Organisation of the Software Development Project 2017–2018
21 September 2017

Table of contents

TABLE OF CONTENTS 1
REFERENCES (ON PYTHON AND DJANGO) 1
TOOLS 1
CASE DESCRIPTION 3
PROJECT ORGANISATION 3
CONTACT PERSONS 4
SCHEDULE 4
DELIVERABLES AND FEEDBACK 9
EVALUATION 12

References (on Python and Django)

- 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, interactive online version of the book just above.
- To learn the *Django* web application framework, there is this interesting course in French on OpenClassrooms:
  - https://openclassrooms.com/courses/developpez-votre-site-web-avec-le-framework-django

  In English, many tutorials can be found as well; some or listed below. Up to you to decide which one you think is best, or to look for another. We tried not to list any paid resources.
  - Writing your first Django app, part 1 – this official Django tutorial is a good place to start.
  - Starting a Django Project – explains how to setup a Django Project from scratch.
  - Learn Django – this portal page lists a bunch of Django tutorials and courses.
  - Django Tutorials – this page provides a curated list of Django tutorials.
  - Django Tutorial – a free tutorial on Django on TutorialsPoint.

Tools

Throughout this year's software development project, we will require the use of the following tools and environments. *If you are not familiar with some of these tools, we advise you to learn about them as soon as possible, during the first weeks of the semester.* Throughout the year we will use many tools and there will be many deliverables, so better start early. Remember that this is not just an ordinary academic course; we will actually be building a real product for a real client, which requires us to act professional and use professional tools.

- The programming language and web application framework that are imposed by the

client's existing application will be Python and Django.

- 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/or 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 or, alternatively, a code quality tool like Landscape integrated into your versioning tool.
- Along with Python style guides, use typical Python coding idioms and avoid common mistakes. More information on what to do and what not can be found here:
    - Code Like a Pythonista: Idiomatic Python
    - Python Tricks: Common mistakes and Warts
    - How not to write Python code
- Use the Selenium web browser automation tool for testing the user interface and functionality of the web application.
- Use GitHub as a versioning tool, from the very start, not only when deploying your prototypes. As you will see in the organisation of project, the use of Github will be essential to integrate and merge your code with the existing application as well as 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:
    - 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 scenario
  - Object-role model or Entity-relationship model
  - Gantt chart or PERT diagram
  - 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.
- 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:
  - 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 follow 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 and structure provided by the client.
2. Each team of students will work separately and locally on their own clone, on their own module to be added to the client's application.
3. All teams guided by a same assistant (3 or max. 4 teams) will need to merge their 1st 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 each team continues working individually on the 2nd prototype.
5. Once finished, these 2nd prototypes will need to get merged again in the common shared

repository (for all teams of a same assistant)

Contact persons

Contact person	Nom	Rôle
KM	Kim Mens kim.mens@uclouvain.be	Professeur
LF	Laurent Fourny laurent.fourny@oscar.education	Client (Eureduka)
MC	Maher Chemseddine maher.chemseddine@masi.henallux.be	Client (Eureduka)
RD	Robin Descamps robin.descamps@student.uclouvain.be	Client (jobsite pour Eureduka)
BD	Benoît Duhoux	Assistant 1
XG	Xavier Gillard	Assistant 2
MS	Michael Saint-Guillain	Assistant 3
AG	Aoga John	Assistant 4
HQM	Ha Quang Minh	Assistant 5

Schedule

Dates	Activity	Responsible
Week 1 (18–22.09.2017)		
WED 20.09 10 :45-12 :45 SUD11	Professor introduces course briefly	KM
	Client introduces Oscar case	LF
	Client introduces case technology and architecture	RD
Getting acquainted	Read case specification and technical description	students
	Learn technology: Python, Django, GitHub and other tools	
	Get registered: confirm course attendance, register to Moodle, create teams, join Slack	
FRI 22.09	Distribution of case specification	LF
	Distribution of planning document	KM
FRI 22.09	**[F1] Teams created**	**feedback**
Week 2 (25–29.09.2017)		
Getting started	*(Teacher may add new team members during this week.)*	KM
	Continue learning technology and tools: Python, Django, Github, PyCharm, Slack, …	students
	Download and install initial code base, development tools, versioning system.	
	Assign internal team leader (a student).	team
	Contact assigned assistant to fix weekly meetings.	
	Discuss and choose module to implement.	
	Make detailed ORM or ER model of the existing database.	
	Ask client if things are unclear in the case description.	
~~WED 27.09~~	(no course: fête de la communauté française)	

Weekly meeting with assistant	Getting acquainted with your assistant. Discussion about modules to be chosen. Feedback on ORM and ER model.	team + assistant
FRI 29.09	**[D2a]** Module to be developed chosen.	**deliverables**
	[d2b] A detailed ORM or ER model of the structure of the initial database.	
	[F2] Updated teams	**feedback**
Week 3 (02–06.10.2017)		
MON 02.10	Notification of acceptance of chosen module.	KM
WED 04.10 10 :45-12 :45 SUD11	Course on requirements analysis (~1h).	KM
	Q&A session with client regarding case study (~1h)	LF
Requirements analysis	Creation of user stories, user scenarios and estimates for the module to be developed.	teams
Weekly meeting with assistant	User story workshop. Each team creates a set of user stories for its module (1 hour). In the second hour the teams present (some of) their user stories to the other teams.	teams + assistant
FRI 06.10	**[d3]** First draft of requirements document ready: user scenarios, user stories and time estimate of user stories.	**intermediate deadline**
Week 4 (09–13.10.2017)		
MON 09.10	Assistant provides feedback on draft requirements document.	assistant
WED 11.10 10 :45-12 :45 SUD11	Course on activity diagrams, Gantt charts, PERT charts and wireframes.	professor
Requirements analysis (continued)	Refine user scenarios, user stories and estimates based on feedback received.	teams
	Create mock-ups or wireframes of the application.	
	Create activity diagram to describe application flow.	
	Make detailed project planning using Gantt or PERT chart.	
Weekly meeting with assistant	*Assistant discusses and provides feedback on requirements documents produced by the team this week*	teams + assistant
FRI 13.10	**[D4]** Final requirements document (10 pages) : user scenarios, uses cases, mock-up, planning, activity diagram.	**deliverable**
Week 5 (16–20.10.2017)		
MON 16.10	Client reads all requirements documents.	client
WED 18.10 10 :45-12 :45 SUD11	Course on architecture, design and team work.	professor
Meeting with client (and assistant)	During a question-and-answer meeting the client and the teams will interact to verify whether the requirements have been understood well. Based on the outcome of this meeting the client may request an update of the delivered requirements document.	teams + client + assistant
Design phase	In the mean time the teams start working on the design of their application in terms of a class diagram and some sequence diagrams for key functionalities.	
FRI 20.10	**[F5]** Modifications requested to requirements document by	feedback

	the client.	
	[d5] First draft of design diagrams ready: class and sequence diagrams.	intermediate deadline
Week 6 (23–27.10.2017) – "SMART" week		
Updates	Based on the modifications requested by the client, the students update their requirements diagram as well as the class and sequence diagrams already designed, if they are impacted by these requested modifications.	teams
Coding	Take advantage of this course-free week to start implementing the 1st prototype of your module(s).	teams
WED 25.10 10 :45-12 :45 SUD11	(professor absent)	professor
	Permanence and feedback session with the client	client
Weekly meeting with assistant	Discuss and show modifications made to requirements and design diagrams. Discuss and show what has been or will be coded this week.	teams + assistant
FRI 27.10	**[D6a]** Updated requirements document to be delivered by all teams for which the client requested modifications.	**deliverable**
	[D6b] Completed version of design document with improved version of class and sequence diagrams + a detailed description of how the module to be developed will integrate with the existing application.	**deliverable**
	[d6c] First rough implementation of initial prototype, with corresponding tests.	intermediate deadline
Week 7 (30.10–03.11.2017)		
~~WED 01.10~~	(no course: Toussaint)	
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.	teams
Weekly meeting with assistant	Discuss and show progress on implementation.	teams + assistant
FRI 03.11	**[D7a]** 1st prototype ready and working.	**deliverable**
	[D7b] Test suites + document explaining the tests.	**deliverable**
	[D7c] Document detailing what existing code or parts of the database the 1st prototype has used or touched.	**deliverable**
Week 8 (06–10.11.2017)		
Integration	Teams (of a same assistant) try to merge their 1st prototypes together into the main application and resolve potential merge conflicts.	teams
WED 8.11 10 :45-12 :45 SUD11 or slot weekly meeting with assistant or …	**Presentation of 1st prototype to the 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. The client may also provide suggestions regarding how to design the user interface.	teams + client + assistant
FRI 10.11	**[F8]** Feedback from client on 1st prototype and its tests.	**feedback**
	[D8] 1st prototype integrated and merged with main	**deliverable**

	application and prototypes of other teams.	
Week 9 (13–17.11.2017)		
Design of 2nd prototype	Now that the 1st prototype is finished and integrated, the teams can start developing the 2nd prototype of their application. As a first step it would be good to work out the design of the 2nd prototype in terms of class and sequence diagrams.	teams
Coding of 2nd prototype	Now that the 1st prototype is finished and integrated, the teams can start implementing the 2nd prototype of their application + make the modifications requested by the client regarding the 1st prototype.	teams
WED 15.11 10 :45-12 :45 SUD11	(nothing planned)	
Weekly meeting with assistant	Discuss and show design diagrams. Discuss and show what has been / will be coded this week.	teams + assistant
FRI 17.11	**[D9a]** Design document with class and sequence diagrams for 2nd prototype + detailed description of how the module to be developed will integrate with the existing application.	**deliverable**
	[d9b] First rough implementation of second prototype with corresponding tests.	**intermediate deadline**
Week 10 (20–24.11.2017)		
MON 20.11	**[F10]** Feedback from assistant on design document D9a.	**feedback**
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.	teams
Weekly meeting with assistant	Discuss feedback on design document D9a. Discuss and show progress on implementation and tests.	teams + assistant
WED 22.11 10 :45-12 :45 SUD11	Permanence and feedback session with the client.	client
FRI 24.11	**[D10a]** 2nd prototype ready and working.	**deliverable**
	[D10b] Test suites + document explaining the tests.	**deliverable**
	[D10c] Document detailing precisely what existing code or parts of the database the 2nd prototype touched.	**deliverable**
Week 11 (27.10–01.12.2017) : presentation and validation of 2nd prototype		
WED 29.11 10:45-12:45 SUD 11 or slot weekly meeting with assistant or …	**Presentation of 2nd prototype to the client.** Teams demo their 2nd prototype and corresponding tests to the client to receive feedback whether their prototype satisfies the client's expectations. Based on the outcome of this meeting the client will either validate or request modifications to the prototype.	teams + client + assistant
FRI 01.12	**[F11]** Written feedback from client on 2nd prototype and its tests.	**feedback from client**
Week 12 (04–08.12.2017) : deployment and merge of prototypes; technical report		
Improve	If the 2nd prototype wasn't validated in the previous week	

	and the client still requested modifications, make these modifications, have it validated again (either by the client or assistant, as agreed), and then deploy and test it on the production site.	
Weekly meeting with assistant	**[F12]** Discussion with and feedback from the assistant on the requested modifications.	**feedback**
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 issues are still allowed, but no extra functionality can be added anymore.	All teams (of a same assistant) jointly
Document	Write a final **technical** report, in OpenOffice format, dedicated to the client with a summary of all the deliverables of the project (requirements, design, test methodology, …)	teams
WED 06.12 10:45-12:45 SUD 11	Explanation of final deliverables expected	professor
	Permanence and feedback session, for example, on the requested modifications to the 2nd prototype	client
FRI 08.12	**[D12a]** 2nd prototype integrated and merged with the main application and prototypes of other teams	**deliverable**
	[D12b] Final technical report	**deliverable**
	[D12c] Report on the team organisation	**deliverable**
Week 13 (11–15.12.2017) : finalisation of user manual, tests and documentation		
Test	Finalisation of unit tests, automated tests with Selenium and documentation.	teams
Document	Writing of the final user manual.	
WED 13.12 10 :45-12 :45 SUD11	Short recap of final deliverables expected	professor
	Permanence and feedback session (last one before defence)	client
Weekly meeting with assistant	**[F13]** Teams give a live demo of their application to their assistant.	Team, Assistant
FRI 15.12	**[D13a]** Fully documented code.	**deliverable**
	[D13b] Final overall version of the tests (unit tests and Selenium), updated according to the two feedbacks.	**deliverable**
	[D13c] Final user manual in OpenOffice format, to be handed over to the client (production quality).	**deliverable**
Week 14 (18–22.12.2017) : project defences		
MON 18.12	**[D14a]** Short video (YouTube link) with a demo of the working prototype.	**deliverable**
WED 20.12 10 :45-12 :45 SUD11 + additional slots (TBD)	**[D14b]** Final project defence including a demonstration “in vivo” on the “production” site.	Team, Assistant, Teacher and 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.* **[D]** *refers to evaluated deliverables.* **[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 of feedback is expected according to this planning*[1].

[F1] Team creation.

[D2a] Choice of the module that will be implemented by your team.

[d2b] Detailed ORM (Object Role Modelling) or ER (Entity Relationship) model of the database structure, as a preparatory document to get to know the database of the client's existing web application.

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

[d3] First draft of the requirements document including a set of user stories, user scenarios as well as an initial time estimation of the user stories.

[D4] Final requirements document of maximum 10 pages[2] containing (a representative selection of) the user scenarios, uses cases, mock-ups or wireframes, activity diagram(s) and detailed planning.

[F5] Modifications requested by the client to the requirements document based on the oral discussion the teams had with the client and based on the client's reading of the document.

[d5] First draft of design diagrams: class and sequence diagrams (or other types of design diagrams if relevant) of the module to be implemented.

[d6a] 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.

[D6b] Complete and final version of design document with up-to-date version of class and sequence diagrams (or other types of design diagrams[3] if relevant). This design document should also include a detailed description of *how and where* the module to be developed will integrate with the existing application, as well as an *updated planning* of what and when you will implement for your 1st and 2nd prototypes.

---

[1] 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.

[2] 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.

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

[d6c] First intermediate implementation of an initial prototype. 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.

[D7a] Final version of the first prototype working on the team's own clone.

[D7b] 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. If possible and relevant, in addition to unit tests, try to include already Selenium tests[4] for your first prototype.

[D7c] 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.

[F8] Written feedback from the client on the 1st prototype, based on a meeting with the client where you discussed, executed and demoed your 1st prototype and its tests.

[D8] 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.)

[D9a] Design document with the class and sequence diagrams for the 2nd prototype (may include other types of design diagrams, if relevant). This design document should be complemented with a detailed description of *how and where* the module to be developed will integrate with the existing application, as well as an *updated planning* of what and when you will implement for your 2nd prototype.

[d9b] First intermediate implementation of a 2nd prototype. You will still have time to complete it in the next week, but try to have a first working version ready this week. This intermediate implementation should be accompanied with a set of tests, including Selenium tests, and should pass all those tests

[F10] Feedback from your assistant on the design document deliverable D9a.

[D10a] Complete version of the 2nd prototype finished and working on the team's own clone.

[D10b] 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. In addition to unit tests, the tests should include Selenium tests for your 2nd prototype.

[D10c] 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.

[F11] Written from the client on the 2nd prototype, based on a meeting with the client where you discussed, executed and demoed your 2nd prototype and its tests.

---

[4] Selenium tests will be imposed for the second and final prototype, so better start exploring this testing technology now already.

[D12a] 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. (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.)

[D12b] Final technical report of max. 15 pages containing an updated and final version of (a representative selection of the) the different requirements documents (user 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 OpenOffice format for compatibility reasons and to make it easy for the client to make changes to this document later.

[D12c] 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), …

[F13] During the week *before* the defences each team should give a live demo 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] Fully documented code should be made available to the client.

[D13b] This final version of the code should be accompanied with comprehensive tests (both unit tests and Selenium tests).

Obviously, tests and documentation shouldn't be written only at the end but all throughout the project. The deadlines [D13a] and [D13b] above are thus only a finalisation and clean-up of the already existing documentation and of the already existing unit tests and Selenium tests which you created before, updated according to the two feedbacks and the results of the integration process, so that they can be handed over in a final clean state to the client as part of the final application to be delivered.

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

[D14a] Short video of the final working version of the application in action. 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 or the application.

[D14b] Final project defence with a demonstration "in vivo" on the production site. This must be a live demonstration, not just showing the video; we want to "see" that it works.

Evaluation

Requirements document [D4]		**10%**
Design documents [D6b]		**5%**
First prototype [D7*]		**20%**
Design documents [D9a]		**5%**
Second prototype [D10*]		**20%**
Documentation		**15%**
Technical report [D12b]	5%	
Organisational report [D12c]	5%	
User manual [D13c]	5%	
Video [D14a]	bonus	
Final product		**25%**
Code [D13a]		
Tests [D13b]		
Demo and presentation [D14b]		
