CS 151 Project Report

Improving BSCS Program Sheets

Michael Nixon, Ishan Khare, Soham Konar

Problem

As students, it is important that we can plan out the courses that we will be taking over our undergraduate career and so we need a system or document that makes that process easy for us. The current system for computer science majors is a set of program sheets, one for each track in the computer science major. While these program sheets are functional, they have a few issues: they are not interactive, they are unclear about or don't list all courses for each requirement, and they only list the course numbers for the majority of classes. Our goal is to make the BSCS program sheets as interactive, informative, and accessible as possible while also making the user interface clean and dynamic. The features we are introducing in our new BSCS program sheets are that they will be interactive, they will list all courses for each requirement, they will list all course names and numbers, and the web page will update to reflect your course selections and unit counts as you fill out the sheet.

User Interface

Our user interface balances ease of use, flexibility, and extensibility. Through the employment of color as a visual aid, users are guided through the worksheet's various steps towards the completion of their track requirements. Incomplete requirements are made salient by the color red and return to black once they have been met. At a glance, users can immediately gauge their progress and identify areas that require completion.

The flexibility of the interface allows users freedom in how they provide information and engage with our interactive worksheet. With all fields displayed and available to the user at all times, there is transparency in the length of the worksheet, how the results will be presented, and the overall unit progress towards graduation requirements. Progress updates automatically as users input and change data, giving immediate feedback.

Excitingly, this interface design is easily extensible to other applications. The minimal visual design, tailored to the brand language of Stanford University, means the worksheet design is relevant to other subjects and/or departments at Stanford. Our general, reusable predicates and rules in the underlying logic make it simple for administrators to customize the worksheet to match their specific degree requirements.

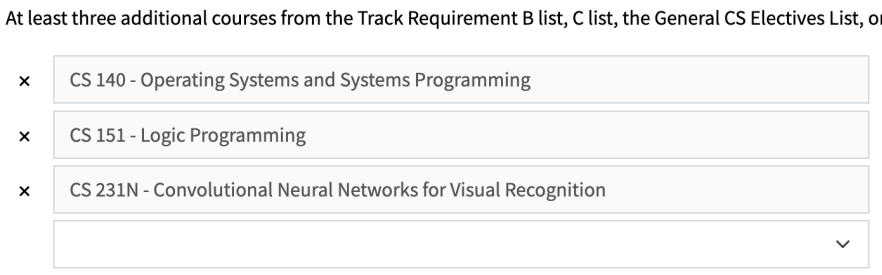
Logic Programming Implementation

We decided to use much of the data obtained from the MSCS data sheets as a starting point. Then, we scraped additional information from the Stanford Course Catalog (Explore Courses). We stored this information/dataset using the `course(id,department,number,title,num_units)` predicate. For example, this is what the predicate for CS 151 would be in our dataset: `course(cs151,"CS","151","Logic Programming",3)`. In addition to the data, we used logic programming to represent the different requirements users would have to meet. These included subgoals that had to be satisfied as well as view definitions that calculated a minimum number of units. One problem we had to overcome was to ensure that students could not apply two classes to the same requirement. For example, students are not allowed to take both CS107 and CS 107E and use both to satisfy the requirement. Instead, they must choose one or the other. To ensure this occurred, we made use of `exclusive()`. Most of the other technical logic programming details can be found in the source code itself.

Overall, it is not necessary to utilize logic programming for the implementation of program sheets. However, logic programming simplified the development process quite a bit. Since the majority of our project consisted of validating requirements, using Epilog was a perfect fit. Styling the various HTML elements was quite easy as we rarely had to worry about the underlying system and could work at a more abstract level. If we used typical programming, we would have had to include numerous if-then statements. As a result, fewer lines of code were needed to implement the core functionality of the project with logic programming.

Challenges

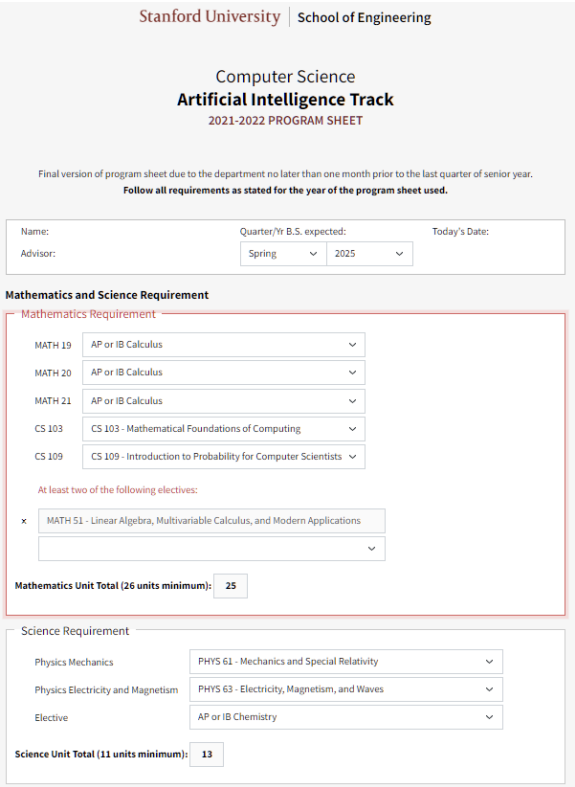
Although logic programming made developing our project easier, we still faced a considerable number of challenges. First and foremost, the large datasets made our coding very cumbersome. In certain situations we had to create hundreds of options for a single HTML dropdown and implement logic for each of those dropdown options. In general, the smallest of typos could sometimes cause the functionality of the worksheet to fail. We also had moderate difficulty allowing for multiple-select to satisfy electives as shown below:

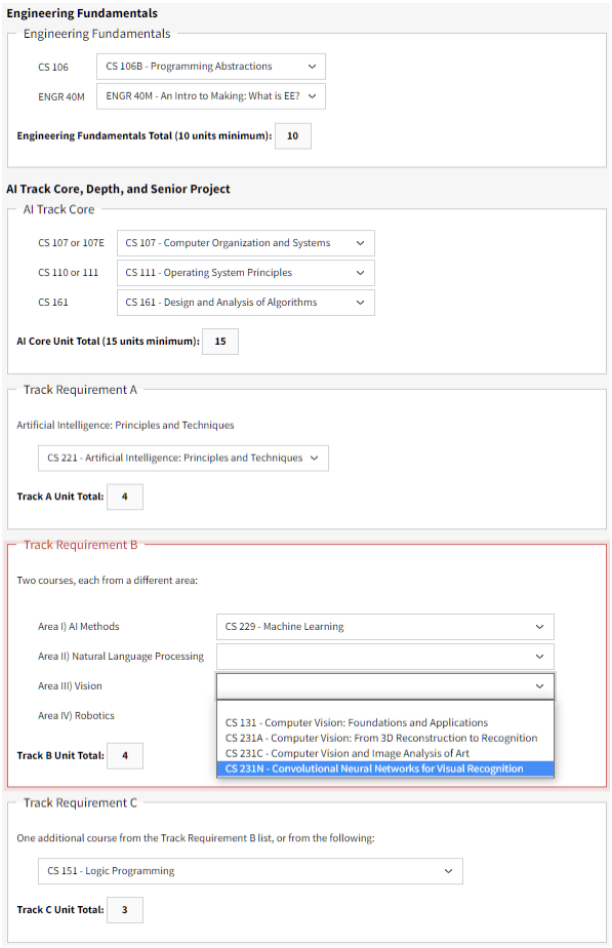


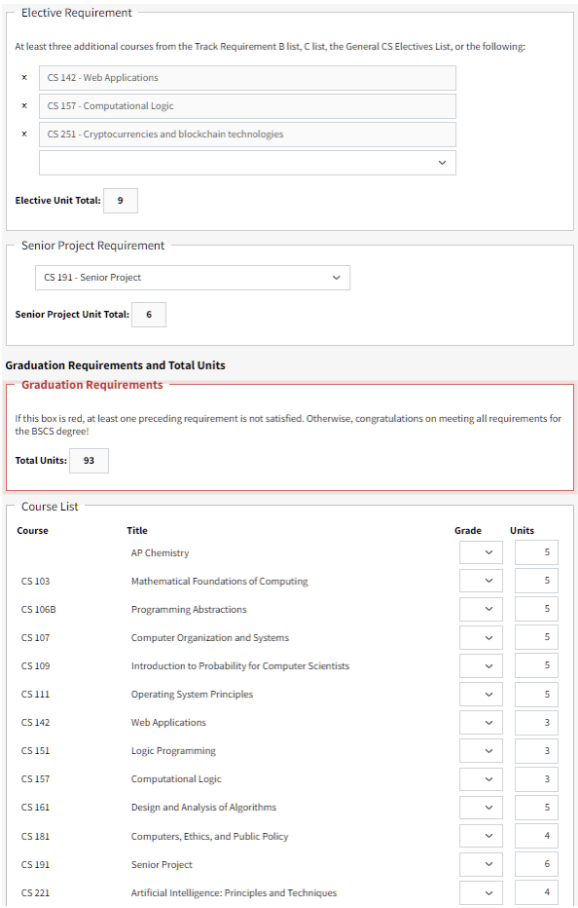
Future Work

Our current project provides students with a way to check their progress for the Bachelor's degree in Computer Science — AI track. However, there are several other tracks that we should also consider, namely, Biocomputation, Computer Engineering, Graphics, HCI, Information, Systems, Theory, and Unspecialized. Upon having completed one program sheet, creating very similar ones for the other tracks should not be too difficult. Additionally, our current program sheet has no real backend for data collection and submission. This would be another feature to implement to increase the utility of our project. A possible extension would allow students to specify which quarters during their Stanford experience they would take each course. Depending on course conflicts and scheduling, the program sheet would then provide feedback on whether or not the proposed four-year plan was feasible. In general, there are various ways to expand upon our work, a few of which we have just elaborated upon above. We are excited to see where this leads us in the future!

Project Screenshots




Engineering Fundamentals
Engineering Fundamentals
CS 106
CS 106B - Programming Abstractions
ENGR 40M
ENGR 40M - An Intro to Making: What is EE?
Engineering Fundamentals Total (10 units minimum):
10
AI Track Core, Depth, and Senior Project
AI Track Core
CS 107 or 107E
CS 107 - Computer Organization and Systems
CS 110 or 111
CS 111 - Operating System Principles
CS 161
CS 161 - Design and Analysis of Algorithms
AI Core Unit Total (15 units minimum):
15
Track Requirement A
Artificial Intelligence: Principles and Techniques
CS 221 - Artificial Intelligence: Principles and Techniques
Track A Unit Total:
4
Track Requirement B
Two courses, each from a different area:
Area I) AI Methods
CS 229 - Machine Learning
Area II) Natural Language Processing
Area III) Vision
Area IV) Robotics
CS 131 - Computer Vision: Foundations and Applications
CS 231A - Computer Vision: From 3D Reconstruction to Recognition
CS 231C - Computer Vision and Image Analysis of Art
CS 231N - Convolutional Neural Networks for Visual Recognition
Track B Unit Total:
4
Track Requirement C
One additional course from the Track Requirement B list, or from the following:
CS 151 - Logic Programming
Track C Unit Total:
3


Elective Requirement
At least three additional courses from the Track Requirement B list, C list, the General CS Electives List, or the following:
CS 142 - Web Applications
CS 157 - Computational Logic
CS 251 - Cryptocurrencies and blockchain technologies
Elective Unit Total:
9
Senior Project Requirement
CS 191 - Senior Project
Senior Project Unit Total:
6
Graduation Requirements and Total Units
Graduation Requirements
If this box is red, at least one preceding requirement is not satisfied. Otherwise, congratulations on meeting all requirements for the BSCS degree!
Total Units:
93
Course List
Course
Title
Grade
Units
AP Chemistry
5
CS 103
Mathematical Foundations of Computing
5
CS 106B
Programming Abstractions
5
CS 107
Computer Organization and Systems
5
CS 109
Introduction to Probability for Computer Scientists
5
CS 111
Operating System Principles
5
CS 142
Web Applications
3
CS 151
Logic Programming
3
CS 157
Computational Logic
3
CS 161
Design and Analysis of Algorithms
5
CS 181
Computers, Ethics, and Public Policy
4
CS 191
Senior Project
6
CS 221
Artificial Intelligence: Principles and Techniques
4