Team Project

Your team will work on the project over roughly 8 weeks, and you will receive grades both for your team's performance as a whole and for your individual contributions to the project. The project will be divided up into an initial planning period (roughly 2 weeks) and two development iterations, *Alpha* and *Beta*, (roughly 3 weeks each). First, this document describes the team deliverables and grading criteria for each stage of the project. Then, it describes goals and grading criteria for your individual contributions throughout the project.

1 Team Deliverables: Initial Project Planning

The main deliverables to come out of the initial planning are a collection of artifacts regarding project requirements, planning, and design. These artifacts are collectively call the *Initial Planning Milestone*.

1.1 Deliverable: Initial Planning Milestone Submission

For the Initial Planning Milestone, you will submit the following artifacts:

1. a set of user stories,
2. a sitemap for your web app,
3. user interface sketches, and
4. a class diagram of your model.

The artifacts should satisfy the following grading criteria:

- **Tagged submission in GitHub.** To grade your code, I will clone your team's GitHub repo, and checkout a tagged version of your code. You must tag your code as "plan".

- **Artifact quality.** All your artifacts must be of high quality.
    - **User story quality.** Your USs must follow the guidelines/principles described in lecture. For example, they must follow the templates given and must satisfy the INVEST criteria.
    - **Sitemap quality.** I don't care about the exact notation you use for this diagram. However, it should be easy to figure out what pages your site will have and how those pages will be interconnected.
    - **UI-sketch quality.** Your UI sketches should show the basic page elements (fields, buttons, etc.). They may or may not be styled (e.g., with colors/graphics). Such sketches are sometimes referred to as wireframes.
    - **Design quality.** Your designs should follow principles of good design, such as the SRP (Single Responsibility Principle) and DRY (Don't Repeat Yourself) principles.
    - **Class diagram quality.** Use proper class diagram notation (as given in lecture). Label all associations, and include all multiplicities. Include attribute types (e.g., "name : string").
    - **Consistent naming.** You must use consistent names for things across artifacts. That way, for example, we can tell where each UI sketch fits into the sitemap.

- **Diagram format.** Diagrams need not be anal-retentively typeset; they simply must be readable. Thus, you may hand-draw diagrams on paper/whiteboard, and scan them into a digital format (PDF preferred). Of course, if you really want to typeset them, that's OK too.

- **Diagram submission.** Keep the diagrams and USs with your web app code. Specifically, add a folder "misc" to the top level of your project, and place the file(s) in there.

Keep in mind that these artifacts will evolve as the project rolls along. Plan for such evolution.

Note that there is an A&B eligible role below (Quality Assurance Czar) with special responsibilities regarding the milestone submission.

2 Team Deliverables: Alpha Iteration

There are three main deliverables for the Alpha Iteration: a demo video, a collection of project artifacts, collectively referred to as the *Alpha Milestone*, and a live in-class demo session.

2.1 Deliverable: Alpha Iteration Demo Video

Your team will be responsible for creating a demo video of your software. This video is mainly to assist the course instructors in grading your progress on the project. The video must also have an accompanying text document that lists who built each of the demoed features. The demo video and document must meet the following grading criteria:

- **Criterion: Demonstrate the progress that the team has made so far.**
  - **All the new features.** Include all the latest features in the demo. Don't leave any out. A big point of this exercise is to demonstrate all the wonderful progress that the team has made. Note that this criterion does not mean that you should skip re-demoing old features. It just means you shouldn't skip the new ones.
  - **What's new?** Clearly state which features are new as they are being demoed. You need not explicitly state which features are old, unless you think there may be confusion.
  - **Backend too.** Although UI features are a high priority, you may also demonstrate that backend functionality is working, even if it's not yet connected to the frontend. The key thing is to prove that the code runs and works! Along those lines, you may demo automated tests.

- **Criterion: Make clear who contributed what to the project.**
  - **Who made it?** In the text document, give an entry for each new/updated feature demoed. Each entry must include the name of the feature, the time offset in the video where the feature was demoed, and the U of M username(s) of who built the feature. If multiple people contributed to a feature, say who did what. Don't forget to credit anyone. Be sure not to make any factual errors.
  - **New or old?** If it's not clear, make it clear what part of the work is new.

- **Criterion: Display the team's work in the best possible light.**
  - **Story form.** Any demonstration of UI must take place in the context of a cohesive story. That is, the presenter must describe one or more characters (with names, like Alice and/or Bob) and relate a story about the character using the software. The presenter must stick to this story format. The story and accompanying demo must be well thought out, and not leave the audience with the impression that the presenter is making it up as he/she goes along. Use realistic names for things and not made-up placeholders, like "foo" and "slafjsd".

    - **UI first.** Since the UI is generally most interesting, you should lead with that.
    - **General audience.** Don't forget that not everyone is as familiar with your project as you are. To be on the safe side, explain it as if you are talking to someone who has never seen it before.
    - **No special effects or fancy editing.** The video should clearly show a user (or users) interacting with your web app. Don't add special effects or sound effects, which distract or detract from the authenticity of the interaction.

- **Criterion: Length and format constraints.**
    - **Time limit.** The video must be no more than 10 minutes long.
    - **Fill the Time.** Your video should be at least 8 minutes; otherwise, you're probably doing it wrong.
    - **Video format.** The video must be in a format playable in VLC (http://www.videolan.org/vlc/), which accepts most common formats.

Note that the creators of the demo video and accompanying document are eligible for A&B points (see below).

2.2 Deliverable: Alpha Milestone Submission

For the Alpha Milestone, you will submit the following artifacts:

1. a copy of your code (tagged in GitHub), and
2. up-to-date versions of your Initial Planning artifacts (i.e., USs, sitemap, UI sketches, and model-class diagram).

The artifacts should satisfy the following grading criteria:

- **Tagged submission in GitHub.** To grade your code, I will clone your team's GitHub repo, and checkout a tagged version of your code. You must tag your code as "alpha".

- **Code builds and runs.** I should be able to build and run your code using the usual approach from the Boot Camp projects. If any special instructions are required to build/run your software, include them in the README file in your project's top-level directory.

- **Replicable demo.** I should be able to replicate your demo video. If seed data is required to do so, you must somehow make that data available to me (possibly giving instructions in the README).

- **Artifact quality.** All your artifacts must be of high quality. The criteria from the Initial Planning Milestone still apply, with the following addition.
    - **Code quality.** Your code must follow common style guidelines and be well organized and readable. For example, all code must be properly indented, and class/variable/method names must be sensible. You should also do your utmost to avoid bugs and other sloppiness.

- **Customer satisfaction.** Your customer will provide feedback on how well your team has satisfied the requirements he/she gave you and how well aligned your team's prioritization of the work has been with the customer's priorities.

Note that there is an A&B eligible role below (Quality Assurance Czar) with special responsibilities regarding the milestone submission.

2.3 Live In-Class Demo Session

For this session, each team will operate a demo booth. One member of your team (the "demo-booth operator") must run the booth, providing visitors with an interactive demo of your team's software. The remaining members of your team will circulate about the other booths, acting as visitors. The interactive demo must meet the following grading criteria:

- **Clearly explain your project to visitors.** Assume that visitors have never seen your project before. Thoroughly and clearly explain what problem your project solves and how it does so.
- **Display the team's work in the best possible light.** Use presentation techniques discussed this semester to present your team's software in an engaging and compelling way. Also, think about the best way to set up your booth. What equipment will you need? Extra monitors?
- **Allow visitors to use your project.** This is an interactive demo, which mean that visitors should be allowed to try out your project.
- **Time limits.**
  - **Don't go too long.** The demo must be no more than 10 minutes long. Be sure to allow enough time for users to "play" with your system.
  - **Fill the time.** Keep your visitors engaged throughout the 10 minutes.

Note that demo-booth operator is an A&B eligible role (see below).

Figure 1 below provides an idea of what the floorplan for the demo session will be like.

3 Team Deliverables: Beta Iteration

The deliverables for Beta Iteration will be mostly the same as for Alpha Iteration, with only a couple changes:

- **Demo video.** Follow the instructions above for creating the video. This time, the video may be up to 15 minutes long. Make the demo a good overall tour of your system, including both new and old features. Also, make the who-did-what document the same as before, except only credit work done in the Beta Iteration. To clarify, the demo should include work done in Alpha Iteration, but don't say anything about it in the document. Make sure that the who-did-what document is specific and precise so that no Alpha Iteration work might be accidentally credited as Beta Iteration work.

- **Milestone submission.** Follow the instructions above for creating your milestone submission, except this time, tag your code version as "beta". Make sure that all your submission documents are up to date (esp. your user stories and class diagram).

- **Live in-class demo session.** Follow the instructions above for presenting the demo. This time, the demo should 15 minutes long (instead of 10).

This iteration has the same A&B opportunities as the Alpha Iteration.

4 Individual Productivity

4.1 Regular Productivity

The majority of your individual productivity points are associated with regular productivity. Each team member is assigned certain tasks for each iteration, and it is expected that he/she will complete his/her assigned tasks in a timely manner. It is also expected that team members will be continuously productive, and not to put off their work, rushing to slap something together at the last minute. We will use a combination of the following criteria to assess your regular productivity:

- **Make it into the demo!** A key indicator of whether you are producing value for your team is whether your work is visible in the demo. If your contributions are nowhere to be seen in the demo, it will almost certainly count against you.

- **Minimum acceptable contribution.** Your completed work (as reported with the demo video) will be compared against the minimum acceptable work in the iteration plan.

- **Teammate evaluations.** Your teammates will provide their subjective assessment as to your performance.

- **GitHub metrics.** GitHub tracks how much each team member has contributed to the code base. These metrics will provide further evidence of your contributions.

If a team member's productivity is found to be particularly unsatisfactory, they may receive the following additional penalty:

- **Milestone Deduction for Unproductiveness.** A student who demonstrates unsatisfactory productivity may also lose points on the associated milestone. This deduction is meant to account for the lack of contribution made by an unproductive team member to the project.

4.2 Above and Beyond Productivity

To achieve the highest grades in the course, you will need to go above and beyond the call of duty; thus, your individual productivity grade also accounts for Above and Beyond productivity. You may earn A&B points in the following ways.

- **Play a special role.** Each iteration, you will have an opportunity to play a special role. The team should assign these roles democratically, since the roles serve key functions on the team.

    - **Project Manager (PM).** This role is concerned with customer and instructor communication, and with task planning and teammate coordination and communication. Here some additional details:
        - **Reliable.** Must be highly reliable (e.g., attends all classes/meetings and responds quickly to emails).
        - **Organized.** Must be organized and detail oriented. For example, this person will be held responsible if class procedures are not followed correctly.
        - **Deciding vote.** Although most team decisions should be made democratically, the PM does have authority to break any stalemates that should arise.

        - **Compensation:** 1 A&B point per iteration (1 person only).

    - **Quality Assurance Czar.** This role is concerned with ensuring that the milestone artifacts are of high quality. Here are some additional details:
        - **Code gatekeeper.** This person must make sure that the submission instructions are followed, and that contributors adhere to good design and coding standards.
        - **Repo organizer.** He/she must ensure that the repo stays well organized.
        - **Class diagram creator.** He/she is responsible for producing the class diagram, and making sure that the code is consistent with the design.
        - **Compensation:** 1 A&B point per iteration (1 person only).

    - **Video-Demo Creator(s).** The team member(s) who create the video demo will receive special compensation for their extra effort.
        - **Compensation:** 2 A&B points per team per iteration, to be divided based on contribution (multiples of .25 or .33) to whoever worked on the video.

    - **Demo-Booth Operator.** The team member who operates the demo booth will receive special compensation for their extra effort.
        - **Compensation:** 1 A&B point for the booth operator (1 person only).

- **Be a top contributor.** Each iteration, I may identify one or two members of each team who have made exceptional contributions in the past iteration. These "10Xers" will receive an A&B bonus for their exceptional work.

- **Negotiate additional work.** In addition to your planned work in a given iteration, you may take on additional work for A&B points. You must negotiate the work and compensation with me, and I must approve it. Such work often involves adding bonus features to the project; however, I am open to your ideas.

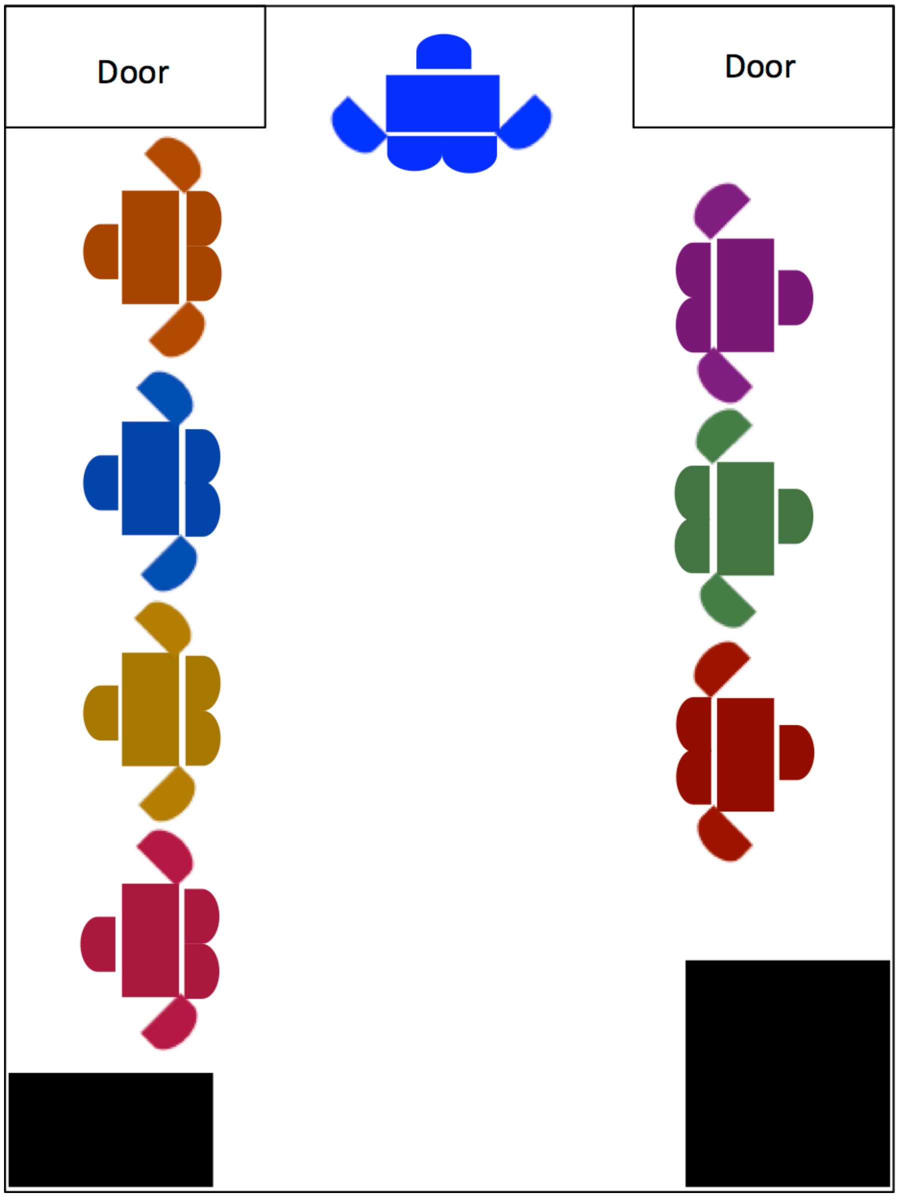

Figure 1. Demo session floorplan.