**Deliverable 4 Rubric**

Read the grading rubric below. Submit each file as a link to a Google Document and share it with [gregory.nelson@maine.edu](mailto:gregory.nelson@maine.edu) with full editor privileges and all group members (do not share it using "Anyone at UMaine with the link can..."). Note that any revisions/resubmissions will also be done in those same Google Documents, to make it easier to understand what has changed. Make sure to download these documents and check them into your GitHub repo as well; this is also good practice in the Industry as a backup.

For example, your text submission should look like this:

* Product Backlog: <web link>
* Sprint Backlog: <web link>
* Software Requirements Specification (SRS): <web link>
* Architecture Design: <web link>
* Use Case Diagrams/Models and Descriptions: <web link>
* Sequence Diagrams: <web link>
* Domain Model: <web link>
* Sprint Review: <web link>
* Github Repo: <web link>
* TDD Evidence and features developed walkthrough video: <web link>
* Focus Group / Usability Testing documents and results: <web link>

You should have developed and passed tests for 2 to 3 of the most valuable features for your users.

Grading Rubric:

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 that being not present.

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, create a task 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 4 (/5)**

* The comments from Deliverable 3 must be addressed.
* For any comment that was not addressed, deduct -2 points.
* Contains only the stories (user stories and technical and non-technical stories) you commit to working on and delivering for Deliverable 4.
* Should reflect the product backlog. That is, all the user stories that are assigned to Sprint 4 in the product backlog, should be shown in the Sprint Backlog 4.
* For any missing user story, deduct -2 points.
* If the Sprint Backlog 4 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.

### 

### **GitHub (/5)**

* Contributions should be consistent and all students should 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)**

* The 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 submittable for regrade.
* There should be 20-50 tasks for current Deliverables 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 4 (/5)**

* You 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 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.
* The document should contain all prior Sprint Reviews as well, with the most recent at the top, otherwise -5.

### 

### 

### **SRS + Use Case Models and Descriptions + Architecture design (/15)**

*If no updates/comments need to be addressed for this, this part will be graded 0/0 and 5 points added to Sequence Diagrams (/10 -> /15), and 10 points added to the New version of your application (/25 -> /35)*

**SRS**

* Need to ensure the comments from prior deliverables 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**

* The comments from the prior Deliverable 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.

### 

### 

### **Architecture Design**

* The comments from Deliverable 3 must be addressed.
* For any comment that was not addressed, deduct -2 points.
* This will most likely be a MVC architecture. You can use this as a template, 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.

### 

### **Sequence Diagrams (/10 or /15)**

* The comments from the prior Deliverable must be addressed.
* For any comment that were not addressed, deduct -2 points.
* 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 diagrams can 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.

### 

### **Domain Model (/15)**

* Here is a [good example](https://drive.google.com/file/d/1B1vu1JrqSvrKM-xQnx5ARZ5mWU4htksh/view?usp=share_link) and [another example](https://drive.google.com/file/d/19Er1uxVuEscF5h6mQagTlO12WBDaGdKk/view?usp=share_link)
* Review relevant class sessions and slides before doing this. Feel free to search online for class diagrams and domain models for similar domains as yours for inspiration, just cite them in your final domain model.
* Submit a model of the domain with links and relationships. The expectation is that you have at least 8 classes in their diagram. If your domain is actually very small, you may have less.
* If the number of classes is below 8, per each missing class -2.5 points.
* If the model is too large, break it down in to 2 – 4 smaller models.
* Otherwise, -5 points overall.
* UML domain model needs to include attributes, links, methods.
* For each mistake, -2 points.
* Create a task to compare your sequence diagrams and domain model and harmonize them (consistent method names)
* If the domain includes something like database or UI deduct -2 points per mistake.
* Links should be correct. If not, each mistake -2 points.
  + Represents reasonable relationship
  + Cardinality is reasonable (e.g. 0..\* zero to many, …)
* Students also need to have a description for the domain model as a whole and how each smaller model is related to the rest.
* If the description is missing, deduct -10 points.
* If the description is not completed, deduct -5 points.

### 

### **New version of the application (/15 or /25)**

* If there is no progress compared to Deliverable 3, give 0.
* Any comments from Deliverable 3 must be addressed.
* For any comment that was not addressed, deduct -2 points.
* Your team must develop 1 to 2 core user stories/features of your app.
* At least 1 new feature is started, and at least 1 automated test written for the feature (which may be failing i.e. is commented out)
* If no feature implementation is started, or there are no automated tests, give 0.
* Use the process Dr. Greg demonstrated in class for test-driven web development for a use case. *0 points for non TDD code. Not recoverable by regrade because you can’t go back in time to use TDD. Do not commit or squash your commits such that you can’t tell TDD was used. Write some tests, then write some code to make the tests pass, rinse and repeat.*
* 15 points: Record a short 1-10 minute video showing the progress your team has made, walking through, for each feature, the user story/use case/mockup, tests, then code, then website in the browser, and people contributing to the development for each.
* This is meant to simulate the demo that happens with the Product Owner and stakeholders in an industry sprint review
* If this video is missing, -15
* If there is no code/HTML/scripts, and no changes from last time, this part will be graded as 0.
* If no UI resembling your application, -5 points
* If login is not finished, -5 points
* 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.
* All team members are expected to 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. 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, -10 points.