COMS 4170: User Interface Design

Fall 2019 – Prof. Brian Smith

Team 6 – Final Project Report

WURPIT – When Ur Registration Process is Terrible

Spencer Bruce – sgb2145 Nadia Jurkovich – nmj2120

Vishnu Nair – vn2287 Sam Ross – shr2118

Table of Contents

Project Concept and Introduction

WURPIT is a streamlined, organized, and individualized support system for semester course selection and long-term major planning, currently targeting Computer Science (CS) students at Columbia. The current systems in place for course planning are not particularly useful to CS students, as they are not specifically tailored to the complicated CS major system. WURPIT will allow students to easily navigate between the six tracks of the CS major, allowing them to see what classes they have left in each track and how the courses they have already taken fulfill various requirements. It will clearly display important course information like specific track classes and prerequisites, making it easy for students to plan their courses and pick their track based on what they have taken/want to take in the future. By streamlining the process of selecting courses, WURPIT will ease the logistical stress of planning out a CS major and provide advising support for students, which is in high demand.

There are over 1,000 different combinations of courses that will fulfill a major in Computer Science at Columbia, and there are around 300 undergraduate majors at Columbia. There are only ten dedicated Computer Science advisors, and the all-around academic advisors admit to not understanding the CS curriculum enough to offer trustworthy guidance. This is an important design problem. Though some advising is available, it is infeasible to expect the current faculty advisors to be able to sit down with *all* of their assigned students to talk through their individual course plans. Students need assistance with the ins and outs of the CS curriculum, but are generally provided little support for both long- and short-term course planning. Many students find this aspect of being a CS major difficult and frustrating, and they often resort to the construction of ad-hoc spreadsheets to try and make sense of their various graduation requirements. This only adds to the stress one may experience as a student here, which can often be overwhelming for some. WURPIT will make this process simpler and more straightforward, increasing a user's enjoyment in their education.

There are 93 individual courses offered in CS this semester (Fall 2019). Although this list of offerings may seem lengthy, not all courses offered by the department are available each semester, extending the list even further. This makes it difficult for a student to predict what their schedule may be like in the future, as course offerings can be unpredictable. Our design solution in

WURPIT is ambitious because it seeks to transform the current course listing system. The current resource for CS students is a long list of classes and the information associated with that course's reference code. It has no innate prerequisite or track structure, forcing students to switch back and forth between many sites just to figure out what they must take and, more importantly, if it will be offered. To make our solution useful, we need to come up with a way to succinctly – but informatively – display ~100 different courses each semester. It is necessary to not make this amount of information overwhelming, and we must show users exactly what they need while making it easy for them to try out many versions of a schedule. What is missing from the present system(s) is a visual interface that makes this information easy to absorb. It is simple to list a bunch of classes, yet it is difficult for a student to absorb this list. However, if it were simple to design an interface to make the CS curriculum easier to maneuver, the CS department would have done it already. It is a challenge to clearly display such a volume of intricate information. There are administrators at Columbia who spend large portions of their time answering the questions of confused CS students who have trouble figuring out their courses. Our design should lessen this burden – not solely the one felt by the administration, but also that of the students.

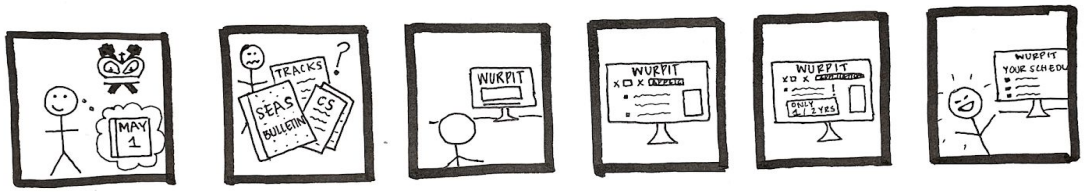
Target User Population

Our chosen target user population is students in the Columbia Computer Science curriculum who require support in their schedule planning and course selection process. These students may be new to Columbia, interested in the Computer Science major/department, or perhaps just curious about what classes the CS department offers. As we detailed in the introduction, there are many hurdles that *every* student in the Columbia CS curriculum must overcome in order to graduate with a degree from the program. Along the way, they must keep track of prerequisites and exemptions for classes, major requirements, track requirements, frequently changing course offerings, a frustrating waitlist system, and more. Our target user population may not understand how to maneuver these multiple moving parts, or perhaps they have some experience with the department and its specifics, but would benefit from added support aside from the advising already offered through the department.

Although all of the members of Team 6: WURPIT are students in Computer Science here at Columbia, we feel that this does not detract from the benefits that can be created through the

construction of our project. With the perspective of three undergraduate juniors and one PhD student, we all remember times where additional structured documentation would have made our lives as students easier. With Computer Science being the second most popular major here, the potential positive impact from our design is quite substantial. We choose this target user population with the hopes that, in future design revisions and project implementations, the user population can expand to *all* students.

Storyboard



Alex is a sophomore in SEAS at Columbia and is studying computer science. He is preparing for his next (5th) semester class registration.

When researching classes, he is overwhelmed by the number and possibilities of major requirements, track requirements, and electives. "I have no idea what to take. This is so overwhelming!" he yells.

Because Alex's CS adviser doesn't have time to meet with him, and Vergil and CULPA are flooded with a lot of information, Alex is stressed out. So, he goes onto WURPIT to get some suggestions and a better structured layout.

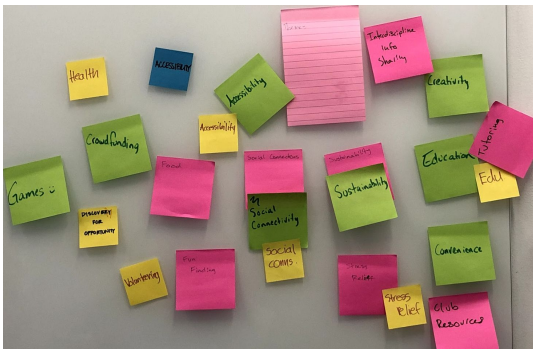
He wants to pursue the applications track, so he filters the CS courses by track ("applications"), and views all of the required and elective courses for that track that are available next semester.

He notices that one of the classes is only offered once every two years, and that he should take it.

After spending some productive time on WURPIT, Alex now has a better organized idea of classes to take and is all prepared for registration!

Brainstorming Process

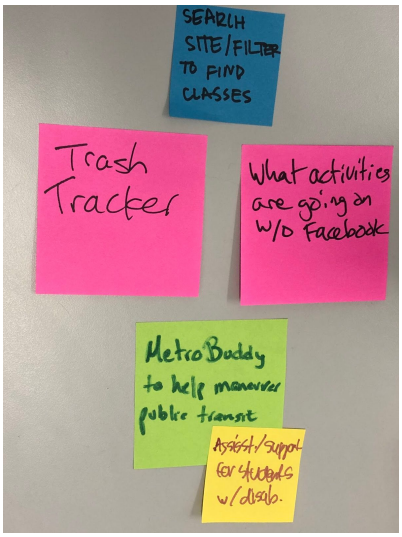
First, our team brainstormed various possible themes for our project (e.g., sustainability, social connection, and education). We documented our ideas by writing each theme on a sticky note and placing each note on the wall in a group study room. Multiple sticky notes with the same theme indicate that multiple members of our group thought of the same general theme.



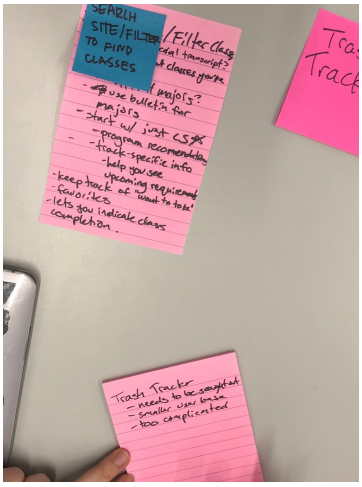
Next, we grouped the themes by topical similarities and moved the groups to different parts of the room.

Then, we each stood up and walked around the room, looking at the ideas for themes. As we walked, we brainstormed more specific project ideas related to each theme. These project ideas were written on more sticky notes and placed on the wall near their accompanying theme.

Afterwards, we discussed all of the notes added for each theme. The author of each note described their thinking process that spawned their idea. After listening to each description, we each picked a favorite topic and discussed what features the user interface could potentially include.



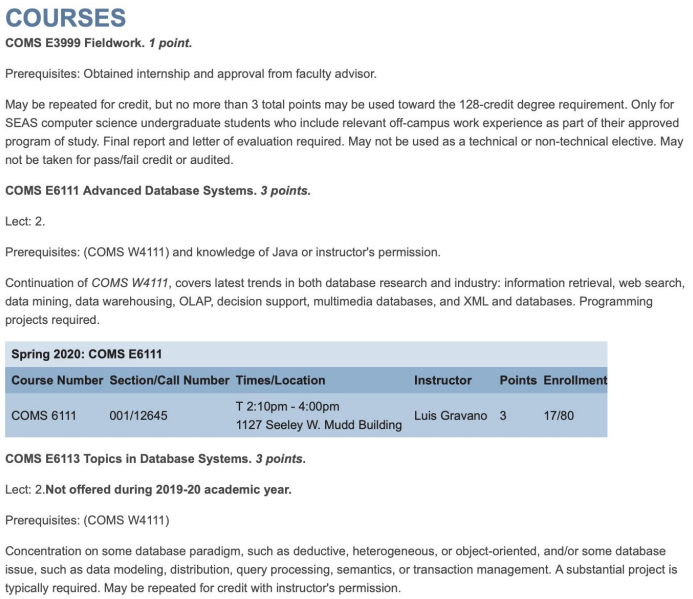
As we discussed ideas for the more specific interfaces, one sticky note began to stand out with an idea in which we all shared interest, sparking excitement and curiosity in the group. After further discussion, we all agreed that we were most excited to make a user interface that would help students easily search for their classes. Concluding our brainstorming, we started to write out some specific bullets describing what UI elements we thought up during our conversations.



Comparative Analysis

Existing design solutions are often tedious to use and have very few tools to help students plan out their semester and major. Some specific existing solutions are:

School/Major Course Bulletin *(CS course bulletin)*: The course bulletin is a great resource for finding what classes *could be* offered in a certain major at Columbia. However, the issue lies in how tedious it is to peruse the bulletin. The list is generally static, with *all* classes listed regardless of whether or not they are offered the current or the following semester. Getting a list of classes (or moreso, classes relevant to the student) is tedious and time-consuming. Although prerequisites are labeled, there is neither a clear hierarchy nor order displayed for these prerequisites. Our solution will need to make it easy for students to get an up-to-date, ordered list of classes (with their prerequisites) being offered that can be easily perused. However, we will also need to include (and clearly delineate) courses from recent, previous semesters so that students can plan ahead for courses that may not be offered every semester.



COURSES

COMS E3999 Fieldwork. ***1 point.***

Prerequisites: Obtained internship and approval from faculty advisor.

May be repeated for credit, but no more than 3 total points may be used toward the 128-credit degree requirement. Only for SEAS computer science undergraduate students who include relevant off-campus work experience as part of their approved program of study. Final report and letter of evaluation required. May not be used as a technical or non-technical elective. May not be taken for pass/fail credit or audited.

COMS E6111 Advanced Database Systems. ***3 points.***

Lect: 2.

Prerequisites: (COMS W4111) and knowledge of Java or instructor's permission.

Continuation of *COMS W4111*, covers latest trends in both database research and industry: information retrieval, web search, data mining, data warehousing, OLAP, decision support, multimedia databases, and XML and databases. Programming projects required.

Spring 2020: COMS E6111

Course Number	Section/Call Number	Times/Location	Instructor	Points	Enrollment
COMS 6111	001/12645	T 2:10pm - 4:00pm 1127 Seeley W. Mudd Building	Luis Gravano	3	17/80

COMS E6113 Topics in Database Systems. ***3 points.***

Lect: 2.**Not offered during 2019-20 academic year.**

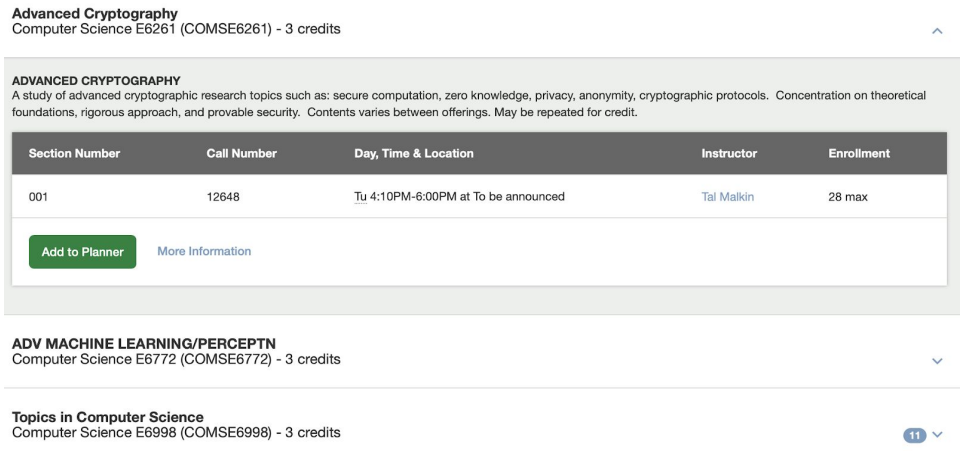
Prerequisites: (COMS W4111)

Concentration on some database paradigm, such as deductive, heterogeneous, or object-oriented, and/or some database issue, such as data modeling, distribution, query processing, semantics, or transaction management. A substantial project is typically required. May be repeated for credit with instructor's permission.

Columbia Directory of Classes: The school's course directory is a comprehensive resource for viewing a list of classes that will be offered next semester. However, it neither visually shows prerequisites for classes nor does it show a hierarchy of classes according to prerequisites and other requirements, making it difficult to plan out a path for future semesters. Like the course bulletin, getting a list of relevant classes can also be a time-consuming process. Our solution will need to offer multiple (convenient) visualizations of class hierarchies, specifically in the context of their prerequisites, by clearly laying out paths that students can take to fulfill these requirements. There also should be a clear delineation between previously-offered courses and currently-offered ones.

Spring 2020 Department: Computer Science	
Spring 2020 Computer Science W1004 **INTRO-COMPUT SCI/PROG IN JAVA**	
Section 001	*INTRO-COMPUT SCI/PROG IN* **Call Number:** 12611 **Points:** 3 View in Vergil **Day/Time:** TR 1:10pm-2:25pm **Location:** To be announced **Enrollment:** 209 students (300 max) as of November 26, 2019 **Instructor:** Adam H Cannon
Section 002	*INTRO-COMPUT SCI/PROG IN* **Call Number:** 12612 **Points:** 3 View in Vergil **Day/Time:** TR 2:40pm-3:55pm **Location:** To be announced **Enrollment:** 199 students (300 max) as of November 26, 2019 **Instructor:** Adam H Cannon
Spring 2020 Engineering E1006 **INTRO TO COMP FOR ENG/APP SCI**	
Section 001	*INTRO TO COMP FOR ENG/APP* **Call Number:** 19671 **Points:** 3 View in Vergil **Day/Time:** TR 5:40pm-6:55pm **Location:** To be announced **Enrollment:** 188 students (189 max) as of November 26, 2019

Vergil: Class listings in the directory often link to Vergil, which is touted as a "course planning guide." Vergil offers a more visually attractive, compact view of classes that are/will be offered during the current or next semester. However, many of the deficiencies of the class directory can still be found here as well (no visual prerequisites/hierarchies, tedious, etc.). Although Vergil has class planning features, they are limited to the current and next semesters, and often are not updated with special topics courses or newly-added sections for popular CS classes. The needs raised with the class directory still apply here. However, in addition, we must also offer class planning features that span a student's entire career here at Columbia.



Student Services Online: Student Services Online, or SSOL, works well during the final part of the class planning process: registration. It features a "Wish List" where users can save/favorite classes they are interested in and features functionality to import classes from the planner in Vergil. It also allows searching for classes. However, SSOL cannot track favorites over each semester because the wish list is purged when registration officially ends and, like the others, finding classes can become tedious, especially with the listing of individual research/project sections amplifying the size of the list. Again, our solution will need to allow planning over multiple semesters and allow visually convenient ways of keeping track of classes.

ADD 12612	COMS 1004 W sec:002 **INTRO-COMPUT SCI/PROG IN**	Adam Cannon	Tu Th	2:40pm-3:55pm BTBA RTBA	01/21/2020 05/04/2020	Open
MORE 20074	COMS 3101 W sec:001 **PROGRAMMING LANG (PYTHON)**	Lawrence Stead	We	6:10pm-8:00pm BTBA RTBA	01/21/2020 05/04/2020	Waitlist
ADD 20135	COMS 3101 W sec:002 **PROGRAMMING LANG (JAVASCRIPT)**	Ramana Isukapalli	Mo	6:10pm-8:00pm BTBA RTBA	01/26/2020 03/14/2020	Open

Overall, the four existing solutions generally do the same thing: they provide a list of classes alongside very limited planning features. However, they do not provide a convenient interface with which a user can plan out their projected career as a student in CS at Columbia, leading to confusion and frustration. These needs are what we plan to address in our solution.

Risks to Mitigate

Pursuing such a project is inherently risky from design and technical standpoints. Any mistake in how information is rendered or initially entered can cause the system to communicate incorrect information to the student, which could lead to accidental or incorrect registration and may consequently put them behind on fulfilling degree requirements. Since the use of this system can affect a student's academic trajectory, we must take the utmost care in creating our solution in mitigating some of these risks. Such considerations include:

The recency, completeness, and flexibility of class information: We must ensure that the system includes *all* classes that are offered in CS. We must also ensure that the status of these classes and corresponding sections are as up-to-date as possible, especially during high-volume periods (such as the start of registration and the change-of-program period). We must also guarantee that this part of the system is as flexible and responsive as possible, easily allowing for the addition of new classes and sections.

The correctness of prerequisites and other requirements: We must ensure that prerequisites and major/track requirements are 100% correct in the system. Any mistakes can negatively affect a student's registration, making the process more difficult for the student or even putting them behind. Thus, we need to ensure that the information we pull from Columbia sources is indeed pulled and processed correctly. This information must also be clearly displayed to the user so that there is a clear understanding of these requirements.

Clarity of information communicated: We must ensure that the system communicates the information it has to the user clearly and effectively. Registration begins toward the end of the

semester when students have final exams, papers, and projects due. We do not want our system to cause students additional anxiety during this already-stressful period. Thus, we need to ensure that our interface is as clear and easy-to-use as possible.

Availability of course information from previous semesters: We need to figure out how we can pull course information from previous semesters. Such data can be helpful for students as some (required/elective) classes may only be held during certain semesters/years. It can also be helpful for us (i.e., the system) to possibly predict when a certain class may be next held, which can help a student's planning process. Finding this information in a timely manner to avoid setbacks is ideal, as this information is integral to the functionality of our design.

In general, the risks associated with building out this system lie mainly in communicating the class, major, and track data that we pull. As mentioned before, we do not want this system to be responsible for any students registering incorrectly and being left behind. Thus, our design process will take the utmost care in ensuring that the information communicated is clear and correct.

Prototyping Process

Before we began designing and prototyping WURPIT, we needed to evaluate the needs of our target user population. Given that all of our team members belong to the population of computer science students at Columbia and Barnard, we need to take special care to ensure that our own biases do not affect the design process. In order to evaluate whether or not our proposed idea indeed meets the needs of the population we are targeting, we went through needfinding in two different ways: by sending out a survey to undergraduate CS students and by executing a contextual inquiry user study with two current undergraduate CS students.

Needfinding (Survey)

We created a simple survey that assessed an undergraduate student's satisfaction with the current class scheduling/planning process and with planning tools currently available at Columbia. The survey (included in our package as "Needfinding_Survey.pdf") asked the following:

- The student's year and undergraduate school **[multiple choice]**.
- The student's current track (even if they are undecided) and how confident they are that they will remain in their current track **[multiple choice]**.
- The perceived level of usefulness of certain currently available tools **[5-element Likert scale with an extra option to indicate that they do not use the tool in question]**.
    - *Tools:* SSOL/MyBarnard, Vergil, the CS Bulletin, the student's adviser, MICE, personal organization system/spreadsheet.
- The perceived availability of the student's CS adviser **[5-element Likert scale with an extra option to indicate that they do not meet with their adviser]**.
- The perceived ease of the current scheduling process **[5-element Likert scale]**.
- The perceived ease of navigating the different tracks **[5-element Likert scale]**.

We posted the survey on Tuesday, December 3 in the "CS @ Columbia" Facebook group and the "COMS GU4170" Piazza group and received a total of 17 responses (representative of all four schools: CC, SEAS, GS, Barnard) by EOD Saturday, December 7. According to the results (included in our package as "Needfinding_SurveyResults.pdf"), users generally:

- find the current scheduling process difficult (average: 2.6/5.0).
- find it relatively difficult to navigate tracks (average: 2.8/5.0).
- do not find their adviser useful (average: 2.2/5.0 of those who do meet with their adviser).
    - 10 out of 17 respondents indicated that they *do not* meet with their adviser at all.
    - Those who do meet their adviser indicate that their adviser is generally unavailable (average: 2.1/5.0).
- find their personal spreadsheet/organization system the most helpful (average: 3.8/5.0 for those who do use it),
    - followed by SSOL/MyBarnard (average: 3.2/5.0 for those who use it),
    - and then the CS Bulletin (average: 3.1/5.0 for those who use it).
- do not find MICE to be useful (average: 1.2/5.0).
    - Only 6 out of 17 respondents indicated that they use MICE.
    - These numbers are notable in that MICE is the only existing system out of those mentioned that officially acknowledges the existence of tracks in CS.

Needfinding (Contextual Inquiry Study)

We executed a contextual inquiry with two current CS undergraduate students (one Columbia student, one Barnard student). The goal was to observe how a typical student plans out classes and navigates between tracks using currently available tools. The outline of the study is included in our package in "Needfinding_Study_Outline.pdf," and the actual handout given to each participant can be seen in "Needfinding_Study_Handout.pdf."

In short, we asked each participant to adopt the persona of "Lola," a first-semester junior studying computer science (the choice of school is dependent upon what school the participant is in). Lola is trying to evaluate her performance in both the *Vision and Graphics* and *Intelligent Systems* tracks. Given the CS-specific courses that Lola has already taken (along with a blank computer and some scratch paper), we asked the participant to use whatever existing class planning tools are available to evaluate Lola's progress in both tracks and pick the track that is better suited to her current progress. After picking the best track, we then asked the participant to craft a realistic schedule for Lola for the Spring 2020 semester. We created the Lola persona so that the participant did not base their activity on their already-planned schedule for next semester. The participant's screen was recorded for future reference and audio purposes, and signed consent was received before the study progressed. The user's notes taken from the study are included in our package in "Needfinding_Study_ParticipantNotes.pdf".

Afterwards, we interviewed each participant to learn how they feel about the current scheduling/planning process: What were their difficulties? What were the pros/cons of the tools and methods they used? We also asked them what changes they would make if they had the ability to make the process easier. Notes we took during the interview and user study have been uploaded as "Needfinding_Study_OurNotes.pdf" in the compressed .zip file.

Both participants picked the *Intelligent Systems* track as the best track for Lola and picked mostly similar classes for her Spring 2020 schedule. Participant 1 found the process frustrating, saying that the information needed was "all over the place," which thus made information hard to find. They also said that the process was not very approachable and that it took longer than it should. If they could improve the process, they would have everything in one place and have

instructions for each tool's functionalities (for example, they did not know that one could search words in SSOL's class search box).

Participant 2 called the process "stressful" and clearly relayed the fact that they were struggling throughout the test itself. They did not like how the necessary information was so disorganized nor how difficult it was to view prerequisites for some courses. They also remarked on how cumbersome it was to switch between various tools and how the process was overwhelming. If they could change the process, they would like a personal checklist and a timeline function for viewing the order in which to take certain courses.

User Notes from Needfinding (Contextual Inquiry Study)

Refer to "UserStudyNotes.pdf".

Takeaways from Needfinding

Looking at the results of our survey and contextual inquiry, we can conclude that the status quo *does not* effectively meet the needs of our target population. Our survey shows that students have a relatively hard time navigating the class planning process. Even resources that are meant to help students in this process (e.g., advisors) have proven themselves to be unreliable and underdeliver in terms of their intended purpose.

We specifically found a problem when it comes to deciding and registering for courses in a track. Of the 17 participants in our survey, 14 indicated between a 1 and a 3 (out of 5) on the ease of navigating the tracks, with a 1 being very difficult and a 5 being very easy. Here our data skews left and shows many students having trouble navigating the various tracks the CS major offers. In addition, MICE is the only resource of the many available that allow users to note their CS track requirements. In our survey, only 6 of 17 participants indicated that they use MICE; of those six, five declared it as a 1 (out of 5) on the level of usefulness ("not at all useful"), and one declared it as a 2. This indicates that, while CS students at Columbia and Barnard experience difficulty when researching and registering for track courses, there are no available resources that even mildly assist with this specific aspect of the registration process. Overall, our contextual inquiry showed firsthand the difficulties that CS students face in planning and scheduling classes.

However, we also found that there is room for improvement. From our survey, the most helpful tools are the tools that do not exist publicly (i.e., they were created by the students themselves in the form of spreadsheets and other personal systems). Both of our contextual inquiry participants also gave specific suggestions on how they would make the process easier. This information will be used throughout our design process, first with the creation of prototypes.

Lower-Fidelity Prototypes

Using what we learned from the needfinding process, each member of the team worked on their own hand-drawn low fidelity prototype. This was so we could prototype over multiple points of the design space and gather a mix of ideas for our final higher-fidelity prototype. The prototypes are included in our .zip file as "Prototypes_Lo-Fi.pdf" and a brief explanation of each follows:

Nadia (pg. 1):

For our user interface, I envisioned the home page being a list of all available courses, with the option for users to filter courses by school, major, track, track electives (given track), tech electives, and non-tech electives. Given that our user population is CS students at Columbia and Barnard, the home page would list all available CS courses for either the current or next semester (depending if there are classes available to view for the next semester). From there, the student can select which track they're on and can view the available track requirements and qualifiable electives. They can also view elective courses not relating to track, but qualify for completion of the CS major (divided into tech and non-tech). The second main page shows the user's profile, with tabs "my profile", "my class schedule", "checklist", and "wishlist". In "profile", the user can indicate what school they're in, their major, track, and year, so they don't have to continually filter by these options when browsing courses. In "my class schedule", the user can view what courses they've added to their upcoming semester's schedule, and can view a visualization of it as well. In "checklist", the user can see which courses they've already taken, and what others they need to take to fulfill their CS requirements (in addition to being able to directly check/uncheck classes from the list). Lastly, "wishlist" shows which courses the user has favorited, indicating the user may not have been able to take the course previously or currently, but wants to in the future. (S)he

can reference this in the future when deciding they have room to take a course they've wanted to take before.

In terms of mitigating risks, one of the largest risks addressed in Progress Report 1 was the lack of clarity and organization of an abundance of information in already available registration resources, emphasized in survey and contextual inquiry results. To mitigate the risk of our design being overwhelming in terms of content, I made sure to address the visual aesthetic of my prototype, curating tools like a directly-manipulatable checklist and visual schedule. In our higher-fidelity prototype it is important to continue to emphasize a simple visual design, where users aren't bombarded with an abundance of text and lists of courses.

Spencer (pg. 2):

I wanted my design to be a card-style layout to add some direct manipulation to the classes that one plans for. The status quo implementations for course scheduling do not allow means of visually distinguishing which classes one has taken, which classes one wants to take, and future plans. I chose to break up my design into two main parts: one part where the classes are represented by interactive cards in columns separated by priority and one part where a weekly schedule is shown. On the side of these two parts exists a navigation bar that lists *My Schedule*, *Track Progress*, *All CS Courses*, *Professor Review*, *Future Planner*, and *Explore*.

The *Explore* page works as a search interface and references all of the courses available in the computer science departments of Columbia and Barnard. From these searches, the user can add results to their wishlist to be added to their schedule later.

The *All CS Courses* tab lists all of the CS Courses offered for the current and next academic semesters, sectioned off by topic and prerequisite. This addresses the problem where prerequisites are not always listed nor prioritized. This also shows the user only the courses that are *currently* offered in CS, as there are many courses that are no longer being offered that still exist in the status quo bulletins.

The *Professor Review* scrapes reviews from CULPA.info and places them in an easy-to-reach place, allowing CS students to check on professor ratings before scheduling for courses. Although not a necessary addition, it is a nice one nonetheless.

The *Track Progress* tab is the 4 column idea I expressed earlier. The user can drag and drop different wishlist or required classes into their respective columns to see them reflected in the *My Schedule* tab. After one scrolls down, they can see a checklist and a pie chart that displays track and requirement completion.

The *My Schedule* tab shows the weekly projected schedule for the current registration period (Spring of 2020 for the Fall of 2019, etc). This reflects the additions added to the "Future" column in the *Track Progress* tab. The user can hover over a class, and a tooltip will appear displaying the description, meeting time, professor, and location of each class, along with any prerequisite or registration cap information. A progress percent and requirement fraction is displayed along the bottom to remind the user of how much of their major is left to be completed. This page is considered the "home base" for my low-fi design, as I believe it delivers the most frequently-referenced information .

The *Future Planner* page would give a large overview checklist of the student's major progress, similar to the one found at the bottom of the *Track Progress* tab. This gives a good view of what each class translates to with degree requirements, something that is not seen in the status-quo implementations.

The risk of displaying the frequently-changing course information for the whole university is mitigated through only referencing the CS curriculum. Additionally, the clarity and uniformity of displaying each class as a box allows for a colorful and interactive way to avoid information overload. I mitigated the risk of searching through previous semester offerings by removing the function of viewing when an individual course was last offered at Columbia. This function cannot be feasibly implemented under the time constraint and does not add much utility to our interface, so I believed it was okay to remove.

Vishnu (pgs. 3-6):

I focused on classes and major/track requirements in my prototype. Here, I envisioned the interface as conforming to a simpler list format that lists out all of the requirements for the student's degree program, including core classes and elective/track classes. This list would show what classes the student has already taken, what classes they are taking right now, what they still need to take, and how all of these classes count toward their degree program. The goal of such a

list format would be to clearly delineate what a student needs to take and what a student still has to do in order to make progress.

I also prototyped a class info screen that visually shows a network of prerequisites for the class (and what progress the student has made toward these prerequisites). The screen also puts the class in the context of the student's degree program: Will this class count toward anything? If so, how could it count specifically and why? With this adviser-like approach, students can rest assured that they are making a fully-informed decision when registering for a class. The screen also features regular class information pulled from SSOL so that the student does not have to jump between programs to view this information.

Some risks we cited in Progress Report I include the correctness and completeness of class information and prerequisites, the clarity of information communicated, and the availability of information about previous semesters. My prototype ensures that class information is correct and complete by pulling it directly from SSOL. It also shows this information clearly in the class info screen and shows the prerequisites visually. The interface depicted also briefly mentions how often the class is offered.

Sam (pgs. 7-8):

For my prototype, I wanted to focus on making the scheduling process less stressful to students. To achieve this, I made the main, initial screen display the student's progress on the right side, and a checklist of everything the student has to do on the left side. On the right are three pi charts displaying how far through the user is with the core, with their track, and with the major overall (including various math requirements that are not part of the core or track classes). This allows students to quickly and easily view their progress. A large source of stress for students is feeling like they don't know how much progress they have made, so displaying that clearly was a priority. During our needfinding process, our participants expressed a desire for a checklist that shows them the courses they have done and still have to do. So, on the left hand side is a checklist specifically tailored to their track, so they can see within the track what classes they have done and what they still have to do. What is unique about this interface is that users can easily toggle between tracks to see what their progress would be if in another track if they were to switch. If they selected a different track they would have different parts of the checklist checked and

unchecked based on their completed classes, and the pi charts on the left would shift to reflect the change. This makes switching between tracks much simpler. For students who are unsure what track to pick is a useful feature that no current system available to students provides. From the main page users can go to a scheduling page that displays a potential schedule for their upcoming semester. Required classes are listed at the top allowing students to quickly and easily identify courses to take. Below that students can search from the total list of available courses. When the user clicks on a class they can view it in their schedule. One of our biggest risks described in progress report 1 was the challenge of displaying this much information clearly and concisely. For my prototype, I tried to minimize the amount of information displayed at once, by having most of the options either minimizable, or part of a drop down menu.

Lower Fidelity Prototypes: Takeaways

After presenting our prototypes to each other, we talked about what features we liked most from the collective group of prototypes we created. Our favorite features were as follows:

- Direct manipulation of classes
    - Allowing the user to directly interact with (e.g., drag and drop) classes when working with their schedules and future planning
- Incorporating a checklist into the scheduler
    - Communicating degree program requirements to the user via a checklist they can interact with in the scheduler
- A persistent and customizable wishlist
    - A wishlist that persists between semesters so that a student does not lose track of classes that they cannot take now but want to in the future
    - Allow the addition of parameters to the wishlist, such as a specific professor. The student would then get a notification the semester this parameter is satisfied
- Visual progress tracking
    - Create a visual representation of the student's progress through their entire degree program (and through specific components, like the core and their track)
- Advisor-like functionality

    - Communicate to the student whether or not a class will count toward their major/track as well as how specifically it will count
    - Put in visual symbols (for example, a traffic light: green, yellow and red) to signify eligible, incomplete prerequisites, and ineligible classes for the track that the student has selected
- Detailed class information (pulled from SSOL)
    - Allow users to see detailed information from SSOL on a class when clicking on one

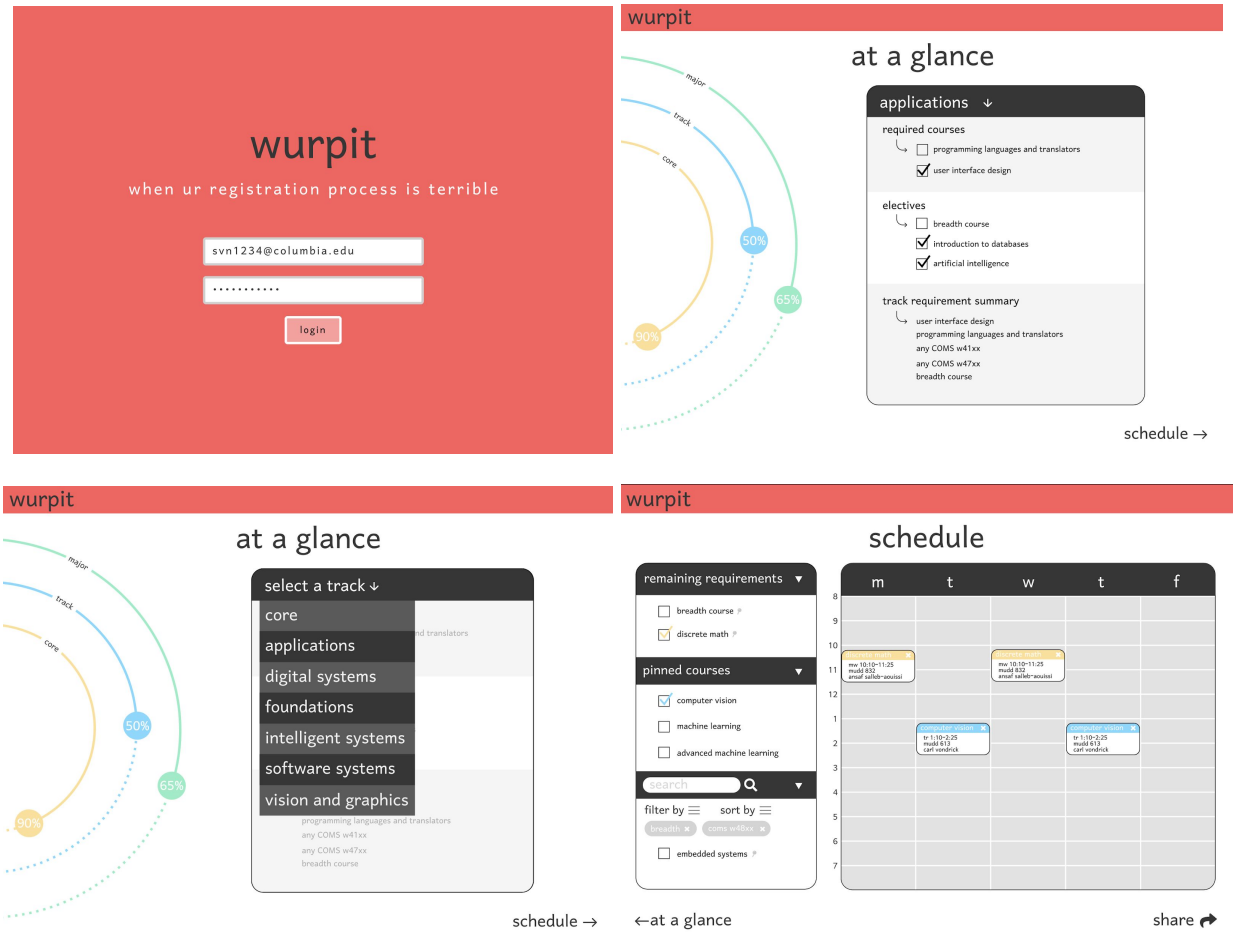
We will take these favorite features into consideration when designing our higher-fidelity prototype and final implementation.

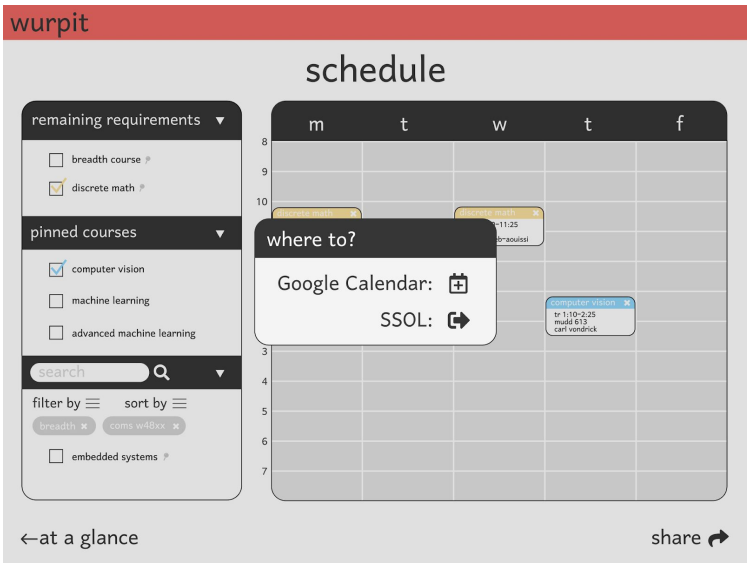
Preliminary Higher-Fidelity Prototype

After we designed low fidelity prototypes individually, we met and discussed our favorite features (described above). Once we decided what features we wanted, we created a Figma document and started building up features altogether. We began with a simple login, making boxes for users to enter their information and enter the system. The first page is “at a glance,” which shows users their progress in a cropped, nested donut chart. We decided on this chart format after agreeing that three separate pi charts would take up too much space and not be the most condensed option. The “at a glance” page also shows users a checklist of their selected track courses. This was in response to the participants of our user study expressing frustration with there being no simple checklist in any of the available resources, forcing users to either be confused, or open up a spreadsheet in yet another window of their computer to make a checklist of their own. In our design, users can quickly select a track from a drop-down menu at the top of the checklist to see their progress in that track.

Another risk we identified through our needfinding is that users have to open up many tabs over the course of one scheduling. It was our priority to reduce the back-and-forth users have to do to schedule their classes, so there are only two pages, the “at a glance” page and the “schedule” page. From the “at a glance” page, users can transition to the “schedule” page to plan courses for their upcoming semester. On the “schedule” page, users can pick classes to add to a schedule. The

classes are displayed in a way that mitigates the information overload risk. The section for adding classes is grouped by not yet completed required classes, classes the user has pinned to save for the future, and an area to search for classes with filters designed to streamline the class search process. This prevents users from ever just scrolling through a long list of courses with little clarity or direction. Instead, they can see their requirements, classes they have saved, and the results of searches for times, course codes, keywords, and more. Through the whole Figma design process, the whole group was together, giving each other feedback on designs in real time. From this, we came upon a design that we feel is the most concise, streamlined way to view and add classes, taking into account the intricacies of the CS curriculum.





Prototype 1 User Feedback

We followed-up with the two students we interviewed for our contextual inquiry study for their feedback on our preliminary higher-fidelity prototype. Both students had highly positive reactions to the design and liked how everything was in one place. Student 1 suggested that the "share" button should be renamed to "export" to make the button's function more immediately obvious to the user.

Student 2, who is a Barnard student, suggested that waitlist and registration information should be shown somewhere on the schedule screen. Waitlist information is not shown on MyBarnard; thus, Barnard students must log onto SSOL to see this information. Thus, having this information in WURPIT will be very helpful for Barnard students. In a similar vein, Student 2 also suggested adding HW0/Google Form information to the class information tooltip as this information is also not shown in MyBarnard. The student also suggested more advisor-like functionality, such as a warning if the user reaches their registration credit limit in addition to a waitlist simulator.

Progress Report 1 Feedback Follow-Up

We received the following feedback from Kenneth Yuan:

Feedback	Response
Because our target user population is similar to us, it is important to evaluate that the design meets the needs of others from our target population, and minimize the effect of our own biases.	To ensure our design meets the needs of those in our target user population, we 1) created and collected responses from an online survey, and 2) performed contextual inquiry on two users in our target population. We analyzed the total 17 responses from our survey, and more closely evaluated the needs of our target user population by observing two different students go about registering a sample persona. The persona was the same in both cases, and the two students were observed and interviewed at different times, with no means of being able to share study information. After observing the two students in their own personal registration processes, a short interview was conducted to clarify any ambiguous thoughts and understand the potential for how the process might be improved.
To avoid the risk of making a "clone" of any of Columbia's available registration sites, our user interface should address the gap between these sites (most prominently SSOL/MyBarnard and Vergil), while offering unique tools that are both usable and useful.	In order to also make our user interface a "one stop shop" where users don't have to consult other registration resources (aside from SSOL/MyBarnard where classes must be formally registered), in addition to the tools already offered, such as a visual schedule and list of added courses, we created unique tools, such as a checklist and progress bars (or a "nested donut chart"). In our contextual inquiry, we found that our target users were mainly frustrated with registration information not being all in one place, and consequently felt overwhelmed with lack of organization. To address both of these overarching needs, we not only made a user interface that keeps the majority of registration information and the registration process in one place (taking into consideration SSOL/MyBarnard must be

	used to formally register for classes), but also emphasized the visual aspect. We did this by creating progress bars that show the user's progress for 1) core classes, 2) track classes, and 3) all major-required classes, as well as a directly-manipulatable visual (Mon-Fri) class schedule and a directly-manipulatable checklist.
The "track" element of our user interface is the most unique aspect and separates our design from other registration resources. It is therefore important to emphasize this element.	We found in our contextual inquiry that users have a difficult time navigating between CS tracks, and keeping a personal spreadsheet can be overwhelming and a lot to keep track of. So, the first screen that pops up after the login page shows the track the user is on, with a dropdown option to change tracks as well as an option for "core" (to view a checklist of core classes). The direct manipulation of these various checklists will correspond to a change in the progress bars, and allow users to view how much of the core and each track they've completed.

Final Design Implementation Details

Our group used Figma for our high-level implementation. After realizing that code implementation and interactivity would be very time-consuming, and not as important as following-through on the results from both rounds of our contextual inquiry, we spoke with our TA Kenneth Yuan and Professor Smith to confirm coding that would not be necessary. We chose to proceed with Figma to make sure that our design would be as presentable as possible and so that all group members would be able to collaborate on it simultaneously.

Important Aspects of Our Design

I. At a Glance Page

A. In the dynamic course list on the right side of the page, the courses have one of three "key" characteristics:

1. Green arrow: the user has taken the course. Strikethrough text indicates this for those who are colorblind.
2. Yellow arrow: the user hasn't taken the course, but is eligible to (i.e., the prerequisite(s) have met).
3. Red arrow: the user hasn't taken the course, and is ineligible to (i.e., the prerequisite(s) have *not* been met). The indented arrow indicates this for those who are colorblind.

B. Key

1. A key icon is present in the upper right hand corner of the course list. It indicates what each of the above characteristics represents.

II. Schedule Page

A. Checklist

1. Both of the users we observed and interviewed as part of our contextual inquiry wished for an interactive checklist. We implemented this suggestion as follows:
    a) Remaining requirements
        (1) This checklist will always show the required Core courses, but will change depending on which track the user selects in the prior page. If a user added a track class to their schedule, but goes back and changes tracks, and then returns to the "schedule" page, that class will still be present on the schedule, but may not be in the "remaining requirements" checklist (depending on if that track class is a requirement for the newly selected track or not).
    b) Pinned courses
        (1) This checklist shows courses the user has "pinned", either courses they've searched in the search bar, or required classes. The functionality of this "pinned" section is for users to essentially "save" classes they know they may want to take in the future, but may not be able to currently. "Pinned"

classes will always remain in this section unless the user "unpins" them.

c) Search bar

(1) The user can search all Columbia and Barnard courses currently available in these search engines. The purpose of this is for users to have one place to resort to for scheduling classes, and not have to refer to other materials or resources that pertain to scheduling (other than the required SSOL/MyBarnard). Courses can be filtered and sorted, and the user can click on a course to get more information about it. From here, the user can "check" a course to add it to their schedule, or "pin" it to add it to their list of "pinned" courses.

B. Dynamically visual schedule

1. The purpose of the schedule on the right side of the page is for users to have a visual representation of what their schedule will look like for the following semester. It is dynamic and directly manipulatable, meaning that the schedule changes as the user directly interacts with it. For example, the user can click on a course that will open up a new window with information detailing course specifics. Clicking on the "x" in the upper right hand corner of a course slot will remove it from the schedule; however, if that course is a required or "pinned" course shown in the list on the left side of the page, it will not be removed from its list.

Instructions For Our Design

1. To log onto WURPIT, the user types in their Columbia/Barnard email and password, and clicks "login" to login.
2. The "at a glance" page allows the user to view the requirements for every computer science track, as well as the Core. To view the requirements for a specific track, and what

classes the user has taken so far, they can click on the down arrow and select one of the following tracks: 1) core, 2) applications, 3) digital systems, 4) foundations, 5) intelligent systems, 6) software systems, or 7) vision and graphics.

3. To view progress within a certain track or the Core, once the user clicks on their desired option, a "nested donut" chart appears on the left side of the screen. A more literal breakdown of classes that are required appears on the right side of the screen. Here the user can view which required classes they've taken and what other classes are required for that track or the Core. The user can click on the "key" icon to view what the colors correspond to in the required course lists.
4. To view a schedule for next semester classes, the user clicks on the "schedule" button in the lower right hand corner, and is taken to another screen.
5. The "schedule" page allows the user to view a dynamic list of required courses for the option they chose in the page prior, "pinned" courses, and allows the user to search for classes (any available course through SSOL/MyBarnard), with options to "filter" and "sort". Whenever a user searches for or finds a class, they can "pin" it to add to their list of "pinned" courses. Under "remaining requirements" and "pinned courses", the user can add or remove classes to the schedule shown on the right side of the screen. Removing, or rather "unchecking", a class will not remove it from the list. This allows the user to compare and contrast various schedules, while having all the courses they're considering in a condensed list to the left of the schedule.
6. A user can also directly manipulate the schedule by clicking on the "x" shown in the upper right hand corner of each course slot. This also will not remove the course from the list, but will rather "uncheck" it. A user can also interact with the schedule by clicking on a course slot, which will prompt a new window detailing the course specifics.
7. After playing with their schedule and testing various course options, the user can export their schedule (all "checked" courses) to SSOL/MyBarnard and/or Google Calendar by clicking on the "export" button in the bottom right hand corner of the screen.

Progress Report 2 Feedback Follow-Up

We received the following feedback from Kenny:

Feedback	Response
When a user chooses a course, make sure that the check marks/symbols clearly show whether or not the course counts for a category.	On the schedule screen of our final design, courses (and their corresponding checkmarks) are colored either yellow or blue. These indicate core and track courses, respectively. A progress bar at the bottom of the schedule screen (similar in nature to the one on the "At a Glance" page) indicates the meaning of the colors to the user.
Make sure to differentiate between courses whose prerequisites that the user is trying to fulfill versus courses whose prerequisites have already been fulfilled.	On the "At a Glance" page, courses are marked with either green, yellow, or red check marks to indicate completed courses, uncompleted courses (whose prerequisites have been satisfied), and uncompleted courses (whose prerequisites have not been satisfied), respectively. Furthermore, on the schedule screen, each course in the checklist and the schedule grid has a tooltip that contains prerequisite information. This information is either red or green depending on whether or not the user has fulfilled the course's prerequisites. Furthermore, if the tooltip header is shaded, then the prerequisites have not been fulfilled.
Make sure to emphasize the automatic tracking of requirements.	As mentioned previously, all of the pages and tooltips in our design indicate the category that a course will count towards as well as whether or not a course's prerequisites have been satisfied. Furthermore, the progress bars on both the schedule and "At a Glance" pages change depending on the track/courses selected.

Tasks

Task	Assigned Member(s)	Actual Time for Completion (hours)	Completion Date
Group Brainstorming	Vishnu, Spencer, Nadia, and Sam	2	Saturday, November 23rd
Progress Report 1 Planning	Vishnu, Spencer, Nadia, and Sam	0.5	Saturday, November 23rd
Progress Report 1	Vishnu, Spencer, Nadia, and Sam	1.5	Tuesday, November 26th
Meeting #1 with Kenny	Vishnu, Spencer, Nadia, Sam	0.25	Monday, December 2nd
Team Meeting #3: Planning for User Study	Spencer, Nadia, Sam, Vishnu (remotely)	3.5	Tuesday, December 3rd
Team Meeting #4: Conducting User Study	Spencer, Nadia, Sam	2	Friday, December 5th
Creation of Hi-Fi Prototype + Progress Report 2	Vishnu, Spencer, Nadia, Sam	9	Thursday, December 5th
Creation of Lo-Fi Prototype	Vishnu, Spencer, Nadia, Sam	0.5	Thursday, December 5th
360 Evaluations	Vishnu, Spencer, Nadia, Sam	0.5	Sunday, December 8th
Meeting #2 with Kenny	Vishnu, Spencer, Sam	0.25	Monday, December 9th
User Follow-Up Discussion	Spencer, Nadia, Sam	1	Friday, December 13th
Final Design + Report Construction	Vishnu, Spencer, Nadia, Sam	10	Sunday, December 15th
360 Evaluations	Vishnu, Spencer, Nadia, Sam	0.5	Monday, December 16th
Final Presentation	Vishnu, Spencer, Nadia, Sam	2	Tuesday, December 17th

Post-Mortems

Nadia:

The process of our designing our user interface, from conception to final implementation, overall went relatively well. I believe we were able to talk through our idea, perform contextual inquiry, analyze our data and results, and implement two iterations of our design in an efficient amount of time. The contextual inquiry specifically went very well, as I feel like we were able to get the data and results we needed, as well as vital feedback and suggestions. We then based our first design off of the problems we saw our users encountered in the first round of contextual inquiry, and then implemented a high-level version of our design in Figma that simulates user interaction, changing our design to reflect the suggestions our users offered in the second round of contextual inquiry.

In terms of what went wrong, because our design required the use of gathering data from several Columbia and Barnard registration sources, we would often have to ensure that when a user navigated our UI, they were being told the correct information. While we didn't necessarily encounter any mishaps or things that "went wrong", in order to prevent something "wrong" happening to a user, we had to do a lot of error prevention.

The main thing we would do differently would be creating a registration resource for all majors at Columbia and Barnard, which was our original intention. After further brainstorming our idea, we knew a registration resource encompassing this large user population would be too much to conquer in a short amount of time, and thus decided to solely focus on computer science majors. If allotted more time, we would expand our UI for use by any Columbia or Barnard student. This would also prompt a more invested approach to our contextual inquiry.

Spencer:

I am very satisfied with the design process. Within the time frame granted, I believe my group was able to design a feasible and functional prototype implementation for what we had planned. Going through a full, formal design cycle was fun, and I would have loved to have done more cycles to get a better experience for what the design process is like long-term. I feel very lucky that the team collaborated well and that we all were able to contribute our own ideas for the

duration of the project. I appreciated getting to work more with Figma, and I can see myself using it in the future for more creative design projects.

I would not say that much went wrong in the design process. I did not feel that my group was disorganized, which was helpful in the long-term planning and preparation that the project required. We ran into some issues with Figma animation, but those were just bugs and issues with the program, not with the design process. If I were to do this project again or continue the project, I would focus on true implementation of the prototype in code, with a backend. Given just about a month is not enough time to reach this ideal implementation, but I feel confident in that such an implementation *would* be attainable given the time and resources. If I were to implement WURPIT for the long-term, ideally it would be usable for all majors, disciplines, and classes at Columbia. Due to constraints and resources, along with the needs we expressed and discovered in the process, computer science students were a wonderful first start.

Vishnu:

The overall design process went very well. During our brainstorming session, we were able to coalesce around an idea very quickly. Considering that we are part of the target population for our idea, we needed to do some needfinding. We received plenty of responses to our survey, and our contextual inquiry gave us many ideas to work with. The prototyping process resulted in a very presentable and detailed prototype as well, and we got even more feedback by showing our contextual inquiry participants our revised prototype. Furthermore, the four of us worked very well together as a team, and we achieved our intended goals.

In terms of what went wrong: Not much. If anything, we were pressed for time and needed to scale down our final prototype from the overall idea to get it done on time. However, other than that, the process went very well. Our initial idea was to create a resource/system for all majors. We would have liked to expand out towards this goal if we had more time. Furthermore, we would have liked to attempt to implement this in code, though this was currently not possible due to our varying skill levels in web development.

Sam:

The process of making our user interface was very straightforward, as we closely followed the framework from lecture. We made sure to do extensive needfinding, and made an effort to quantify our problem. From there we were able to take user feedback and form low and higher fidelity prototypes. Each prototype improved on the previous, making our final result a design solution we were all happy with, that we feel would be extremely useful and usable to our user population. Being able to make our design in Figma was extremely useful, because it allowed us to all work together and quickly, easily iterate on our designs.

Our process went relatively smoothly, with not much going wrong. The main thing is that we had such a complex set of requirements to manage that we had to let some features go, that we did not have time to implement. Specifically, we were unable to tailor the requirements by school, instead opting for showing the UI for a particular student.

Given the opportunity to do this project again, and with more time, I would have wanted to be exhaustive in the way the UI is designed. I would make sure to design for every school-track pairing, and possible inconsistencies, to produce a robust system. I would also have done more loops in the design cycle, and done a thorough user study with a users attempting to schedule their own courses. With the time allowed, we were able to show users our first pass at the design, but it would have been great to do more iterations and continue to improve the design.

Screenshots, Overview, and Permission

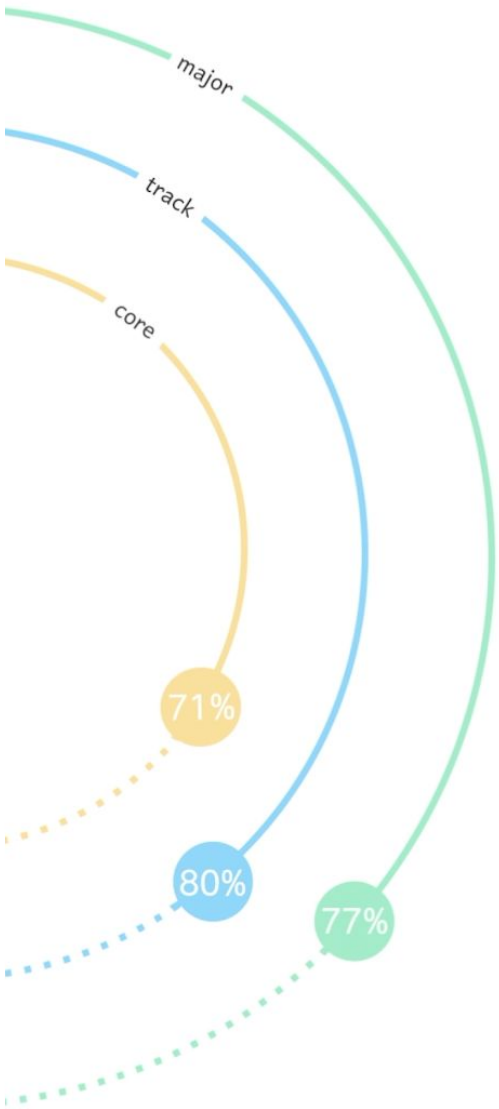
Sam Ross, Spencer Bruce, Nadia Jurkovich, and Vishnu Nair are willing to have their names appear next to their presentation of their work on the project web page for W4170.

See the screenshots attached below or the one titled “WURPIT-Screenshot.png” in the uploaded directory. Additionally, see the overview paragraph below:

WURPIT is a streamlined, organized, and individualized support system for semester course selection and long-term major planning, currently targeting Computer Science (CS) students at Columbia. The current systems in place for course planning are not particularly useful to CS students, as they are not specifically tailored to the complicated CS major system. WURPIT will allow students to easily navigate between the six tracks of the CS major, allowing them to see

what classes they have left in each track and how the courses they have already taken fulfill various requirements. It will clearly display important course information like specific track classes and prerequisites, making it easy for students to plan their courses and pick their track based on what they have taken/want to take in the future. By streamlining the process of selecting courses, WURPIT will ease the logistical stress of planning out a CS major and provide advising support for students, which is in high demand

wurpit



at a glance

applications ↓

required courses

- ~~programming languages and translators~~
- ~~user interface design~~

select 2 from

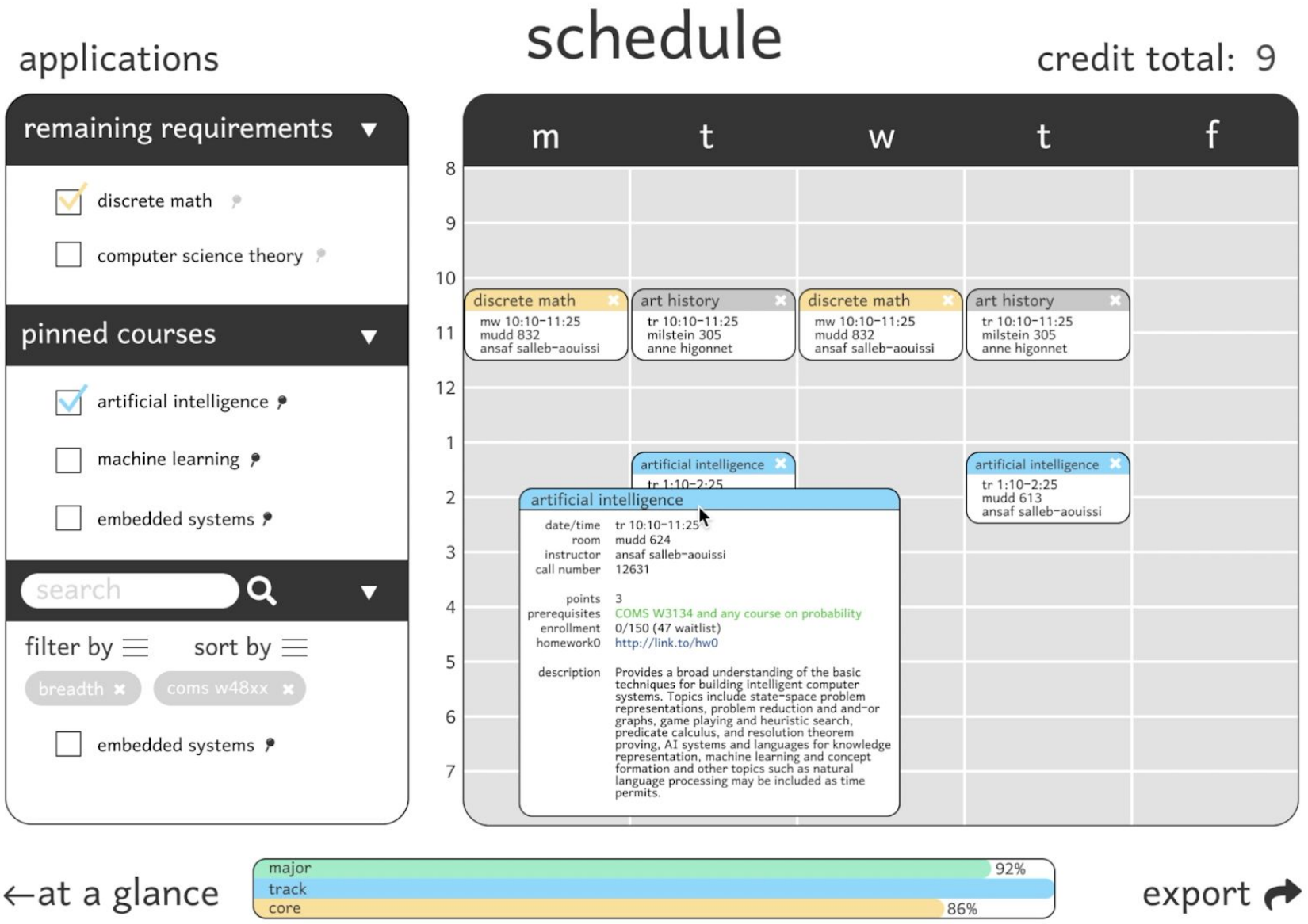
- any coms w41xx
- ~~any coms w47xx~~ (1)
- any coms e69xx
- other advisor approved

other requirements

- ~~breadth course~~

schedule →

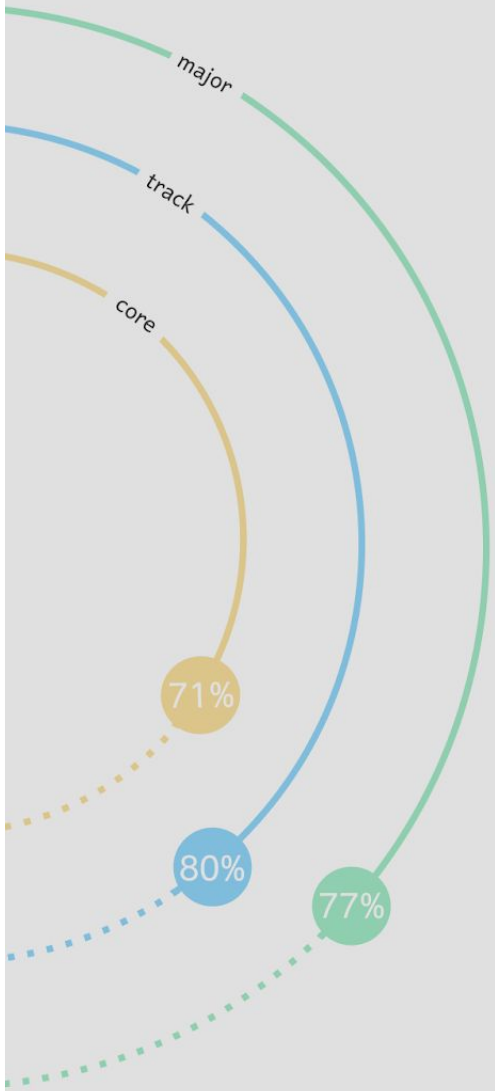
wurpit


applications
schedule
credit total: 9
remaining requirements
discrete math
computer science theory
pinned courses
artificial intelligence
machine learning
embedded systems
search
filter by
sort by
breadth
coms w48xx
embedded systems
m
t
w
t
f
8
9
10
11
12
1
2
3
4
5
6
7
discrete math
mw 10:10-11:25
mudd 832
ansaf salleb-aouissi
art history
tr 10:10-11:25
milstein 305
anne higonnet
discrete math
mw 10:10-11:25
mudd 832
ansaf salleb-aouissi
art history
tr 10:10-11:25
milstein 305
anne higonnet
artificial intelligence
tr 1:10-2:25
artificial intelligence
tr 1:10-2:25
mudd 613
ansaf salleb-aouissi
artificial intelligence
date/time tr 10:10-11:25
room mudd 624
instructor ansaf salleb-aouissi
call number 12631
points 3
prerequisites COMS W3134 and any course on probability
enrollment 0/150 (47 waitlist)
homework0 http://link.to/hw0
description Provides a broad understanding of the basic techniques for building intelligent computer systems. Topics include state-space problem representations, problem reduction and and-or graphs, game playing and heuristic search, predicate calculus, and resolution theorem proving, AI systems and languages for knowledge representation, machine learning and concept formation and other topics such as natural language processing may be included as time permits.
←at a glance
major
track
core
92%
86%
export


wurpit
when ur registration process is terrible
uni@school.edu
password
login

wurpit

at a glance



applications ↓

required cou

~~prograr~~

~~user in~~

select 2 fror

any coms ~~w41xx~~

~~any coms w47xx~~ (1)

any coms e69xx

other advisor approved

other requirements

~~breadth course~~

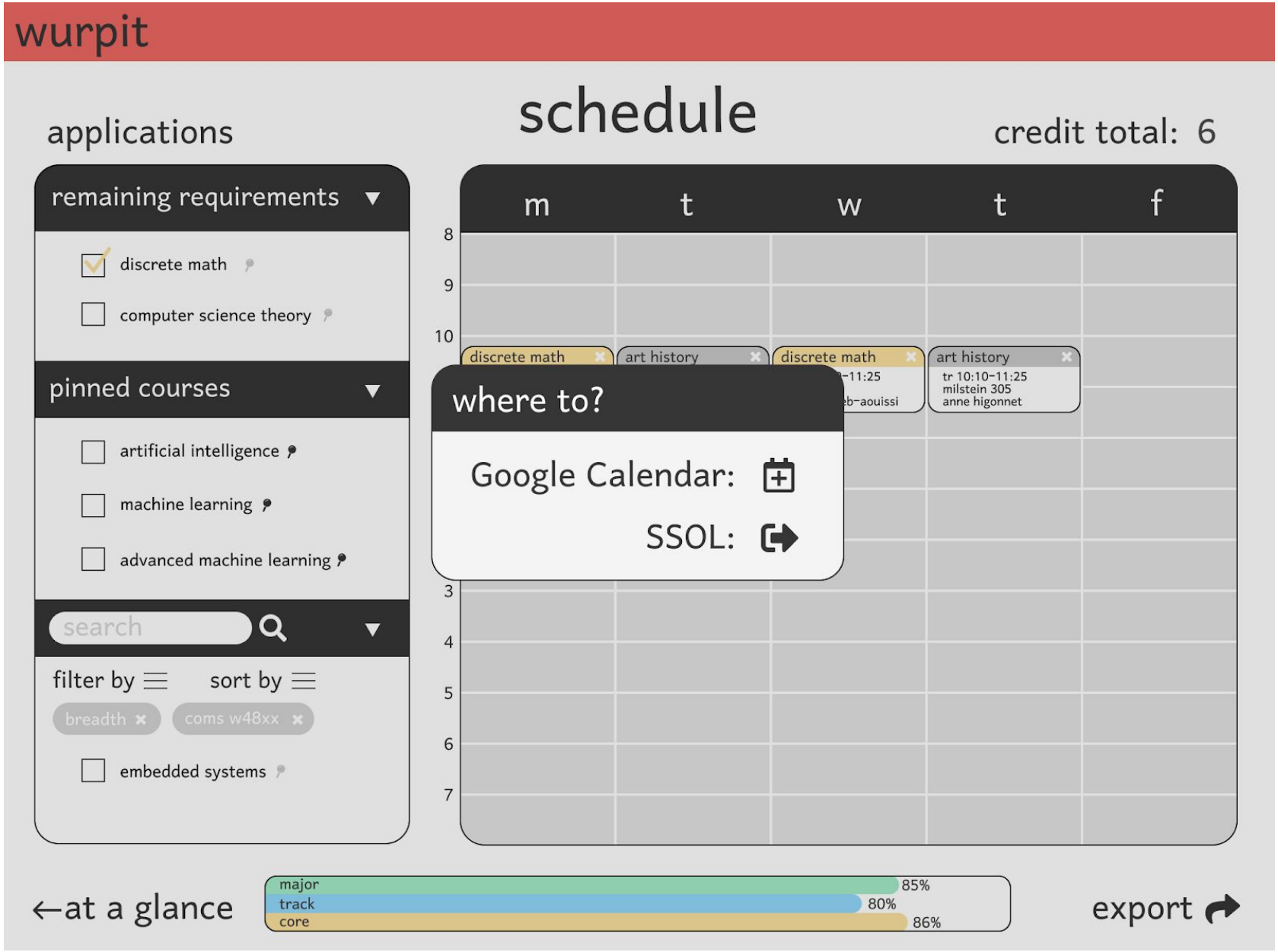
key

~~completed~~

able to be completed

must complete perequisites first

schedule →


wurpit
applications
schedule
credit total: 6
remaining requirements
discrete math
computer science theory
pinned courses
artificial intelligence
machine learning
advanced machine learning
search
filter by
sort by
breadth
coms w48xx
embedded systems
m
t
w
t
f
8
9
10
3
4
5
6
7
discrete math
art history
discrete math
art history
tr 10:10–11:25
milstein 305
anne higonnet
where to?
Google Calendar:
SSOL:
←at a glance
major
track
core
85%
80%
86%
export