# 

[**Design your team 1**](#_vem877q8yih3)

[**Deliverable 0 3**](#_7eb9m1eu5hdw)

[Student-Run Discord 4](#_fy02y8e6kzxu)

[Social Media (Yik-Yak, Facebook, etc.) 4](#_weyje0n4mi80)

[Various Help Labs on Campus (Math Lab, CS Lab, Physics Lab) 4](#_us5shzcwp6ti)

[**Deliverable 1 7**](#_on0ipjfvkgf3)

[**Deliverable 2 13**](#_u804irpazcab)

[D2 feedback on NMD part: 23](#_w7q5qlmd45s3)

[**Deliverable 3 26**](#_ulncrpdvn5v)

[Rubric common between two classes: 27](#_rbvolnwrq6sa)

[NMD rubric (they lead the mockups and refining the mockups): 30](#_yq9ekkdlqmqi)

[COS rubric (they lead the technical design work and initial coding): 31](#_imlxa7ijewdq)

[**Deliverable 4 36**](#_b6k9rxx48uqb)

[**Deliverable 5 36**](#_nfrch2j3c1es)

[**Deliverable 6 36**](#_5gdynpz5gvg3)

# 

# Design your team

**Good job overall! Revision/regrade: 85%\*1.15points back, rounded up to +1 from 3.75. A good first revision and mostly addressing comments was done, so you've got these points. But see the important note on partially addressed feedback below. (No more revisions on this part are asked for, revision regrades are one resubmit)**

(Warning/note: You're generally going to get lower than normal grades for the first submission for each project assignment. Once you revise you're looking at a 4.7/5 or so grade. This can be a little scary but by responding to feedback with revisions you'll make it back and probably end up getting higher grades overall)

* 0.5 points for listing your team members' names and roles
* 0.5 points for your team name and a coherent mission statement
* 3 points for a clear, concise, logically sound problem statement that defines:
* the problem being solved
* its consequences in the world
* the things that cause the problem
* and why existing solutions don't resolve the problem
* Points will be deducted for problem statements that are unclear, verbose, logically flawed, or present a solution.
  + Feedback not in google doc b/c wasn't shared with edit privileges - 0 for next deliverable parts that do that
  + Statement 1 - great core insight that student taking initiative is a problem. Also I'm floored there are tools for this for student athletes but not all students, your team could on the side advocate for that maybe. Could also just build a simple solution for this, but probably a better problem exists to solve. But the "simplicity" is the right level.  
    - **[fixed]**-.3 ends with many different how might we statements, instead of one. "course assignment load" is incoherent/disconnected from rest of statement so needs more on that causality if to be included
  + Problem Statement 2: **[mostly fixed, overall good revisions, generally a good attempt to address was made so you've got these points. But be careful** [**comment1**](https://docs.google.com/document/d/1GwzlO_-GYO1v2LLw5XCmA1Mn4_k9GI7fw_GHJgIXEQY/edit?disco=AAAA6dLVU-0)[**comment2**](https://docs.google.com/document/d/1GwzlO_-GYO1v2LLw5XCmA1Mn4_k9GI7fw_GHJgIXEQY/edit?disco=AAAA6dLVU-4) **got lost I think because some text got deleted, be sure to reply to those or keep them by keeping like a space character the comment is on. You don’t need to fix it now for grading purposes but comments like this should be integrated or responded to. The substance of this** [**comment2**](https://docs.google.com/document/d/1GwzlO_-GYO1v2LLw5XCmA1Mn4_k9GI7fw_GHJgIXEQY/edit?disco=AAAA6dLVU-4) **did get addressed in the ending sentence at least, and somewhat addressed in the part the comment was on in “Many opportunities for social connections were disrupted by the appearance of COVID-19 and there have not been added processes to rebuild that”; at the same time, the soundness issue remained in the final version so -.1 ]**-.7 this statement is a good start but doesn't get to the real causality, which is that particular students or groups ran these things but didn't have a transition process for people graduating, or faculty to help step in if that transition process failed. So the real "why" behind the problem didn't get there. A lack of something is not the cause of the problem, what led to the lack of something? what positively has to exist to make those somethings happen?
  + great job on not being overly verbose. there is more irrelevant detail here
  + ending goes all over the place so doesn't make this a problem statement. it is far better to end like this though than proposing a random solution or a solution with a bunch of features that don't relate to the causality you discussed before though!

**[fixed, very good policies]**-.25 Missing Communication policies / if you don't follow instructions on where to put them, submit a direct link to where they are

# Deliverable 0

**Revision/regrade: Much improved, some points well addressed, some partially addressed. Can improve on logistics on following up on your comments, which were partly my fault. Definitely follow up with me again if I don't respond to a question after a day or two! Let me know if you did that and I somehow missed it!**

Missed or I didn’t see messages on this? There’s a lot of great ideas here in Deliverable 0! The project description has fallen into a common pattern in problem solving, which is to focus on a solution, and kind of work your way back to the problem. This leads to varying causality and features being included that sound abstractly like a good idea, and can be, but it doesn't help make sure you're addressing the most important causes. **Fix your problem statement, then come back to me, then you'll be set up to add the critical parts of those features** and how they address the problem causlity, and make sure you're addressing the most important causes.”

Project Description Document ( 47.2/50):

* Group Name and Team Members - 5/5
* App Name and Category -5 /5
* General Overview and description - 8.7/10
  + Overall this is good! There’s a few key details missing for making explicit how problem causality is addressed “one level deeper”. Mostly at the insight level, what will these useful and insightful insights be, I’m so curious to see where you go!
  + Could use some more details on the feedback receiving end, e.g. the task flow should have the feedback going somewhere and then information about it flowing back
* Similar app’s descriptions and comparison (three apps minimum) - 28.5/30  
  + Each comparison is 10 points. See table below, 2.5 points per part
  + Students need to explain the similarities of the applications as well as how their application differs from them.
  + Students need to outline the unique features of the group’s applications.
  + The comparison for each similar app will be about one paragraph (half a page).

|  | **Student-Run Discord** | **Social Media (Yik-Yak, Facebook, etc.)** | **Various Help Labs on Campus (Math Lab, CS Lab, Physics Lab)** |
| --- | --- | --- | --- |
| Explain the similarities of the applications | yes | yes | yes |
| Outline the differences / unique features of their team’s app | yes | Yes “Yik-Yak is based on a general area, however and not made specifically for class and talking about a specific class/major will prove to be difficult.” | yes |
| Explain why your app is different in that way | Yes “to provide students who are struggling with support, mentorship, and most importantly, empathy.” | “help ease the anxieties of imposter syndrome by connecting with others” | -.5 Explains but jumps from help to “concerns”, loses clarity. The differences and why explanation are about making help more accessible asynchronously and avoid getting discouraged. But it then jumps to voicing concerns. One way to do both is to include the “get help” and also “getting root cause help” or “things that would help that are outside their control” |
| Comparison for each prior solution at least one paragraph | Yes | Yes | -1 needed a couple more sentences |
| Task flow diagram present and not incorrect | Reasonable (black on grey nodes hard to read) | Good, nice and simple | Great! The embarrassed part is great, that’s including the problem causality well |

User story document - 50/50

* About two pages or at least 15 – 20 user stories.
* Each user story follows the structure given in the lecture slides.
* For each user story that does not follow the style, deduct -2 points.  
  + As a *< user, role>*, I want *< feature, functionality>* so that *< some reason/benefit / value >*
  + When the benefit is reasonably obvious, it does not need to be stated.
  + When the benefit is circular or imprecise, up to -1 point, e.g. As a user, I want to upload my profile picture to see myself.
  + When the feature is unclear or imprecise, up to -1 point, e.g. As a user, I want to open a new file.
  + **Great job doing the edits in the document, made it easier to see what changed**
  + **Seems to be missing user stories for reading feedback, taking actions on feedback, basically the admin/prof side. Seems to be focusing on features for like a better anonymous student Discord**

# Deliverable 1

Hi Team A! You’ll do fine on revisions after fixing the below, your revised grade should be a 9.6/10 once you make the fixes below. Great job overall.  
  
**Feedback is in bold below, if a section doesn’t say how regrade points will be given, it means to edit the document to incorporate the feedback.**

**Do @ mention me in order for me to get notifications on comments, when responding to comments or asking questions on Google Docs. (No penalty for this, but do @ me next time)**

**Note: So I can give better feedback, please avoid cutting and pasting in revised versions of a section into a google doc from another doc editor (or from the google doc itself) - instead, make the edits in the google doc directly. Otherwise, it mangles the edit history and I can’t easily see what has changed, which causes me to lose time I would like to use for giving feedback.**

**Summary: (make issues/tasks for these)**

1. **Needs some more on what insights will be generated, what are the types of insights needed to help** 
   1. **students overcome the negative feedback loop**
   2. **feedback receivers**
2. **Story priority - have your designers set these by user value, coherence for a minimum viable solution**
3. **Focus the mockups on the core features, and having the emotional support properties, and other key causality/properties. Make the 3 examples (include these somehow concretely in your mockups)** [**here**](https://docs.google.com/document/d/1fxrl4o1tdInE3tFvIu5DXQxTnZmhgBSOhRQXIoC3uwI/edit?disco=AAAA8KT6rCc)
4. **I'm not sure if you're trying to allow student / prof / admin to give feedback to any of student / prof / admin? Clarify this**
5. **There’s a lot of similarities to an anonymous discord server where students talk with each other, with some form templates for messages. In your user research you should look for secondary research on anonymous student groups for ideas on features and prioritizing features, and to think about the potential positive and negative effects of your design. Or you could do some more secondary research on student feedback systems if you want to design a system for that.**
6. **There’s a lot of work done in group therapy research on how to have a group of people grow together, looking into that would be beneficial. One easy high signal way to do that is to ask any experts on campus to have a 15 minute conversation where they distill the lessons from that that might be relevant for the design of your app. This is just a longer term user research / design task.**

Updated Project Description Document ( 5/5)

* Based on the comments from deliverable 0.
* If the comments were not addressed, the document will be marked as 0.
* If the comments are partially addressed, the document will be marked as 2.5.
* You must document a changelog by replying to the Google document comments when you think an issue has been addressed. You must also reply to comments saying what you didn’t address.

Updated User story document ( 5/5)

* Based on the comments from deliverable 0.
* If the comments were not addressed, the document will be marked as 0.
* If the comments are partially addressed, the document will be marked as 2.5.
* You must document a changelog by replying to the Google document comments when you think an issue has been addressed. You must also reply to comments saying what you didn’t address.

Product Backlog ( 8/10)

* Use [this template](https://docs.google.com/spreadsheets/d/1422Rr_2YRKFydVoz7m8bNor666bmfIwj/edit?usp=sharing&ouid=108788083057609121398&rtpof=true&sd=true).
* 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.
  + **-2 Missing story priority, you want to have the priority for implementing the minimum viable solution, the highest user value parts. These can change but need to get your start.**
* **Points off here will be regraded as fixed if the Product Backlog is edited to fix the issues above.**

Sprint Backlog 1 (6/10)

* Use [this template](https://docs.google.com/spreadsheets/d/18AXZZxBCle6GE3ufvEgtNAKRVHvWBjZV/edit?usp=sharing&ouid=108788083057609121398&rtpof=true&sd=true)
* 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 should have other technical tasks in your backlog
* You should also have other tasks in the backlog, **like documentation, requirements, UI mockups, etc. As covered in class sessions, these are called non-technical stories.**
  + **-2 need other tasks**
  + [**Seems like these are here a bit under story tasks**](https://docs.google.com/document/d/1a4bQlN0szPqVtlyXOhOtP2QOLeH2sWe6vxvGgWUQpq4/edit)
  + **Don’t need to have all the tasks, just the ones for the sprint that you planned to do (and perhaps completed)**
  + **See the sprint backlog example from deliverable 2 rubric**
* 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.
  + **Name and contribution percentage are missing, will need this for D2**
    - **Don’t need to go back in time and add these, just do it going forward**
    - **If there’s a way to generate an equivalent report in your task management system, such that it ties back to the story level, that would be fine also. But it needs to be actual work, not who was assigned it.**
* **Points off here will be regraded as fixed if the Backlog for D2 doesn’t have these issues.**

Github (2.5/5)

* 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. For this deliverable, any people unfamiliar with Github don’t need to make commits yet.
* **-2.5 I see Andrew and Kevin, should have had also Sam M and Samson C. Next deliverable Designers can commit some mockup files.**
* **Points off here will be regraded as fixed if D2 doesn’t have these issues.**

Kanban (5/5)

* Kanban board should be created and shared with the group (on your issue/task management platform e.g. Github Issues + Github Projects, Jetbrains Space, etc)
* Kanban board should show progress and activity (i.e. be updated and used).
* If no progress is shown, the grade for this part will be 0.
* **Points off here will be regraded as fixed if D2 doesn’t have these issues.**
* **Looks okay but for most groups I mention, something to keep in mind:**
  + **Add quality checking tasks and tasks for integrating the deliverables together, with enough time to assign tasks to fix that before submission.**
  + **That integration and quality checking work can be done by people who are not the PM, otherwise it’s too much work for the PM.**

Sprint Review (10/10)

* Should follow the [template](https://docs.google.com/document/d/1QkzW3zqLNUdav9vv93DhmefqulpcLdep/edit?usp=sharing&ouid=108788083057609121398&rtpof=true&sd=true).
* 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)
* **Points off here will be regraded as fixed if D2 doesn’t have these issues.**
* **“*Determining what is “good enough” is still very difficult for our team.”*** Andrew has started addressing this a bit with questions on D2 on Discord. @’ing me in google doc comments is another good method.

SRS ( 22/40)

* Has about 15 – 30 functional requirements   
  + If the number of FR is below 15, -5 points
  + Put them in Section 4
* Has about 10 – 15 non-functional requirements.   
  + If the number of NFR is below 10, -5 points
  + Put them in Section 5
* Begin the requirements by selecting a few of the features that you would like to develop first and write the requirements for them. Continue with the next features if you still need to have more requirements.
* For each wrong requirement deduct -2 points.
* For each conflicting/contradictory requirement, deduct -2 points.
* If there are no UI mockups, deduct -20 points.   
  + This is here so you get started. Put these in Section 3.1
* Use cases are not required.
* Other sections will be filled in in the next deliverable. Here’s an example for the eventual **Deliverable 2** [level of detail](https://docs.google.com/document/d/1WkqFkOZBiUtC19x8w7Zt7qIg8MJishZp/edit?usp=sharing&ouid=108788083057609121398&rtpof=true&sd=true).
* **See Google doc comments, this amount of needed revision is quite normal**
* **I’ve bolded done NFR in the NFR document,** [Non Functional Requirements](https://docs.google.com/document/d/1aOgIL0o7vSpzT5GZXFSTjAPFGBnHsUZVCiJS6EuPK5A/edit)
* **I think there are around 10-15 good / nearly good NFR, just focus on getting to the # asked for in D2**
* **A lot of these are actually FR** [Non Functional Requirements](https://docs.google.com/document/d/1aOgIL0o7vSpzT5GZXFSTjAPFGBnHsUZVCiJS6EuPK5A/edit) **e.g. a monitoring screen is an FR. Implementing some specific external monitoring tool like New Relic or Google Analytics would be an NFR.**

Software (10/10)

* The implementation should have been started.   
  + If there is no code/HTML/scripts in Github, this part will be graded as 0.
  + **Good start, need to have your own app next time, it can be a super simple homepage. You can use create-react-app**
* Setting up the build environments and the necessary dependencies.   
  + You do not need to have any significant progress on developing your application. Focus on setting up the build environment.You can use create-react-app or otherwise create your project structure.
* Your README file should detail the installation and build process for your application. Make sure it works on each person’s machine.  
  + Submit screenshot(s) or a single video showing the app running on each person’s machine
  + If there is nothing, deduct -5 points.

# Deliverable 2

# Grading comments on the rubric are written like this in a different font and color. Thanks for the suggestions on making this easier to read.

There is one round of revisions for each deliverable, so for D1 that is done and all points final there (barring any mistakes I made which you can point out by commenting on them). The formula is 85% of “points back”. For D2 this will be a special one time 100% of “points back”. [Here’s more on how to read the revisions regrade comments](https://docs.google.com/document/d/1Yc1O_8Tk-sjuLXQuaGip43dHEd7GQuQBJ98p4eSn88k/edit) (this is also repeated at the end of this green text so you read it).

For each Deliverable, I aggregate at the top of the rubric any high level issues/tasks for the revisions or next deliverable at the top of the rubric feedback here. Any points to address related to earlier deliverables are detailed/called out here or in the rubric below in green text; you don’t need to make tasks to look for missing ones from me or look back over the prior deliverable’s revisions grading. To make it clear I have labeled tasks to do as [task].

Great job overall for D2! Normal issues with use cases; with revisions you can get to 100% for D2.

Tasks:

(earlier tasks from the use case diagrams and descriptions grading below) **1. [task] Move the use case descriptions into the template from the rubric / class exercises. The main missing part is the user steps, system response. If you want to just paste in the tables for that for each use case, that is enough. The lack of relationships between use cases is perfect, don’t add relationships like includes etc.**

**2. [task] Make tasks for each person with which use cases they will make user step, system response for.**

(new tasks from grading other parts)  
3.[task] add any libraries you are planning to use into the architecture design document in the design description section

4. [task] not so much a task but I don’t see any design sequence diagrams, though there’s plenty of time to still do those. Looking at the tasks on the kanban, this task should be for the design sequence diagrams <https://bemental.jetbrains.space/p/main/issues/113> but it’s not very clear just looking at the task itself.

5. Reminder: All backlogs, deliverable documents in general need to be the same document, do not create new versions by creating a new document and copy pasting into it.

(Earlier grading feedback on use case descriptions and diagrams:

(I’m prioritizing giving feedback on use cases for all the groups before finishing. There will be revisions regrading with the one time 100% of points back rather than usual 85%, at grading of D3. I’ll grade the rest of Deliverable 2 and D1 revisions after I give feedback on use cases for all the teams.)

Excited to see your project coming together! Good progress! The use cases chosen were good, the parts done were done well, and the diagram - description correspondence quite good also! Most points are from the UCDs missing the user step, system response tables. And some slight description and diagrams not matching. The UCD steps should be more detailed, e.g. specifying what data is filled in or shown - you may also reference mockups that have that detail instead of retyping it (i.e. “system displays this screen <direct link to mockup object>” “User fills in elements on form in <direct link to mockup object>”) Focus on adding that detail. EG any form/input validation, options that might not be listed on mockups, etc. Make sure to fix the UCD steps before making your sequence diagrams!

Suggestions:

1. Add a use case for searching feedback and comments using text search
2. After people make user step, system response tables, get a round of feedback from me on them.

)

**How to read the D1 revisions grading:**  
1. I left comments on the rubric feedback in this document. They start D# Revisions: ... A comment of “2.5 back” or +2.5 means 2.5 points back. If no number is mentioned, no points were taken or given.

2. For google docs comments on the deliverable documents with points, here’s how to interpret the revisions regrading (examples below):

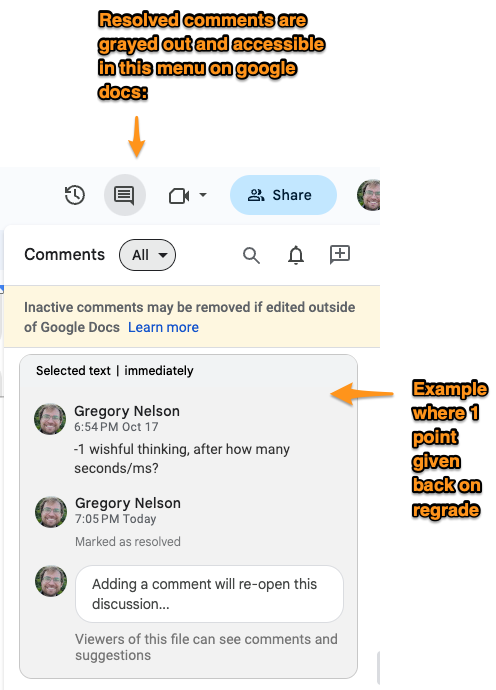
2.1 If a comment was resolved and I left it resolved, it was fixed and full points back.

2.2 If a comment was partially resolved, I left it open or reopened it and left a comment with partial points back.

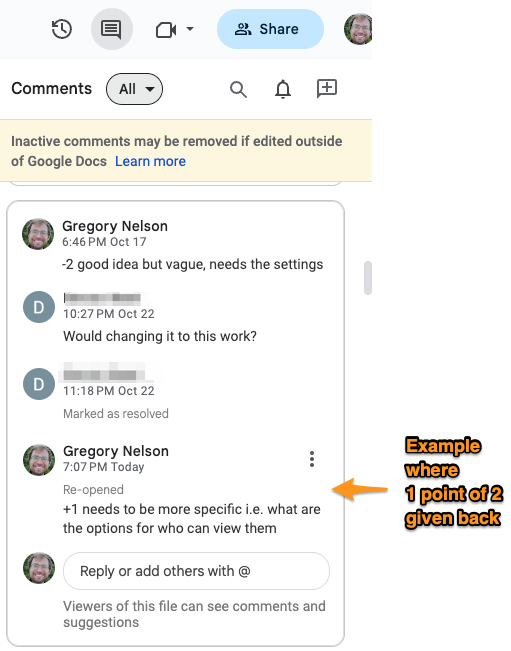
2.3 If a comment was left open/unresolved or I marked it unresolved without replying to it, it was not fixed and no points back.

**Examples:**

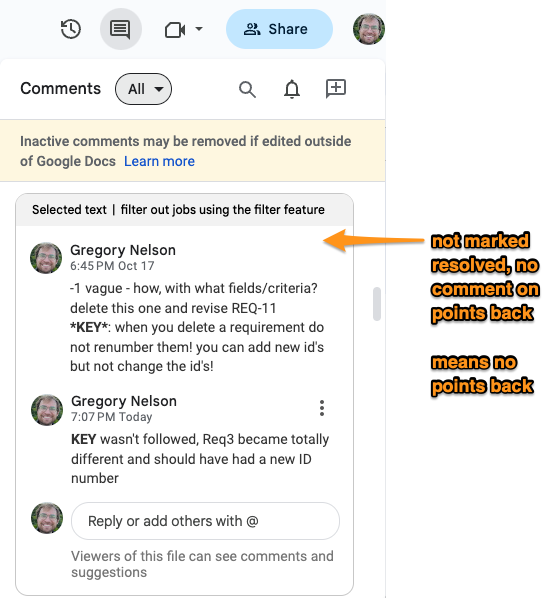
**2.1 If a comment was resolved and I left it resolved, it was fixed and full points back.**



2.2 If a comment was partially resolved, I left it open or reopened it and left a comment with partial points back.



2.3 If a comment was left open/unresolved or I marked it unresolved without replying to it, it was not fixed and no points back.



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 above. If that’s not available yet, create a task for that by copy-pasting this paragraph.

Product Backlog (2.5/2.5)

* 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 ( 2.5/2.5)

* 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 (5/5)

* 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 (10/10)

* 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
  + **Points off here will be regraded as fixed if the D4 Kanban has these fixes made.**
  + **DO NOT WASTE TIME REVISING YOUR D2 KANBAN TASKS. As per the sentence above, points for revisions will be given by doing your D4 kanban better.**

Sprint Review 2 (10/10)

* 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)
* Should follow the [template](https://docs.google.com/document/d/1QkzW3zqLNUdav9vv93DhmefqulpcLdep/edit?usp=sharing&ouid=108788083057609121398&rtpof=true&sd=true).
* 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 ( 15/15)

* 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 ( 15/15)

* Developers lead this
* Examples: [Team C 2022](https://docs.google.com/document/d/1OG6txMwHzCfog6YLM_vxo4DIkhL-nvrx/edit) and [*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.

**(I’m prioritizing giving feedback on use cases for all the groups before finishing. There will be revisions regrading with the one time 100% of points back rather than usual 85%, at grading of D3. I’ll grade the rest of Deliverable 2 and D1 revisions after I give feedback on use cases for all the teams.)**

**Excited to see your project coming together! Good progress! The use cases chosen were good, the parts done were done well, and the diagram - description correspondence quite good also! Most points are from the UCDs missing the user step, system response tables. And some slight description and diagrams not matching. The UCD steps should be more detailed, e.g. specifying what data is filled in or shown - you may also reference mockups that have that detail instead of retyping it (i.e. “system displays this screen <direct link to mockup object>” “User fills in elements on form in <direct link to mockup object>”) Focus on adding that detail. EG any form/input validation, options that might not be listed on mockups, etc. Make sure to fix the UCD steps before making your sequence diagrams!**

**Tasks:  
1. [task] Move the use case descriptions into the template from the rubric / class exercises. The main missing part is the user steps, system response. If you want to just paste in the tables for that for each use case, that is enough. The lack of relationships between use cases is perfect, don’t add relationships like includes etc.**

**2. [task] Make tasks for each person with which use cases they will make user step, system response for.**

**Suggestions:**

1. **Add a use case for searching feedback and comments using text search**
2. **After people make user step, system response tables, get a round of feedback from me on them.**

Use Case Diagrams/Models and Descriptions ( 12.5/30)

* 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.
  + **Good job here overall!**
  + **-.5 each (2.5 total) for not exact match (easy fix):**
    - **Submitting Feedback is Submit Feedback on the diagram**
    - **And the 4 other use cases in** [Feedback System](https://docs.google.com/document/d/1bv4fhnbfZL1HLRI5Qr9JZUStCmYaokHY/edit)

* 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.
  + **-15 all use case descriptions missing user step, system response tables**
* 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.

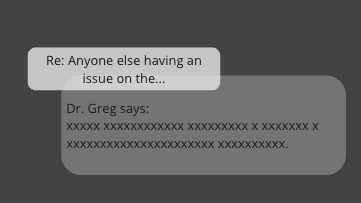
New version of the application ( 10/10)

* 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.

## D2 feedback on NMD part:

**Overall great job getting to D2 level of completeness! Your app is really coming into being! The existing design is comprehensive for the features included and well-detailed. At the same time with real/concrete values in D3 I think many of the font size choices and relative sizes of page elements will need to be changed, but that’s normal.   
  
The usability testing script showed a lot of care for users, which was great! In classtime we get critique but generally haven’t on project 1 done usability testing as in** [**class 10-2**](https://miro.com/app/board/uXjVNfZUHeo=/?moveToWidget=3458764565609348730&cot=14)**. The usability testing script showed a lot of care for users; for the revision add the tasks like from class 10-2 to the script, then do the usability testing. So you got good feedback and design ideas but it’s different information from basic usability for people that are unfamiliar / never seen your project. Use tasks from #3 and #4 below, and you should make 1-2 other basic tasks for using the app. Besides that, the script is quite good, see also** [**this script**](https://docs.google.com/document/d/1QIzA53ESoLBhmdWs4qHZ9ofnRN6uD4ai/edit) **from Team C for other ideas/comparison (you want bigger tasks, the tasks there are a little too much stepping the user through what to do).  
  
Points will come back on revisions when addressed (100% this one time for D2, 85% for D3). Do your D3 work to the D3 standard and you’ll get the maximum points back. You’re really well set up for that!**

**Tasks for D3 and D2 revisions:**

1. **[task] From D3 rubric: “Most important/valuable screens have concrete actual values in them instead of filler text (I will note the specific screens in the grading feedback)”**
   * **[task] concrete example of putting in a query and what the behavior is like, for searching posts/questions/feedback. Similar to course search that you did very well, will require 2-5 screens to show that.**
   * **[task] have course page and notifications page screens have concrete examples of the 3 scenarios we discussed in last deliverable’s feedback**
   * **[task] homepage is fully concrete/actual data**
2. **[task] Change some design choices to increase contrast/readability e.g. black on gray is hard to read. With the full fidelity mockups the colors there should be actual colors in the app. In industry you’d update all the mockups but for this project you just need the **
3. **[task] In usability testing add a task for “Imagine you are feeling imposter syndrome in <class name> and have a question you feel embarrassed about - use the app to ask it” - then at the end ask them if they’d really post their question if they were in that situation, why / why not, how they felt. Listen to the think aloud from people and note whatever information they want to use that isn’t on the page, emotions/concerns, etc.**
4. **[task] In usability testing add a task for “Imagine you are feeling imposter syndrome in <class name>, use the app to feel better”**

SRS (primarily the prototype/mockups) ( 37/40)

1. Example Mockups: [Team C 2022 (this is at D3 level of detail)](https://miro.com/app/board/uXjVNaxNZpA=/?share_link_id=773411667516)
2. If there are no UI mockups, deduct -40 points.   
   * This is here so you get started. Put these in Section 3.1
3. You should have wireflows or a series of mockups or an interactive prototype in Adobe XD, Figma, or similar software, for the most valuable features in the application. The application should be complete, but the mockups can be in varying levels of fidelity. Do higher level fidelity for the most important/user valuable features.
   * **-3 missing the searching of posts from the course page (could have been very low fi)**
4. Use cases are not required.
5. Other sections will be filled in in the next deliverable. Here’s an example for the eventual **Deliverable 2** [level of detail](https://docs.google.com/document/d/1WkqFkOZBiUtC19x8w7Zt7qIg8MJishZp/edit?usp=sharing&ouid=108788083057609121398&rtpof=true&sd=true).

Usability testing ( 13/25)

* Write a usability testing script ( 6/10)
  + **Script shows a lot of care for the people you’re working with, very good**
  + **-4 missing tasks to ask the user to do, need those tasks to be usability testing (i.e. using your app’s interface with a minimum amount of help / extra information) - for example, the existing wireflows are good for design but for the usability testing would need to be wizard of oz’d like we did in class on 10-2 removing the explicit arrows etc that the real app would not have.**
* Do usability tests with at least 3 other people, and write up consolidated notes and planned revisions ( 7/15)
  + **Did you somehow use the usability testing script during class? I don’t think that would have been possible in the time available, but correct me if I’m wrong. If not, you need to do usability testing like we did in 10-2 class using your script. The main thing to add to the script is the tasks for people to accomplish. IE instead of “\*Participant now runs through the UI mockups\*” have some tasks like “post a question you have about your class NMD358” “...”**

# Deliverable 3

## Rubric common between two classes:

[Examples for the Project Deliverables through Deliverable 6](https://docs.google.com/document/d/1mlv7hvrquPboTr49aAdA6OhdGsB3Up1OhBvobqXYTEk/edit)

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.

Product Backlog ( /5)

* The comments from Deliverable 2 must be addressed.  
  + For any comment that was not addressed, deduct -2 points.
  + All unaddressed comments are carried forward by default to the next deliverable as well
* 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.
* The backlog should have been updated with respect to sprint numbers, story points, etc.   
  + If there is no update, deduct -2.5 points.

Sprint Backlog 3 (/5)

* The comments from Deliverable 2 must be addressed.   
  + For any comment that was not addressed, deduct -2 points.
* Should reflect the product backlog. That is, all the user stories that are assigned to Sprint 2 in the product backlog, should be shown in the Sprint Backlog 2.  
  + For any missing user story, deduct -2 points.
  + If the Sprint Backlog 2 does not reflect the product backlog, deduct -5 points.
* You must have other technical tasks in your backlog otherwise -5.
* 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. Otherwise -5.
* Sprint backlog must have accurate information about who did what % of the work. The sprint retrospective is a good time to fill that in, so you can see the difference between planned and actual effort.

Kanban (/5)

* 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 3 (/10)

* Should follow the [template](https://docs.google.com/document/d/1QkzW3zqLNUdav9vv93DhmefqulpcLdep/edit?usp=sharing&ouid=108788083057609121398&rtpof=true&sd=true).
* For any missing item from the template, deduct 2 points.
* Should be written for this past sprint / week (i.e. for Deliverable 1, Deliverable 1 and Revisions for Deliverable 0) but reflect about the whole project experience so far
* 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, otherwise -5.

## NMD rubric (they lead the mockups and refining the mockups):

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, or upload the mockups, etc.
* You may use the Github GUI ([how to guide](https://docs.google.com/document/d/1gzfsdIgxpR_ipdTFF0xPWNLhsXQWhBIkOdp-ITgeEFI/edit)) to upload files for this, recommended if you're less interested in the technical development.

Mockups ( /50)

* Example Mockups: [Team C 2022 (this is at D3 level of detail)](https://miro.com/app/board/uXjVNaxNZpA=/?share_link_id=773411667516)
* If there are no UI mockups, deduct -40 points.
* You should have wireflows or a series of mockups or an interactive prototype in Adobe XD, Figma, or similar software, for the most valuable features in the application. The application should be complete.
* All screens need to be high fidelity wireframe/wireflows or made as an interactive prototype in Figma, Adobe XD, etc.
* Most important/valuable screens have concrete actual values in them instead of filler text (I will note the specific screens in the grading feedback)
* Look through the requirements and use case descriptions and diagrams for differences with the mockups. Incorporate good ideas/details from the use case descriptions, but you can make better designs than they describe.

Usability testing revisions ( /15)

* Revise the mockups based on usability testing results

New version of the application (/5)

* The implementation should have been started and advanced from the last deliverable.
* You should be able to run the app on your machine. See [this video walkthrough](https://2189801-2.kaf.kaltura.com/media/NMD442%20Fall%20'23%20Video%20walkthrough%20of%20first%20git_react%20textbook%20interactive%20chapters/1_kbmv67ha) of [the interactive textbook](https://greglnelson.github.io/react-hooks-typescript-tome/1-setup/environment.html) on setting up git and node on your machine, and get help as needed from developers on your team.
* You'll participate in the making of a video of your team members each showing their machine running the latest version of the app.

## COS rubric (they lead the technical design work and initial coding):

Architecture design (/5)

* Developers lead this
* The comments from Deliverable 2 must be addressed.   
  + For any comment that was not addressed, deduct -2 points.

SRS ( /5)

* Need to ensure the comments from Deliverable 2 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)
* If the number of FR and NFR were lower than 15 and 10 in Deliverable 2, you must meet those minimums in this deliverable. If not met, give 0.
* For each wrong requirement deduct -2 points.
* For each conflicting/contradictory requirement, deduct -2 points.
* If the UI is missing, give 0.
* If the UI is not updated, deduct -3 points.

Use Case Diagrams/Models and Descriptions ( /10)

* The comments from Deliverable 2 must be addressed.   
  + For any comment that was not addressed, deduct -2 points.
* Update Use Case Models and Descriptions document with potential new use cases and based on the feedback given on prior Deliverable(s)
* If the number of UCD were below 10 or the models/diagrams were below 2 in Deliverable 2 and they did not change in Deliverable 3, deduct -5 points for each.
* If the models/diagrams are missing entirely, an additional -5 points
* Use this [use case description](https://docs.google.com/document/d/1jBaLHNLAAEPAqN7BCqZspq04X625YcIY/edit?usp=sharing&ouid=108788083057609121398&rtpof=true&sd=true) template. Here is an [example of Use Case Diagrams and Descriptions](https://drive.google.com/file/d/1_r0T_X7xg1zB9Vnt0CAREqIjXAOmi0MV/view?usp=sharing). Here is [another example](https://docs.google.com/document/u/0/d/1HsytILdXk_8f-2vJjSbXwhj9OYWIcvVP/edit?fromCopy=true).
* 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.

Sequence Diagrams (/35)

* Pick a few of the main features of your application and draw the sequence diagrams based on those. Prioritize drawing ones with the highest priority.  
  + For each of the sequence diagrams, start by writing a table describing the steps/messages, with columns for subject, subject action (a verb), parameters, and object acted upon. See class/slides for more information and examples.
* About 7 - 15 analysis sequence diagrams plus 3 -7 design sequence diagrams.  
  + If the number of analysis/design sequence diagrams is below 7 and/or 3 respectively, deduct -3 points per each. For example if there are only 5 analysis SD, deduct -6 points.
  + See [this example of sequence diagrams](https://docs.google.com/document/d/16aZE4B08wakokbiZmZOn2_nkfWlb59rq/edit?usp=sharing&ouid=108788083057609121398&rtpof=true&sd=true).Analysis sequence diagramscan be less formal (not showing method calls and formal types for arguments). Design sequence diagrams show message arguments and types (e.g. function call arguments and types), as covered in the class lecture.
* Each diagram needs to have correct variables, methods and parameters.  
  + For any mistake, deduct -2 points.
* For each diagram you need to write a description of the steps.  
  + For each description that is missing, deduct -5 points.
  + For each incomplete description, deduct -2.5 points.

New version of the application (/15)

* If there is no progress compared to Deliverable 2, give 0.
* Any comments from Deliverable 2 must be addressed.   
  + For any comment that was not addressed, deduct -2 points.
* The implementation should have been started.   
  + Aim to have developed at least 1 of the core features of your app.  
    - You should use test-driven development. +5 for your application if you used TDD. Next deliverable TDD will be required.
    - At least 1 feature is started, and there are at least 3 automated tests written for the feature (some of which may be failing and thus commented out)  
      * If no feature implementation is started, or there are no automated tests, give 0.
  + The developers should lead this and are expected to, as needed, to 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. Developers are expected to help any other team members with technical questions.
  + If there is no code/HTML/scripts, and no changes from last time, this part will be graded as 0.
  + If there is no HTML/CSS UI (this may include React components or just plain static HTML/CSS ) that looks similar to at least one of your UI mockups (or you have no UI mockups in your SRS, even if there is some UI), deduct -10 points  
    - Your designers will provide wireframes/mockups. In general it is best practice to make a simple UI mockup/sketch before implementing any UI more complicated than a single button or HTML element
* Setting up the build environments and the necessary dependencies.   
  + 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.
* You must upload a video to your team Discord channel in the COS420 Discord, from a synchronous team meeting/screenshare on Discord or Zoom, where each team member shares their screen showing the app working on their machine, to show that this has been set up and works for each team member. No video editing, one person on the team should record their screen continuously through the meeting (you can use the Zoom record meeting feature to do this easily).  
  + If this is not present, -10.
  + For each team member not showing it works on their machine, -2.5

# Deliverable 4

# Deliverable 5

# Deliverable 6