EECS 448 GitGud Project 2 - Scrum Meeting Logs

9/27 - 11:30 AM, Eaton 2

- We agreed on a time to have a longer, more formal meeting meeting about the project, Thursday (9/28), at 7:30.
- We noted that the Flask framework is similar to Rails, so learning the structure shouldn't pose too much of a challenge.
- Mason is out of town, so we'll have to keep him in the loop, possibly through video conference.
- Our goal until then is to familiarize ourselves with Flask and Python.

9/28 - 7:30 PM, LEEP2

- Due to everyone being busy or absent, this meeting never happened.
- Mason still absent.

9/29 - 11:30 AM, Eaton 2

- No progress has been made on the project, although everyone is a little bit more familiar with Python now.
- We planned a meeting for Saturday, 9/30, at Andy and Gage's place.

9/30 - 2:00 PM, Andy and Gage's house

- First order of business is database structure. Our inherited project uses Postgres. Our project 1 also used Postgres. We are all familiar with it, but setting up a brand new database is always painful. A lot of that meeting was getting it working, and we did not fully complete that.
- We also split up assignments in the following manner:
  - Gage - Database Rework
  - Mason - Views/Styling
  - Zach - Tasks
  - Andy - Multi-day scheduling
- Mason was present at this meeting. Welcome back.

10/2 - 11:30 AM, Eaton 2

- We talked about changes we wanted in the database structure. That is, how do we incorporate multi-day and task into their project without changing too much?
- Task was simple, since it was its own thing and independent of any work the old team has done. Multi-day was a bit more difficult. Ultimately we would have to change Times to DateTimes.

10/3 - 5:00 PM, Spahr Library

- Talked about Github. A few observations:
    - Since we forked a repo that's not ours, we cannot create issues. This was our primary organizational tool in Project 1. We will need something else.
    - However, we can still create projects. Also however, this is not an ideal replacement.
- We created branches similar to our structure for project 1:
    - Master (Protected, someone else must approve your pull request)
    - Exp (Protected, but you can approve your own pull request)
    - Everyone else's individual branches, which will merge into Exp ultimately.
        - When exp is in a good state, we will occasionally merge it into Master.
- We also made sure Postgres was running, Flask was running, and that they were working together. In essence, everyone had a working local environment to run.

10/4 - 11:30 AM, Eaton 2

- All previous changes mentioned in database structure are now implemented.
- Mason is absent, due to illness.
- First official "jobs" assigned:
    - Mason - Feel better, and, if necessary, work on styling/css/views.
    - Gage - Implement tasks into the db.
    - Zach - Make tasks interact with db.
    - Andy - Work on multi-day scheduling.
- No hard benchmarks were established, since everyone is extremely busy this week. We just want to get as far as we can. At this point all of us (excluding Mason) agreed the weekend would be the easiest time for us all to meet up and "grind it out"

10/6 - 11:30 AM, Lindley 412

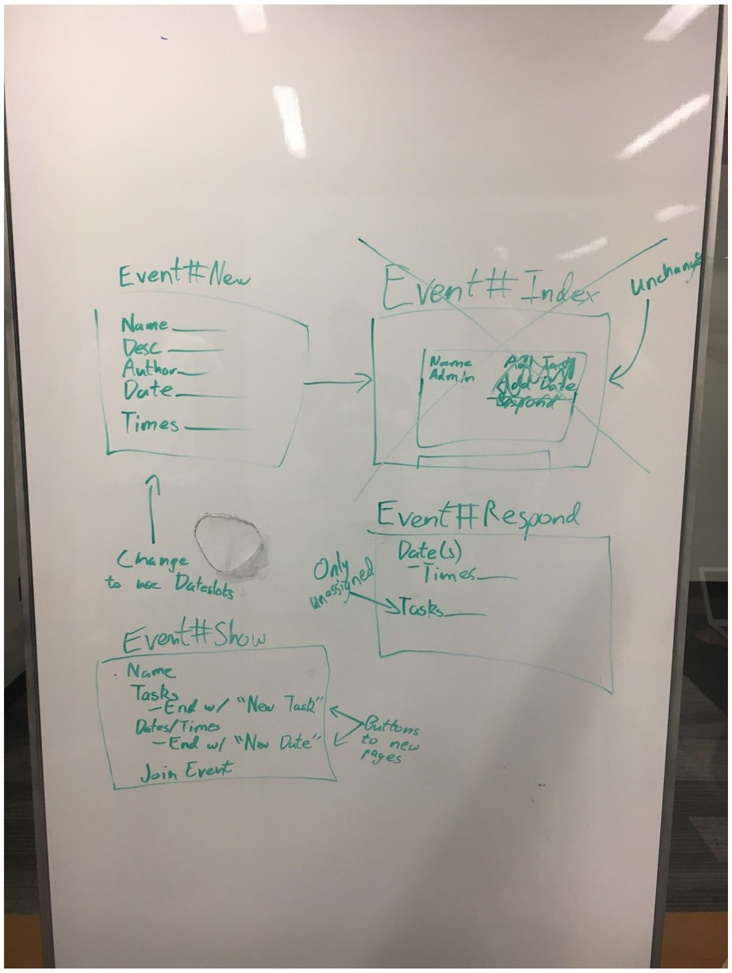
- Very little progress was made on this project.
- Mason absent due to ever-increasing, worry-inducing illness.
- The db was set up according to our specifications.
- The team decided that Task should be its own table, and should have a column for assigned/unassigned, and should link to the event id and the id of the person who claims the task, if it is assigned. Otherwise, it's value will be some null type.

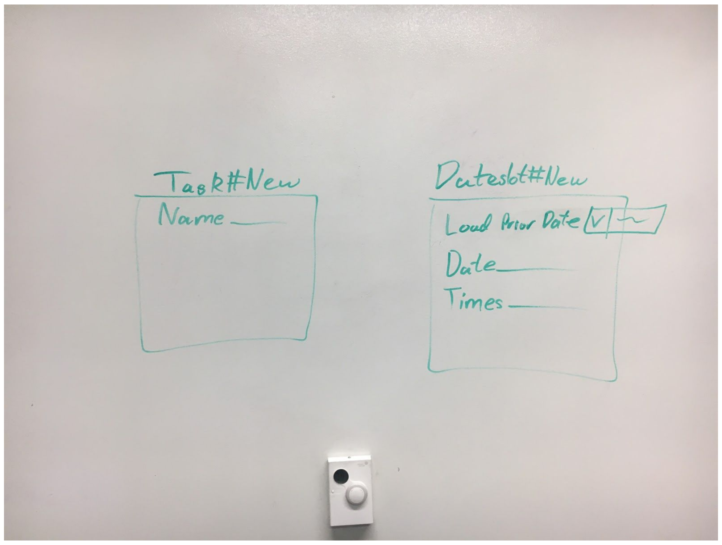
10/7 - 8:00 PM, LEEP2

- This is crunch time.
- What has happened so far: A beta version of the updated event#new page was made.
- What everyone is working on currently:
    - Gage: Tasks working right with the database

    - Andy: Multi-day scheduling, and understanding the form that the old team used, called WTForm
    - Zach: Data validation, specifically for tasks, since tasks are somewhat up and running.
        - Also - it was discovered that it is possible to create an event with zero timeslots. If time permits at the end, we may fix that.
    - Mason: Styling the events#new page, and ensuring that it works as intended.
- Problems so far:
    - WTForms has *very* poor documentation.
    - Some parts of the code, specifically for the timeslots, seem very hacky and unorganized. None of the team quite understands how it works.
    - The timeslots are very inextensible. We might have to find a different approach than switching everything to DateTimes.
    - Custom validation, like what we need for timeslots, or for a **list** of tasks (as opposed to a single object), is very complicated with WTForms.
- This caused a significant roadblock.
- Ultimately the team decided on the following structure, which will allow us to keep the current code as is, and working fine, without needing to modify it.
    - The idea behind it is, if it is difficult to incorporate the new requirements into what's already there, we split it up into it's own thing.
    - Their code interacts with the database. And we can change the db structure and it will still work fine.
    - If our code does not touch their code at all, and instead only interfaces with the db, we can create our own classes and files.
    - We concluded that interfacing with the db will be a "relatively easy" task, and we should do that as much as possible, instead of editing already existing code.

- We drew some diagrams of the high-level concept:





- What we need to do now:
    - Revert changes back to a point where we have the new db structure, but the original code for the forms is unchanged (and therefore working)

    - Change the backend of the form to a Date -> Timeslots structure (Assigned to Andy), push it up as soon as it's ready. This will be the base of the project going forward.
    - Create all our new classes.
    - If done correctly, it should not matter how hacky the original code was; ours will still work and look beautiful.
- In terms of new files/classes, we need:
    - Modify Event#Show, currently it displays the times, and the ability to join (in same page)
        - We need to change it to dates -> times
        - We need to add buttons which allow new tasks to be created, and allow the user to create new dates for the event
    - Dateslot#New - Expand the event to another day
        - Give the option to copy times from a different day (accessed from db)
        - Otherwise, create new times, potentially using the same obfuscated code that the old team had, if it works.
            - Other-otherwise, use checkboxes, since we know how to implement that easily.
    - Task#New - Allows you to create a SINGLE new task, and puts it in the db
        - Time permitting, we may allow multiple, if it's easy.
    - Event#Post(Respond) - Able to respond to an event with:
        - For each date, any times allowed from that date
        - Taking any unassigned task (has null value in db)
        - Following our philosophy of "if it's hard to do, split it up into different classes", if it's hard to keep dates and tasks together, split it up.
    - Event#New - Change backend, which Andy said he would do
- At this point in the meeting, Mason feels like he is on the verge of death. When he missed meetings earlier, he had a ***very*** valid reason for it. He is unconscious for a bit of this discussion. Seriously I hope you get better dude.
- In terms of who's doing what right now:
    - Zach: Validation, error checking, and potentially anything else that needs to be assigned. Also making sure we meet every requirement (including this document in a presentable form).
    - Andy: Rework backend (as discussed earlier) for easier multi-day, then work on multi-day.
    - Mason: Feel better, and if we have styling for you to do we'll get it to you asap.
    - Gage: Tasks. Creation, distribution, etc.

10/8 - 11:00 AM, Spahr Library

- Over the night, the following got done:
    - Significant work on multi-day events. The ability to add another day to event is finished, although the “copy another day’s times” feature is yet to be implemented.
    - The ability to create new tasks is almost complete. Some errors arise when we create one without a name.
    - Validation for tasks (old way), and the old team’s timeslots is working.
    - Mason is feeling quite a bit better.
- Things to work on, during this meeting:
    - Finish creating tasks, and add them to the db.
    - Join events on multiple days.
    - Style everything.
    - Validate everything.
- Same overall roles as usual. We all have work to do.
- At the end of this meeting (around 3:00 pm), more validation is complete. Joining multi-day events is *almost* there, but some validation issues were arising because WTForms is a wonderful library which makes perfect sense. Task creation is done. Task assignment is close; we can take one at a time, and there’s no validation. Trying to validate either crashes the program. Also, everything looks all styled thanks to Mason.
- This validation issue for task/multiday has the other three going crazy. If we don’t do validation, it works, but obviously that’s not ideal.
- Needing a break, the entire team splits off, and agrees to reconvene at 6:00 PM.
- Todo list created at the end:
    - Validation for joining events, creation of an empty task, assigning tasks, and potentially creating first/additional days.
    - Ability to copy timeslots from previous days.
    - Ability to view signups and tasks for an event concisely.

10/8 - 6:00 PM, Andy & Gage’s place

- During the time between meetings, Zach and Gage got everything with tasks working, from a user standpoint. The backend is a little messy, so if we have time it might be worthwhile to clean it up. All validation is in place for taks, both creating and assigning. As far as requirements go, task is done.
- Mason is working remotely, due to illness.
- In the meeting time we accomplished copying times from one day to the next. Only problem is now you can duplicate days. Validation guy (Zach) is working to make sure that can’t happen.
- Progress is being made on event joining.

- Things left to do:
  - Event creation, and viewing
  - When copying dates, our strategy is to reload the page and blank out the date input field. Currently we reload the page but don't blank out the field.
  - Validation, specifically right now to avoid duplicate dates.
  - Generate documentation
  - Works cited
  - Styling
- After significant progress made in an hour, we are all really close to being done, and we all head home and finish the project communicating via Slack.
- Things left to do now: A few things on validation, generate documentation, styling the last few pages, and rewriting the works cited and readme.
- At 10:26 PM, we send our final submission email to the Product Owner (Gibbons) with our team info and Github link

10/9 - 9:30 AM - RETROSPECTIVE MEETING

- We all met up before class to plan our presentation and have our Scrum Retrospective Meeting (Which conveniently ties in with what we need in the presentation)
- Overall, we all agreed we could/should've started a bit earlier.
- We're all grateful for documentation, now that we've all had to experience poor documentation and unreadable code.
- Never use WTForms again.
- Try to avoid having Saturday Night revelations in the future, requiring us to change our entire plan for the project
- Overall our weekend of hacking away was very successful.
- Created Google Slides, split up individual slides between us.
- Also, where do we turn this document in?