# Dillinger

# Official Requirements Document

Authors:Albera Roberto, Alloa Giorgio, Cimino Giuseppe, Maina Alberto, Spasaro Matteo

Date: 12/06/2019

Version: 1

Change history

| Version | Changes |

| ----------------- |:-----------|

| 1 | First file writing|

| 2 | Abstract modified following changes to the program

# Contents

- [Abstract](#abstract)

- [Stakeholders](#stakeholders)

- [Context Diagram and interfaces](#context-diagram-and-interfaces)

+ [Context Diagram](#context-diagram)

+ [Interfaces](#interfaces)

- [Stories and personas](#stories-and-personas)

- [Functional and non functional requirements](#functional-and-non-functional-requirements)

+ [Functional Requirements](#functional-requirements)

+ [Non functional requirements](#non-functional-requirements)

- [Use case diagram and use cases](#use-case-diagram-and-use-cases)

+ [Use case diagram](#use-case-diagram)

+ [Use cases](#use-cases)

+ [Relevant scenarios](#relevant-scenarios)

- [Glossary](#glossary)

- [System design](#system-design)

- [Deployment diagram](#deployment-diagram)

# Abstract

The software allows to simulate a classic garage operation, like the wheels sostituations, this virtual experience keeps the environment and the work realistic under every points of view.

At the start of the simulation the operator will find a car supported by four box without wheels and, obliuvisly, four wheels on four mini-box (one for each);the simulation will take place in this way: the operator will have to hit one of the wheels in some way, at which point the wheel will automatically position itself on the area of the machine that competes with it, moreover, to fix the wheel to the car, it will be necessary to hit the wheel once its run to the vehicle is over.

At the start of the project the first idea was a little more complex: there were a wrench, some bolts and adjustable cricks.

The operator should have to take manually the wheels and attack them at the car using the

wrench and one bolt for each wheel. Unfortunally the time available and the conditions have made it impossible.

# Stakeholders

| Stakeholder name | Description |

| ----------------- |:-----------:|

| Administrator |Uses the application to train the employees in a small office (5-10 people)|

| Colleagues |Uses the software directly (under directive of the Administartor) to improves their mechanical skills.|

# Context Diagram and interfaces

## Interfaces

| Actor | Logical Interface | Physical Interface |

| ------------- |:-------------:| -----:|

|Administrator|GUI |Screen, keyboard|

| Employee | GUI | Screen, keyboard|

# Stories and personas

John works in the garage and he’s a little unprepared about some tasks.

So his boss decided to buy our virtual simulator to learn how to do, for exemple, a wheels’s sostituation without waste materials. On the money side, John’s boss will not ever spend again for materials to train his

employyes.

Morover every colleague has a virtual account to have a

private update course about the new technologys.

All colleagues trust each other, knowing the mechanical skills of each one they know who they’re working with.

# Functional and non functional requirements

## Functional Requirements

| ID | Description |

| ------------- |:-------------:|

| FR1 | Realistic simulation |

| FR2 | Virtual reality |

| FR3 | Record that a colleague has recharged x euros on her account |

| FR4 | Private accounts |

| FR5 | Wireless comunication |

## Non Functional Requirements

| ID | Type (efficiency, reliability, .. see iso 9126) | Description | Refers to |

| ------------- |:-------------:| :-----:| -----:|

| NFR1 | Usability | Application should be used with no training by any colleague in the office | All FR |

| NFR2 | Performance | All functions should complete in < 0.5 sec | All FR |

| NFR3 | Portability | The application runs on MS Windows (7 and more recent) | All FR |

| NFR4 | Portability | The application (functions and data) should be portable from a PC to another PC in less than 5 minutes | All FR |

| NFR5 | Localisation | Decimal numbers use . (dot) as decimal separator ||

# Use case diagram and use cases

## Use case diagram

```plantuml

1) The user can open the gate at the start of the simulation

2) The user can shoot buttons and wheels to perform certain functions, such as mounting wheels

```

## Use Cases

### Use case 1, UC1 - FR1 Record usage of capsules by colleague

| Actors Involved | Administrator - Employee |

| ------------- |:-------------:|

| Precondition | Simulator exists, operator exists |

| Post condition | The operator has worn the equipment - The program has been launched |

| | The operator shoots the button to open the gate |

### Use case 2, UC2 - FR2 Record usage of capsules by visitor

| Actors Involved | Administrator - Employee |

| ------------- |:-------------:|

| Precondition | Simulator exists, operator exists |

| Post condition | The operator has worn the equipment - The program has been launched - The gate is open |

| | The operator shoots a wheel |

| | the wheel makes a run towards the car, stopping where it must be mounted|

### Use case 3, UC3 - FR3 Record recharge of account of colleague

| Actors Involved | Administrator - Employee |

| ------------- |:-------------:|

| Precondition | The wheel has finished its run |

| Postcondition | A second wheel must be able to be struck |

| |By shooting at the wheel at the end of the stroke the latter sticks to the machine|

# Relevant scenarios

## Scenario

| Scenario ID: SC1 | Corresponds to UC1 |

| ------------- |:-------------|

| Description | The user wants to simulate a change of wheels|

| Precondition | The user wore the necessary equipment and the simulator was started|

| Postcondition | Account of the user updated |

| Step# | Step description |

| 1 | User hit the button to open the gate |

| 2 | The user shoots at the wheel to make him ride |

| 3 | At the end of the run, the user shoots at the wheel to have it fixed to the car|

| 4 | Repeat for all 4 wheels |

# Glossary

```plantuml

class LaTazza

class Colleague {

+ name

+ surname

}

class PersonalAccount {

+ balance

}

class CapsuleType {

+ name

+ price

+ quantity

}

class LaTazzaAccount {

+ balance

}

class BoxPurchase {

+ quantity

}

class Transaction {

+ date

+ amount

}

class Recharge

class Consumption

LaTazza -- "\*" Colleague

LaTazza -- "\*" CapsuleType

LaTazza -- LaTazzaAccount

LaTazzaAccount -- "\*" BoxPurchase

LaTazzaAccount -- "\*" Consumption

CapsuleType -- "\*" Consumption

CapsuleType -- "\*" BoxPurchase

Colleague -- PersonalAccount

PersonalAccount -- "\*" Transaction

Transaction <|-- Recharge

Transaction <|-- Consumption

Transaction <|-- BoxPurchase

```

# System Design

Not really meaningful in this case. Only one personal computer is needed.

```plantuml

class "Personal Computer"

```

# Deployment Diagram

As a stand-alone application only one node is needed.

```plantuml

artifact "Simulator" as a

node "Personal Computer" as n

a -- n

```