Guideline to PRO1000B
Spring 2019

1. Introduction

1.1. General information

This bachelor-level course PRO1000B Practical Project deals with a project assignment that is mandatory for all IT students. Students will apply the knowledge from first year subjects in the study to implement a practical system development project. Emphasis is placed on the students' ability to reflect on the subject matter, and the ability to make a critical assessment of the connection between theory and practice.

This guideline contains all the necessary information for this course, the assignments (one for each project-group), and a suggested outline for the final project report. This guide aims at helping you complete and document your project as a comprehensive phenomenon. Practical information regarding project-group composition, dates etc. can also be found on Canvas. All students should check this web page for updates. In case of mismatch between information in this compendium, information given during lectures, by email, and on the above web page, the last updated information should be regarded as correct.

1.2. Course Responsible:

Anh Nguyen-Duc. Course PRO1000B responsible, email: angu@usn.no

Further information is found at https://www.usn.no/english/about/contact-us/employees/anh-nguyen-duc-article209091-7531.html

1.3. Group arrangement

Students will be divided in project groups of 6-8 students. Groups will be formed based on expressed interest by students for particular projects, geographical location, or student age and working experience. We also ensure a diverse group of students with regard to, for example, academic performance, location, gender, and nationality. Each group will be allocated a regular supervision from the course responsible. The supervision serves for keeping students on track, and oversees that sufficient contact with the customer is maintained. Therefore, a regular update with meeting and status reports is required. In addition, the students should have several, weekly internal meetings, and regular customer meetings. Overall summary of group assignment, customer and project description is given in Canvas.

1.4. Goal and rationale of PRO1000B

The **goal** of course PRO1000B is to teach *you* and your fellow students – by working in groups – software engineering (SE) and project management (PM) skills in the context of

a *development project* to make a realistic *prototype* of an Information System (IS) *"on contract"* for a real-world customer. Each project group is initially given a one-page *project assignment* from an external customer. All the phases of a typical IS/IT project are covered, e.g. project management and planning, pre-study, requirements, design, programming, testing, evaluation and documentation, but no "maintenance".

Each group will produce a final project report (either in English or Norwegian), and hold a presentation and demonstration of the final *prototyped* product for the customer, while an external examiner (censor) is present.

If a fundamental disagreement with the customer arises, the group has, if needed, "the final word" since the group members gets the credit through a final report worth 7.5 Sp. However, the group and their advisor should do what is possible to resolve any major disagreements. Conflicts are to be explained, negotiated and resolved (managed), as this is part of the real world work. **It is therefore crucial that the group is focused and has a good dialog with the customer.**

1.5. Learning objectives

Three major area of learnings are expected after completing the course: (1) project management, (2) team work and (3) software engineering. Regarding to Project Management, the team should be able to:

- Perform fundamental project planning activities, such as project planning, management of scope, time, cost, effort and risk.
- Risk management You should document all delays, overruns, and weaknesses, so that they can be explained and argumented.
- Execute the plans. This involves actual activities and registration and monitoring of effort and resource usage.
- Close the project. This involves finalizing the product implementation, preparation of final deliveries, i.e. prototype, manual, presentations, etc.
- To present the final prototype to the customer / external examiner. Under the final presentation and demonstration, it is important to give the customer a complete and good impression of the system delivered.

Regarding to teamwork, after the course, the students should be able to:

- Assign, coordinate efforts and distribute work and responsibilities among team members
- Understand the importance as well as challenges of having a good team collaboration
- Make a group decision. The students would experience collaborative working environment. Earlier in your studies, the assignments have been smaller and more well-defined. In this project, there are (conflicting) decisions to be made.
- Perform problem solving skills, i.e. creativity, decision making under uncertainty, resolving conflicts among team members
- Ability to adapt to no-ideal working situations.

Regarding to Software Engineering (Society et al. 2014), after the course, students should be able to:

- Ability to handle difficult customers. They can be unreliable and/or unavailable. They might change directions, come up with new ideas, and have an unclear picture of

what they really want. An important part of this course is to manage the group project, so that the results match the customer's needs, even though the situation may turn difficult.

- Understand the process of requirement explication.
- Document requirements properly, i.e. as user stories, putting them into Sprint Backlogs
- Provide architectural solutions for the given task, visualizing them as architectural diagrams, i.e. use case diagrams, architectural views, database diagrams, workflow diagrams, etc.
- Apply programming skills with both front-end and back-end development into a real project.
- Provide a test plan to ensure the quality of delivered products. The test plan might include test scenarios, test cases and test reports at different levels of testing.
- Document the whole project report. The final project documents must be complete, well structured and target the technical knowledge level of the customer.

1.6. For completing the project

No prior knowledge is required. However, the team will plan, execute and close a software development project. Projects can be carried on with the accumulating progress of the four knowledge area:

1. Software development. Most of the project will be the development of client-server web applications. Not compulsory, but this might include:
    a. Front-end development: HTML5, CSS3 and Javascript
    b. Backend development: NodeJS[1], Python, PhP and C#
    c. User experience: User-centered design principles, Photoshops, Illustrator, etc
2. Project infrastructure. Not compulsory, but we recommend these tools:
    a. Project management tools: Trello[2]
    b. Project communication: Facebook Messenger, Slack[3]
    c. Source code version control: Gitkraken[4], Bitbucket[5]
    d. Document management: Google Doc[6], Dropbox, Google Drive
    e. Process modelling (chart drawing): Visual Paradigm[7]
3. Software engineering principles. The team is encouraged to perform Agile methodology. Scrum (Schwaber et al. 2016) is a preferred option, which offers you several useful team practices:
    a. User story
    b. Product backlog and prioritization
    c. Sprint
    d. Sprint backlog

[1] https://nodejs.org/en/
[2] http://trello.com
[3] https://slack.com
[4] https://www.gitkraken.com/
[5] https://bitbucket.org
[6] https://www.google.com/docs/about/
[7] https://www.visual-paradigm.com/

e. Sprint planning meeting
f. Sprint retrospective meeting

4. Domain knowledge. Each project comes from a customer with different knowledge domains. You might need to do your own research about the domain knowledge or keep close contact with the customers to be able to provide a solution that fits to customers' problems.

1.7. Rating of project work

Project work is a team effort, hence you will receive the same evaluation for every team member of a group. The project work will be evaluated on the basis of :

1. Effectiveness of teamwork. For instance, how well team members work with each other, effectiveness of communication and negotiation with customers, and teams' process improvement efforts.
2. Quality of the project report. For instance, layout and structure readability, quality of architectural diagrams, product backlogs, and test plans. Reasonable grounds for decisions taken, logical flow in the report.
3. The functioning prototype of the system. For instance, whether the group has solved the given assignment, according to the customer's objectives of the project.
4. The final presentation delivered at the end of the course
5. Students' reflections on the project work. For instance, what students learn from the course, how the course can be improved, visibility of limitations imposed.

The project is intended to be conducted as a teamwork effort. This also means that the team dynamics will have an impact on the final grade. The final grade is a composition of the abovementioned components.

The project report, the prototype and the presentation count towards the grade in an integrated way (they are not formally weighted against each other). How the group actually has worked, technical problems, customer behavior and availability, etc. are part of the section of reflections in the report. This section should be about their experiences during the project and reflecting upon what they had learned and how they could have done things differently. The *focus of this part is on the process rather than the product*.

Note that since the presentation counts towards the grade, it is important that you maintain a functioning version of your program in case you (the group) appeal the result (grade). If an appeal is made, you will have to make your presentation for the new examiner, including demonstration of the program.

1.8. Supervision and meetings

Supervision meetings are arranged from Week 7, after all lectures are finished. Each group is expected to have one hour meeting with the supervisor. The meeting will be hold every two weeks. The team should prepare for a Sprint demonstration, status update and raising all team concerns.

Customer meetings are held when needed, starting on the kick-off day. The next customer meetings arranged in dialog with the customer, but the group is responsible for booking a room and other logistics. We recommend taking more contact with the customer, especially verbal communication (face-to-face or virtual meeting).

1.9. Deadlines

Project deliverables to examiners and jury on April 30th via Wiseflow. All code and intermediate material need to be handed-in at this time. Note that there is no printed deliverables in the course. The date for final presentation and demonstration is updated later in Canvas.

2. Adoption of Software Engineering processes and techniques

2.1. Requirements Specifications and Analysis

In the requirements specification phase, it is important to explicitly state the system requirements and link them to the business requirements from the pre-study phase. Typically, requirements are divided into **functional** and **non-functional requirements**. Structure the requirements such that the presentation is well organized.

Some persons like to enumerate requirements (R1, R2 ...), which may create "boring" reading where it is easy to lose track of the content. The advantage with numbering is that it is then easy to separate the requirements from the rest of the text, each becomes explicit, and you achieve traceability and structure.

Before requirements are stabilized, you should give an overview of the software architecture. This means including figures that show how separate modules are related. Prototype-diagrams of the user interfaces are often very helpful for communicating with the customer.

Non-functional requirements should be well understood and documented. Even though the implementation of such requirements are excluded from your project scope, you should be able to document and present the solutions to them in your design.

User story should be used for capturing requirements. Agile introduced the practice of expressing requirements in the form of user stories, short descriptions of functionality–told from the perspective of a user. This directly goes into product backlog and sprint backlogs. An example of user stories:

- FR1: As an admin I would like to add events to a calendar so that everyone can see upcoming events.
- NFR1: As a user, I would like to get to the task board within three clicks since the web is opened.

2.2. Design

Understanding the difference from design and its previous activity (analysis) and post activity (programming). Design is all about getting a vague system description, to a specific and detailed description that can be implemented and realized. A system can be viewed from multiple perspectives. One way to represent design is the 4+1 view model[8] (Kruchten 1995).

You can use a lot of figures! Good figures say more than a thousand words. We strongly recommend making use-case diagrams here, also because we then can make quick and reliable estimates of the ensuing design, programming and test effort. As you can see in the Figure 1, a student group created an activity diagram for a requirement "Uploading document".

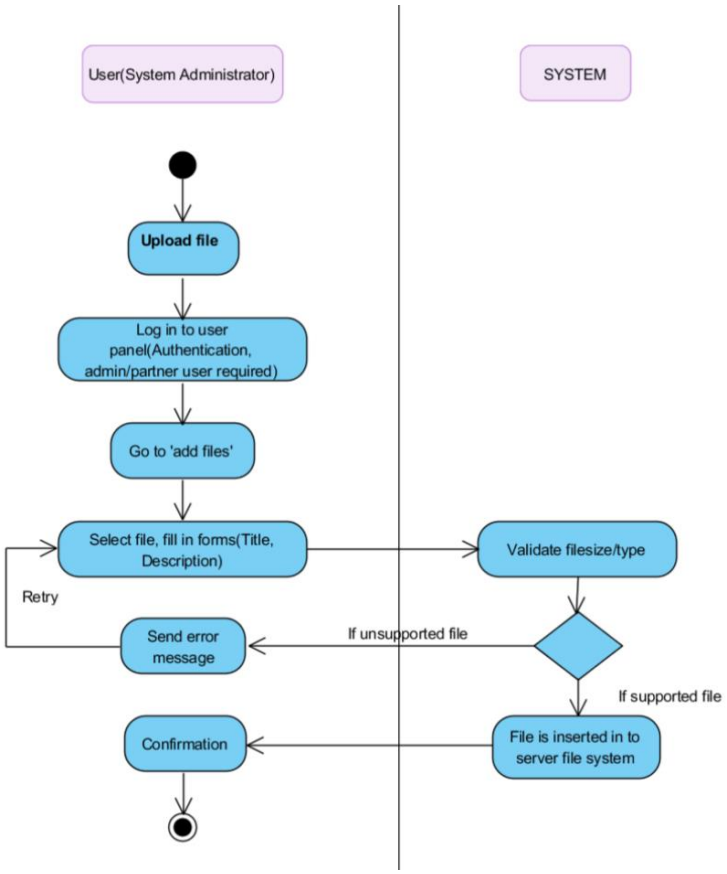


Figure 1: Activity diagram of Uploading document

Or in Figure 2, another student group created an entity-relationship diagram for their database:

[8] https://ieeexplore.ieee.org/document/469759

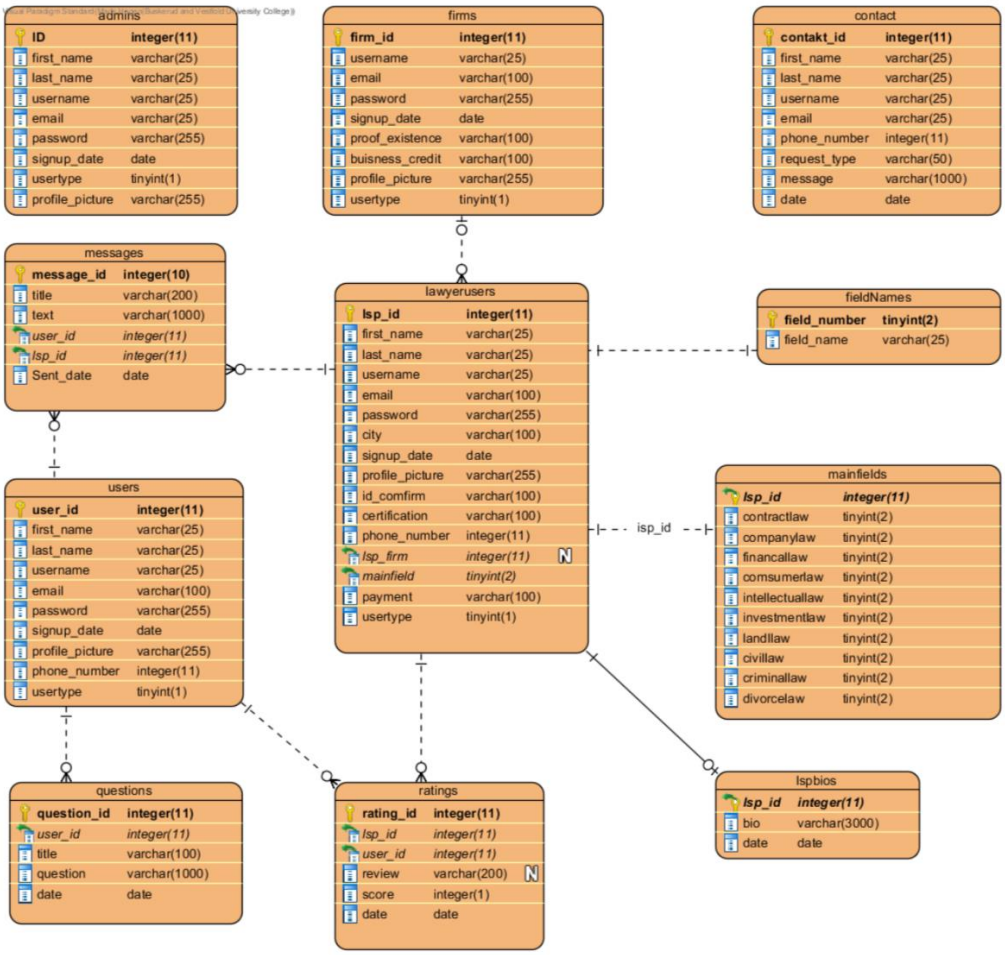


Figure 2: ERD diagram

There are some existing and popular software architecture, which is called software design patterns. Three-tier patterns, for instance, includes User interface, Business or application logic and Database or file system.

UML are useful for describing solutions in the design phase. Regardless of which type of development strategy is chosen, (waterfall, Agile) most software implementation projects start with system architecture and a sketch of the desired design, in order to ease later division into parts. The project most likely will be further developed later, so a "modular" design is to prefer. Designing a modular system also makes testing much more easily, since defects can be tracked down to individual modules.

In the very beginning of the project a lot of important technical decisions should be made. This is vital for the remainder of the project, but there are certain practices that make it all easier.

2.3. Implementation

General knowledge about programming is expected to be covered in previous courses. Depending on the actual project commitment, this project might require that the team members learn new programming language(s), new concepts of programming, various

technical skills etc. The group has to plan how to obtain this knowledge, maybe in cooperation with the customer and the advisor.

Best practices:

- The code you write might be used as a base for further development, and may also be used by the other team members.
- Inside the group, it will also be practical to have common design and code conventions that all group members understand and practice. If the customer has a coding convention, they probably would like you to use it also.
- Please observe that some freeware or trial ware licenses of code editors etc. states that is it prohibited to use them to write code for commercial use. Check the license of the software that you decide to use in the project, and discuss it with the customer if there are such clauses that you might be in conflict with.
- Versioning control and backup tools, such as Git should be used for all technical artefacts (UML-diagrams, code, test data, etc.) and documents in the project. When you set up a development environment, make sure to set up a functional versioning system with backup as well.

2.4. Testing

Testing, or quality assurance, is usually planned and carried out in four parts: (1) Overall test plan, (2) detail test case specification, (3) execution of testing and (4) approval of test results. When you create a test plan, it is important to specify:

- Which tests should be carried out?
- Which tests should contain checklists? (Checklists are most common for entity and module testing)
- Which tests should contain detailed specifications? (Detailed test specifications are common for testing systems, integrations, usability and acceptance)
- Who are the test persons? (Project, customer, others...)
- When should the tests be carried out?

The level of detail should fit the nature of your project.

- The detailed test specifications should contain:
- Test descriptions (the operations that should be carried out)
- Data that will be tested (input and expected output)

Tests carried out by developers/ testers	
Unit test (programming phase)	Testing of the smallest units in the projects, i.e., user interface, methods, stored procedures, objects, classes, etc.
Model test (design phase)	Entities integrated into bigger software components. Modules are tested to assure that the coordination and communication between the entities are as expected.

System test (requirement phase)	All modules that together form a complete version of the system should be tested. The system are tested to assure that the coordination and communication between models are as expected.
Integration test (design phase)	This is a complete test of the system and its interfaces to the world around. The last defects should be found and it should be verified that the system behaves well according to the requirement specifications. In some projects integration and system tests are merged.
Tests carried out by the end users	
Usability tests (non functional)	These are tests that assure that the interaction between users and the system is as expected. The goal is to get user friendly applications.
Acceptance tests (non functional)	Here, the end users should test if the system and its user interface to its environment are as expected. Based on this acceptance test, the management or customer make decision on whether the product should be used or not.

3. Project closure and evaluation

Project closures include major activities:

1. Evaluate if the project delivered the expected benefits to all stakeholders
    - Was the project managed well?
    - Was the customer satisfied?
2. Assess what was done wrong and what contributed to successes
3. Identify changes to improve the delivery of future projects

With in the course, project closure requires customer involvement and feedback, supervisor's feedback, delivery of both Project report and Source code and final presentation.

Project evaluation can be viewed and documents from different perspective:

1. Process perspective: the internal process and results. How have you worked together as a team? Conflicts that arose and how these were handled? Did you reach the project goals?
2. Individual perspective: self-reflection. What did you learn? What have you done well? What have you not done so well? What would you do differently?
3. Team perspective: What did you learn? What have you done well? What have you not done so well? What would you do differently?
4. Project perspective: The customer and the project task: How was the communication with the customer? How did you experience the project assignment?

We will NOT look at the product evaluation as a major input for the project evaluation in the context of these courses. We also expect you to write some thoughts about what can be done for future works.

4. Report guideline

4.1 Report length and format

There is not strict requirement on the length of the report. However, the main content of the report should be less than (**MAXIMUM**) **60** A4 pages (excluding Appendixes). You can provide links to the external materials related to your project. The report should be submitted in the **PDF** format, following the instruction from the Exam office.
The report should not contain:

- Large images that contain unreadable texts. All images inserted in the report need to fit to its layout (either in horizontal or vertical way).
- Images that copy and paste directly from i.e. Trello, Kanban board or screenshots of other Software tools. Images should be exported properly from those tools, or redrawn to fit to the report.

4.2 References

It is important to properly and appropriately cite references in scientific/ technical documents (such as books, websites, blogs, papers) in order to acknowledge your sources and give credit where credit is due. Examples of how citations and references should be used can be seen from this Guideline. There are many different formats for your references. You need to be consistent with one formatting style. See more here:

http://tim.thorpeallen.net/Courses/Reference/Citations.html

We recommend students to use one of the reference management software: Zotero[9] or EndNote[10]. The software is easy to use and offer plug-ins for Microsoft Words.

4.3 Report Structure

The following structure needs to be followed:

Part 0: Meta information

0.1. Name of the group
0.2. Group members
0.3. Report version, created/ last updated date
0.4. Individual contribution breakdown

Part 1: Pro1000B

1. Introduction

[9] http://zotero.org/
[10] https://endnote.com/

1.1. Project background
1.2. Goals and objective
1.3. Project organization
1.3.1. Team roles and description
1.3.2. Stakeholders and user description
1.4. Scope planning (what include and what not include in the project, WBS)
1.5. Time planning (Gantt Chart)
1.6. Risk planning
1.7. Resource planning
1.8. Communication planning

2. Pre-study of the problem space vs. solution space

2.1. Research on the problem spaces
2.2. Project challenges and constraints
2.3. Possible solutions for the problem

3. Software Development Method

3.1. Development Methods selected for the project
3.2. Description of actual uses of practices and techniques
3.3. Description of tools and technology stack

4. Product backlog

4.1. List of functional requirements + estimation
4.2. List of non-functional requirements + estimation

5. Design

5.1. Reason for architectural solutions
5.2. General architecture design (client-server/ n-tier/ cloud)
5.3. Architectural views
5.3.1. Class diagram/ Domain model
5.3.2. Database diagram (at a simple level)
5.3.3. User interface design with screenshots

6. Testing

6.1. Overall test plan.
6.2. Test cases for at least two functional and two non functional feature/ use case/ user story
6.3. Test report: the result of execution the described test cases

7. Sprint report

7.1. Sprint goals
7.2. Sprint backlog items
7.3. Summary of Sprint review meeting
7.4. Summary of Sprint retrospective meeting

8. Project evaluation

8.1. What the team learn from the project
8.2. What can be improved for next year

9. References

Reference

Kruchten, P. B. (1995). The 4+1 View Model of architecture. *IEEE Software*, *12*(6), 42–50. https://doi.org/10.1109/52.469759

A Guide To The Project Management Body Of Knowledge (PMBOK Guides). (2004). Project Management Institute.

Society, I. C., Bourque, P., & Fairley, R. E. (2014). *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0* (3rd ed.). Los Alamitos, CA, USA: IEEE Computer Society Press.

Scharning H. S., Aakre J. (2016). *Prosjekthåndboka 3.0*. Universitetsforlaget AS.

Schwaber K., Sutherland J. (2016) The Scrum Guide™ (Web Version)

Appendixes

A1. Project plan

Recommended content of the project plan ("project directive"):

- **A1.a. Project name**
- **A1.b Customer and organization**
- **A1.c. Background for the project**
- **A1.d. Scope planning:** Work Breakdown structure. If you are doing Agile, you can use Epic and user stories for work breaking down.
- **A1.e. Time planning:** Gantt chart
- **A1.f. Risk planning:** what are risks associated with the project? How severe are they? How likely they can happen
- **A1.g. Resource planning:** how much time should your team and team member spend on the project? how you organize the team effort?
- **A1.h. Communication planning:** how you plan to communicate within your group, with the customer. Which communication tools to use? Which project management tools to use? How to store documents, code etc ...?

A2. Sprint report

Recommended content for the sprint report:

- **A2.a**. **Sprint goals**: A sprint goal is a short, one- or two-sentence, description of what the team plans to achieve during the sprint
- **A2.b. Sprint backlog items**: A sprint backlog is a list of the product backlog items the team commits to delivering plus the list of tasks necessary to delivering those product backlog items. Each task on the sprint backlog is also usually estimated
- **A2.c. Summary of Sprint review meeting**: a short summary of what is shown to the customers and their feedback.
- **A2.d. Summary of Sprint retrospective meeting**: a short summary of what went well during the sprint cycle, what went wrong during the sprint cycle and what could do differently to improve?

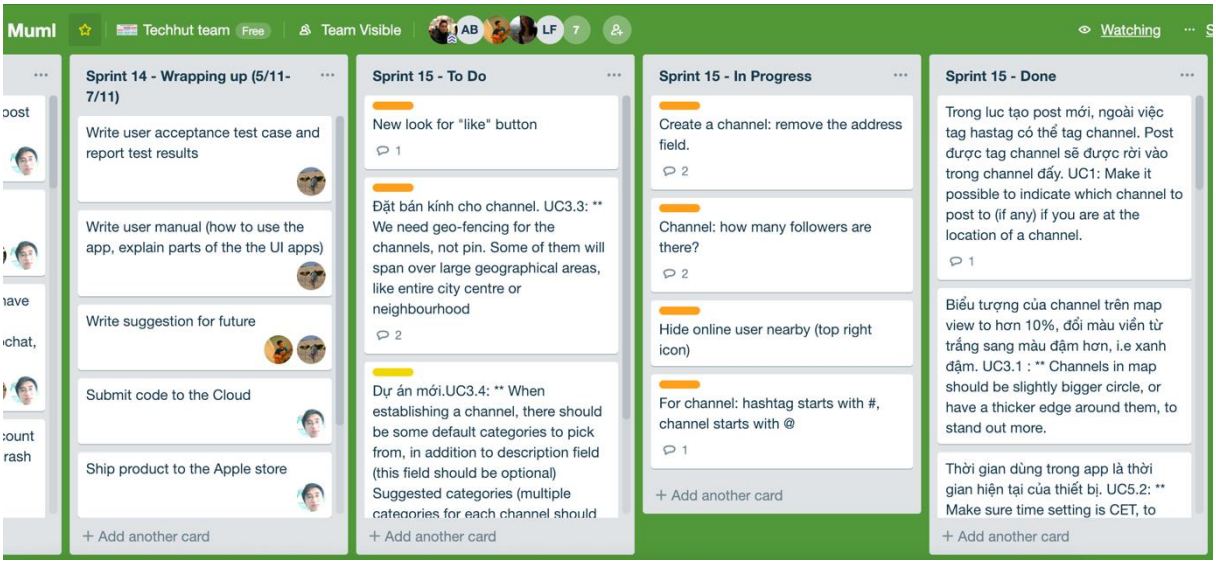
A3. Sprint backlog template

ID	Backlog items (User stories)	Estimated.	Priority
FR01			
FR02			
NFR01			
NFR02			
…			

A4. Risk analysis table

Risk Id	Risk Description	Frequency	Severity	Action plan
R01				
R02				
…				

A5. Kanban board with Trello



Project link: https://trello.com/b/uBnDxhKQ/muml