Schedule-4-Me

Team Retrospective Report

The Data Minors

Team Members:

Gage Aykroyd

Andrew Whitman

Dean Orenstein

Department of Computer Science and Engineering

Texas A&M University

11/23/2020

Table of Contents

1 Executive summary

Many scheduling applications exist, but very few or none plan your week for you. In order to benefit those who struggle to prioritize their to-do lists and complete tasks on time, our group decided to create a product which simplifies, streamlines, and automates the scheduling process. Schedule-4-Me is intended to plan out a user's weekly schedule with ease. A user simply defines recurring activities and creates tasks with appropriate due dates, priorities, and durations. Then, the application optimizes the user's weekly schedule in a preset format.

Our final product is a web application that functions as a one-time visit from the user, functioning to automate the task of creating an efficient schedule (starting Monday) from the tasks you have to accomplish over the week. Schedule-4-Me allows users to connect a Google account if desired. If the user does not have a Google account or does not wish to login with it, he or she will still be able to create a schedule and download it as a text file. However, if the user does login, he or she will be able to import a Google Sheets To Do List which auto-populates tasks. This makes the task creation process easier and much quicker. Additionally, a logged-in user will be able to export their final schedule as a Google Doc.

Our team began the project with fairly poor communication and cooperation. This hindered our progress in the early stages of development because some team members were putting in more effort than others, and certain areas of the project were well developed while others were under developed. However, as development continued, our communication improved. This created a better work environment and increased the quality and speed of production. By the end of the final sprint, our team had learned how to communicate and collaborate effectively. We understood each member's strengths and how to capitalize on each one.

2 Project overview

2.1 Needs statement

Many people have difficulty creating a schedule for their week that is easy to follow. It can be easy to forget to write down each task that needs to be completed. Additionally, it can be hard to know the best time to schedule each task depending on when a person eats, sleeps, goes to class, and goes to work. Most scheduling applications that exist require users to manually create tasks and place them in the desired time slot. This works well for some people, but for others it can be overwhelming.Therefore, our product provides a simple, easy to use auto-scheduling system which plans out the week ahead for the user.

2.2 Engagement analysis

Our team was at different levels of engagement during the entire development process. Three team members were at a high level of engagement during all of development while one member was back and forth with their contributions. This team member was not involved in writing code--whether that was HTML, Node.js, or Python--and she did not attend multiple lab sessions and scrum meetings. Although she did help write some sprint and user study documentation, this was not enough to be evaluated as an equal workload. The rest of the team was constantly working on the code, studying API documentation, and testing the product.

Overall, our team was slightly dissatisfied with the unequal engagement of team members. This created the challenge of redistributing tasks and putting in more effort in order to complete the product in the given time frame. One thing that added to this challenge was the fact that the reasoning behind this lack of engagement was not communicated early. This caused the team to be frustrated, confused, and less excited about the project. Once we understood the situation at hand, the challenge was approached with a new point of view, and we were able to handle the workload better.

We had many opportunities to innovate. These included the design of our website, design of our database, and how we would incorporate the Google APIs. The design of our website is simplistic and clean to promote an environment of focus and productivity. We constructed each page with a single idea at hand: login, task creation, schedule view, and success response. This allows the user to clearly understand the process flow and where they are at in the creation of their schedule. The design of our database created a backend which stores user schedule information in a format that is intuitive and easy to follow. In the code, it is easy to see how data is obtained from different tables. Additionally, this backend supports our stretch goal of being able to save a user schedule for future reference. Finally, our API integration allows for excellent use of Google Sheets and Google Docs. A user can simply paste a Sheets To Do list URL to import existing tasks, and select the Google Docs export option to receive their completed schedule in their Google Drive. At the conclusion of the project, we were mostly satisfied, yet one or two of us were still wanting to implement some features we originally planned but had not gotten to, including using the Google Maps and Google Calendar APIs.

2.3 Product analysis

Our final product is one which we are satisfied with and proud of. The working Google API functionality is something that was difficult to implement, but ultimately we created a high-quality application that uses those functions successfully. However, we were unable to complete the product to be representative of the original design; due to multiple challenges, we could not get to the implementation of features like a dashboard page. Our team learned the value of clear and frequent communication. We found that this was the foundation for our success. When we did not communicate, our progress slowed down and we were not on the same page. However, when we did communicate, development was at its highest performance and we all understood what was needed from each person. Our team could have saved time and effort in the early

stages of development by communicating all aspects of the project and our understanding of it. Instead, we did not feel the need to be proactive and learn about our APIs and each others' skill sets. This is something that we could have done better.

2.4 Work and Work Effort analysis

We are proud of our final product because it is something that took great effort and concentration to create. The use of burndown charts, scrum meetings, sprints, and GitHub all contributed to our success. Being able to quickly view the remaining tasks and time needed to complete them helped our team know whether we were ahead, on track, or behind our goal, which changed the pace at which we worked. More often than not, we were behind schedule. However, this allowed us to prioritize certain tasks and move others into the stretch goal category. Defining the timeline as three sprints allowed us to break down our tasks into smaller categories. These were mainly frontend development, Django implementation, and finally API implementation. This helped us focus on specific parts of the project which relied on each other, and created a good development plan. Additionally, GitHub allowed us to share code with one another and work remotely. One issue we encountered with GitHub was a particular sqlite file that got overridden with certain commits, causing the next person who pulled the code to have to either delete files manually or migrate changes and then redo some of the admin configuration. This, as well as committing more frequently and carefully, could have gone better.

When looking at individual development versus group development, the effectiveness of each varied from person to person. Dean tended to be more focused and productive when working alone whereas Gage and Andrew often produced a lot of code when pair programming. At the end of sprint three, all team members agreed to use Visual Studio's Live Share feature in order to save time, pushing and pulling through GitHub to avoid merge conflicts.

At the onset of the project, the work was evenly divided between the four members of the team. This was done in a way that tailored to each members' strengths. However, as time progressed, the tasks of one team member were picked up by other team members in order to complete them on time. This created an unequal balance of work between team members. Also, after Mary Ashley informed us of her situation, the workload distributed across Dean, Andrew, and Gage increased by about 25%. Thus, the workload we initially conceived differed drastically by the end of the project; the work done per day increased from about 1 or 2 hours to 5 or 6 hours.

2.5 Future projects

Considering future projects, our team would likely research the technologies that we plan to use more. We were caught off guard by the difficulty of the Google APIs, and we would not want to enter a similar situation. More research about coding languages and libraries would help us understand which would be best for our intended product at our level of skill. Additionally, as we have mentioned many times, communication is the ultimate skill of a good teammate. Those who communicate well are the ones who contribute the most to the success of a product.

Therefore, we would emphasize frequent and clear communication among all team members. We also learned, and would share with the class, that it is important to approach people who are struggling with grace and gentleness, not only for the sake of the project but also for the sake of our humanity. As a project manager, Dean learned that it is difficult to stay on top of the backlog, burndown charts, and scrum meeting agendas. Although it was really hard to listen to people and consider everyone's perspectives as equally important as his, Dean became more competent in his leadership abilities and interpersonal skills through his shortcomings.

3 Final design

3.1 System design

Because our application was web-based, a large portion of our design involved the various web pages that would need to be designed and how they would interact. Additionally, we considered the wide variety of frontend and backend frameworks we could have used and ended up going with Bootstrap for the frontend and Django for the backend, both of which were designed to make web development much easier (which was needed given our experience). We also figured that representing the logical flow of our application using a flow chart was necessary in the design stage. Because of the nature of full stack web development, design diagrams associated with object-oriented programming were not very applicable. Ultimately, the design of our system rested more on the logical flow and UI more than anything, for this was sufficient to guide the creation and prioritization of future tasks in the product backlog, and much of the project was a learning process.

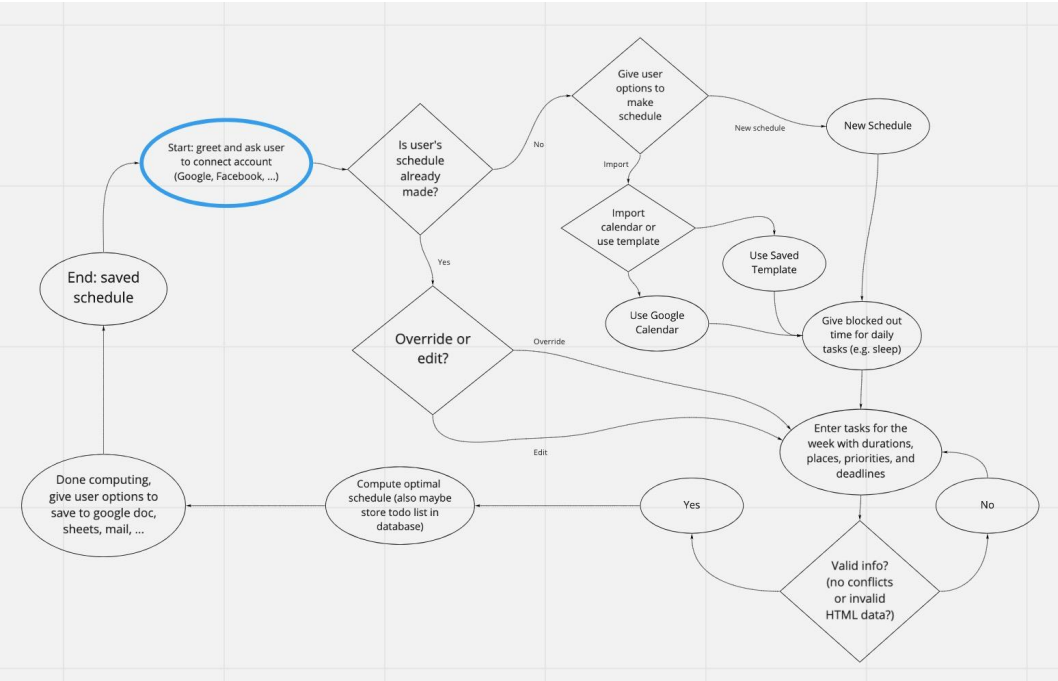


Figure 1: The Logical Flow of Schedule-4-Me

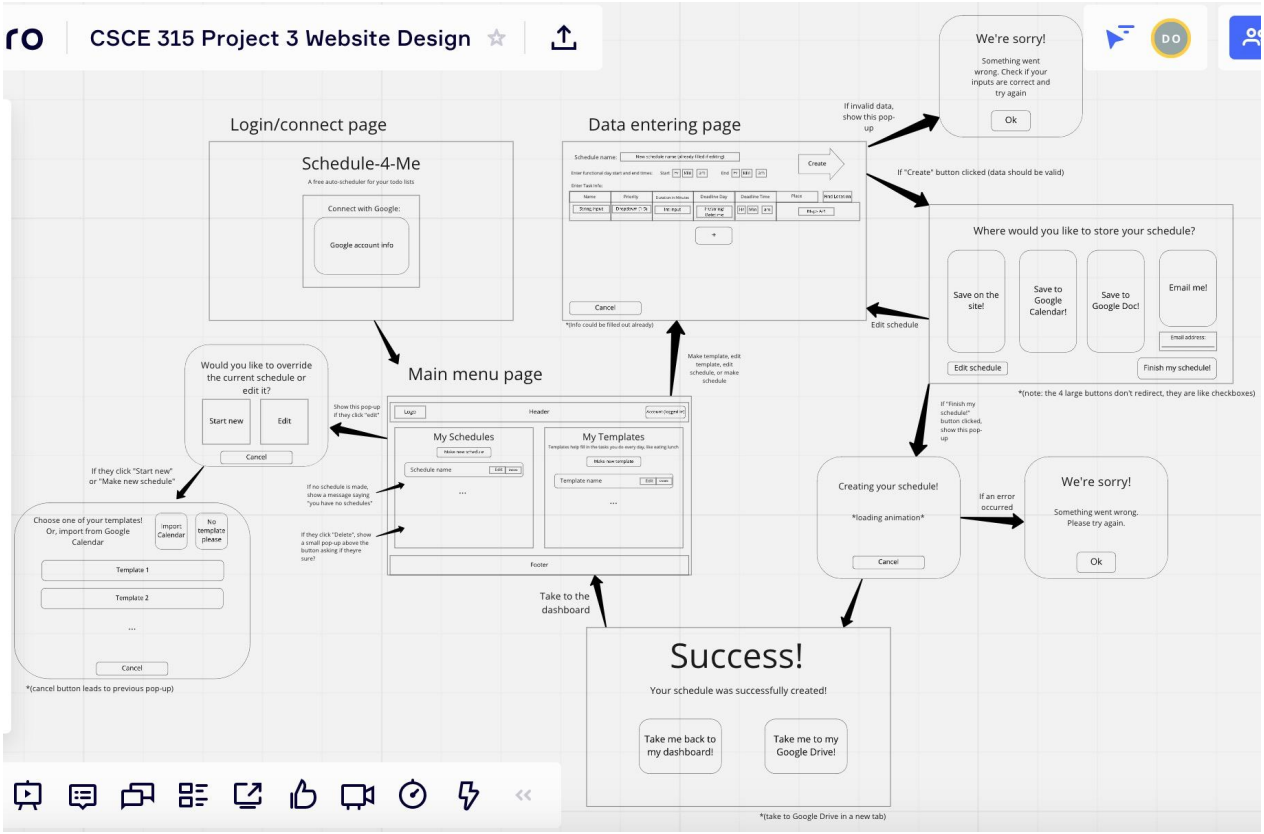


Figure 2: Initial UI Design and Page Interactions

3.2 Approach for design validation

Explain your internal team testing, then your user and usability testing. Point out changes to the system based on your user testing.

From the beginning of sprint 1 to the middle of sprint 3, much of our testing was done directly after one of us coded a new feature. This was typically done solo and involved running the python server, entering the localhost URL into our Browser, and then navigating to the page with the new feature to see if it worked appropriately and/or looked suitable. Eventually, entering a placeholder task on the data entering page got so tedious that we used hardcoded values to save time (starting in sprint 3). Error messages came from either the browser console or stack traces from Django's debug mode being on. Most of the testing we performed as a team was within the last few days of sprint 3 as we were testing API functionalities and ensuring that our system could go through the steps of logging in with Google, creating tasks after importing from Google Sheets, creating a schedule, saving the schedule to Google Docs, and going back to the data entering page to create another new schedule (though this is not the only route one can take when using the app, so we tested other routes as well). Collaborating over a shared virtual workspace in Visual Studio Code proved useful in this stage, because a team member could

write the code to fix a bug, which is often far easier than trying to explain it verbally to another team member.

User testing was performed via our user study. Conducted towards the end of sprint 2, our user study involved asking group 16 to join a Zoom call and collectively test our product. One of their members created some tasks for a test schedule and then created the schedule. Before, during, and after this study, we gained valuable feedback that helped make the application more understandable, easy to navigate, and intuitive.

3.3 Updates from testing, usability testing, and accessibility

Some of the features/changes we added to Schedule-4-Me post-user study included an “Instructions” button on the data entering page which clarified parts of the to-do list entering process that were not obvious at first glance (such as “passive” tasks), and we built an output page which shows the user their computed schedule, wherein they can get a visual representation of it before saving it to a particular medium. We also made some smaller adjustments, such as changing the “Cancel” button to a “Home” button and fixing spacing and alignment issues.

A large challenge we faced as a team was being short-handed and not knowing why for about two thirds of the project; Mary Ashley was undergoing some extenuating circumstances which greatly hindered our productivity. Thus, we did not get done with nearly as many features as we had hoped at the conclusion of sprints 1 and 2, and so we had to move some tasks to the stretch goal category or remove them entirely. This had quite drastic inadvertent effects on our design, both in terms of the logical flow and UI. Any application states from figure 1 that involved templates could no longer be considered, because by sprint 3, we still were needing to get the Google APIs working properly. Consequently, we did not have time to create a dashboard page, saved templates, or templates to assist in productive workflow. Not only was this part of the logical flow removed from a lack of time, but some of the other small features, such as a back button to edit tasks once a schedule is made and saving a schedule to Google Calendar, also had to be removed for the same reason. The major changes to figure 2 include changing the page containing toggle buttons for ways to save the schedule to an output page that also displays the computed schedule, not implementing some of the pop-up windows, connecting without a Google account on the landing page, and removing the main menu page and any buttons that would lead to that. These changes were needed because we were short on both people and time (especially in sprint 3), and there were a couple large bottleneck circumstances that hindered progress for up to a few days.

4 User’s Manuals

There are two paths a use can take, one which you must sign in to your google account and one where you do not. In the path that a user chooses to not sign in, on the homepage the user must click the skip button. This takes them to the data-entering page. On this page there is a place for the user to type the name of their schedule. Then there is the task-entering input fields. These input fields include name, priority, duration, passive, deadline day, and deadline time. All of these inputs are required except the passive checkbox. There is a green plus button that allows the user to add more tasks and each task can be deleted by clicking the corresponding red “X” button by the task. Additionally there are four buttons at the top right of the screen. A green “Instructions” button that pulls up a pop up with instructions. A blue “Time Block” button that pulls up a pop up that allows the user to enter in repeating tasks. There are three preset ones, and much like the task, a user can add and delete timeblocks (except for the preset ones). For a time block, the user can set the time, and the days that it occurs. The third button is a yellow import button which in this path (non-signed in) the user cannot use. A pop up telling the user to sign in will appear when clicked. The last button is a green outlined create button which submits the schedule to the algorithm. All input fields must be filled otherwise a pop up appears telling the user to finish filling out the input options. Once this button is pressed, it takes the user to the output page which displays the schedule. The user has two options, export to a Google doc or download to a txt file. On this path, when the user clicks on the download toggle button, a pop up will appear telling the user to sign in. Once the user has toggled their options, they can press the green outlined finish button which will do the toggled task and take the user to the success screen. On the success screen the user can go back to the landing page or be sent back to the data-entering page to create a new schedule. On the second path (when signed in using their Google account/gmail) all of the process is identical except the import feature and export features are available. When a user chooses to sign in, since our application is not verified, a Google warning page will appear. On this page the user will have to click advanced, then continue, so that they can sign in. This will be explained in more detail in the project debriefing section. A user can choose to not import a schedule on the data-entering page and follow the same steps as listed before. If the user does import they must give the full url of their Google sheet that is in the same form as the Google Sheets To Do template. If there are no styling/name changes of the sheet a user will have a page identical to the data-entering page (minus the import button) with some values pre populated. On this page the user can do all the same things as before. On the output page the user can now toggle the export to Google Doc option and receive their schedule in their Google Docs/Drive.

5 Project debriefing

Our team followed the basic Agile way of managing a team. We as a group picked tasks that we felt that we could accomplish, and any left over task were assigned by our team leader. We held four to five meetings a week with some being full scrum meetings and some just being quick informal meetings to catch up or understand something. We prioritized tasks in three levels, high, medium, and low. Additionally, we had tasks that were assigned as stretch goals. These were tasks that we hoped to accomplish if we had the time. If a task was not completed in the assigned sprint, then it would be pushed into the next one. As a group we prioritized the HTML

and JavaScript side of the project first since it was a part we as a team were more comfortable doing. After the majority of that was done, we then started to work on the API's and Django side of the project. Our team had three forms of communication: Discord, Zoom, and text messaging. Urgent things or general updates were sent in our group text message as that was what all of our team members routinely checked. Specific work things like helpful links and pictures were sent in the discord to help us while working as well as most of our meetings were held over the voice chat feature of Discord. Recorded scrum meetings were held over Zoom. All of our documentation was stored in a shared group Google Drive.

If we as a group were to do the project all over again, most of what we did would stay the same. The two main things that would change would be how we assigned tasks, and what order we focused on in our project. Instead of just picking tasks we felt comfortable with, we would try and focus on areas more. This happened naturally at the end of the project, but at the beginning of the project some members were working on one HTML page and other on another. This meant that we had to go back and forth and make stylistic changes to make things uniform. As for how we organized the priorities of the project, instead of focusing on what we already know at the beginning, we would try and mix it so within each sprint, a member's work would be fifty-fifty, what they are familiar with and what they are unfamiliar with.

At the moment there are a couple of slight safety and security concerns. The first is that our application is not verified by Google. What this means is that we have not submitted it to Google to review and make it a verified application. There is no reason that we should not be verified as we are simply reading and writing and not breaking any terms and conditions, we had just not yet submitted as we were going to wait until the site was hosted. This way we would not resubmit when we went from development to production. What this looks like to the user is when they click the sign in button on the landing page, a caution page from Google comes up saying that our application is not verified and that you may not want to trust it. There is a button that says advanced, and when you click on it, it gives the user the option to continue and trust our application. Once we are hosting the application, we will submit to get verification and the user will no longer have to worry about this.

The security issue we have is not so much of an issue, but a warning. The user does have the option (and is encouraged to) sign in with their Google account/email. Because we used the Google provided authentication, there should not be an issue. That being said, in today's world there is always a chance that a data leak or malicious attack can occur, especially when it comes to large companies. If Google were to have a security issue, that would affect our product, and therefore possibly our users. This is why we give the option to not sign in so that a user can still use our product, but they don't have to link their account if they do not feel comfortable. The only things a user would not be able to do is import and export from their google drives.

The only additional step that we could take is again to submit for verification. We could do this earlier so as soon as the product is live, the user does not have to worry about the trust alert from Google. This is something that we would do more research on earlier in the project so that

we could have it ready as soon as possible. The verification process can take anywhere from a couple of days to a couple of weeks.

Our product is in a place where it can be hosted and deployed in a live environment for anyone to use. It is stable, but it does have some issues. These issues are mostly really obscure things that a user could do that might break the program. For instance, if a user has “Monday” as one of their task names, it could create an issue with splicing and lead to scheduling issues. Another issue is if when submitting a url to be imported from. If the user does not submit a valid Google sheets url, the application will show an error page. So while the product is in a stable and working condition, there are some validation things that still need to be done and updated. That being said, there are still features that our team would like to implement in the future. For instance, we want to connect the Google Calendar API to the application so that a user can import their schedule directly into their calendar. We also want to add the ability to save and edit user schedules depending on their Google account. These are things that we feel could be added in website updates in the future. Lastly, like mentioned above, we need to do the verification process so that users know that our application is trusted by Google.

6 Retrospective Meeting Agenda

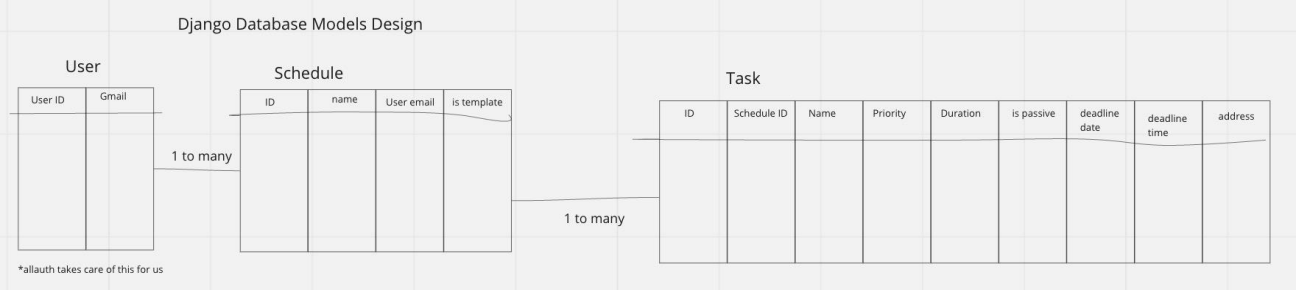
Time and date: 9:00 PM on November 24, 2020
Place: Discord call
What we talked about:

- When went right? What went wrong?
- How well did we assign tasks? Was anyone too overwhelmed?
- How well did we communicate? Did anyone ever feel left out or not up to speed?
- How was overall organization and management?
- Final thoughts

7 Appendices

Database Design for Future Features:

The following figure illustrates the few Django database entities (called models in Django) we would have used in the dashboard page and perhaps in other future features:



Useful Links / Help:

The following link shows the majority of the Django walkthrough we followed as we set up Django for the project. It also includes helpful links to videos and documentation, some steps we took to try to move the code to production, and links to the Miro design drawing and GitHub:

https://docs.google.com/document/d/1JXJSUZngKAXCLmA3l3_7QLQAe-sSz8Bc2AigHfW_hLA/edit