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

Alexander Skafte
tfy13ask@student.lu.se

Contents

1 References

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
- Integration testing
- System testing
- Acceptance testing

3 Test process

3.1 Process overview

3.2 Unit testing

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 line of code is tested

Stop rule: No errors found

3.3 Integration testing

Integration testing is performed in a similar way to unit testing, but larger and more inclusive modules are tested. Each module is tested in software through the use of a test suite library.

Performed by: Developers

Type of test: Structural

Criteria: Every API method is tested completely

Stop rule: No errors found

3.4 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

3.5 Acceptance testing

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

4 Tested items

5 Test recording procedure

5.1 Unit testing

5.2 Integration testing

5.3 System testing

5.4 Acceptance testing

6 Test cases for system testing

6.1 Test cases

6.2 Requirements coverage and traceability

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. 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. 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. 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. 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:	User is registered; bicycle is associated with user.
Postconditions:	Bicycle is not associated with user nor is it present in the system.

Main success scenario:

1. 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.