advisr

Final Written Report
Anna Carroll, Radha Jain, Aditi Poduval, Matthew Stewart
Human 2.0, CS147, December 9th

VALUE PROPOSITION

Advisr equips students with the optimal four-year plan to accomplish their goals.

PROBLEM AND SOLUTION OVERVIEW

Between balancing requirements, exploring potential majors, and scheduling around extracurricular, academic planning can be a daunting task. Advisr generates personalized, four-year plans that allow students to explore different majors and stay on track to accomplish their goals.

TASKS & FINAL INTERFACE SCENARIOS

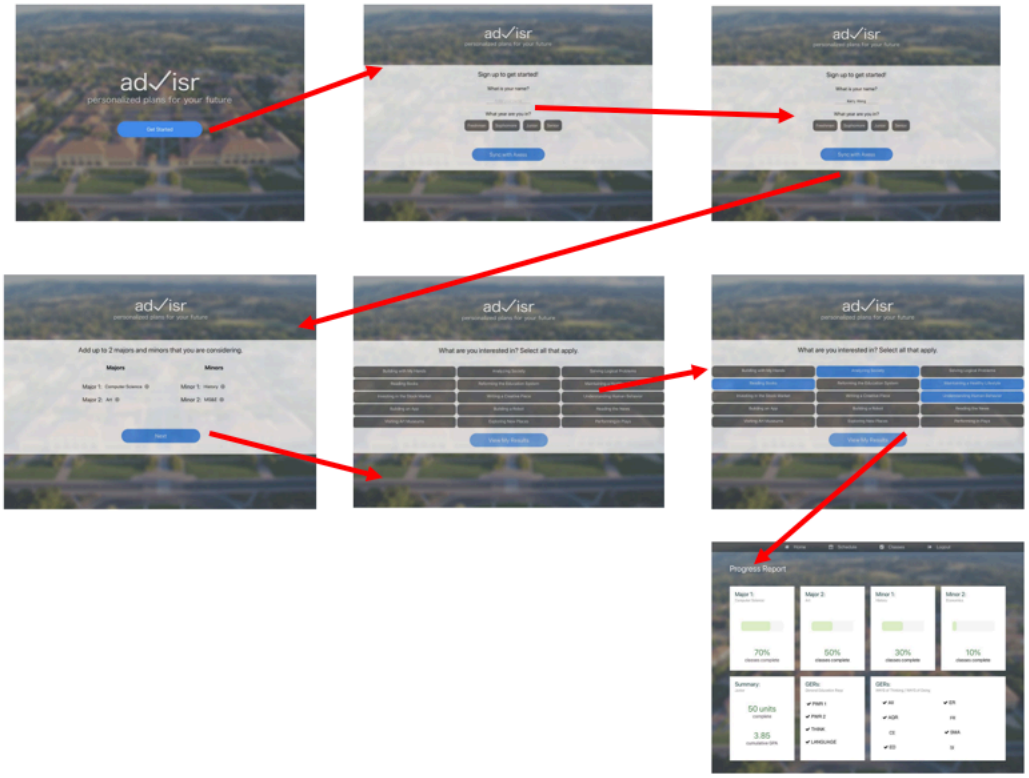
Task 1: Choose potential major(s)/minor(s) and track progress towards multiple majors *(simple)*. This task encompasses the onboarding process starting with signing in through Axess then inputting potential majors/minors and finally selecting interests. After the onboarding, the user is taken to the dashboard view where they can see a progress overview of each of their tracked majors as well as GERs and other requirements. This is an important task because every user must complete the onboarding process and tracking progress towards a major is one of our main functionalities.

Task 2: Track requirements completed to each potential major(s)/minor(s) *(simple)*. From the dashboard view, the user can click on a major/minor and is taken to a more detailed view of the requirements necessary for that major/minor. We decided to make this into a separate task from Task 1 because it provides the user with a more detailed view of the specific requirements for each major.

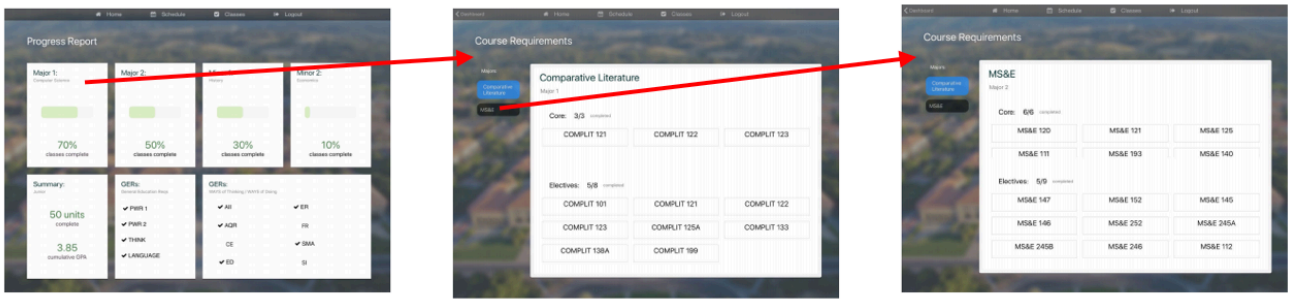
Task 3: Select courses based on interests and preferences *(moderate)*. In this task, the user chooses which elective classes they would like to take. The program suggests classes to the user based on the interests that they input during task 1. Mapping user interests to suggested classes is one of the main applications of AI in the program. This task is important because it encompasses one of our other main functionalities, suggesting classes based on interests. We have yet to implement the drag and drop functionality of this feature fully but in a final version, the user would drag the suggested class onto the quarter in which they want to take it.

Task 4: Personalized four-year plan *(complex)*. This task involves viewing the four-year (or one-year) plan, adding or removing classes, and dragging classes from one quarter to another. Providing students with a clear, customizable map to graduation is a cornerstone of Advisr. This task is the most complex for users but provides the most value.

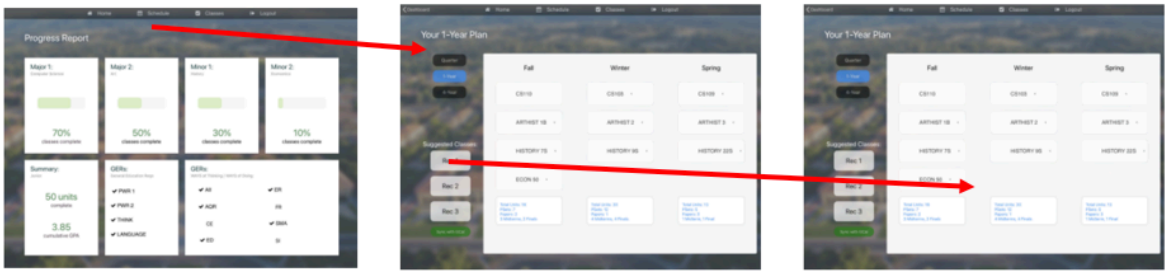
Task 1 Storyboard:



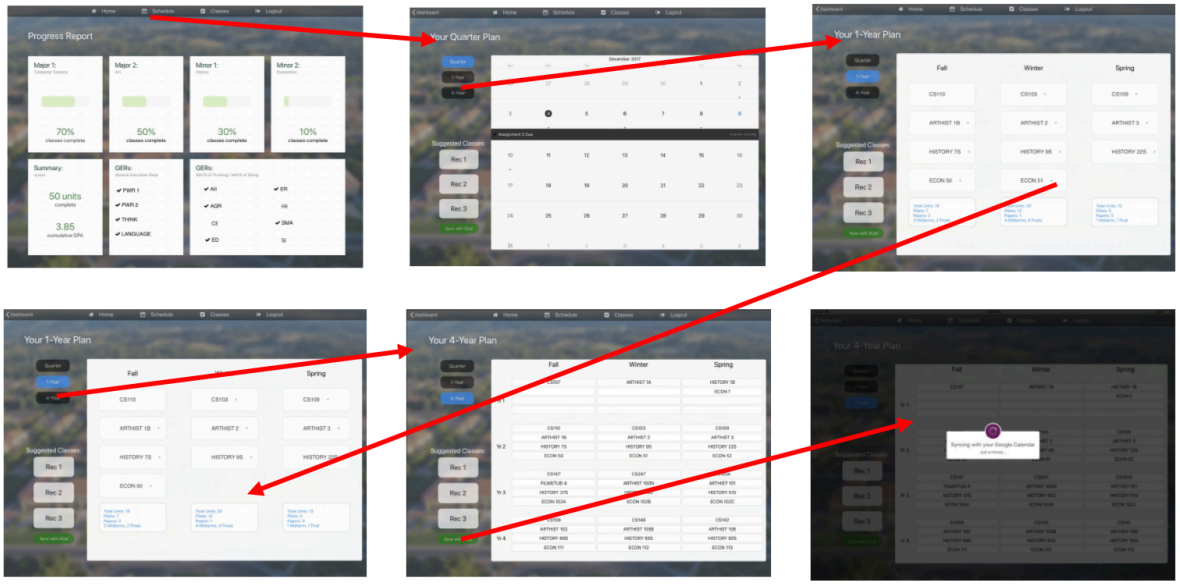
Task 2 Storyboard:



Task 3 Storyboard:



Task 4 Storyboard:

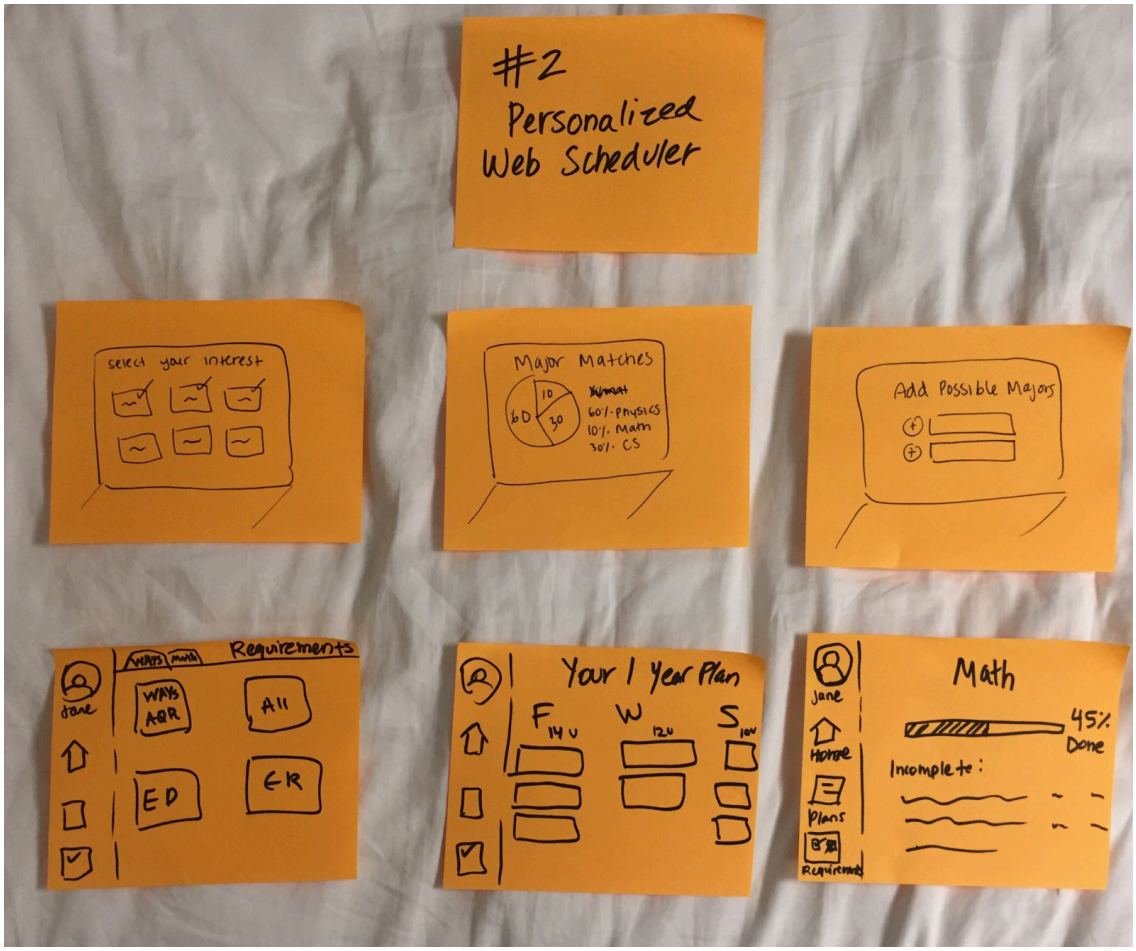


DESIGN EVOLUTION

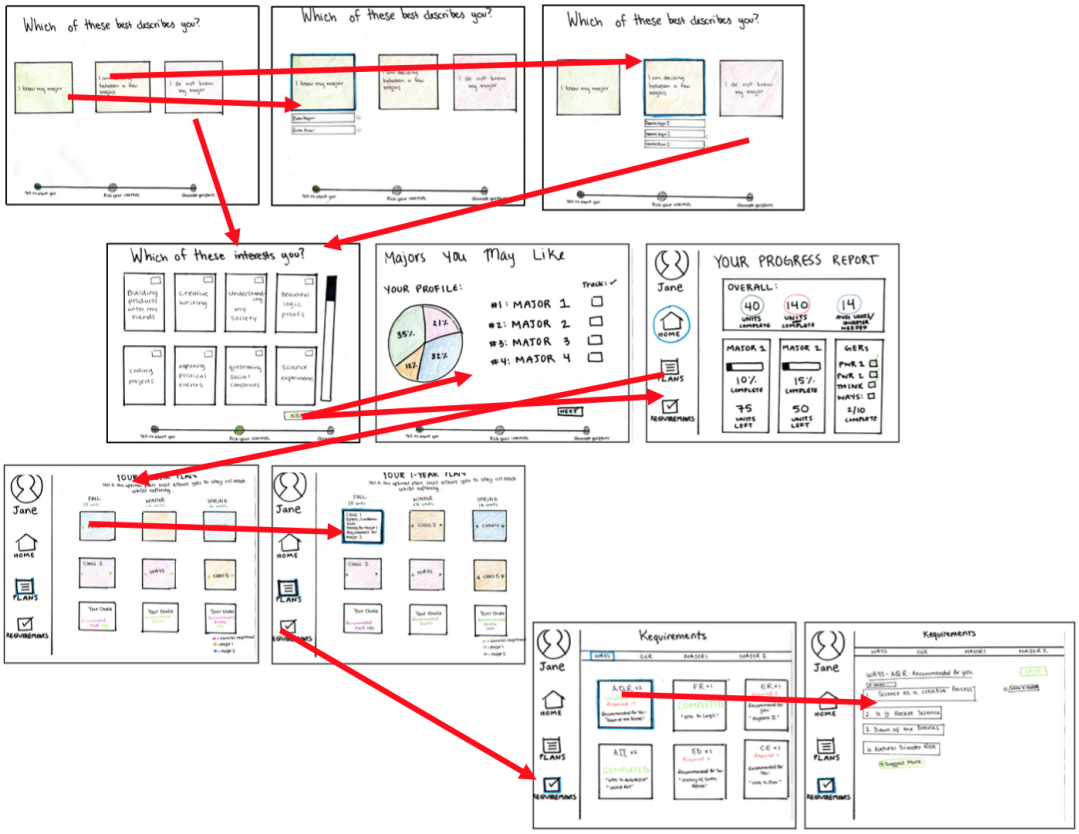
Needfinding:
Our initial needfinding focused on academic planning both at Stanoford and at other schools across the country. Our findings suggested that students find the process of academic planning to be difficult and often frustrating. In addition, when interviewing upperclassmen, we found that students regularly regretted the classes they had taken and wished that they had planned more carefully.
Preliminary POV:
We met a regretful senior who was struggling to complete her major. We were amazed to learn that no one told her to create a four-year plan. It would be game changing if students learned early on that academic organization was as important as academic exploration.

Sketches:

1. **Initial UI Brainstorming:**
   While brainstorming initial UI designs, we came up with the idea of having a personalized web scheduler that creates four-year plans based on user inputs. This was Advisr in its earliest stage.

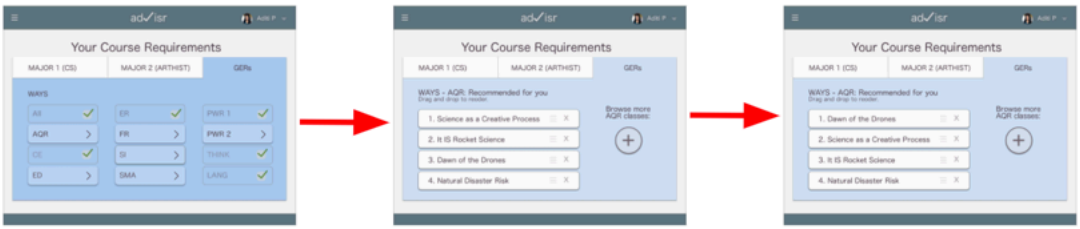


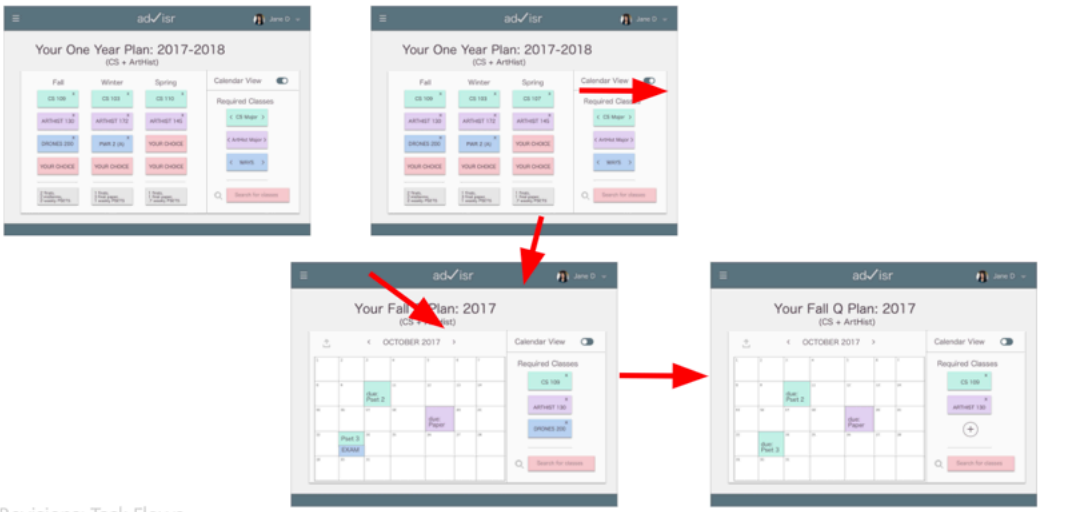
2. **Low-fi prototype:**
   Initially we chose a web based application. Below is our initial low-fi prototype with arrows indicating flow. This initial design, although revised continuously, provided the backbone for our application UI and structure. We conducted user testing on this paper prototype by asking participants to interact with the prototype using their fingers. The main feedback we received from participants was to include a toggle between four-year, one-year, and weekly calendar views, clarify what the pie chart was showing, and to associate the dashboard tiles with an action. We ended up removing the pie chart during the next iteration of our design, but incorporated the other feedback.



3. **Medium-fi prototype:**
   We used Marvel to create our medium fidelity prototype. The major design changes we implemented during this stage were to add toggles between one-month, one-year, and four-year plans, make the dashboard tiles clickable, and finally to add functionality in the calendar to include assignment and exam due dates. In addition, we had to switch our platform from a web application to an iOS application so we chose to use an iPad in order to display as much information as possible. We also added screens showing the interface with Axess to make it clear that the program would pull data from the users course history.

UI Revisions: Task Flows





UI Revisions: Task Flows

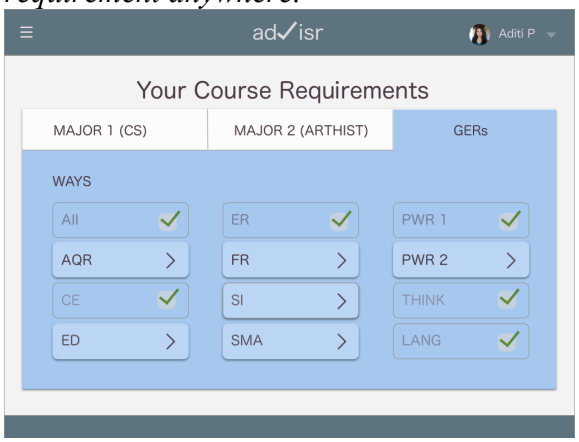
MAJOR USABILITY PROBLEMS ADDRESSED

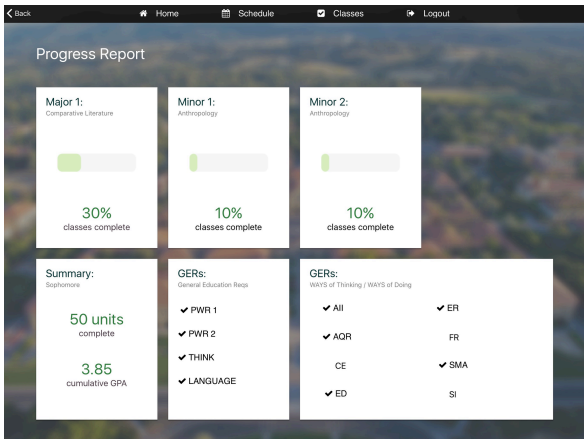
Heuristic Evaluation Usability Problems (severity 3-4):

1. **H4. Consistency and standards (Severity 3)**
   Changes made to the class schedule in the calendar view do not save and changes are not reflected on the non-calendar view. This leaves the user wondering what's going wrong and why the views show different information.

   Fix: Since this was just a prototype testing UI, we had yet to implement the back end to support updated changes to the calendar view. In real implementation this functionality would obviously be implemented.

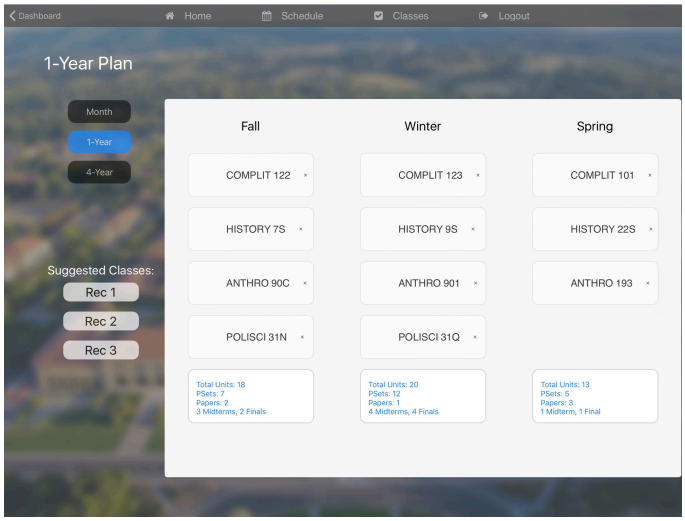
2. **H2. Match between system and the real world (Severity 3)**
   The "Requirements" screen doesn't display the number of units/classes needed for each requirement anywhere.





Fix: We streamlined the entire flow by creating a more comprehensive "Dashboard" screen that holds all the information relevant to course requirements, units, and progress towards major/minor completion. This change took care of this usability issue. In order to view the specific requirements of each major, the user simply clicks on the major tile and is taken to a more in depth view of the required courses.

3. **H7: Flexibility & Efficiency of Use (Severity 3)**
   One of the key tasks you have for the product is allowing the user explore new classes they may have otherwise never considered. Being a senior using this product, I'd want to access this immediately, rather than have to dig through a menu to "explore" new classes.



   Fix: We added a "Suggested Classes" side bar to the "Schedule" view. This way users can view suggested classes and input them into their schedules directly. We did not use the evaluators suggestion to include an "explore" button because we thought that it increased complexity passed what was necessary for our application. The functionality of exploring classes is achieved in the "Schedule" view. Above is our final version with suggested classes shown on the left.

4. **H8: Help Users Recognize, Diagnose, and Recover from Errors (Severity 3)**
   If I'm playing with my schedule on the calendar view and accidentally hit "X" on a class I actually want, there's no immediate way to undo the change I just did on accident.

   Fix: We never got around to implementing this change, however in a final version there would definitely be an undo button when editing course schedules.

5. **H10. Documentation (Severity 4)**
   As a user who know nothing about the undergraduate requirement system (new students using this app might have the same problem), I would appreciate this app have some introduction page to tell me some background.

   Fix: Given the time frame and scope of our app, we chose to leave this documentation suggestion for future versions and assumed that our users would have some idea of how the requirements at Stanford worked.

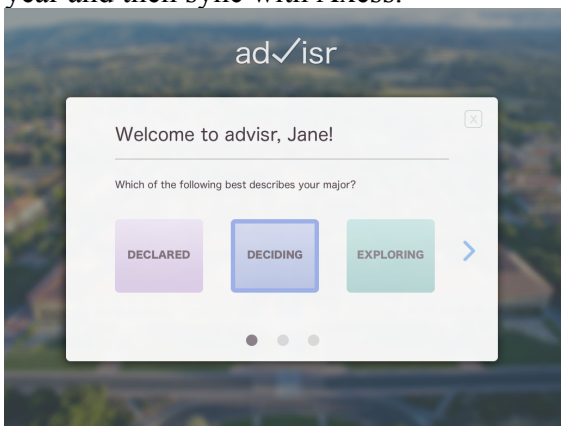
6. **H3. User control and freedom (Severity 3)**

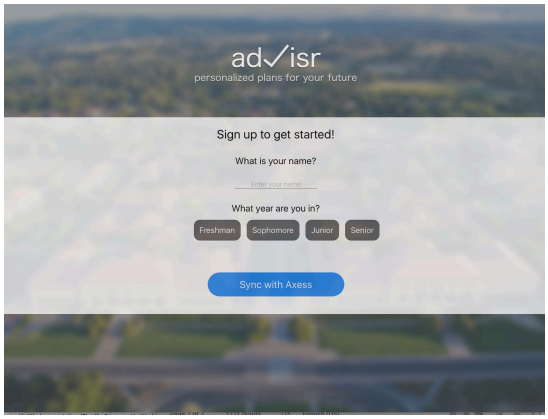
Wish the user could change majors they've chosen, i.e. have a way to modify the information inputted in the welcome pages.

Fix: Again, we focused on core functionality instead of implementing this feature. In the current prototype, the user can simply scroll back to the beginning and input their interests again.

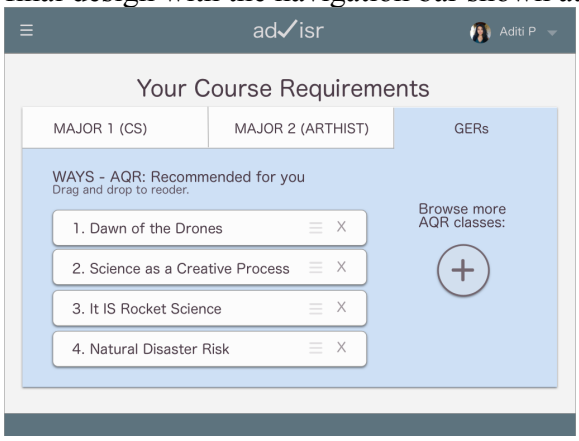
Other changes we made:

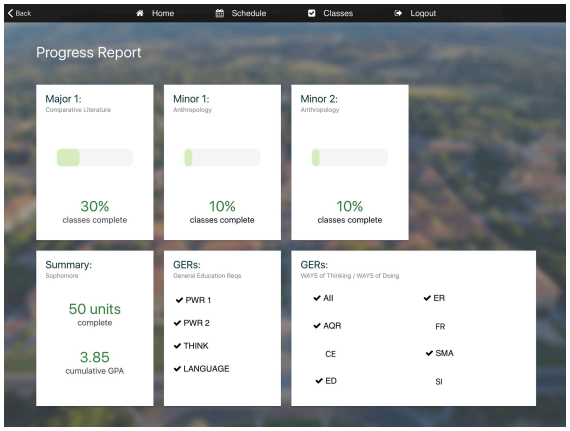
1. We found that the onboarding experience was generally confusing and simplified it by removing the 3 large buttons shown below and having a single process for all users to follow. On the right is the final version where all users are prompted to enter their name, year and then sync with Axess.





2. In order to increase navigation ease, we added a top navigation bar that allows users to easy toggle between different views. In addition it allows users to quickly return to the dashboard from a more in depth view. On the left is the original design, on the right is the final design with the navigation bar shown at the top of the screen.





PROTOTYPE IMPLEMENTATION

Tools:

1. **NativeScript:** We used NativeScript to create our application. We chose this tool because our group had the most experience with it and it provides an easy-to-use toolkit for creating applications quickly. One drawback with using NativeScript is that the HTML based language lacks sophisticated data structures that make other tools more powerful for developers.
2. **Github:** We used Github to manage version control. Setting up a NativeScript repository on Github proved to be a more challenging task than we anticipated. We had a lot of issues getting the software to run on everyone's computers. Ultimately, Github was necessary because it allowed multiple team members to work on the application at the same time and from their own computers.
3. **XCode:** Finally we used XCode in order to run our simulation. We ran our simulation on an iPad Pro. This worked seamlessly.

Wizard of Oz/Hardcoded Data:

1. Axess: The biggest Wizard of Oz technique we used was pretending to sync with Axess. We show a spinning wheel to indicate that the application is working and to simulate the actual process of syncing with Axess that would take place in a fully functional version. Because we do not sync with Axess, the Dashboard displays hardcoded data about progress, GERs, units, and GPA.
2. Progress Bars: The progress bars that are visible on the Dashboard are actually static images. Based on what year the user inputs during onboarding, we display progress bars that are more or less complete.
3. Majors: We had to hardcode each of the majors into the program since there is no interface with Axess. We also hardcoded a long list of classes that correspond to each major in order to display those on the "Course Requirements" page.
4. Four-year plan: Currently we display classes that are relevant to the major that the user inputs, however these classes do not actually fulfill all requirements needed to graduate.
5. Calendar Assignments: The assignments that are shown on the "Month" view are also hard coded to demonstrate the functionality.

Unimplemented Features:

1. The suggested classes feature that takes user interests and uses some sort of sophisticated mapping to prompt the user with classes they may be interested in is currently not implemented. We have a place holder demonstrating where this feature would be located in the "Schedule" view.
2. When the user removes a class in the "1-year" calendar view, it does not remove it from the "4-year" view.
3. Classes are not clickable. In a fully functional version, the user would be able to click on a class and see more details about the class like when it is offered, how many units, what type of work it requires etc. This is information that could be pulled from Axess.

Future Additions:

1. Interface with Axess in order to pull the users actual course history.
2. Create an algorithm that is able to match user interests and past course history with classes that they might find interesting.
3. Create an algorithm that takes course history and desired major and creates an optimal path to graduation that minimizes unit-heavy quarters.
4. Allow more customization by adding features that allow students to input preferences such as having lunches free, not having morning classes, or not taking certain classes together.
5. Add a feature that shapes schedule around extracurricular activities such as sports or clubs.
6. Add a feature that allows students to share their classes with their friends and find friends who are in their classes.
7. Interface with existing technology like Carta in order to pull class reviews, grade distributions, and weekly hours of work.

Summary

Academic planning can be an arduous and complicated task. A common theme that we encountered during needfinding was that upperclassmen frequently regretted not planning their academic careers more carefully. After identifying this need, we sought out to create a platform that allows Stanford students to plan their academics more carefully in order to accomplish their goals more easily. Advisr is an iOS application that aims to provide Stanford students with a simple, easy to use tool to create personalized four-year plans to accomplish their goals. Using Advisr, users can select multiple majors to track, input interests, view and track their progress towards graduation, and generate customized four year plans. The application as it stands is still in the prototype stages. The underlying functionality of some of the features of the application have not yet been implemented. However, the foundations of the application are solid and could be built upon to create a program that truly revolutionizes the way that Stanford students go about academic planning.