CSCC01 Introduction to Software Engineering - Midterm Test

27 October 2018

DO NOT OPEN THIS BOOKLET UNTIL YOU ARE TOLD TO DO SO

In the meantime, read and complete this front page.

Last name:	____________________
First name:	____________________
Student number:	|_|_|_|_|_|_|_|_|_|_|
Student email:	____________________`@mail.utoronto.ca`

Rules:

1. Before the exam starts, you are allowed to logon to the exam computer, open a web browser and check that you have access to `https://cms-git.server.utsc.utoronto.ca/` using your UTORID and password.
2. When you receive the signal to start, please make sure the copy of your examination is complete.
3. Legibly write your name and student number on this page.
4. There are a total of 100 marks and 7 questions.
5. Total time is 90 minutes.
6. Write and submit all of your answers as instructed for each question using the computer in front of you.
7. The exam is closed book, and no aids of any kind are allowed.
8. At the end of the exam, return this examination copy to the exam staff.

Exam Setup

There is no question in this part. However, read carefully the informations below regarding the exam setup.

In this exam, you will be asked to complete a project documentation and source code. You will use the computer in front of you to do this work.

Similarly to what you have done so far in this term, you will use `git` and a private exam repository for submitting your work. Your answers will be graded **if and only if they have been pushed to your exam repository**. No file will be retrieved from the computer once the exam is completed.

Using `git`

You will use `git` and a private exam repository (see below) to 1) retrieve the initial documentation and source code and 2) submit all of your answers. Throughout the exam, here are some good practices that you should adopt:

- **Push early, push often.** It is strongly advised not to wait for the end of the exam for pushing your answers to the repository. You should push each of your answers as soon as they are ready and/or updated.
- All of your commit must have meaningful commit messages.
- Do not push automatically generated files.
- Make sure to adopt good branching and merging management as seen during this term.
- All of your answers must be on the branch `master` unless specified otherwise.

Here is a brief reminder of the most important `git` commands. For the commands below, all keywords surrounded by double underscores such as `__email_address__` must be replaced with a proper input.

description	git command
setup a global variable email	`git config --global __email_address__`
clone a repository	`git clone __repository_url__`
pull changes	`git pull`
staged a file	`git add __filename__`
commit staged files	`git commit -m "__commit_message__"`
push latests commits to the repository	`git push`
list all branches	`git branch -a`
create a new branch	`git branch __branch_name__`
change branch	`git checkout __branch_name__`

push/pull on a new branch
(first time only)
	`git push origin __branch_name__`
	`git pull origin __branch_name__`

merge a branch	`git merge __branch_name__`

Using the exam repository

In this exam, you will use a `git` repository that is different from the one you use for your term work. This repository is **not on github** but on a dedicated `git` server. You can use it from the **Git CMD** Windows terminal directly or from its web interface located at `https://cms-git.server.utsc.utoronto.ca/`. It is a private repository and you should use your UTORID and password to access it.

To know the url of your repository, replace `__your_email_prefix__` accordingly. For instance if your email is `john.doe@mail.utoronto.ca` your email prefix is `john-doe` (notice all dots have been replaced by dashes).

```
https://cms-git.server.utsc.utoronto.ca/sansthie/__your_email_prefix__.git
```

If you are not able to clone this repository (assuming you use the right command), open a web browser to `https://cms-git.server.utsc.utoronto.ca/` and connect to the web interface using your UTORID and password. Once you are logged-in, try again to clone your repository from the terminal.

If you still cannot clone your repository, contact the exam staff. Otherwise, proceed to Part I.

Part I
Product Backlog

The GoodHealth Medical Center needs your help to develop an application to register new patients and manage the medical records of its patients. At first, new patients must fill the "New Patient Intake Form" directly through the application or have someone to fill the form for them. If needed, the Medical Office Assistant seating at the frontdesk can also do it for the patient over the phone or in person at the medical center. Once the form has been submitted, the Medical Office Assistant will create a new medical record for that patient and assign that patient to one of the doctors working at the medical center. A medical record cannot be created without being assigned to a doctor. When a patient meets with a doctor, the doctor can add notes to the patient's medical record and see all the previous ones (related to previous visits). The application must be robust because without access to the medical record, the doctor will not be able to give an appropriate diagnostic to the patients.

Question 1. [15 MARKS]

Task: Complete the file `product-backlog/personas.txt` by briefly describing up to four (but not necessarily 4 if it is not adequate) Personas that would be useful for the design and development of your system.

Question 2. [15 MARKS]

Task: Complete the file `product-backlog/user-stories.txt` by writing up to ten (but not necessarily 10 if it is not adequate) user stories based on the description of the required system. Choose the most relevant system features for your user stories and order them by build priority (the first one being the first feature to be built). Include user stories for all Personas that you have described in the previous question.

Part II
Sprint Backlog

GoodHealth has a team of three developers: Alice, Bob, and Charlie. Their sprint length is 5 days and 1 story point corresponds to 1 developer-hour. Each of them contributes equally to the project — 3 story points per day.

Question 3. [10 marks]

These three developpers have come up with a plan for sprint 01 and recorded it in their sprint backlog (`sprint-backlog/sprint01/plan.csv`). According to this plan, they should be able to release the user stories `U1` and `U2`. Unfortunately things did not go as planned during the sprint.

Bob starts working on task 1 on Monday. However, later that day, he announces that he is giving up and hands in his resignation without completing any work. Alice and Charlie will have to work on sprint 01 without him.

So, after finishing T4 on Monday, Alice decides to take over tasks 1 and 2 on Tuesday. She starts from scratch and finishes both of them by the end of the day. Then, she spends the rest of the week working and finishing T3 by Friday night.

Meanwhile, Charlie is cruising. He finishes T5 on Tuesday night. Then, he spends the rest of the week working and finishing T6 by Friday night.

Task: Assuming that you are tracking the sprint execution and re-allocating tasks on day-by-day basis, complete the file `sprint-backlog/sprint01/execution.csv` by producing an actual sprint execution report.

While re-allocating tasks, you should follow these constraints:

- the user stories with higher priority should be completed first whenever it is possible
- each task should be completed by one developer and one developer only
- for a developer to be able to work on a given task, each dependency must have been completed either 1) at last the day before if the dependency has been assigned to a different developer or 2) the same day if the dependency has been assigned to the same developer

Question 4. [10 marks]

Task: Complete the file `sprint-backlog/sprint01/provisional-burndown-chart.txt` by producing an adequate provisional and actual burndown chart. Instead of drawing an actual chart, complete the given table by filling it with all x and y values for that sprint including the value at origin ($x = 0$) for both the provisional and the actual burndown chart.

Question 5. [10 marks]

After sprint 01, the management has not found a suitable replacement for Bob. Alice and Charlie will have to work on Sprint 02 alone again. Yet, they believe they can finish the user story `U3` by the end of the week.

Task: Create the file `sprint-backlog/sprint02/plan.csv` and produce an adequate plan for Sprint 02. While allocating tasks, you should follow the same constraints as for Sprint 01.

Part III

Feature Development and Release

In this exercise, you are going to work on the source code of the product.

Currently, there is one new feature called `patient` that has been developed but not released yet. According to the person who worked on that feature, the code should be complete however there is a bug that makes the unit tests fail. This person is asking for your help to debug the code and release the feature.

Question 6. [15 marks]

Task: Fix the bug. While doing so, do not change the design and signature of the existing code. When the code is working, release that feature as the latest version of the product.

Question 7. [25 marks]

Then, you are going to work on a new feature to manage the medical records. During the team meeting before sprint 02, you and your team mates agreed on the overall design of that feature. By the end of the meeting, you have drafted three classes that can found in the `draft` directory of sprint 02:

- the first one is the class `MedicalRecord` that implements the medical record data structure
- the second one is the class `MedicalRecordApi` that has two API functions so far:
  - `createMedicalRecord` to create a medical record and then add it to the existing collection of medical records
  - and `getMedicalRecord` to retrieve a patient's medical record based on its `HealthCardID`
- the third one is the class `PatientNotFoundException` that implements the exception raised when retrieving a medical record for patient ID that is not in the collection.

For the sake of this exam, we assume that we will never create two medical records for patients with the same `HealthCardID`. Therefore, no need to check nor test that in your code.

Task: Based on the design draft, create this new feature called `md`, complete the class `MedicalRecordApi` and implement its corresponding Java unit tests class `MedicalRecordApiTest`.

Indeed, do not work directly on the `draft` directory but use these drafts to populate the existing code base.

While working on the feature, follow the design that was decided during the team meeting. This means that you should not modify the design and the signature of the classes given as draft. Yet, you can add additional methods if needed.

Finally, when this feature is ready, release it as the latest version of the product.

Use this page for rough work.

Use this page for rough work.

Use this page for rough work.

Total Marks = 100