Test Plan for Bike Garage Pro (Group 33, 2015)

Current version: 0.9.1

Alexander Skafte
tfy13ask@student.lu.se
Dennis Jin
desuvader@gmail.com
Petter Berntsson
dat14pbe@student.lu.se
Emelie Löthman
pol14elo@student.lu.se
Adam Mzrozek
dat14amr@student.lu.se

Contents

1 References

- *Examples and Exercises in the Software Engineering Process*. ETSA01 VT 2015. Deparment of Computer Science, Lund University. March 10, 2015.
- *Software Requirements Specification for Bicycle Garage Pro*. ETSA01, Group 33, 2015.

2 Introduction

2.1 Tested system

The system described in this document is the software for a public bicycle garage. This software is responsible for managing the authentication of users and the management of their information and their bicycles.

This document provides a specification for testing the bicycle garage software. The test process consists of the following phases:

- Unit testing
- System testing
- Acceptance testing

3 Test process

3.1 Process overview

The tests of the software are carried out on several levels of abstraction:

- At the lowest level, *unit testing* is performed which aim to test each line of code inside the software.
- These units are grouped into modules, which would normally be tested through the API or *Application Programming Interface*, a process known as *integration testing*. For this software however, integration testing will not be performed due to the smallness of the software.
- On an even higher level of abstraction, so called *system testing* is performed which ensures that the requirements provided in the *Software Requirements Specification* are fulfilled.
- Lastly, *acceptance testing* is performed. However, acceptance testing is performed by the customer and thus not within the scope of this document.

3.2 Unit testing

Unit testing aims to test each line of software code. Every non-trivial function is tested in software through the use of a test suite library.

Performed by: Developers

Type of test: Structural

Criteria: Every important line of code is tested

Stop rule: No errors found

3.3 System testing

During system testing, all requirements specified inside the *Software Requirements Specification* are tested.

Performed by: Developers

Type of test: Functional

Criteria: All requirements inside the SRS are fulfilled

Stop rule: No critical errors found

3.4 Acceptance testing

Acceptance testing is performed by the client and not by the developers, and is thus not discussed in this document.

4 Tested items

TODO

... perhaps this section should be removed? In the example found in the green booklet for the course, this section is nothing but a repetition of what has already been said.

5 Test recording procedure

5.1 Unit testing

Every non-trivial piece of software code is tested individially by the developers. These tests are automated and performed incrementally as the code base grows. The results of these tests do not have to be recorded; the rationale for this being that the tests provide enough assurance on their own and that the size of the test log would soon grow out of hand.

5.2 System testing

For each test session carried out, a new requirements coverage matrix shall be produced according to the one shown in *Appendix B: Requirements Coverage Matrix*. In a document containing this matrix, the result of each test case shall be displayed. Any faults found shall be reported to the developers, either directly or through the use of a shared file where error reports are logged.

5.3 Acceptance testing

Acceptance testing is performed by the client and not by the developers, and is thus not discussed in this document.

6 Test cases for system testing

6.1 Test cases

Since unit testing is performed in the software environment and acceptance testing is performed by the customer, only system testing will be fully described in this document. For this purpose, please refer to *Appendix A: Test cases*.

6.2 Requirements coverage and traceability

All functional requirements provided in the *Software Requirements Specification* should be tested by at least on test case to ensure that all functionality is tested. For this purpose, a *requirements coverage matrix* is used. For this matrix, please refer to *Appendix B: Requirements Coverage Matrix*.

A Test cases

Test case 1:	**Registration of a new user**
Primary actor:	Operator
Preconditions:	User is unregistered
Postconditions:	User is registered

Main success scenario:

1. The operator provides the required user information to the control interface.
2. A new PIN code is generated for the user.
3. The user is added to the system.

Test case 2:	**Registration of an already registered user**
Primary actor:	Operator
Preconditions:	User is registered
Postconditions:	User is registered

Main success scenario:

1. The operator provides the required user information to the control interface.
2. The system responds with an error message, e.g. ”The user is already registered.”

Test case 3:	**Unregistration of a registered user**
Primary actor:	Operator
Preconditions:	User is registered

Postconditions:	User is unregistered

Main success scenario:

1. The operator provides the required user information to the control interface.
2. All bicycles associated with the user are removed from the system.
3. The user is removed from the system.

Test case 4:	**Association of a new bicycle with a user**
Primary actor:	Operator
Preconditions:	User is registered; garage is not full
Postconditions:	Bicycle is associated with user

Main success scenario:

1. The operator provides the required user information to the control interface.
2. A unique 5-digit identification number is generated and associated with the bicycle.
3. The bicycle is added to the set of bicycles owned by the user.
4. A barcode associated with the 5-digit ID is printed and given to the user.

Test case 5:	**Disassociation of a user's bicycle**
Primary actor:	Operator
Preconditions:	The user is registered. The bicycle is associated with the user.
Postconditions:	Bicycle is not associated with user nor is it present in the system.

Main success scenario:

1. The operator provides the required user information to the control interface.
2. The bicycle is disassociated with the user.
3. The unique 5-digit identification number associated with the bicycle is returned to the pool of available ID's. As a consequence, the barcode is rendered invalid.

Test case 6:	**A valid PIN code is entered**
Primary actor:	User
Preconditions:	The PIN code entered is associated with a registered user, who has at least one bicycle stored in the garage.
Postconditions:	-

Main success scenario:

1. User enters their PIN code at the PIN code terminal.
2. The green LED lamp is lit for 4 seconds. Simultaneously, the entrance door opens and stays open for 10 seconds.

Test case 7:	**An invalid PIN code is entered**
Primary actor:	User
Preconditions:	The PIN code entered is not associated with any registered user.
Postconditions:	-

Main success scenario:

1. User enters the PIN code at the PIN code terminal.
2. The red LED lamp is lit for 4 seconds.

Test case 8:	**Recovery of a lost/forgotten PIN code**
Primary actor:	Operator
Preconditions:	The user is registered.
Postconditions:	-

Main success scenario:

1. The operator provides the required user information to the control interface.
2. The operator requests the PIN code from the system, and gives it to the user.

Test case 9:	**Modification of a user's PIN code**
Primary actor:	Operator
Preconditions:	The user is registered and there are available PIN codes left.
Postconditions:	The user's PIN code is updated.

Main success scenario:

1. The operator provides the required user information to the control interface.
2. The operator asks the system to generate a new PIN code.
3. A PIN code is successfully generated and associated with the user.

Test case 10:	**An invalid PIN code is entered**
Primary actor:	User
Preconditions:	The PIN code entered is not associated with any registered user.

Postconditions: -

Main success scenario:

1. User enters the PIN code at the PIN code terminal.
2. The red LED lamp is lit for 4 seconds.

Test case 11:	**An invalid barcode is scanned**
Primary actor:	User or other person
Preconditions:	The barcode is not a part of the system.
Postconditions:	-

Main success scenario:

1. The invalid barcode is scanned.
2. The red LED lamp is lit for 4 seconds.

Test case 12:	**Creation of a new barcode for an existing bicycle**
Primary actor:	Operator
Preconditions:	The user is registered and the subject bicycle is associated with them.
Postconditions:	A new barcode is associated with the subject bicycle's 5-digit identification number.

Main success scenario:

1. The operator provides the required user information, including the bicycle's 5-digit identification number, to the control interface.
2. The old barcode associated with the ID is rendered invalid and a new one is generated, associated with the ID and then printed.

B Requirements Coverage Matrix

In the requirements coverage matrices shown below, requirements are shown on the vertical axes and test cases on the horizontal axes.

	T1	T2	T3	...	Tn
R 3.3.1.1					
R 3.3.1.2					
R 3.3.1.3					
...					
R 3.3.1.n					

Figure 1: A minimal example of an empty requirements coverage matrix.

	T1	T2	T3	...	Tn
R 3.3.1.1	X				
R 3.3.1.2		X			
R 3.3.1.3					
...					
R 3.3.1.n					X

Figure 2: A minimal example of partly filled requirements coverage matrix.

	T1	T2	T3	...	Tn
R 3.3.1.1	X				
R 3.3.1.2		X			
R 3.3.1.3			X		
...				X	
R 3.3.1.n					X

Figure 3: A minimal example of a filled requirements coverage matrix.