

Title: SystemName! A system for ... capstone courses...

Capstone courses

Instructors

Grading

Team

Abstract

Write this last

Introduction

- *What problem are we solving?*
- *Why is that problem important?*
- *What is our solution?*
- *How is our solution better / good ?*
- Goals / Research Questions

- Our idea is to make System Name!
 - Capstone Gradinator
 - Cornerstone
 - It's a Far, Far Grader Thing That I Do
 - Grade Expectations
 - The Grade Collector
 - Grade Decoder
 - Capstone Companion
 - The Informant
 - The Capstone Informant App (CIA)
 - Capstone Pal
 - Capstone Team Assessor
 - Capstone Assessment Aid
 - Capstone Assessment Tool (CAT)
 - Grader Things
 - The "Grade" of Honor
 - Oz: The "Grade" and Powerful
 - "Grade" Scott!



-
- SurveySays
- TAG - Tool-Assisted Grading
- TA-Assist

- MinuteGrade
- Blue “Grade” Shoes
- It’s better because (all info is in one place, multiple sprints, team stuff, peer ratings...)

At the University of Memphis, undergraduate computer science students must take a capstone course in which they work in groups to create software engineering projects. Given the nature of this course, traditional methods of evaluating student progress are not ideal. Instead of well-defined tests, homework problems, etc., this course is centered around teamwork, implementation of Agile best practices, and progress toward a final product. The problem then becomes how to measure students’ success in these areas, so that instructors can give feedback, identify areas of struggle, and assign grades.

This is a problem that affects not just capstone courses at the University of Memphis, but any college course that is centered around team-based software engineering projects. Professors and teaching assistants (TAs) must find a way to provide meaningful feedback in a consistent and time-efficient manner. Current methods of grading for such classes at Memphis involve monitoring the students’ GitHub project repositories and gathering student input through use of surveys.

The shortcomings of the existing solutions are that the necessary data exist in multiple places and are not always easy to collate and visualize. Our idea was to create the Capstone Gradinator [real name pending], a tool that professors and TAs could use to access all the needed information in one place, neatly organized according to teams.

Literature Review (10 cites) Cisco AnyConnect VPN:

<https://www.memphis.edu/umtech/solutions/vpn.php>

All capstone courses do peer ratings survey

-

Existing survey systems don’t visualize peer ratings well

- Google forms, Qualtrics don’t visualize peer ratings well

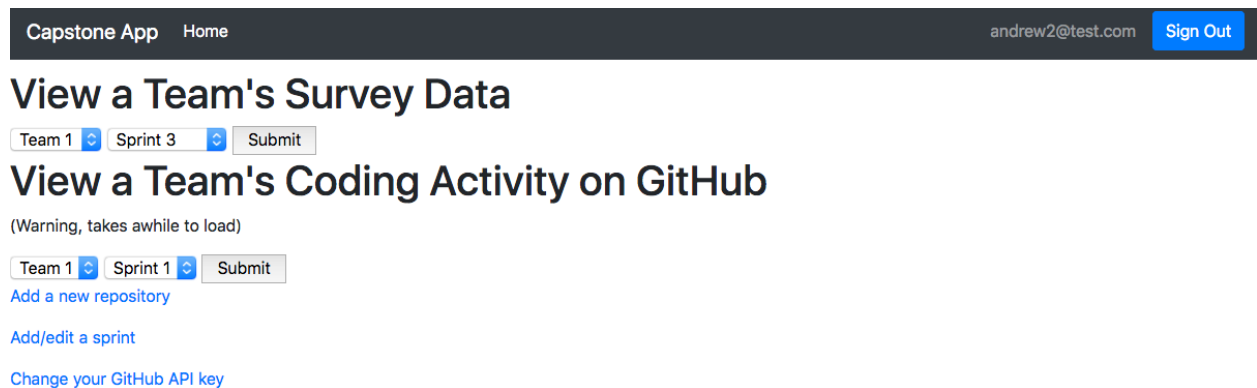
System Design

- Description of the system
- Screenshots
- Design choices or technical decisions

The system went through two major iterations. The first focused on incorporating data from GitHub and from Qualtrics surveys, as well as launching the app on Heroku. The second

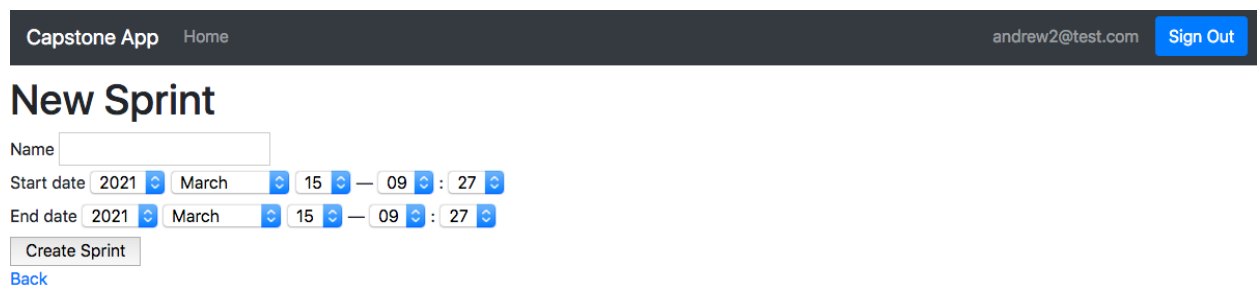
focused on a rehaul of the system in order to try to implement the ability for survey creation and completion to occur completely in-app.

Version 1 of the app implements a standard sign-in using the *Devise* gem. Upon signing in, the user is greeted by their homepage, where they have the options to view survey data or GitHub statistics (Fig. 1). We will focus on the GitHub data first. Each user can create many sprints and repositories. The sprints have start and end dates (technically, they also include times), as well as a name, such as Sprint 1, Final Sprint, etc. (Fig. 2). The repositories include information about their GitHub names and owners, which are used to make calls to the GitHub API. They also include the name of the Team that is using the repository, so that the instructors can later reference by team instead of having to remember the repository name. (Fig. 3)



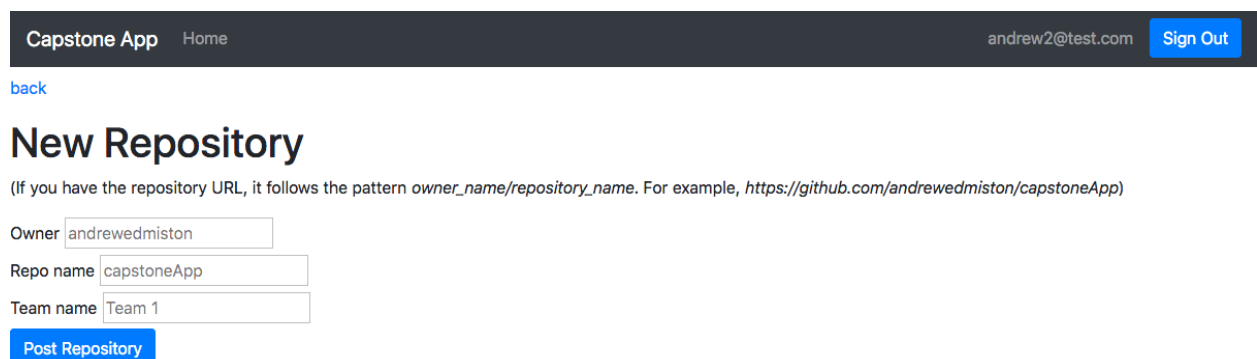
The screenshot shows the Capstone App homepage. At the top is a dark navigation bar with "Capstone App" and "Home" on the left, and the user email "andrew2@test.com" and a "Sign Out" button on the right. Below the navigation bar is the heading "View a Team's Survey Data". Under this heading are three dropdown menus for "Team 1", "Sprint 3", and a "Submit" button. Below that is the heading "View a Team's Coding Activity on GitHub". Under this heading is a warning message "(Warning, takes awhile to load)", followed by another set of dropdown menus for "Team 1", "Sprint 1", and a "Submit" button. Below the "Submit" button are three links: "Add a new repository", "Add/edit a sprint", and "Change your GitHub API key".

Fig. 1



The screenshot shows the "New Sprint" form. At the top is the same dark navigation bar as in Fig. 1. Below the navigation bar is the heading "New Sprint". Under the heading is a form with a "Name" input field, a "Start date" field with dropdowns for "2021", "March", "15", and "09 : 27", and an "End date" field with dropdowns for "2021", "March", "15", and "09 : 27". Below the date fields is a "Create Sprint" button and a "Back" link.

Fig. 2



The screenshot shows the "New Repository" form. At the top is the same dark navigation bar as in Fig. 1. Below the navigation bar is a "back" link. Below the link is the heading "New Repository". Under the heading is a note: "(If you have the repository URL, it follows the pattern *owner_name/repository_name*. For example, *https://github.com/andrewedmiston/capstoneApp*)". Below the note are three input fields: "Owner" with the value "andrewedmiston", "Repo name" with the value "capstoneApp", and "Team name" with the value "Team 1". Below the input fields is a "Post Repository" button.

Fig. 3

Capstone App

Home

andrew2@test.com

Sign Out

[back](#)

Change GitHub API Key

Api key

Change Key

Fig. 4

The last piece of information that is needed is a GitHub Personal Access Token (PAT), also known as an API key. This can be generated by the user directly on their GitHub account and copied into the app (Fig. 4). It allows the app to run API queries with the permissions of the user; thus, it can have read access to private repositories, as long as the user already has access to those repositories.

The use of a GitHub PAT raised a security concern, because the PAT would have to be stored unencrypted on Heroku servers. The usual solution to this problem is to provide a time to live (TTL) for tokens, but GitHub (at least at the time of design) did not support this option. Their tokens would not expire; it would last until manually revoked by the user. We looked into alternate options, and the only way to get tokens with a TTL was to create a “GitHub App,” which was out of scope for this project. However, users of our app could potentially choose to use a token with restricted access to their GitHub account, perhaps with read permissions but without write permissions. In the end, due to the small user base of the app, the security threat was deemed minimal.

After adding the necessary sprints, repositories, and GitHub PAT, the user can choose a team and a sprint under the heading *View a Team’s Coding Activity on GitHub* and click the submit button. This will take them to a page like that of Figures 5 and 6. The app makes several calls to the GitHub API, pulling metrics about that team’s repository for the date range indicated by the sprint.

The table near the top of the page shows all pull request data for all of the repositories “contributors” (even if none of them have worked on any pull requests). *Pull Requests Reviewed* shows how many pull requests a user has officially reviewed. It is typical to have at least one person review each pull request for quality control before accepting it into the master branch. *Pull Requests Assigned* is the flip side of the coin. Unless the team is using GitHub in an unintended way, this metric will usually be a good identifier of how many pull requests each user contributed code to. Under the table are some notes explaining these pull request metrics.

At the bottom of the page is a graph of commits by each user over time. Note that student 4, who had 0 pull requests reviewed and assigned, also contributed 0 commits during this sprint. This is a red flag that could be further investigated by the instructors. The user can hover the mouse over individual data points to see mouseover text with specific date, user, and number of commits. The user can also click the names of the students in the legend above the graph in

order to add/remove them from the graph; the graph will update its scaling automatically based on the range of the remaining lines.

The user can return to the homepage from the GitHub metrics page using either the back button or the navbar.

[back](#)

Team 2's Metrics from Sprint 2

From 2019-03-15 10:04:00 UTC to 2020-03-15 10:04:00 UTC

	Student 1	Student 2	Student 3	Student 4	Student 5
Pull Requests Reviewed	13	5	11	0	0
Pull Requests Assigned	7	14	4	0	6

Note:

- A Pull Request review indicates that the user actually reviewed code for the PR, not just that their review was requested. A user will typically review other team member's PRs, not the ones that he/she is assigned to.
- If a pull request is "assigned" to someone, that typically means that they were the one who initiated the PR, or at least contributed a significant portion of code to it. A Pull Request may be assigned to more than one person; in that case it will count as 1 PR for each person it is assigned to.

Commits

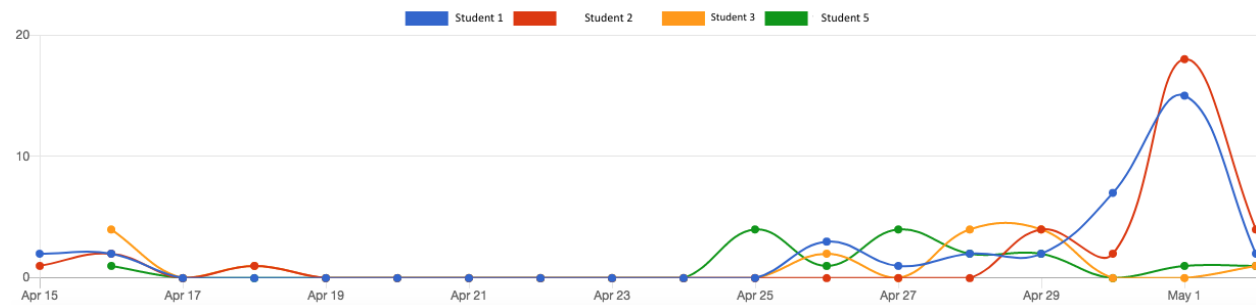


Fig. 5 & 6 (Note, these are real screenshots of results using a GitHub repository, except that the names were manually overwritten in editing for privacy.)

Next, we turn our attention to the portion of the app dealing with survey data. From the homepage, under the heading *View a Team's Survey Data*, the user can once again select a sprint and team. This part of the app was admittedly brittle, which will be brought up again during the discussion of the app's second version.

Error! Unable to process survey data. Make sure that the file is stored in the root directory of the app and is named "survey.csv".

[back](#)

Team 1, Sprint 3

Qualitative Questions

No survey data found for this sprint.

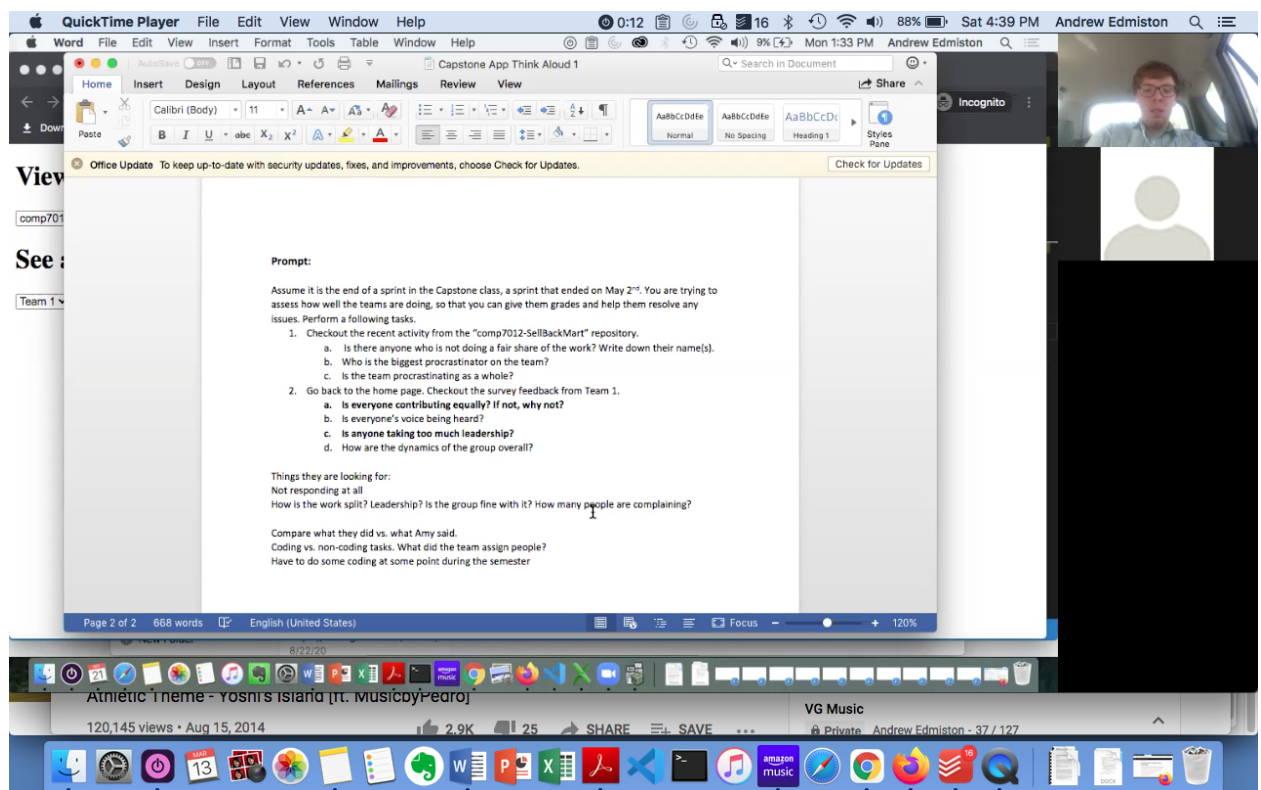
Preliminary System Evaluation

- Method
 - Think aloud protocol
 - Anonymously describe participants
- Results
 - Quotes from participants

TA 1

"I like the fact that on yours it actually tells you which student's saying it, whereas when I go to the analytics site, it does not. ... that helps a lot."

- Other stuff around 29 minute mark



- Thematically organize 'findings'

Discussion & Reflection

- What went well
- What went wrong
- If you had a do-over
- What did you learn
- Next steps / future work

Conclusion

Write this last