# Deliverable 2 Instructions

Start by going through any uncompleted tasks from the prior deliverable, and assign them with new due dates. Then go through the feedback on the prior deliverables and create & assign tasks for any unaddressed comments/feedback, both comments on documents and also in the overall feedback on Brightspace. If that’s not available yet, create a task for that by copy-pasting this paragraph.

***Bolded text*** *inside the rubric is used to show any points off or other comments.*

*For example, if “****\*2 points: description of expectation\*****” is bolded, that means 2 points were deducted for being not present.*

## 

## Product Backlog ( /2.5)

Use [this template](https://docs.google.com/spreadsheets/d/1422Rr_2YRKFydVoz7m8bNor666bmfIwj/edit?usp=sharing&ouid=108788083057609121398&rtpof=true&sd=true)

Example: [Team C 2022](https://docs.google.com/spreadsheets/d/14DN41vuV5kV_J6hwGp40sGkVDdGYiAt5/edit#gid=944386808)

* Product backlog and sprint backlog should match i.e. product backlog items in the sprint backlog need to have the same IDs etc.
* If they do not match deduct -2 points.
* Make sure that all of your user stories are in the product backlog. For any missing user story deduct -1 point.
* All columns based on the projects’ details document should be in the product backlog and must be filled out. For any missing column or missing information deduct -2 points.

## 

## 

## Sprint Backlog 2 (/2.5)

Use [this template](https://docs.google.com/spreadsheets/d/18AXZZxBCle6GE3ufvEgtNAKRVHvWBjZV/edit?usp=sharing&ouid=108788083057609121398&rtpof=true&sd=true)

Example: [Team C 2022](https://docs.google.com/spreadsheets/d/19WeMzVr-R33PSxbN50U8Z6GwOdp9VWvh/edit#gid=869056032)

* Should reflect the product backlog. That is, all the user stories that are assigned to Sprint 1 in the product backlog, should be shown in the Sprint Backlog 1.
* For any missing user story, deduct -2 points.
* If the Sprint Backlog 1 does not reflect the product backlog, deduct -5 points.
* You must have other technical tasks in your backlog You must have other tasks in the backlog, like documentation, requirements, UI mockups, etc. As covered in class sessions, these are called non-technical stories.

## 

## Github (/5)

* Contributions should be consistent and all students contribute.
* If a student does not contribute to Github, the student will be graded as 0. You can each commit different files, or make changes to the README.

## 

## Kanban (/10)

* Kanban board should show progress (on your issue/task management platform e.g. Github Issues + Github Projects, Jetbrains Space, etc)
* If no progress is shown, the grade for this part will be 0.
* If no link to the project management platform showing the Kanban is submitted, 0 and it’s not resubmittable for regrade.
* There should be 20-40 tasks for current Deliverable and prior deliverable revisions, each task should be small.
  + If < 20 tasks, -2.5 points
  + If any tasks are nonsensical or made just to get to 20, -10 points
  + If there are more than 1-2 tasks with the entire team assigned to them, -2.5 points for each
  + Each task should have the grading feedback/rubric information inside the task, for that task. The task assignee should have everything they need to know to do the task well, by just looking at the task. (Also for the quality checking for the task). -2 for each task missing this

## Sprint Review 2 (/10)

Should follow the [template](https://docs.google.com/document/d/1QkzW3zqLNUdav9vv93DhmefqulpcLdep/edit?usp=sharing&ouid=108788083057609121398&rtpof=true&sd=true).

Examples:

* [Team C 2022](https://docs.google.com/document/u/0/d/1oKKcyWRzvgNYZk8482eMJBYPHfwSkYf4/edit?fromCopy=true)
* [Team F 2022](https://docs.google.com/document/d/1Oc4fAFQFwNm4flCDaYdD9mBWpcdTX_nz/edit) (from end of project so all sprint reviews are here)
* This document is generally 1-2 pages.
* Should be written for this past sprint / week (i.e. for Deliverable 1, Deliverable 1 and Revisions for Deliverable 0)
* Use the template as an agenda for the sprint review meeting. Step through each point as a team, have a discussion and have someone take notes. The team should review the notes during the meeting in a shared Google Doc to make sure everyone is heard, and anyone can edit the notes.
* For any missing item from the template, deduct 2 points.
* Document should contain all prior Sprint Reviews as well, with the most recent at the top (the first sprint review happens in Deliverable 1)

## 

## 

## SRS ( /15)

Examples:

* [Team E 2022](https://docs.google.com/document/d/102TYJwTMRb3uJG8bkk1XLBtnxU4dJYYT/edit)
* [Team F 2022](https://docs.google.com/document/d/1U4bD9EDc3W9SLv9QI0Lev2ebVLBtD7JO/edit)

Example Mockups: [Team C 2022](https://miro.com/app/board/uXjVNaxNZpA=/?share_link_id=773411667516)

* Need to check if the comments from Deliverable 1 have been addressed.
* For any comment that is not addressed (every single one), deduct -2 points.
* Fill in the rest of the sections, using information from the project description document.
  + See [this example](https://docs.google.com/document/d/1WkqFkOZBiUtC19x8w7Zt7qIg8MJishZp/edit?usp=sharing&ouid=108788083057609121398&rtpof=true&sd=true) for an example of the minimum level of detail required.
  + Unlike this example, your section 4 should be organized into subsections, grouping by general feature area or top-level use cases (i.e. don’t have a sub section for every single use case that is an extension, or instance of)
* Has about 15 – 30 functional requirements
  + If the number of FR is below 15, -10 points
* Has about 10 – 15 non-functional requirements.
  + If the number of NFR is below 10, -10 points
* For each wrong requirement deduct -2 points.
* For each conflicting/contradictory requirement, deduct -2 points.
* If the UI is missing deduct -10 points.
* If the UI is not updated, deduct -5 points.
* Use cases are required in a separate document.

## 

## 

## Architecture design (/15)

*Developers lead this*

Examples:

* [Team C 2022](https://docs.google.com/document/d/1OG6txMwHzCfog6YLM_vxo4DIkhL-nvrx/edit)
* [*Team F 2022*](https://docs.google.com/document/d/1xVi9zW4GadxJmQ8z1thv0pwIRkUIy-ZB3q2WFEO9JGQ/edit#heading=h.s6v4xgruuay5) *(this has more detail and is closer to a detailed technical design, you don’t need to do this much)*
* This will most likely be a MVC architecture. You can use the examples or [this as a template](https://docs.google.com/document/d/1S7R3Tt6wKMN1nIgka4z8HZBGtxmpXdrA/edit), but your team will need to fill in your specific technologies and other information specific for your app. The example is for an online shopping system.
* A UML package diagram showing the system architecture, where the basic classes/system entities are located, and associations.
  + If the classes, associations, system architecture are not properly shown, deduct -7.5 points.
  + The name of the classes, packages, etc should match the application under the development. If not, deduct -2 points for each mismatch.
* The description of the architecture and the justification of the chosen architecture.
  + If the description is missing or does not make sense, deduct -7.5 points.
  + If the reason to choose this specific architecture does not make sense, deduct -7.5 points.

## 

## 

## Use Case Diagrams/Models and Descriptions ( /30)

Use this [use case description](https://docs.google.com/document/d/1jBaLHNLAAEPAqN7BCqZspq04X625YcIY/edit?usp=sharing&ouid=108788083057609121398&rtpof=true&sd=true) template.

Examples:

* [UMaine Connect](https://drive.google.com/file/d/1_r0T_X7xg1zB9Vnt0CAREqIjXAOmi0MV/view?usp=sharing)
* [Team C 2022](https://docs.google.com/document/u/0/d/1HsytILdXk_8f-2vJjSbXwhj9OYWIcvVP/edit?fromCopy=true)
* [*Team F 2022*](https://docs.google.com/document/d/1Y8GUXKtYVYOHeUMbw9VIdRacyOoaIGko/edit#heading=h.dj302yy7ur3p) *(some errors in arrows but overall good)*
* Make the use case diagrams first, then assign individual people to making each use case’s description.
* About 10 - 20 use case descriptions and 2-4 use case diagrams/models each including 3-5 use cases.
  + If the number of use case descriptions (UCD) is below 10, deduct -3 points for any one missing. For example, if there are only 8 UCD, then deduct -6 points.
  + If the number of use case diagrams/models is less than 2, deduct -10 points.
* Use case diagrams/models should be correct and have the correct links.
  + If they have unnecessary arrows, deduct -2 points per each mistake.
  + If they are complicated, deduct -5 points per each diagram/model.
  + Use cases should start from the actor. If not, deduct -2 points for each mistake.
  + Only have include, extend and generalize links between the use cases.
    - If the links are not correct or they do not have a type, deduct -2 points for each mistake.

* + Use case descriptions should follow the templates/guidelines from slides or the given template.
    - If they do not match, deduct -5 points for each mistake.
  + The name of the UCD should match with the use cases in the diagram.
    - If they do not match, deduct -1 point per each mistake.
  + The related use cases should be shown correctly in the UCD
    - If they do not show correctly, deduct -1 point per each mistake.
  + For every other mistake, deduct -1 points for each.
* If the quality of the UCD or diagrams varies substantially, deduct -10 for the team not internally reviewing/quality checking their work before submission. Not recoverable with regrade.

New version of the application (/10)

* The implementation should have been started. It is recommended to begin with the UI and some classes such as sign-up, login, etc. The developers should lead this and are expected to, as needed, review/complete parts of the [Interactive React Textbook](https://greglnelson.github.io/react-hooks-typescript-tome/) earlier than other team members, do self-directed learning using Google, StackOverflow, other online resources, and ask people or other classmates questions.
* If there is no code/HTML/scripts, and no changes from last time, this part will be graded as 0. As a reminder, review the class policies about copyright and AI use. You can even use code from prior projects to help start your project.
* Setting up the build environments and the necessary dependencies.
  + If there is nothing, deduct -5 points.
* Your README file should detail the installation and build process for your application. Make sure it works on each person’s machine.
  + If there is nothing, deduct -5 points.