COM1001 Introduction to Software Engineering – 29th January, 2021

Team Software Project:
Instructions for the Spring Semester

1 Structure, Deadlines & Deliverables

In the spring semester each team will implement a web system according to the particular stories they collected from their client in the autumn semester. Because each team has a different client, the requirements/stories for each team will be slightly different. The system will be delivered to the client over two iterations, as per an agile software development strategy.

You will demonstrate progress on implementing and testing the system to the client at the end of each iteration, and the client will feed back to you (and us, so that we can assess your work) as to whether the system meets their expectations as described to you in the autumn semester. In particular, following the first iteration, your client may request further changes based on what they've seen and in addition to changes to the original set of user stories. The end of the second iteration marks the finalisation of development on the project and its delivery/handover to your client.

We cover the technical aspects behind the development of the system during lectures and labs in weeks 1–6. Please see the MOLE/blackboard site for lectures (including resources), labs and forum.

The schedule and deadlines for the project are as follows:

Iteration	Deadline	Semester Weighting
First	Mon, **22nd March**, 16:00 (week 7)	40%
Second	Mon, **10th May**, 16:00 (week 11)	60%

The following artifacts will be assessed as following:

Iteration 1 and 2:

- Exemplar code (see Section 1.1) to be
    a. Iteration1 - committed to your team repository
    b. Iteration2 - submitted through blackboard
- Client demonstration - see also Section 3. This will be recorded and also viewed by the markers. Please ensure you:
    1. Introduce yourselves individually, at the beginning, saying very briefly what your roles/responsibilities were.
    2. At the end (possibly after your client has left), you need to show your automated tests running.
- The Team Contribution will need to be submitted by each individual team Member (see Section 1.4)

Iteration 2 will also include:

- Inspection of your Code committed to your team repository (section 1.2)

- A **Team Report**, (also containing your exemplar code), submitted through Blabkboard (see Section 1.3)

1.1 Exemplar Code

For both iterations, you will submit a short PDF document containing 'your best code'. The document is limited to 3 sides for iteration 1 and 4 sides of your Team Report for iteration 2.

This exemplar code allows you to show off your ability and receive specific feedback. You should choose your code to represent different aspects of your solution. At least 80% of the code should be ruby code; you may also include some html/css/javascript.

You must use source code colouring, line numbers and name each listing. **Take a screenshot** and paste it in the document, e.g.

```
1  module Util
2    def Util.days_in(month, year)
3      result = 31
4      if ([4,6,9,11].include? month)
5        result = 30
6      elsif (month == 2)
7        result = 28
8        if (year % 4) == 0
9            result = 29
10           if ((year % 100) == 0) && ((year % 400) != 0)
11               result = 28
12           end
13       end
14     end
15     return result
16   end
17 end
18
19
```

*Listing **A** src/example/days_in_month.rb*

Please note:

- The Caption needs to include:
  - an alphabetic listing 'letter', e.g. 'Listing A' above allows feedback to simply refer to A.11 (i.e. listing A, line 11).
  - the folder and filename, from the root directory, to allow the marker to see more detail in your repository when necessary, e.g. the caption above shows 'src/example/days_in_month.rb', which identifies the source file and it's location.
- You should not (typically) include a whole file listing.
- You may choose to 'fold' parts of your code.
- Standard code quality applies, e.g. **appropriate** comments should be included – that typically say **what** the code does, not how it does it (except when it's necessary).
- *Do not include pointless comments – e.g. puts "begin" ~~# output "begin" to the console~~*
- Code should, when possible, explain itself and not need many comments.
- I recommend you do not use code colourizing within the document – it frequently creates issues when opened for marking – a screenshot works much more reliably.

In Iteration 1, the exemplar code needs to be submitted by adding it to your team's repository. So that we can find your report and read it, you *must* produce it in **PDF format**. You must name it as 'exemplar-code.pdf' and place it in the *root directory* of your repository.

For Iteration 2, include the exemplar code in the Blackboard submission of the Team Report.

1.2 Code Committed to the Team Repository

We may inspect your code at the end of iteration1, but this would only be for clarification.

At the end of iteration 2, we will also be looking at the **last commit to the team's Git repository before the deadline** and assessing the amount of work you have completed, the folder structure and its quality. Code will only be checked out to Codio for running and checking – it is strongly recommended that you develop on Codio – and at the least it is essential that you deploy often to Codio – this is the required deployment platform.

Ensure **all the necessary files are committed to your repository by the deadline/s**, otherwise you will not receive the mark you deserve. Late submission will doubtless mean we will not be able to obtain copies to assess. (Do not send files after the deadline that you forgot to commit, or instructions to log into your Codio box to find files)

Please note the following important points in regard to the use of the Git repository:

1. **Team repository — no other.** Ensure you're committing to your team repository (refer to the relevant lab) — not your personal repository from last semester, or another Git repository.
2. **All members should be making regular commits.** In marking your work, we will be checking that you have been making regular commits — not just one or two commits or a sudden rush of commits just before the deadline.

In order to mark your work, we need to be able to properly install and deploy your application. Therefore:

3. **Ensure that your** `Gemfile` **is up to date.** If we cannot run your application due to missing gem information, this will severely negatively affect your mark.
4. **Attempt a clean install before your final commit.** Finally, I would advise that you attempt a clean install of your system before the final commit (testing your deployment instructions that you will write in your team report), so that you are sure that we will be able to deploy your system. If the clean install does not work, then clearly you have some dependency problems or other issues that need to be rectified before submission.

For both iterations, you need to update your README.md to detail the following information:

1. Details on how to run and access your system, including:
   (a) **The exact command(s) needed to install (from scratch)** - including which directory you need to be in.
   (b) Any setup (scripts) that need to be run.
   (c) **How to and start** (and access) **your system** — so we can assess it.

Because of all of the above, **I suggest your team organises itself with a 'soft' deadline 1–2 hours (or more) before the hard deadline of 4pm.** After your soft deadline, make final checks to ensure everything is committed and deploys properly. If you've forgotten anything, you'll still have time to correct the mistake. Don't use the period after the soft deadline to make further tweaks or do more implementation and testing — that defeats the point!

1.3 Team Report

As well as the details in the README.md, at the end of iteration 2, the Team report should also detail the following information:

1. Details on how to run your tests, including example runs (see later)

   (a) **The exact command(s) needed to install (from scratch) and start your system** — we need to know how to start your application in order to assess it.

   (b) The following standard usernames must be used (this is not meant to be secure…):

Username:	Password:
admin	admin
mentee1	mentee1
mentee2	mentee2
mentor1	mentor1
mentor2	mentor2

   If you need to include more usernames, then you will need to detail them in your team report.

   (c) **Any other special details** particular to your application, that we will need to know in order to deploy, run and assess it.

2. **Resubmission of your stories, with changes annotated and highlighted.** Changes to stories are allowed at any time, and can happen for a variety of reasons — for instance: responses to autumn semester feedback, changes made by the client, or changes made due to technical reasons (e.g., something promised in the stories was not actually feasible, or could not be made to work as originally planned). You should highlight any changes (so that it is clear to us *what* has changed), and annotate the change with the reason for the alterations (so that it is clear to *why* it has changed). I strongly recommend that you use **bold** or green highlight for additions, and ~~strikethrough~~ or red highlight for deletions. **Note that failure to adequately annotate and highlight your stories, or unclear/confusing changes, will result in reduced marks.**
3. **The stories you planned to tackle in each iteration.** You should document which stories you planned to tackle in each iteration, along with an account of who did what. You should aim to split the workload as evenly as possible across the two iterations. Trying to deliver everything in the first iteration defeats the point of an agile methodology — the client may well change their mind on aspects of the project once they have seen it for the first time, leaving you with a lot of re-engineering to do.
4. **Your burndown chart, showing progress over all of your stories over each iteration.** You need to include a burndown chart of your progress in developing your stories. This means you need to keep an accurate log of how your project has been advancing over time.
5. **Testing and test coverage.** You will need to include a section on how you went about testing, e.g. unit testing and manual. You need to show the output from your tests, most likely as screen shots. The report should contain details of how to execute the tests you have written on the system (or you may put this in the README.md). Your tests should be named or described for their purpose and you should report why you used different types of testing for different aspects of the system where appropriate. If you have had to resort to manual tests, e.g. for browser interaction, full documentation of these tests should be present, including identifying the purpose of tests and the expected results and any failures.
   Note: It is better to show tests that fail than to delete them – especially in iteration 1.
6. **Exemplar Code.** *Please see previous description.*

The team report needs to be submitted through Blackboard in **PDF format**.

1.4 Team Contribution

Since Web PA is no longer available, further details will be announced when they are ready.

As with last semester, you will need to rate yourself and your team members on the following criteria:

1. Attendance and punctuality (to group meetings, sessions with clients, etc.).

2. Ability to work effectively with other team members.
3. Contribution to content and organisation of project deliverables.
4. Quality of contributions.
5. Timeliness of contributions.

Your final individual mark will be formed from a scaling factor that is applied to your overall team mark. The scaling factor is determined by your own self and peer assessment, and those returned by your team members.

2 Team Supervision Meetings

It is important that all team members are present at each meeting, barring exceptional circumstances. Supervisors will take a register: if you cannot make a session, let one or more of your teammates know. If you persistently miss meetings, your tutor will be informed and you will be reported to the faculty for non-engagement with your studies.

3 Client Meetings

The Client meetings will be held in weeks 7 and 11 as following

- Friday, 26th March, between 1 and 3
- Friday, 14th May, between 1 and 3

The schedule for each team is posted on Blackboard. All meetings will **take place online through your team's Blackboard collaborate** at the same times as the supervisory meetings detailed on MOLE/Blackboard.

In these meetings, you will demonstrate the current state of your software to your client. For this, you should prepare a **10 minute demonstration** of your software (this will be recorded when you demonstrate to your client).

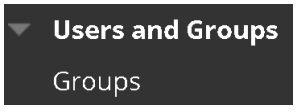
Prepare the demonstration of your software, in its current state, if iteration 1, by rehearsing beforehand. The best demonstration will involve all or as many team members as possible saying and demonstrating something.

Let your client try out the system - **you should send the codio url to access your web application to the client – they can then share their screen with you**. They will give you feedback and, if it's the first iteration, they may request changes to be made for the second iteration.

Blackboard Collaborate Team 'room'

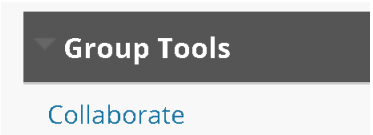
In blackboard, under Users and groups

1 click on groups:



2 You will find your Team there – click on your team (please let me know if you are in the wrong team – mistakes happen). You will notice that your team contains:
   - Your team supervisor
   - The demonstrators
   - Your clients

3 You can then choose the Collaborate option under 'Group Tools'



4 Here you can view recordings (if you have made any). You should record both demonstrations.
5 Select 'Join Room' (top right).
6 When Collaborate starts – you will need to **enable audio** (and video).
7 The bottom right 'arrows' allow you to open a panel – the middle option allows you to share content – you can then Share your application screen; sharing 'Entire window' is easiest.
8 This will allow you to demonstrate to your client. You can also 'chat' and send the 'live' ruby url for the client to use.

Note: Follow the instructions for how to deploy a Client Demonstration within Codio (see Blackboard - 'Assessment - Sem 2'). You should exit the/kill the running application after the client has left.
N.B. DO NOT make your development Codio boxes 'Public' - this would be insecure - YOU HAVE BEEN WARNED.

4 Team Planning

You should schedule stories to implement according to your client's priorities. However, your implementation plan may also be necessarily constrained by the order in which technical material is covered in lectures and labs. So check out the lecture schedule and plan the stories you plan to implement for each checkpoint/iteration accordingly. As stated above, do not try to deliver everything in the first iteration — teams will be marked down if they do this.

With all this in mind, here is a suggested week-by-week list of pointers for your team that you can use to check your plan of action:

Week	Suggested Activities
1-3	Study the lecture notes and practice lab class material on web application development in Sinatra. Reflect on last semester's feedback on your stories — are there any changes that need to be made? Start to think in your team about which stories will be implemented first, and who will do what. Ensure each member can access the team repository and that you're all set up on Slack and GitLab. Implementation should be underway now. You should be constructing the basic views for the system, thinking about the forms that need to be writing and writing the controller code to handle them. It's time to start planning how you will unit test your application, including how you will 'separate your concerns' and make use of test driven development. Do you have units to test? If not, can you refactor some of the core functionality of your system so that it's testable? If you are copying and pasting code, then you should instead be refactoring to make that code modular - and aiming to unit test the refactored method/function.
4	Think about which stories may require a database and how those may be tackled before the first iteration. Make sure you have designed your tables to reduce redundancy and duplication.
5	You should have a good set of Unit tests by now and you should also identify the Acceptance tests (if you haven't already).
6	Write tests to cover as much of your iteration 1 stories as possible. As you incorporate more stories into your implementation, ensure that you are testing as you are implementing, or before, as opposed to leaving it a long time afterward. Start finalizing the

	first iteration code as the iteration 1 is due next week. Prepare for your team demonstration next week.
7	Collate your exemplar code and create a PDF document containing colourised code. Ensure your repository is up to date. Prepare and rehearse your team demonstration of your system for your client meeting this week.
	Easter Vacation
8	Decide, as a team, how you're going to make changes as a result of client/supervisor feedback and any new requests.
9-10	Time to meet up and make sure that everything integrates well and is consistent. Check your code style, appropriate commenting, tests, etc. Get the final parts of the system implemented and tested. Prepare and rehearse for your Client demonstration next week.
11	The second iteration deadline is this week. Time to finish up on implementation and testing. Submit your final team report. Present to your Client.
12	Don't forget to complete the peer and self assessment by Friday

5 Marking Criteria

The following table details the criteria against which your team's work will be marked. You will receive a mark sheet containing a table, with marks and feedback against the aspects detailed below. Each aspect is marked out of 100. The overall mark is the average of each of the marks for each aspect.
Note: 2 indicates for iteration 2 only, 1+2 indicates for both iterations (highlighted in grey).

Aspect (iterations)	Indicative Criteria
Code (1+2)	The quality of your code, including good coding principles, such as useful variables names, code comments, etc. The extent to which your code defends against simple common vulnerabilities. The basic organisation of your code into units, and its overall structure. **Iteration 1**: Your exemplar code will be considered. **Iteration 2:** Also, the Team repository will be examined.
Overall Product (1+2)	The strength of the feedback from your client. **Iteration 1**: The amount of work completed to date, the extent to which you are on track to delivering a good quality product and the appropriateness of stories implemented in this iteration. **Iteration 2**: The completeness and quality of your final system. The extent to which it implements the client's requirements, how usable it is, how polished it is, the extent to which it crashes and/or throws errors.
Testing (1+2)	The quality of your automated tests. The extent to which you have justified the level of coverage that you have obtained. The evidence of your tests runs recorded following the Client demonstration. **Iteration 2:** Your test scripts will also be run by the marker according to your instructions.
Documentation and Deployment (2)	The quality of your team report, the extent to which it details all the requested information, including changes to stories, basic information about using your system etc. The extent to which it was easy and straightforward to deploy and get your system running — on the basis of your instructions and the implementation of your deployment process — from Git checkout to the commands needed to start the application.