Organisation of the Software Development Project 2018–2019
(last update: 16 September 2018 – version 1.0)

Table of contents

References (Python and Flask)

- To *learn Python*, we advise the students to follow, for example, the following online course in French on OpenClassrooms:
  - https://openclassrooms.com/courses/apprenez-a-programmer-en-python

  Or, alternatively, the following book in English, or corresponding online course:
  - « How To Think Like a Computer Scientist – Learning with Python 3 », Peter Wentworth, Jeffrey Elkner, Allen B. Downey and Chris Meyers, August 2012.
  - « How to Think Like a Computer Scientist: Interactive Edition », Brad Miller and David Ranum, an interactive online version of the book just above.
- To learn the *Flask* web application microframework, there is this interesting tutorial:
  - https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world

---

Tools

Throughout this year's software development project, we will *require* the use of the following tools and environments. *We advise you to learn about them as soon as possible, starting from the first week of the semester.* In this project you we will use many tools and there will be many deliverables, so better start early. This is not an ordinary academic course; you are building an actual product for a real client, which requires you to act professional and use professional tools.

- The programming language and web application framework required by the client are Python (3) and Flask.
- Use Pycharm, JetBrains' Python development environment, as IDE. JetBrains provides educational licenses for free to students and teachers from recognised universities such as Université catholique de Louvain. UCL staff and students can get individual access to the software (to install it on their PCs/laptops), using the free individual student/teacher license packs. Anyone who applies with an @uclouvain.be email address will instantly get a free personal subscription to all of JetBrains' desktop tools, including PyCharm.
- Use Trello as project management and organisation tool.
- Document your code in detail with docstrings.
- Use standard unit testing and doctests (test code snippets included in docstrings) for

testing your code quality. Don't wait until the end to start testing. Remember the credo "*Test early, test often"*: testing should be done frequently and from the very beginning.

- Respect Python coding standards and best practices like those listed in the PEP8 style guide for Python code.
- Use a code metrics tool like PyLint to verify that your code respects such conventions.
- Along with Python style guides, use Python coding idioms and avoid common mistakes. Code like a Pythonista.
- Use GitHub as a versioning tool, from the very start, not only when deploying your prototypes. The use of Github will be essential to integrate and merge your code with the modules developed by other students. GitHub also has the advantage that it allows to plugin many other cool tools and services.
- In particular, we *strongly encourage* you to integrate the following services into your GitHub repository:
  - Travis CI, a distributed and cross-platform continuous integration service used to build and test projects hosted at Github. It automatically detects when a commit has been made and pushed to a GitHub repository that is using Travis CI, and each time this happens, it will build the project and run the tests.
  - Coveralls, a test coverage service that shows you which parts of your code aren't covered by your test suites yet.
  - Landscape, based on PyLint, can give you early warnings when your Python codebase is of bad quality, so that you can improve it as soon as warnings occur.

  All of these services are very easy to set up, not intrusive, free to use and can be hooked directly into your GitHub repository. As such, when you have these kinds of continuous integration tools enabled you always know whether your code works or not, you have an easy overview of whether or not your code is tested enough, and you know at all time whether the quality of your software improves or decreases.
- Many online tools exist to create **mock-ups** or **prototypes** of web applications. For most of these free versions with limited functionality exist. We advise you to use one of these tools, or a generic drawing tool (of which many online versions can also be found easily) to create a mock-up of the application you need to develop. Here is an incomplete list of such tools, but feel free to use whatever tool pleases you most:[1]
  - https://www.mockplus.com/
  - https://marvelapp.com/
  - https://www.axure.com/
  - https://www.lucidchart.com/pages/website-mockups
  - …
- Throughout this course we will also use the following modelling notations.
  - User stories
  - Usage scenarios
  - Object-role modeling
  - Gantt charts or PERT diagrams
  - Activity diagrams
  - Class diagrams
  - Sequence diagrams
  - or any other diagram you deem relevant to communicate certain aspects of the requirements analysis or design of your application to your team members, the client or teaching staff.

[1] If you would find a particularly cool tool that is not listed here, don't hesitate to let us know so that we can add it to our list.

- Several tools can be found online to create such diagrams or you could use a generic drawing tool which often has templates for such diagrams as well. Here is an incomplete list of such tools, but feel free to use whatever tool pleases you most:[2]
  - Dia
  - draw.io
  - Google Drawings
  - OmniGraffle
  - Lucidchart
  - Visio
- Finally, we will use Slack as a communication tool and Moodle as course management tool. Get registered to these as soon as possible. They will be the main means of communication throughout this course.

Case description

The case description and list of modules to be implemented will be provided in a separate document written by the client.

Project organisation

Although we won't strictly impose the software development method to be followed by your team, we suggest using an agile-inspired software development method (such as many of you already experienced in the context of your Android project in third year bachelor).

1. All students will start from a same initial code base, structure and database provided by the client.
2. Each team of students will work separately and locally on their own clone, on their own modules to be added to the client's application.
3. All teams guided by a same assistant (4 teams per assistant) will need to merge their working prototype back into a common shared repository, merge it with the prototypes of the other teams, and resolve all conflicts.
4. After a successful merge they branch again and continue working on their second and/or third module, repeating the steps 2 and 3 above.
5. The client will be in charge of the final merge of all shared repositories of the teams of the different assistants.

Contact persons

Contact person	**Nom**	**Rôle**
KM	Kim Mens kim.mens@uclouvain.be	Professeur
LT	Ludovic Taffin ludovic.taffin@uclouvain.be	Client (INGI Sysadmin 1)

[2] The same remark holds here: let us know if you would find another interesting tool to add to this list.

ND	Nicolas Detienne nicolas.detienne@uclouvain.be	Client (INGI Sysadmin 2)
AG	Anthony Gego anthony.gego@uclouvain.be	Client (INGI Sysadmin 3)
XG	Xavier Gillard xavier.gillard@uclouvain.be	Assistant 1
BD	Benoît Duhoux benoit.duhoux@uclouvain.be	Assistant 2
QH	Ha Quang Minh quang.ha@uclouvain.be	Assistant 3
MS	Michael Saint-Guillain michael.saint@uclouvain.be	Assistant 4

Schedule

Dates	Activity	Responsible
Week 1 (17–21.09.2018) – Course and Case Introduction		
MON 17.09	Distribution of case specification.	client
	Distribution of planning document.	professor
THU 20.09 14:00-16:00 A.03	Course organisation and introduction.	professor
	Case introduction.	client
	Explanation of case technology and architecture.	
Getting acquainted	Read case study specification.	student
	Learn technology: Python, Flask, GitHub.	
Course logistics	Get registered: confirm course attendance, register to Moodle, create teams, join Slack.	
FRI 22.09	**[F1] Teams created**	**feedback**
Week 2 (24–28.09.2018) – Getting Started		
Course logistics	*(Team members may still be added during this week.)*	professor
	Assign internal team leader (a student).	team
	Contact assigned assistant to fix weekly meetings.	
	Discuss and choose modules to implement.	
Getting acquainted	Continue learning technology and tools: Python, Flask, Github, PyCharm, Slack, …	student
Getting started	Download and install initial code, development tools, versioning system.	team
	Make an Object-Role Model of the existing database.	
	Ask client if things are unclear in the case description.	
~~THU 27.09~~	(no course: French Community of Belgium Day)	
FRI 28.09 13:00-14:00 SUD11	Technical Q&A session about code, installation and technology used (1h)	client
Weekly meeting with assistant	Getting acquainted with your assistant. Discussion about modules to be chosen. Discuss Object Role Model of the database.	team, assistant
FRI 28.09 08:00	**[d2a]** Modules to be developed chosen.	**deliverables**
	[d2b] Detailed Object Role Model of structure of the initial database.	

FRI 28.09	**[F2]** Updated teams with assigned team leaders	**feedback**
Week 3 (01–05.10.2018) – Requirements Analysis		
MON 01.10	**[F3]** Notification of acceptance of chosen modules.	**feedback**
THU 04.10 14:00-16:00 A03	Lecture on requirements analysis. Lecture on Gantt charts and PERT diagrams.	professor
Requirements analysis	Creation of user stories, usage scenarios, estimates and an activity diagram for all modules to be developed.	team
Weekly meeting with assistant	*Requirements workshop.* During 1 hour, each team creates a usage scenario and a set of user stories for one of its modules. In the second hour, the teams present and discuss these with the other teams.	team, assistant
FRI 05.10 08:00	**[d3]** First draft of requirements document: usage scenarios, user stories, estimation of user stories and activity diagram to describe application flow.	**intermediate deadline**
Week 4 (08–12.10.2018) – Requirements Analysis (continued)		
SAT 06.10 SUN 07.10	Client reads all submitted requirements documents.	client
MON 08.10 TUE 09.10 30 min. / group	Question-and-answer meeting with the client to verify whether the requirements have been understood well. Based on the outcome of this meeting the client and assistant may request modifications to the requirements document.	team, client, assistant, ~~professor~~
THU 11.10 14:00-16:00 A03	Introduction to the use of Git	Xavier Gillard
Requirements analysis (continued)	Refine usage scenarios, user stories, estimates and activity diagram based on feedback received. Create mock-ups or wireframes of the application. Make detailed project planning using Gantt or PERT.	team
Weekly meeting with assistant	*Assistant discusses and provides feedback on requirements documents produced by the team.*	team, assistant
FRI 12.10 08:00	**[D4]** Final requirements document: usage scenarios, uses cases, activity diagram, mock-up, planning.	**deliverable**
Week 5 (15–19.10.2018) – First Sprint		
SAT 13.10 SUN 14.10	Assistants read teams' final requirements document in detail. Client verifies if requested changes have been made.	assistant client
Weekly meeting with assistant	**[F5]** Each team meets individually with their assistant to discuss and receive feedback on the produced requirements document.	**feedback**
Update requirements	Based on the modifications requested by the client, the students update their requirements documents.	team
Design of 1st prototype	Design of 1st module to be implemented in terms of a class diagram and some sequence diagrams for key functionalities.	
Coding of 1st prototype	At the same time the team can start implementing a prototype of a 1st module.	

THU 18.10 14:00-16:00 A03	Course on architecture, design and team work.	professor
FRI 19.10 08:00	**[d5a]** Updated requirements document (only for teams for which the client requested modifications)	**deliverable**
	[d5b] First draft of design diagrams ready: class and sequence diagrams.	intermediate deadline
	[d5c] First rough implementation of initial prototype, with corresponding tests.	intermediate deadline
Week 6 (22–26.10.2018) – First Sprint (continued) – "SMART" week		
MON 22.10	Assistants read updated requirements documents and drafts of design diagrams.	assistant
Weekly meeting with assistant	Discuss and provide feedback on design diagrams.	team, assistant
	Discuss and show what has already been coded and what remains to be coded this week.	
Update design	Based on the modifications requested by the assistant, the students update their class and sequence diagrams.	team
Coding and testing	Continue coding of 1st prototype. A final version of the 1st prototype and its corresponding tests should be ready by the end of this week.	team
~~THU 25.10~~	(no course)	
FRI 26.10 08:00	**[D6a]** Completed version of design document with class and sequence diagrams.	**deliverable**
	[D6b] 1st prototype ready and working.	**deliverable**
	[D6c] Test suites + document explaining the tests.	**deliverable**
	[D6d] Document detailing what existing code or parts of the database the 1st prototype has used or touched.	**deliverable**
Week 7 (29.10–02.11.2018) – Integrate and merge		
MON 29.10	Client and assistants look at deliverables **D6*** in detail.	assistant, client
Integration	Teams (of a same assistant) merge their 1st prototypes together into the main application and resolve potential merge conflicts.	team
Start next module	Teams who successfully finished the integration of their 1st prototype can start designing and implementing their 2nd module, and thus take a head start on the next sprint.	
TUE 30.10 30 min. / group 3 parallel series	**[F7] Feedback on 1st prototype from client.** Teams demo their 1st prototype and corresponding tests to the client to receive feedback whether their prototype matches the expectations. Based on the outcome of this meeting the client will either validate or request modifications to the prototype.	**feedback** from client
~~THU 01.11~~	(no course: All Saints Day)	
FRI 02.11 08:00	**[D7]** 1st prototype integrated and merged with main application and prototypes of other teams.	**deliverable**
Week 8 (05–09.11.2018) – Second Sprint		
Design of 2nd prototype	Now that the 1st prototype is finished and integrated, the teams can start developing a 2nd module that was assigned to them. As a first step it would be good to	team

	work out the design of this 2nd prototype in terms of class and sequence diagrams.	
Coding of 2nd prototype	Implementation of 2nd prototype in accordance to the design.	
~~THU 08.11~~	(no course planned)	
Weekly meeting with assistant	Discuss and show what has been designed and coded this week.	team, assistant
FRI 09.11 08:00	**[d8a]** First draft of design diagrams of 2nd module ready: class and sequence diagrams.	intermediate deadline
	[d8b] First rough implementation of initial prototype of 2nd module, with corresponding tests.	intermediate deadline
Week 9 (12–16.11.2018) – Second Sprint (continued)		
MON 12.11	Assistants read updated requirements documents and drafts of design diagrams.	assistant
Weekly meeting with assistant	Discuss and provide feedback on design diagrams.	team, assistant
	Discuss and show what has already been coded and what remains to be coded this week.	
Update design	Based on the modifications requested by the assistant, the students update their class and sequence diagrams.	team
Coding and testing	Continue coding and testing of 2nd prototype. A final working and tested version of the 2nd prototype is due by the end of this week.	team
~~THU 15.11~~	(no course)	
FRI 16.11 08:00	**[D9a]** 2nd prototype ready and working.	**deliverable**
	[D9b] Test suites + document explaining the tests.	**deliverable**
	[D9c] Document detailing precisely what existing code or parts of the database the 2nd prototype touched.	**deliverable**
Week 10 (19–23.11.2018) – Integrate and merge		
MON 19.11	Client and assistants look at deliverables **D9*** in detail.	assistant, client
Integration	Teams (of a same assistant) try to merge their 2nd prototypes together into the main application and resolve potential merge conflicts.	team
Start next module	Teams who successfully finished the integration of their 2nd prototype can start designing and implementing their 3rd module to take a head start on the next sprint.	
TUE 20.11 30 min. / group 3 parallel	**[F10] Feedback on 2nd prototype from client.** Teams demo their 2nd prototype and corresponding tests to the client to receive feedback whether their prototype matches the expectations. Based on the outcome of this meeting the client will either validate or request modifications to the prototype.	**feedback** from client
FRI 23.11 08:00	**[D10]** 2nd prototype integrated and merged with main application and prototypes of other teams.	**deliverable**
Week 11 (26.11–30.11.2018) – Third Sprint		
Design of 3rd prototype	Design of 3rd assigned module in terms of class and sequence diagrams.	team
Coding of 3rd	Implementation of 3rd prototype in accordance to the	

prototype	design.	
THU 29.11	(nothing planned)	
Weekly meeting with assistant	Discuss and show what has already been designed and coded this week.	team, assistant
FRI 30.11 08:00	**[d11a]** First draft of design diagrams of 3rd module ready: class and sequence diagrams.	intermediate deadline
	[d11b] First rough implementation of initial prototype of 3rd module, with corresponding tests.	intermediate deadline
Week 12 (03–07.12.2018) – Third Sprint (continued)		
Weekly meeting with assistant	Discuss and provide feedback on design diagrams.	team, assistant
	Discuss and show what has already been coded and what remains to be coded this week.	
Update design	Based on the modifications requested by the assistant, the students update their class and sequence diagrams.	team
Coding and testing	Continue coding and testing of 3rd prototype. A final working and tested version of the 3rd prototype is due by the end of this week.	
~~THU 06.12~~	(no course planned)	
FRI 07.12 08:00	**[D12a]** 3rd prototype ready and working.	**deliverable**
	[D12b] Test suites + document explaining the tests.	**deliverable**
	[D12c] Document detailing precisely what existing code or parts of the database the 3rd prototype touched.	**deliverable**
Week 13 (10–14.12.2018) – Integrate and merge; technical report and documentation		
MON 10.12	Client and assistants look at deliverables **D12*** in detail.	assistant, client
Weekly meeting with assistant	**[F13]** Teams give live demo of their 3rd prototype and corresponding tests to their assistant. (The client will be present at the final project presentation next week.)	team, assistant
Improve	Make final improvements to 3rd prototype where needed, as requested by the assistant.	team
Integrate	All teams (of a same assistant) work together to verify and ensure that all developed prototypes are merged and integrated seamlessly with the original application and the prototypes of the other teams. From this point on, small bug fixes to fix merge conflicts are still allowed, but no extra functionality can be added anymore.	team, assistant
Document	Write a final **technical report**, in Latex format, dedicated to the client with a summary of all the deliverables of the project (requirements, design, test methodology, …)	team
	Writing of the final **user manual**, in Latex format, dedicated to potential users of the application.	
THU 13.12 14:00-16:00	Short recap of final deliverables expected	professor
FRI 14.12 08:00	**[D13a]** All prototypes integrated and merged with main application and prototypes of other teams.	**deliverable**
	[D13b] Final technical report.	**deliverable**

	[D13c] Final user manual.	**deliverable**
	[D13d] Report on the team organisation.	**deliverable**
Week 14 (17–21.12.2018) : project defences		
MON 17.12 8:00	**[D14a]** Screencast (YouTube link) with highlights of the implemented modules.	**deliverable**
	[D14b] Slides of your presentation in PDF format.	**deliverable**
MON 17.12-WED 19.12 slots to be determined	**[D14c]** Final project defence with live demonstration.	team, assistant, professor, client

---

Deliverables and feedback

Below we summarize, in chronologic order, all deliverables your team will need to create throughout this course. There are many of them, because in the end the goal is to come up with a well-documented and well-designed application that will actually be used by the client. Assign a team leader (a.k.a. "scrum master" if you adopt an agile approach) as soon as possible to get yourself organised and ensure that your team is always in time and on schedule for all deadlines and deliverables. But don't blame the team leader if things go wrong, blame yourself.

In the list below **[d]** *refers to intermediate deliverables that will not be evaluated but on which you may get feedback from your assistant or from the client.* **[D]** *refers to deliverables that will be scored.* **[F]** *indicates moments where you will receive information or feedback from either the teaching staff or the client. The number following each letter indicates the week in which this deliverable or feedback is expected according to this initial planning*[3].

[F1] Team creation.

[d2a] Choice of the modules that will be implemented by your team.

[d2b] Detailed ORM (Object Role Modelling) model of the database structure, as a preparatory document to get to know the client's initial database.

[F2] Updated version of the created teams (late arrivals).

[F3] Notification of acceptance by the professor of chosen modules by teams.

[d3] First draft of the requirements document including a set of user stories, a usage scenario for each of the chosen modules, an initial time estimate of the user stories, and a first activity diagram of the entire application including the team's modules to be implemented.

[D4] Final requirements document of maximum 10 pages[4] containing:

- a detailed description, in your own words, of the modules that will be implemented by your team;
- a representative selection of user stories describing the functionality of those modules, together with an initial time estimation of those user stories;

---

[3] We will try to stick to this detailed planning as much as possible. In case of unforeseen changes to this planning you will be informed via the established means of communication.

[4] Additional information relevant to the client may be added in appendices if the 10 pages do not suffice, but will not be evaluated by the teaching staff.

- one to three key usage scenarios illustrating the functionality of your modules;
- some mock-ups or wireframes of the modules to be implemented;
- an activity diagram of how the different parts of the application and the modules to be implemented are connected;
- a detailed planning (using a Gantt or PERT chart) including a schedule of the deliverables to be produced;
- in relation with the planning, a brief description of the development methodology that will be adopted by the team and how (agile, waterfall, hybrid, other);
- your team number, team members (last name, first name, NOMA, e-mail), team leader and course assistant;
- some remarks on the organisation and distribution of work in the team.[5]

[F5] Modifications requested by client and assistant to the final requirements document based on the discussions the teams had with them during a face-to-face meeting and based on their reading of the document.

[D5a] Updated requirements document to be delivered by all teams for whom the client requested modifications. This document should *highlight clearly the changes* with respect to the previous version, for easy reference.

[d5b] First draft of design diagrams ready: class and sequence diagrams for key functionalities (or other types of design diagrams if relevant) of the 1st module to be implemented.

[d5c] First intermediate implementation of an initial prototype of the 1st module. You will still have time to complete it further in the next week, but try to have a first working version ready this week. Also, as mentioned before, you should test often and early. This intermediate implementation should therefore come with a set of tests and pass all tests that you have written so far.

[D6a] Complete and final version of design document with up-to-date version of class and sequence diagrams (or other types of design diagrams[6] if relevant) for the 1st module to be implemented.

[D6b] Final version of the 1st prototype working on the team's own clone.

[D6c] Test suites that have been used to test the prototype, plus a document explaining how the tests can be executed, and what parts of the code they cover.

[D6d] Since this prototype will be merged into the main application together with the prototypes of other teams, a detailed document highlighting precisely what existing code and what parts of the database this prototype has touched. This document may make it easier to resolve potential merge conflicts.

[F7] Feedback from the client on the 1st prototype, during a meeting where you present, demo and discuss your prototype and its corresponding tests. The client will provide oral feedback on whether the presented prototype matches its expectations. Based on the outcome of this meeting the client will either validate or request modifications to the prototype.

---

[5] Including a discussion of who exactly worked on what part, especially when certain members did not participate actively. We cannot take appropriate measures in case of non-participation if we are not informed about this.

[6] Such as an architectural diagram, a state chart, a data-flow diagram, …

[D7] This 1st prototype should be integrated and merged with the main application and prototypes of other teams. The tests of all teams will be useful to check whether everything still works well after the merge. It is essential to verify that nothing will break after this integration. (This integration can be done either before or after the meeting with the client. In case the integration is not yet finished, the application can be demoed on the version of the prototype before merging.)

[d8a] First draft of design diagrams for the 2nd prototype, corresponding to the 2nd module to be implemented by each team: class and sequence diagrams (or other types of design diagrams if relevant) of the module to be implemented.

[d8b] First rough implementation of the second prototype with corresponding tests (after this deadline you still have a full week to complete it). This intermediate implementation should be accompanied with a set of unit tests, and should pass all those tests.

[D9a] Complete version of the 2nd prototype, corresponding to the 2nd module to be implemented, finished and working on the team's own clone.

[D9b] Test suites that have been used to test the prototype, plus a document explaining how the tests can be executed, and what parts of the code they cover.

[D9c] Since this prototype will be merged into the main application together with the prototypes of other teams, a detailed document highlighting precisely what existing code and what parts of the database this prototype has touched. This document may make it easier to resolve potential merge conflicts.

[F10] Oral feedback from the client on the 2nd prototype, based on a meeting with the client where you discuss, execute and demo your 2nd prototype and its tests. Based on the outcome of this meeting the client will either validate or request modifications to the prototype.

[D10] This 2nd prototype should be integrated and merged with the main application and prototypes of other teams. The tests of all teams will be useful to check whether everything still works well after the merge. It is essential to verify that nothing will break after this integration.

[d11a] First draft of design diagrams for the 3rd prototype: class and sequence diagrams.

[d11b] First rough implementation of the third prototype with corresponding unit tests.

[D12a] Complete version of the 3rd prototype finished and working on the team's own clone.

[D12b] Test suites that have been used to test the 3rd prototype, plus a document explaining how the tests can be executed, and what parts of the code they cover.

[D12c] Document detailing precisely what existing code and what parts of the database this 3rd prototype touched.

[F13] Whereas the client will only be present at the final project presentation in the last week, during the week *before* the defences each team should give a live demo of their 3rd prototype to their assistant. This live demo serves as a kind of backup in case the live demo during the final defence would go wrong. At least we have some kind of verification by the assistant then that the application did work. This earlier deadline also forces you to be ready with your application in time and not at the very last minute just before the defence.

[D13a] All prototypes should have been integrated and merged with the main application and prototypes of other teams (of the same assistant). All code should be fully commented and accompanied with comprehensive unit tests.

[D13b] Final technical report of max. 15 pages containing an updated and final version of (a representative selection of the) the different requirements documents (usage scenarios, user stories), mock-ups and activity diagram(s), the different design diagrams (conceptual class diagram, sequence diagrams, and optionally other diagrams you created like an architectural diagram, package diagram, state chart, data flow diagram, …), a description of the testing methodology, test coverage and tests that were used, a discussion of the code quality and use of best practices, and any other technical information that could be relevant to the client. Also, explicitly include a list of changes that were made to, or assumptions that were made about, the original application on top of which your prototypes were created and with which they are integrated. This document should be provided in Latex format for compatibility reasons and to make it easy for the client to make changes to this document later.

[D13c] Final user manual of production quality that can be given to actual users of your modules. This document should be provided in Latex format for compatibility reasons and to make it easy for the client to make changes to this document later.

[D13d] A separate report (max. 3 pages) looking back at the more organisational aspects of the project and what you can learn from it for the future. Respect of the planning (was the planning respected, were there unexpected delays and what caused them, how did you deal with these delays), team organisation (how was the team organised, who did what, what problems were encountered and how were they resolved), lessons learned (positive lessons that you should remember for future projects, negative experiences that you should avoid in the future), …

[D14a] Short screencast (a video of max. 5 minutes) of the highlights of all your implemented modules at work. This video should be delivered as a YouTube link and may be used in the future by either the client or the teacher for publicity purposes in the context of the course (for example, to future students) or for potential users of the application.

The link should be available as a video on YouTube where you clearly mention the following information as a comment: *"Screencast of the Software Engineering Project LINGI2255 of Group XX, academic year 2018-2019, under the guidance of Prof. Kim Mens at UCL, Belgium, for the INGI sysadmin team"*. As title you can use "Screencast Software Engineering Project LINGI2255, Team XX, 2018-2019, SuperForm".

[D14b] Slides of your presentation in PDF format.

[D14c] Final project defence with a live demonstration. Don't just show a video; we want to "see" that it works.

Evaluation

Requirements document	[D4]	**10%**
First module	[D6+7]	**20%**
Second module	[D9+10]	**20%**
Third module	[D12]	**15%**
Documentation		**15%**
Technical report	[D13b]	(5%)
User manual	[D13c]	(5%)

Organisational report	[D13d]	(5%)
Video [D14a]		(bonus)
Final defense		**20%**
Code and Tests	[D13a]	(10%)
Demo and presentation	[D14bc]	(10%)