General Information

This document contains the project description and the requirements engineering checklist that will help you gather the requirements.
As discussed in the labs, every team is formed by 6 students, each one having a different role within the team. The roles are:

- **Client** - Representing the client that will provide the requirements of the project.
- **Team leader** - Responsible for team coordination, GitHub repository creation and management, as well as in charge of creating any additional technical documents.
- **Requirement engineer** - Responsible for requirement elicitation and requirement document creation.
- **2 x Developers** - In charge of converting the requirements in functional code.
- **Tester** - Responsible for making sure that the requirements are met and the software is functional in every step of the development process.

The team allocation has to be chosen internally by every team.

GitHub private repository name should follow the following pattern:

cps406_s[1-5]_group*_w16

s[1-5] is the section that the team belongs (or at least the majority of the team members belong - for mixed teams) and
group* is the number of the group that you have been given.
For example, group number 5 from Section 3 should have the following repository name: **cps406_s3_group5_w16**

Clearly defined in the team's GitHub repository should be:

- team allocation
- project selected
- prefered programming language that the project is going to be developed with

You can define the above in the repository's *README.md* file or a wiki page.

Additionally, every team leader has to add the professor (avm@scs.ryerson.ca) and the TA of their section (meftasadat or SokratisT) in the repository as collaborators - (GitHub usernames in brackets).

Projects Description

1. Innovative Automated Teller Machine (ATM)

This is an example of an ATM.

The ATM has a screen, a card reader, a NFC receiver, a small printer, a cash dispenser and a keyboard. The keyboard has numeric buttons, an enter key, a cancel key and a clear key. On both the left and right of the screen are three buttons that allow selection of any options displayed on the screen. The ATM is connected to the bank over the network.

The ATM provides facilities to:

- Withdraw cash
- Display the current balance.

The user has to:

1. Offer up their card to the reader. The display then asks the user to enter their PIN, via the keyboard. Alternatively the user has also the option to use his cell phone as identification, via NFC.
2. If Pin or NFC is successful, the display presents a set of options.

The system must be highly robust, since it is to be used by untrained bank customers in public places.

2. Patient monitoring system

This is an example of a safety-critical system.

A computer system monitors the vital signs of a patient in a hospital. Sensors attached to a patient continually send (input data stream) information to the computer about:

- Heart rate
- Temperature
- Blood pressure.

Some of the readings require conversion to useful units of measurement (e.g. micro volts into degrees centigrade). The computer displays the information on a screen in X seconds intervals. It also logs the information in a file that can be retrieved and displayed. If any of the vital signs exceed safe limits, the screen flashes a warning and an alarm sounds. The limits have default values, but can also be changed by the operator. If a sensor fails, the screen flashes a warning and the alarm sounds.

Requirements Engineering checklist

- Please keep the total number of use cases between 10 and 20.
- Track each use case in a separate github issue (use ZenHub as well).
- In addition to the issues on GitHub you are required to deliver a document including all the requirements through D2L - only the designated team leader should upload a file.

The outcomes for the Requirements Engineering document include but are not limited to:

- Prioritized or ranked requirements *
- Business scenarios
- Activity Diagrams
- Use case model *
- System functional requirements *
- Non-functional requirements *

* means must be submitted

All your requirements should comply with the following:

Validity

- ➢ Does the system provide the functions which best support the client's needs?

Consistency

- ➢ Are there any requirements conflicts?

Completeness

- ➢ Are all functions required by the customer included?

Realism

- ➢ Can the requirements be implemented given available budget and technology?

Verifiability

- ➢ Can the requirements be tested?

An example of the use cases recorded in GitHub can be found at https://github.com/cps406lab/Introduction-to-git/issues/5

Self validate your requirements

How can I validate/review my requirements?

- Systematically review the requirements as they are being gathered by the requirement engineer.
- Develop tests for each requirement.
- Get all sides involved client/developer/requirement engineer.
- Resolve issues while still at an early stage by doing regular reviews - don't let them accumulate.

Additionally:

- Did you capture dependencies on other requirements for a given requirement (if any)?
- Did you capture non-functional requirements in addition to functional?
- Have you thought of domain constraints?
- Will you be able to implement this requirement within 1-week iteration or should it be split into smaller "line items"?
- Did you prioritize the requirements or ranked them?

Criteria that every requirement should meet:

- Verifiability: Is the requirement realistically testable?
- Comprehensibility: Is the requirement properly understood?
- Traceability: Is the origin of the requirement clearly stated?
- Adaptability: Can the requirement be changed without a large impact on other requirements?

Questions to be asked for each requirement gathered:

- Is it helping in achieving the final goal/objective?
- Is the level notion sufficient - or is it overwhelming?
- Is it fundamental or can it be classified as secondary/supplemental?
- Is it clear?
- Does it conflict with another requirement?
- Will it be testable after its development?
- Does it provide all the necessary information that someone will need to work on it?
- Does it comply with client's requirements?
- Is it achievable (from both technical and knowledge perspective)?