CS 480 —Mobile Apps Spring 2020

**Team Project**

Last edited 3/3/2020

|  |  |
| --- | --- |
| Worth: | 45% of course grade |
| Assigned: | March 10, 2020 |
| Due Dates: | TBD |
| Late Policy: | No late submissions accepted. Source code, Kanban board, issues and demo apps evaluated “as is” at scheduled team meetings. Unless excused in advance, each team member must show up on time for scheduled team meeting to receive credit for that week’s milestone. |

# Overview

The team project gives you an opportunity to apply all of the client-side and server-side web development skills you have been learning in this class to a realistic team web development project. For this project, you will work on teams of **2-3 students**.

# Learning Objectives

In completing this team project, you will

* Develop a mobile app that can be used as a capstone for your digital portfolio.
* Break your project’s Epic up into a series of “sprints” that can be reasonably completed in one-week intervals
* Log new user stories (features) and bugs to be addressed by students who work on this project in the future.
* Develop skills in question-asking and question-answering within a team environment.
* Develop skills in information literacy (performing web searches, identifying relevant results and applying results to problem at hand) within a team environment
* Develop skills in meta cognition (identifying when stuck, reformulating goals to make progress) within a team environment.
* Implement productive programming practices, as described in the “Programming Process Requirements” section below.

# Project Steps

Follow these steps to complete the project.

## 1. Formulate a Project Idea

Think about web apps you’ve always wanted to develop. Perhaps you have a pet project or a project you’re working on in another course that lends itself to a web application? For this class, the best project ideas (a) have broad appeal to a variety of students and (b) are large enough in scope that they can be worked on by student teams over several semesters. Come to class prepared to write down a brief description of your idea, which will be anonymously shared with, and rated by, other students in the class. Your instructor will add the most viable ideas to the official list of course projects that teams can choose from.

## 2. Complete Project Pre-Survey

This team project is part of a multi-institutional research study funded by the National Science Foundation. You have the option to participate in this study by releasing to it your project activities and deliverables (GitHub activities, Slack communications), as well as your responses to a pre- and post-survey. Your choice to participate in the study will not impact your grades in the course; in fact, your instructor and TA will not even know whether you chose to participate until after the semester is over and grades are submitted. This project pre-survey will give you the chance to provide your informed consent to participate in the study and also ask you a series of questions regarding your attitudes. It counts toward your participation grade and is required whether or not you choose to release your data for research purposes.

## 3. Find a Team

You are free to make your own teams; however, all students are required to fill out a team matchmaking survey to help place students who do not have a team. You will complete the team matchmaking survey in class.

## 4. Choose a Project Focus

Teams choose the focus of their project in terms of a target app and “[Epic](https://www.yodiz.com/blog/what-is-epic-in-agile-methodology-definition-and-template-of-epic/)”—a collection of “user stories” that define the features and functionality for the target app under development. Teams officially sign up for a project focus in this (evolving) Google Sheet, which lists approved target apps and epics. These include both the instructor’s ideas and students’ ideas adapted from the “Project Idea Brainstorm and Team Matchmaking” class.

## 5. Set up Your Project Online

Once your team has signed up for a project, you need to set up your project online as follows:

1. Create a GitHub repo for your team project by accepting this invitation link.
2. [Join the Slack Workspace](https://join.slack.com/t/cs480-spring2020/shared_invite/enQtOTY3MDgxNDk3MTM4LWUwMzJlNDI1ZDYzZmQ0NDRmZmI0Mjc3NTVhNjk4ZjZhNDgwNzI4OGU1OGQ0Y2M0MTAxNzUxMzViN2M0OGY4NTQ). Once you have joined, your instructor will add you to a private channel for your team’s communications. Only your team members, your instructor and the TAs have access to this private channel. Please use the channel for all online team communications, since these communications will be considered when evaluating your team programming process for each project milestone.
3. In your GitHub repo, create a [project board](https://help.github.com/en/github/managing-your-work-on-github/about-project-boards) for your repo using the “Basic kanban” template. Add a new “Under review” column to the project board between the “In progress” and “Done” columns. You’ll use this column for items to be reviewed in your weekly team meetings with the instructor.

## 6. Create a Project Plan

Once your team has signed up for a project, please meet with your team to decide which specific user stories/features you will address and how you will divide your work into four one-week milestones (“sprints”). Create this plan directly within GitHub as a set of [issues](https://help.github.com/en/github/managing-your-work-on-github/about-issues) and [milestones](https://help.github.com/en/github/managing-your-work-on-github/about-milestones) as follows:

1. Create an [issue](https://help.github.com/en/github/managing-your-work-on-github/about-issues) for each Epic story/feature that your team will address within the project.
2. [Add each issue to your project board](https://help.github.com/en/github/managing-your-work-on-github/adding-issues-and-pull-requests-to-a-project-board). Initially, they should appear in the “To do” column.
3. Create four [milestones](https://help.github.com/en/github/managing-your-work-on-github/about-milestones)—one for each of the four weeks of the project. Give each milestone a name like “Week 1, “Week 2,” etc. Also provide a brief summary of the set of work items in the milestone [in the “description” field](https://help.github.com/en/github/managing-your-work-on-github/creating-and-editing-milestones-for-issues-and-pull-requests).
4. [Add each issue to one of the milestones](https://help.github.com/en/github/managing-your-work-on-github/creating-and-editing-milestones-for-issues-and-pull-requests) you have defined.

Congratulations! You now have a project plan to discuss with your instructor during your first team meeting (March 31 or April 2).

# Project Epics

This course project provides you with the opportunity to participate in a multi-institutional research project that is exploring new approaches to teaching skills in legacy code development. As the first class to participate in this project, you will be “bootstrapping” legacy code bases that students in subsequent CS 480 classes will build on. Thus, the projects you work on must have both broad appeal and the potential to be worked on over multiple semesters.

# Project Deliverables

The project consists of a proposal document and a series of four weekly milestone evaluations that occur during your weekly meetings with the instructor.

## Weekly Milestones

Your team is required to meet with the instructor once per week, on either Tuesday or Thursday, during the final weeks of the semester.

Think of the meeting as a weekly project review. Your instructor will ask questions about how the previous week went, inspect your code repository (including your commit history, Kanban board and source code), and have you demo the latest version of your app.

The main purpose of the initial meeting is to discuss your project proposal and agree on a project schedule going forward. The main purpose of subsequent meetings is to provide a basis for evaluating your process and progress over the previous week. Your instructor will provide feedback and suggestions as appropriate. In addition, he will complete a weekly milestone grading rubric (see below) based on your team’s process and progress.

Team milestone deliverables will have increasing weights according to the following schedule:

|  |  |
| --- | --- |
| Deliverable | Weight (% of project grade) |
| Milestone 1 | 5% |
| Milestone 2 | 7% |
| Milestone 3 | 10% |
| Milestone 4 | 15% |
| Subsequent Milestones | 15% |
| Reflections & Evaluations | 5% |
| Final Presentation | (whatever is left over based on total # of milestones) |
| Total | **100%** |

## Weekly Team Member Evaluations

Your instructor will assign an individual grade to each milestone. In addition, team members will be required to evaluate each other’s contributions to the project over the past week. These evaluations will

## Weekly Reflections

Reflecting on your software development process and progress is a powerful learning strategy. For each milestone, you will write a brief weekly post in which you reflect on your process and progress over the previous week. Reflections do not have to be long – they only need to clearly communicate your personal ups and downs during the last milestone .

The following prompts are intended to help you write your reflection posts:

* Describe a struggle or issue that you or your team overcame when working on this assignment.
* Assess your programming process or your team’s overall process: What did you do well and what could you improve on to be more effective in the upcoming week and beyond?
* Describe the most fun or challenging aspect of the week’s work.

# Programming Process Requirements

Part of growing as a web programmer involves establishing productive programming practices. These are even more important in a team development setting, where communication and coordination are crucial to success. Here is a list of best practices to apply when working on team milestones:

* Decompose a programming assignment into smaller tasks.
* Manage your time effectively by making regular updates to your code, so that programming tasks are not put off until the last minute.
* Keep track of progress by creating tasks and issues on the project’s Kanban board and Issue Tracker. Regularly update the tasks and issues with progress made and questions to be answered. For example, use checkboxes to create lists of to-do items. , This will help you maintain a sense of what you have done and what still needs to be done.
* Coordinate with team members to get work done efficiently. Make sure everyone’s clear on their responsibilities. Use a Slack channel or other messaging app for team communications.
* Seek help when needed! Ask your teammates, post to the OSBLE Activity Feed, and/or look for resources on the web.
* Write code that adheres to accepted standards and conventions and is well-documented, so that both you and others can understand and maintain it.
* Create a code **branch** off of the master branch in your code repo for each milestone, and then merging your code back into the master branch after each milestone is reviewed.

To develop these practices, you will be required to do the following for all team milestones:

* Maintain a project (a.k.a. Kanban) board of tasks issues and update the board regularly to track progress along the way. Use checklists on each task card to decompose tasks further and track progress. (Note: You can use the same board for all milestones, but you should tag issues by milestone where possible.)
* Make *regular* commits to your team’s GitHub source code repository. You will be graded in part on how regularly you check code in.
* Monitor the activity feed and respond to your peers’ questions and issues when you are able.
* Write code that adheres to style guidelines appropriate to your chosen language and development environment. If you cannot find any, you can always follow the suggested [conventions](https://www.w3schools.com/js/js_conventions.asp), [best practices](https://www.w3schools.com/js/js_best_practices.asp), and [documentation standards](http://faculty.cs.niu.edu/~mcmahon/CS241/c241man/node6.html) for JavaScript.

# Weekly Milestone Submission

Create a new project branch in GitHub for each weekly milestone (milestone1, milestone2, milestone3, milestone4) and issue a pull request on that branch prior to your scheduled team meeting with the instructor. (After the meeting, merge the changes back into master and fork a branch for the next milestone.) *All work done in the weekly milestone branch is due and will be evaluated at the team meeting. Your team’s code repository and latest working app will be evaluated “as is.” No late work will be considered*.

To get credit for a milestone, **each team member must be present at the team meeting with the instructor**. Any team member who is not at the team meeting cannot get credit for the milestone. However, I do realize that legitimate scheduling conflicts may arise, especially if a team meeting is scheduled outside of the regular class meeting time. If a legitimate scheduling conflict prevents a team member from being at a meeting, you must discuss this with me in advance. A failure to clear a meeting absence in advance of the meeting can result in an automatic 0 for the milestone.

# Assessment

We will grade the various components of each team milestone according to the rubrics on the following pages.

**Grading Rubric for Weekly Milestones**

| Criterion | Weight | D-F Level: Emerging (0-6 pts) | B-C Level: Developing (7-8 pts) | A Level: Mastering (9-10 pts) |
| --- | --- | --- | --- | --- |
| 1. Functionality | 35 | Some “Mastering” criteria may be met; however, significant omissions or deficiencies exist | “Mastering” criteria are mostly met; however, there may be minor omissions or deficiencies that prevent criteria from being fully met. | * Milestone app correctly implements functionality targeted in milestone. * There are reasonable explanations if some items are not implemented or incorrectly implemented. * Concerns from previous milestone meetings are adequately addressed. |
| 1. Visual Design | 25 | Some “Mastering” criteria may be met; however, significant omissions or deficiencies exist | “Mastering” criteria are mostly met; however, there may be minor omissions or deficiencies that prevent criteria from being fully met. | * The app’s layout, organization and visual design promote efficient task completion. * Appropriate widgets are used to support user tasks. * Appropriate feedback is provided in response to user input. * App gracefully scales to screen width. * Reasonable explanations are offered for deficiencies. * Concerns from previous milestone meetings are adequately addressed. |
| 1. Code Quality and Style | 10 | Some “Mastering” criteria may be met; however, significant omissions or deficiencies exist | “Mastering” criteria are mostly met; however, there may be minor omissions or deficiencies that prevent criteria from being fully met. | * Code is logically and sensibly organized into files and functions. * Header comment blocks exist for each file and function; inline comments used judiciously to explain code. * Coding and style best practices are reasonably followed. * Concerns from previous milestone meeting are adequately addressed. |
| 1. Programming Process | 20 | Some “Mastering” criteria may be met; however, significant omissions or deficiencies exist | “Mastering” criteria are mostly met; however, there may be minor omissions or deficiencies that prevent criteria from being fully met. | * Project board is used to define programming tasks and identify issues that come up; items are *issues*, not notes * Checklists are used to decompose tasks * Questions and issues that arise are discussed either in issue tracker or on OSBLE * Project board and issue tracker are consistently used to track progress New code is regularly checked in throughout milestone period, with clear descriptions of changes made in each check-in message. * Concerns from previous milestone meeting are adequately addressed. |
| 1. Milestone Submission | 10 | Some “Mastering” criteria may be met; however, significant omissions or deficiencies exist | “Mastering” criteria are mostly met; however, there may be minor omissions or deficiencies that prevent criteria from being fully met | * All team members show up for meeting. Any absences are excused in advance. * All columns of team project meeting schedule doc are completed prior to team meeting OR links are submitted via OSBLE assignment |

**Grading Rubric for Post-Mortem Reflection Posts and Responses**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criterion | Weight | D-F Level: Emerging (0-6 pts) | B-C Level: Developing (7-8 pts) | A Level: Mastering (9-10 pts) |
| 1. Reflection Post | 100 | Post is on time, is at least 100 words and appears to be hastily written; in many places, it is off-topic, difficult to understand, is superficial and/or it has grammar or spelling errors that are distracting. (*Late posts or posts that are not at least 500 words receive a 0*.) | Post is on time, is at least 100 words and makes attempt to reflect on programming process; however, it may be difficult to understand or superficial in a few places and/or have clearly-noticeable grammar or spelling errors. | * Post is on time and at least 100 words * Post makes an earnest attempt to reflect on programming process: what you did, what you learned, and what you can do differently next time * Post is easy to read and follow; grammar and spelling are mostly correct |