is happening correctly (as far as the Java application server is involved). Specifically, we are going to use Arquillian to verify that the right components are injected when dependency injection is specified, that the connections with the database are properly managed and similar container-level tests.
The second tool is the **JUnit framework**. Though this tool is primarily devoted to unit testing activities, it's still a valid instrument to verify that the interactions between components are producing the expected results. In particular, we are going to use it in order to verify that the correct objects are returned after a method invocation, that appropriate exceptions are raised when invalid parameters are passed to a method and other issues that may arise when components interact with each other.
Finally, it should be noted that despite the usage of automated testing tools, some of the planned testing activities will also require a significant amount of manual operations, especially to devise the appropriate set of testing data.

5.2 Test Equipment

All the integration testing activities have to be performed within a specific testing environment.

Chapter 6

Required Program Stubs and Test Data

6.1 Program Stubs and Drivers

As we have mentioned in the Integration Testing Strategy section of this document, we are going to adopt a bottom-up approach to component integration and testing.
Because of this choice, we are going to need a number of drivers to actually perform the necessary method invocations on the components to be tested; this will be mainly accomplished in conjunction with the JUnit framework.

6.2 Test Data

Chapter 7

Effort spent

Team Work	
Task	**Hours**
Planning Architecture	8
Architectural design overview	4
Choosing Patterns	3
Checking document	2
Total	17

Table 7.1: Time spent by all team members

Individual Work					
Diego Avila		**Laura Schiatti**		**Sukhpreet Kaur**	
Task	**Hours**	**Task**	**Hours**	**Task**	**Hours**
Component view	10	Scope	3	Purpose	2
Component interfaces	13	Database view	6	UI design	15
Deployment view	5	Runtime view	8	I and T plan	6
Runtime view	4	R traceability	3	Design decisions	2
Corrections	3	Final user interfaces	7		
		Architectural styles	4		
Total	35	**Total**	31	**Total**	25

Table 7.2: Time spent by each team member

Chapter 8

References

- Requirement Analysis and Specification Document: AA 2017-2018.pdf". Version 1.0 - 26.10.2017
- Henriksen, A., Haugen Mikalsen, M., Woldaregay, A. Z., Muzny, M., Hartvigsen, G., Hopstock, L. A., Grimsgaard, S. (2018)
  Using Fitness Trackers and Smartwatches to Measure Physical Activity in Research: Analysis of Consumer Wrist-Worn Wearables. Journal of medical Internet research, 20(3), e110. doi:10.2196/jmir.9157.
  Retrieved from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5887043/
- IEEE. (1993). IEEE Recommended Practice for Software Requirements Specifications (IEEE 830-1993).
  Retrieved from https://standards.ieee.org/standard/830-1993.html
- Sloane, A. M. (2009). Software Abstractions: Logic, Language, and Analysis by Jackson Daniel, The MIT Press, 2006, 366pp, ISBN 978-0262101141.
- Spark. A Micro Framework For Creating Web Applications - http://sparkjava.com/
- Morphia - http://morphiaorg.github.io/morphia/
- Lettuce - https://lettuce.io/
- Angular - https://angular.io/
- Google Maps Platform - https://cloud.google.com/maps-platform/