CS 471: Software Engineering
Spring 2018
Project Assignment (Part 1 of 3) – Product Backlog

Due date: Monday, March 5, 2018 (at the end of the day)

1. Brief Description

This first part of the group project asks you to build the **Product Backlog**. The last two parts of the group project will focus on the first two sprints (see timeline below), and will emphasize other artifacts and engineering activities.

Part 1 of 3	**Part 2 of 3**	**Part 3 of 3**
Product Backlog	Sprint 1	Sprint 2

While this semester we aim at two sprints, in CS 481 we aim at completing four sprints.
All resources referenced in this file are found on Piazza under "Project".
Below are the steps required to complete this project assignment.

2. Read about GitHub and ZenHub conventions

Get familiar with the CS471_GitHubAndZenHubConventions_QuickStartGuide (henceforth referred to as GitHubConventions) document containing the conventions you will need to follow for the group project, until the end of CS481. You will be given access to a private GitHub repository and you will have to install the ZenHub browser (Firefox or Chrome) extension on your machine.

Personalized feedback based on the GitHubConventions will be provided in a Scrum Linter report and the team should fix all the inconsistencies indicated in the report.

3. Decide Scrum Roles

Decide the scrum roles (Product Owner, Scrum Master and Developers). Unlike real-world projects, you can change roles in each sprint, and you are allowed to have more than one role.

- Your **Product Owner** should setup and lead meetings with your sponsor to capture their needs (see below).
- Your **Scrum Master** should enforce the scrum process.
- Each team member should update their name, role and GitHub username via a commit to the README.md file (see GitHubConventions) of the GitHub repository that will be shared with you.

4. Sponsor meetings

Meet with your sponsor and capture **User Stories** that document their needs. You will likely need a series of meetings.

- Record each story using the **Role/Goal/Benefit** template (e.g., "As a **<role>**, I need to **<goal>** in order to **<benefit>**"). See the "Creating Issues" in GitHubConventions for learning about creating users stories with the correct template.
- At this point in your project, you will likely have some **epics** that have not yet been refined into detailed stories. That is ok as long as you have sufficient detailed stories. Please identify which stories you think are epics.
- A **satisfactory Product Backlog** will have at least a dozen detailed stories before beginning your first sprint
- An **exemplary Product Backlog** may have two or even three dozen detailed stories prior

to the first sprint

- Work with your sponsor to assign a **priority** to each User Story
- Exemplary teams contribute their own ideas to the project to help the sponsor understand what is possible. However, the sponsor gets to decide what stories they want and in which order
- The meeting with the sponsors should be documented in the GitHub Wiki page (see GitHubConventions).

5. Definition of Done

Create a **Definition of Done** document as a wiki page on GitHub (see GitHubConventions). In most cases, this will be a one-page document describing what is required for your team to consider a story "**done**" and may include:

- Unit-Level Tests Passing
- Code Reviews (Simple rule for now: If you don't have unit-level tests for some code, then you need reviews for that code)
- Acceptance Tests
- You will later include other activities discussed in the second half of the semester

6. Acceptance Criteria

Augment your User Stories with **Acceptance Criteria (AC)** (see the "Define templates for issues using Saved Replies" in GitHubConventions for a template of creating user stories with AC):

- Create and record each AC using the **Given/When/Then** template (e.g., "**Given** some context, **when** some event arises, **then** the following happens")
- It is not necessary to define AC for epics (as you likely do not know enough about them until you've refined them into detailed stories later in the project)
- Your AC should define the **boundaries/limitations** for a story.
  - **Example:** A story for creating a password will need an AC for each element of your password policy (e.g., minimum length, upper/lower case if required, numeric digits if required, symbols if required, maximum length if enforced, not the same as the user name if prohibited, etc.). You need lots of detail!
- Review your AC with your sponsor. They may modify or add content to your AC.
- A **satisfactory team** will average of about 5 AC for each detailed User Story. Complicated stories (not recommended) will need far more, but you are encouraged to keep your stories simple.
- An **exemplary team** will average 10 or even more AC for each detailed User Story, using the required template.

7. Poker Planning

Use the **Planning Poker** process to estimate each User Story (i.e., update the estimate field in the corresponding GitHub issue). Your estimate should include the time required for each of the following activities that may apply to your team:

- Obtaining, installing and learning any necessary tools, repositories, frameworks or existing code required to implement the story
- Any specification/design work required, especially if it must be coordinated with other team members or teams. For example, if a server implements an API exercised by mobile apps, budget effort to define/review the API specification with its stakeholders.
- The actual implementation of the product code, database tables, CSS/JSON/etc. files, user interface design, or other artifacts
- The quality activities (unit-level tests, acceptance testing, bug fixing) required by your Definition of Done

8. Submission

The artifacts generated on your private GitHub repository (see GitHubConventions) will be considered as your official "submission" (i.e., there is no need to submit anything via Blackboard). The motto for this project will be:

If it's not on GitHub, it does not exist, and you will not get credit for it.

9. Grading Rubric

The maximum points for this assignment is 100, and the points are distributed as follows:

Product Backlog (35 points)	**Points**
Sufficient Number of Detailed User Stories found in the "`Product Backlog (Stories)`" pipeline (see GitHubConventions)	15
All Stories Prioritized	5
All Detailed Stories Estimated	10
Epics explicitly identified	5

User Stories (30 points)	**Points**
Written in customer's business language	5
Conform to the Role/Goal/Benefit Template	10
Appropriate level of detail in each story	15

Acceptance Criteria (AC) (20 points)	**Points**
Each AC linked to a User Story	5
Conform to the Given/When/Then Template	5
Adequate number of AC per User Story	10

Definition of Done (15 points)	**15 Points**

Team Penalties:

- Team did not email the sponsor, within the first week of the project being available, in order to schedule a meeting (-100% of grade)
- Team did not meet with the sponsor (-100% of grade)
- Team did not use the GitHubConventions or team did not fix all the inconsistencies indicated in the Scrum Linter report (up to -100% of the grade)

Individual Penalties:

- Team member was "absent" from the initial planning, did not participate/contribute or got involved (up to -100% of grade)
- Team member did not submit a commit to update the README.md file with their information (-100% of the grade)