Final Report MaraffaOnline DS

Author 1
matteo.santoro@studio.unibo.it

Author 2
sofia.tosi4@studio.unibo.it

January 2019

Up to ∼2000 characters briefly describing the project.

1 Goal/Requirements

Detailed description of the project goals, requirements, and expected outcomes. Use case Diagrams, examples, or Q/A simulations are welcome.

1.1 Scenarios

Informal description of the ways users are expected to interact with your project. It should describe *how* and *why* a user should use / interact with the system.

1.2 Self-assessment policy

- How should the *quality* of the *produced software* be assessed?
- How should the *effectiveness* of the project outcomes be assessed?

2 Requirements Analysis

Is there any implicit requirement hidden within this project's requirements? Is there any implicit hypothesis hidden within this project's requirements? Are there any non-functional requirements implied by this project's requirements?

What model / paradigm / techonology is the best suited to face this project's requirements? What's the abstraction gap among the available models / paradigms / techonologies and the problem to be solved?

3 Design

This is where the logical / abstract contribution of the project is presented.

Notice that, when describing a software project, three dimensions need to be taken into account: structure, behaviour, and interaction.

Always remember to report **why** a particular design has been chosen. Reporting wrong design choices which has been evalued during the design phase is welcome too.

3.1 Structure

Which entities need to by modelled to solve the problem? (UML Class diagram)

How should entities be modularised? (UML Component / Package / Deployment Diagrams)

3.2 Behaviour

How should each entity behave? (UML State diagram or Activity Diagram)

3.3 Interaction

How should entities interact with each other? (UML Sequence Diagram)

4 Implementation Details

Just report interesting / non-trivial / non-obvious implementation details.

This section is expected to be short in case some documentation (e.g. Javadoc or Swagger Spec) has been produced for the software artefacts. This this case, the produced documentation should be referenced here.

5 Self-assessment / Validation

Choose a criterion for the evaluation of the produced software and **its compliance to the requirements above**.

Pseudo-formal or formal criteria are preferred.

In case of a test-driven development, describe tests here and possibly report the amount of passing tests, the total amount of tests and, possibly, the test coverage.

6 Deployment Instructions

Explain here how to install and launch the produced software artefacts. Assume the softaware must be installed on a totally virgin environment. So, report **any** configuration step.

Gradle and Docker may be useful here to ensure the deployment and launch processes to be easy.

7 Usage Examples

Show how to use the produced software artefacts.

Ideally, there should be at least one example for each scenario proposed above.

8 Conclusions

Recap what you did

8.1 Future Works

Recap what you did *not*

8.2 What did we learned

Racap what did you learned

Stylistic Notes

Use a uniform style, especially when writing formal stuff: X, X, $\mathbf{X}$, $\mathcal{X}$, $\mathtt{X}$ are all different symbols possibly referring to different entities.

This is a very short paragraph.

This is a longer paragraph (notice the blank line in the code). It composed by several sentences. You're invited to use comments within `.tex` source files to separate sentences composing the same paragraph.

Paragraph should be logically atomic: a subordinate sentence from one paragraph should always refer to another sentence from within the same paragraph.

The first line of a paragraph is usually indented. This is intended: it is the way LaTeX lets the reader know a new paragraph is beginning.

Use the `listing` package for inserting scripts into the LaTeX source.