DEGREEFLOW:
A VISUAL COURSE PLANNING TOOL

Adam Langevin

A Capstone Project Submitted to the
University of North Carolina Wilmington in Partial Fulfillment
of the Requirements for the Degree of
Master of Science

Department of Computer Science
Congdon School of Supply Chain, Business Analytics, and Information Systems

University of North Carolina Wilmington

2023

Approved by

Advisory Committee

Dr. Geoffrey Stoker

Dr. Toni Pence

Dr. Lucas Layman, Committee Chair

CONTENTS

Abstract

Course planning is a challenging problem due to its many complex dimensions, such as shifting course and degree requisites, selecting between alternative requisites, and conveying course catalog information to the user in a simple yet informative manner. This capstone project presents DegreeFlow—a full-stack web application—to address these challenges and and provide a visual, user-friendly solution to course planning. DegreeFlow allows users to look up course descriptions, visualize requisites as an interactive flow graph, and select between alternative requisites. DegreeFlow uses a normalized database structure to store course and degree data, converts that data into a flow graph, and provides an interactive visualization of the graph in the web browser. Twelve users, including students and faculty, provided qualitative feedback and rated the DegreeFlow on the System Usability Scale (SUS) after completing a set of predefined tasks. The users found searching and visualizing courses and their requisites intuitive, and provided recommendations for improving other features including alternative course selection. DegreeFlow averaged a 79.75 SUS score which is above the recommended threshold of 68.

List of Figures

LIST OF TABLES

I. Introduction

Planning a multi-semester college curriculum as a student can be difficult due to changing course and degree requirements such as prerequisites, corequisites, and a combination of the two. Some services have tackled the challenge of course planning before, such as Ellucian's Degree Works[1]. However, most planning tools do not prioritize course *requisites*–courses that must be taken before (prerequisites) or concurrently with (corequisites) a desired course. This leads students to ask themselves the same questions every time they plan to take a course:

1) Can this course be taken yet, or must another course or chain of courses be taken first?
2) Does another course, such as a lab, need to be taken at the same time as this course?

As students add courses to their plans, they answer these questions by searching the course catalog and semester schedule, flipping from page to page, and tracking a chain of requisites as one course points to another, which points to another, and so on. Then, should they change their plan, they must search through the course catalog again and repeat the whole process. Furthermore, whenever they ask their advisors questions, the advisors must refer to the course catalog and consult the student's degree audit to ensure a course is a valid choice for that student.

Automation can improve the process of researching courses and their requisites. This paper details a full-stack web-based solution, *DegreeFlow*, that outputs a visual flow graph of course requisites given an input of courses and degrees. DegreeFlow helps students and faculty plan a student's course schedule in a manner left unfulfilled by previous solutions. Specifically, DegreeFlow addresses the following shortcomings of previous solutions:

1) Enabling users to visualize entire requisite chains as a graph and not just direct requisites
2) Enabling users to track their progress through these chains throughout their college careers

DegreeFlow emphasizes usability, including intuitive mechanisms for searching for courses, adding them to a visualization of course requisites, selecting between alternative requisites, and saving and printing visualizations.

This paper is organized as follows: Section II analyzes existing course scheduling tools. Section III outlines DegreeFlow's architecture, design, and challenges. Section IV discusses DegreeFlow's capabilities with images of DegreeFlow's interface, and Section V overviews the feedback attained from DegreeFlow users.

[1]https://www.ellucian.com/solutions/ellucian-degree-works

II. Related Works

A. Web-based Decision Support Tool for Academic Advising

In 2011, Feghali et al. wanted to improve the time students and advisors spent planning a student's schedule by decreasing the time needed to research degrees, collect courses, and resolve issues [1]. After several meetings with advisors, faculty members, administration, and students, the team developed three main pages: the summary page, the degree checklist page, and the degree plan page (Figure 1). All three pages use styling to convey information, such as cell color conveying a student's status in any course. However, they all utilize a table structure which can be more challenging for a user to digest than a graph or visual aid. To add courses to the degree plan page, students must select the specific semester they want to construct a plan. The student then selects courses they want to take. Students can only select from courses within their degree. In the end, students click the 'Update Degree Plan' button at the bottom of the page to update the database with the new information. The team evaluated their results through two surveys based on user satisfaction levels. The first survey asked about the original method, and the second asked about their new web-based tool. The findings were that students saw their advisors more frequently, found them more helpful, and were more satisfied with the new web-based product on average. While Feghali's solution improved course planning in 2011, it did not visualize course requisites.

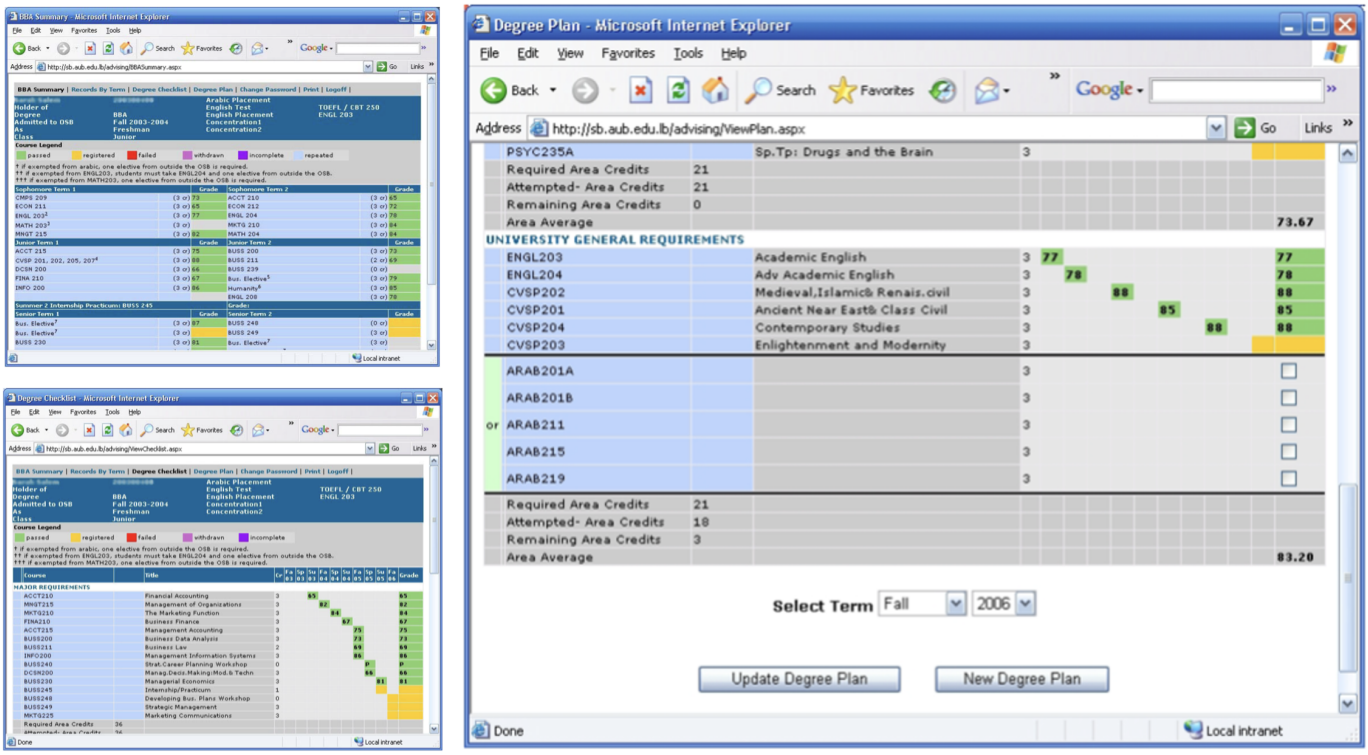

Fig. 1: Feghali's Summary, Degree Checklist, & Degree Plan Pages [1]

B. *Ellucian Degree Works*

Ellucian is a company that develops various Software as a Service (SaaS) products for more than 26 million students and over 2,700 higher education schools [2], including the University of North Carolina Wilmington (UNCW). The Degree Audit section of Degree Works displays a checklist of courses required for a given major, concentration, or minor but also has a ”What-If” feature where students can view requirements for other majors, concentrations, or minors. The Degree Plan, as seen in Figure 2, allows students to plan their schedules multiple semesters in advance using a graphical structure. Semesters are wide columns, and courses are cards that can be dragged from one semester to another. As a system that is still managed today, Ellucian’s Degree Works has evolved with web standards and is more polished when compared to Fegali’s work. This system thrives at tracking a student’s overall progress towards attaining their degree as it keeps track of both courses and credit hours completed. While the system can display course information and requirements, it can only do so one course at a time. To discover a course’s requisites, the user must add the course to their plan by dragging it into a semester, clicking the course’s options menu, and selecting *more information*. Then a modal that covers the plan provides the course information and requirements—an example of this can be found in Figure 3. This means a user cannot view multiple courses’ requirements simultaneously to validate their plans without significant manual effort.

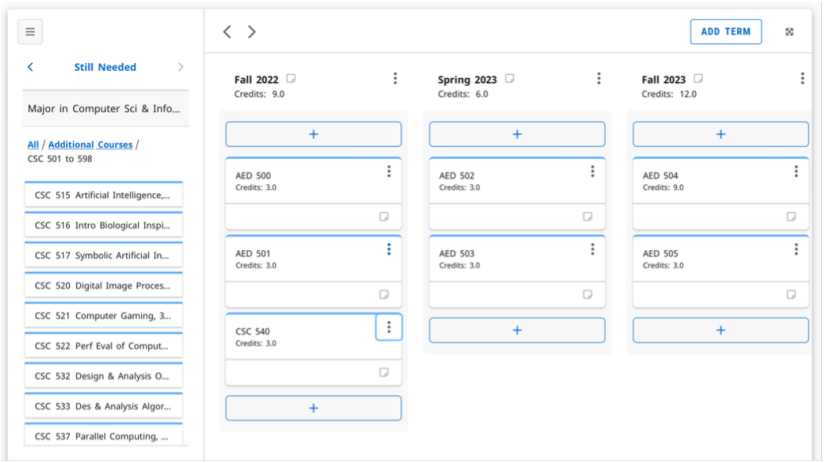

Fig. 2: Ellucian Degree Work’s Degree Plan Graphical User Interface

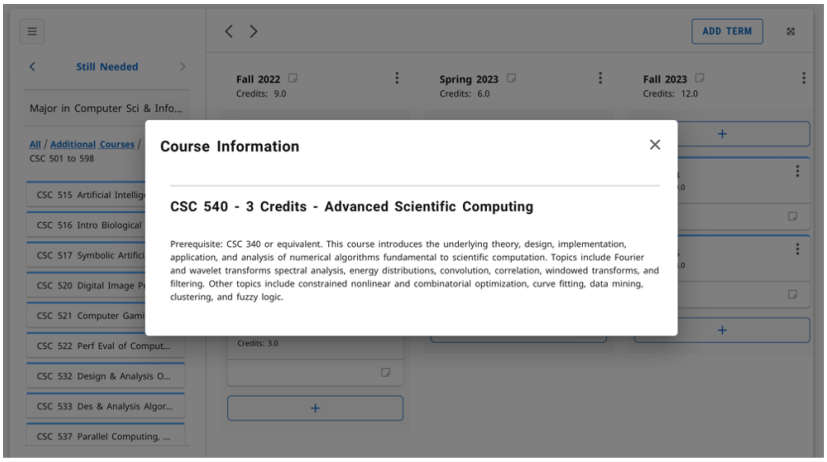

Fig. 3: Ellucian Degree Work’s Degree Plan Displaying a Course’s Information

C. The Curriculum Prerequisite Network

My literature review did not find a functional online system that allows students to visualize course dependencies. However, Aldrich researched how a network of course requirements could be visualized and measured through the perspective of graph theory [3]. He used Python[2] and the Beautiful Soup library[3] to scrape prerequisite course requirements from Benedictine University's online course catalog and NetworkX[4] to convert the information into the "curriculum prerequisite network" (Figure 4). Aldrich researched a variety of course requirements, including prerequisites, hard corequisites, soft corequisites, and cross-listings. However he only visualized prerequisite relationships in his final model to ensure the graph would be acyclic. On top of this, his product was strictly used as a research artifact and lacked a graphical user interface.

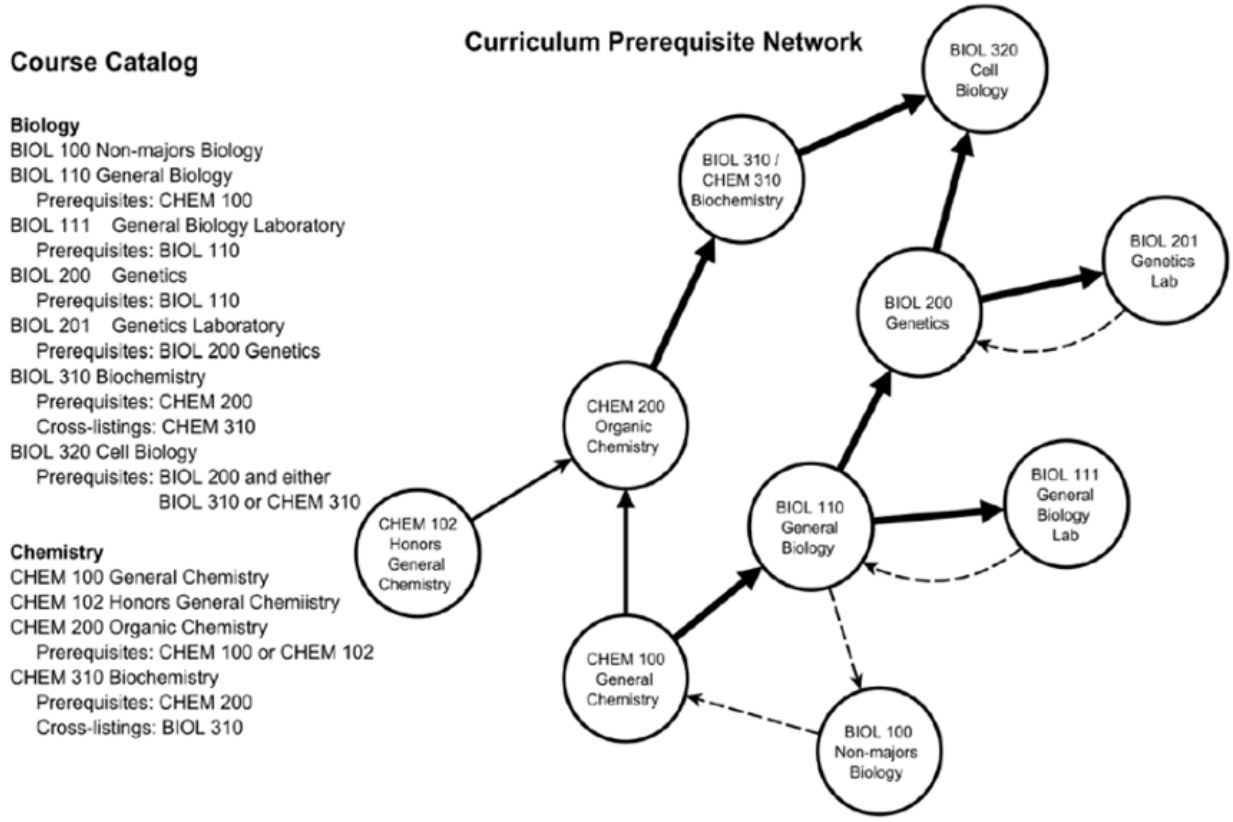

Fig. 4: A segment of Aldrich's Curriculum Prerequisite Network [3]

[2]https://www.python.org
[3]https://pypi.org/project/beautifulsoup4/
[4]https://networkx.org

D. Visual Degree Audit Software

Hom-Nici developed a prototype visual curriculum planner and explored the design side of this issue [4]. She communicated with students and faculty to determine what features they would like improved or added to the current system. She then explored multiple ways to visualize her prototype, including a flow graph and a Gantt chart. Hom-Nici also determined different ways to display information to the user, including text, connecting lines, node shape, and node color, as seen in Figure 5. This capstone uses connecting lines like Hom-Nici did to illustrate requirements, and this capstone uses node color to convey a course's requirement completion. Ultimately, Hom-Nici developed a prototype of a system meant to replace Texas State University's current system. She used inVision to create the prototype and evaluated it with a sample of five students. In a survey, the students answered favorably to many questions aimed at the new visual degree audit prototype. Hom-Nici's solution was strictly a visual prototype, not a functioning web application.

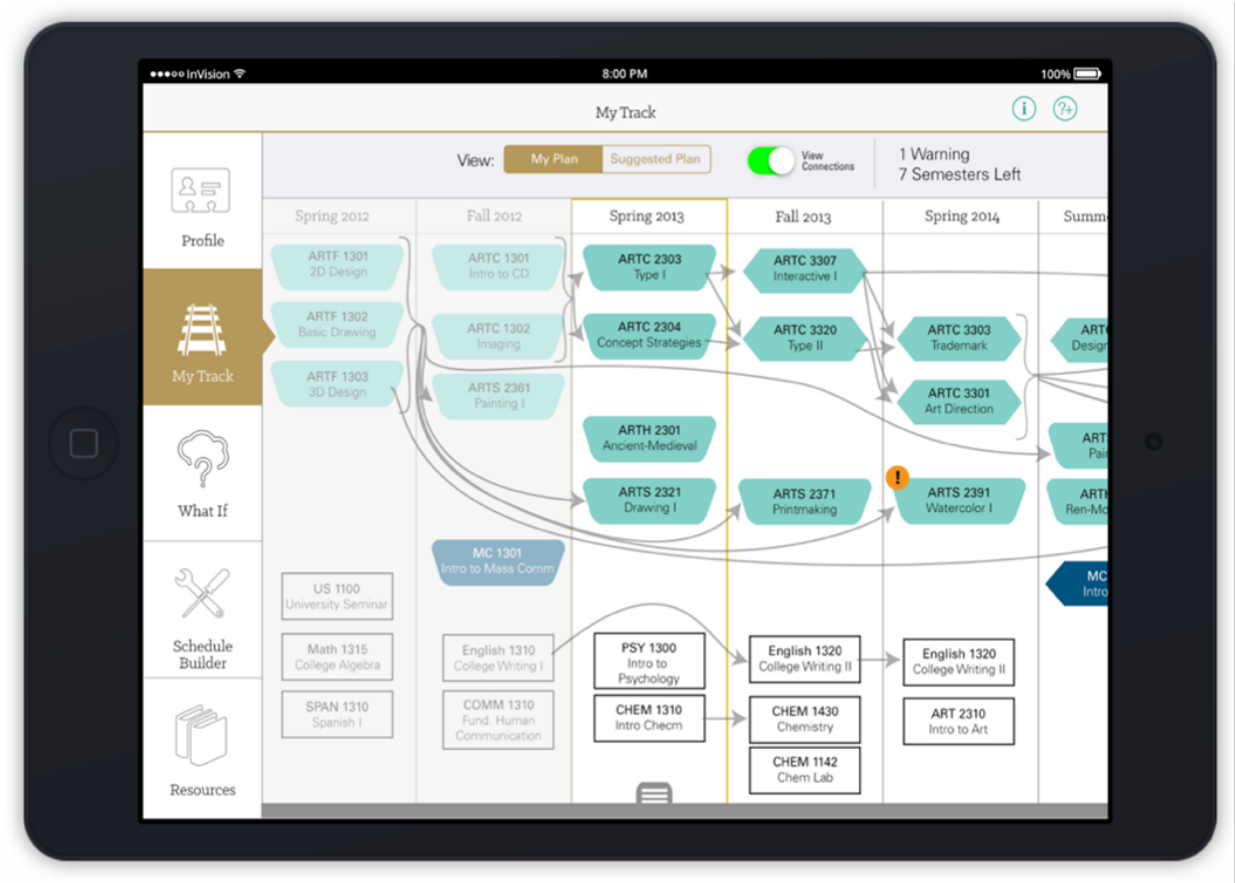

Fig. 5: Hom-Nici's Course Planning Prototype
(https://projects.invisionapp.com/share/7DPRCGNM#/screens/19372587)

E. Challenges and Opportunities from Related Work

The course planning solutions listed above all encountered similar challenges in acquiring course information: course numbers, descriptions, prerequisites, and corequisites. Since Ellucian Degree Works is a SaaS product paid for by the universities that use it, they have direct access to the universities' course data and system administrators that manage it. Hom-Nici's project avoided this challenge by developing a prototype that is more focused on the design and does not depend on the course information. Aldrich's solutions scraped all necessary course information from the online course catalog. As will be discussed in the next section, acquiring and representing course data was also a challenge in this capstone project.

All of these solutions lacked the core functionality essential for a usable system that solves course planning issues. Feghali's solution and Ellucian Degree Works cannot show requisite chains. Aldrich's solution visualized course prerequisite chains; however, it was strictly a research tool with no interactive user interface. Hom-Nici's solution is a well-planned graphical user interface, but it is strictly a visual prototype and not a data-driven end-user system. These deficiencies create an opportunity to build a visualization-focused course planner based on real data that emphasizes usability to help reduce the burden of course planning and academic advising.

III. DegreeFlow Architecture

DegreeFlow is a full-stack web application with a database, server, and client (Figure 6).

- The *database* stores course catalog information, user login data, and user-created visualizations they wish to persist.
- The *server* queries the database and converts courses, degrees, and their requisites into an intermediate representation used by the client to build graph nodes and edges.
- The *client* visualizes courses, degrees, and their requisites as nodes and edges in an interactive flow graph.

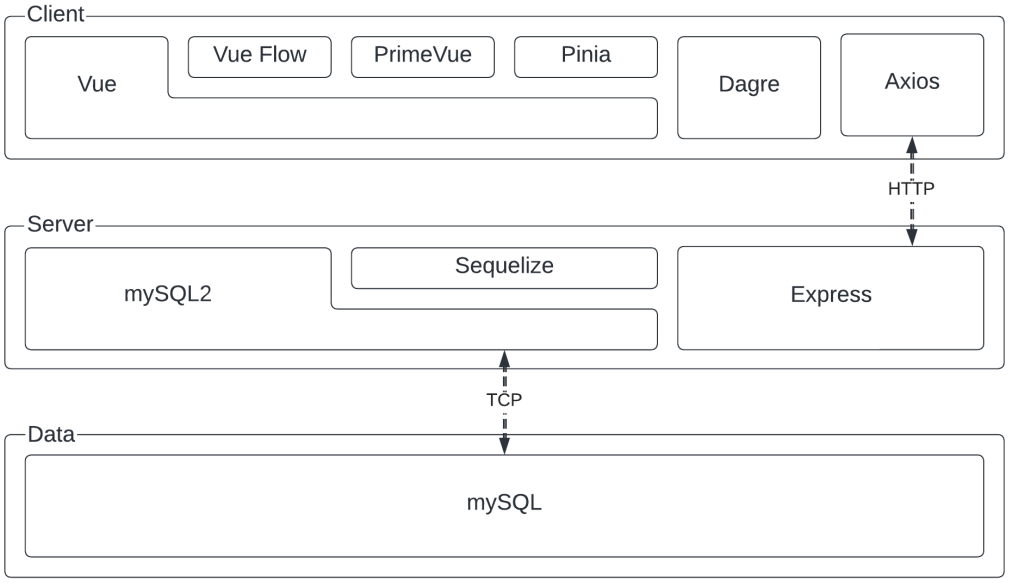

Fig. 6: DegreeFlow Architecture

A. The Data

DegreeFlow's course and degree data is stored in a MySQL database. The database is conceptually divided into two segments—the course catalog segment and the course visual segment.

1) Course Catalog Data: After researching UNCW's online course catalog and a report attained from UNCW's Registrar's office, it was apparent that a new database would have to be developed to store courses and their requisites. The existing database used by UNCW does not follow a normalized schema that would enable efficient traversal of requisite courses. Six course attributes were needed from each course. Table I outlines these attributes along with some examples.

TABLE I: Six Course Attributes Required by DegreeFlow with Examples

Attribute	**Example 1**	**Example 2**
Subject	CSC	CHM
Number	220	212
Title	3-D Computer Graphics Tools and Literacy	Organic Chemistry II
Description	Learning fundamental principles of 3-D computer...	Reactions and reaction mechanisms of organic compounds.
Credit Hours	3	3
Corequisite		CHML 212
Prerequisite	CSC 112	CHM 211 AND CHML 211
Pre or Corequisite	ART 260	

UNCW's online course catalog[5] contains all the required attributes; however, the catalog does not have a consistent format for specifying requisites. I implemented a Python program to scrape course names and descriptions from the UNCW online catalog and output the data to a CSV file. However, the CSV file had to be reviewed manually due to inconsistent course formatting in the catalog. The Registrar's report included all Computer Science and Biology course subjects, numbers, and their prerequisites, as seen below in Table II. However, the report was not easily queried using SQL methods and would have needed to be restructured. As a result, all requirements were manually entered into a CSV file and then converted to a JSON file. A server-side script was run to populate or update the database when necessary.

TABLE II: Portion of Course Database Report Provided by UNCW's Registrar's Office

COURSE SUBJ	*COURSE NUMB*	*DEPT CODE*	**COURSE TITLE**	*COURSE LEVEL*	*SEQ NO*	*CONNECTOR (AND, OR)*	*LEFT PARENTHESIS*	*PRE REQ SUBJ CODE*	*PRE REQ CRSE NUMB*	*PRE REQ LEVEL*	*PRE REQ MIN GRADE*	*CONCURRENCY IND*	*RIGHT PARENTHESIS*
BIOL	140	BIO	Human Physiology Laboratory	UG	2			BIO	140	UG	D-	Y	
BIO	240	BIO	Human Anatomy and Physiology I	UG	2		(	BIO	201	UG	D-		
BIOL	240	BIO	Human Anat & Physiology I Lab	UG	2			BIO	240	UG	D-	Y	
BIO	240	BIO	Human Anatomy and Physiology I	UG	3	O		BIO	204	UG	D-		
BIO	240	BIO	Human Anatomy and Physiology I	UG	3.5	O		BIO	110	UG	D-		)
BIO	240	BIO	Human Anatomy and Physiology I	UG	4	A		CHM	101	UG	D-		
BIO	240	BIO	Human Anatomy and Physiology I	UG	6	A		BIOL	240	UG	D-	Y	
BIOL	241	BIO	Human Anat & Physiology II Lab	UG	2			BIO	241	UG	D-	Y	
BIO	241	BIO	Human Anat & Physiology II	UG	2			BIO	240	UG	D-		
BIO	241	BIO	Human Anat & Physiology II	UG	4	A		BIOL	241	UG	D-	Y	
BIOL	246	BIO	Microbio of Human Diseases Lab	UG	2			BIO	246	UG	D-	Y	
BIO	246	BIO	Microbiology of Human Diseases	UG	2		(	BIO	201	UG	D-		
BIO	246	BIO	Microbiology of Human Diseases	UG	3	O		BIO	110	UG	D-		
BIO	246	BIO	Microbiology of Human Diseases	UG	3.5	O		BIO	204	UG	D-		)
BIO	246	BIO	Microbiology of Human Diseases	UG	4	A		CHM	101	UG	D-		
...	...	...	...	...	...	...	...	...	...	...	...	...	...

For this iteration of DegreeFlow, the database content was limited to a subset of courses from the Computer Science and Biology Departments to save time required to manually review and input course data. Degree selection was limited to a Computer Science Major with a Concentration in Systems, a Computer Science Major with a Concentration in Biology, and a Biology Major with a Concentration in Biology. A few additional courses outside the Computer Science and Biology Departments were also added to satisfy the degree requirements, such as Math and Statistics courses.

The data was stored in a normalized database, as represented by Figure 7. There are two noteworthy aspects of the database. The first is the hierarchical structure of `Coreqs`, `Courses`, and `Listings`. This minimizes the redundancy of course information. The database's second noteworthy aspect is how requisites are stored as `Combos`. This scheme can represent any combination of corequisites with conditionals such as *or* and *and*. This structure enables the database to handle nested conditional such as `(A AND B)OR (C AND (D OR E))`.

[5]https://catalogue.uncw.edu/index.php

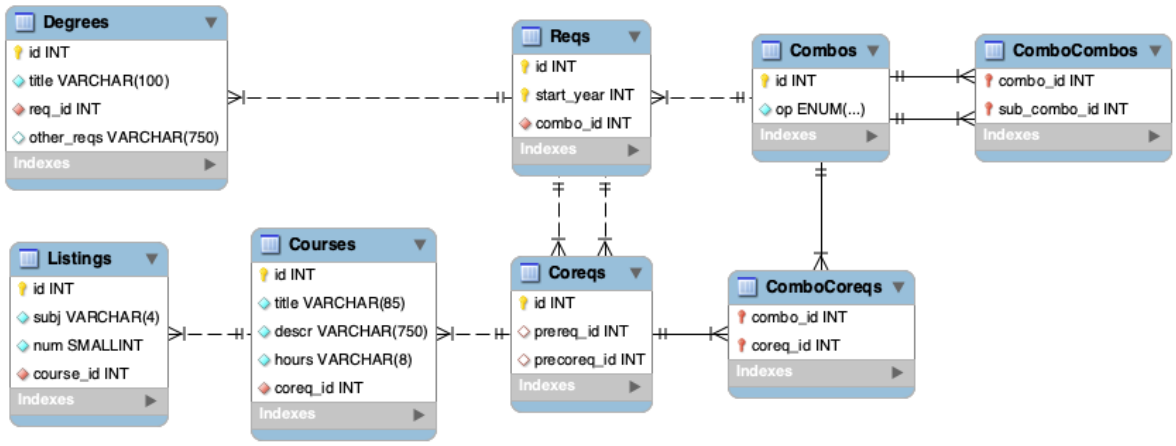

Fig. 7: ERD of the Course Catalog Portion of the New Database.

2) Course Visual Data: The course visual section of the database stores user information and the visual information each user has saved to their account. All the nodes and edges of the visual are exported to a JSON object on the client side and stored in the database as visual elements. A noteworthy aspect of this part of the database is the inclusion of the start year on the visual so that it can be tailored to a specified catalog year.

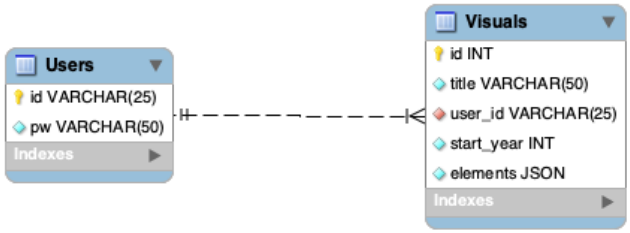

Fig. 8: ERD of the Course Visual Portion of the New Database.

B. The Server

DegreeFlow's server is responsible for querying the database and converting course catalog data into data used by the nodes and edges on the client side. The server is a Node.js[6] application written in TypeScript and built using Express[7]. HTTP requests passed to the server from the client are read by Express, and then Sequelize[8] utilizes MySQL2[9] to query the database for the requested data.

1) Server-side Data Objects: DegreeFlow converts course catalog data into one of three objects: `CoreqNodeData`, `DegreeNodeData`, or `EdgeData`. The server passes these objects to the client, where they are used to populate nodes and edges.

- `CoreqNodeData` objects contain data about a course, including title, description, and hours. It also contains a flag to identify if the course was searched for and added manually to the graph by the user or was added as part of a prerequisite

[6]A Javascript runtime environment (https://nodejs.org/en/)
[7]A minimal Node.JS framework that provides features for web development (https://expressjs.com)
[8]An object-relational mapping tool (https://sequelize.org)
[9]A MySQL driver (https://www.npmjs.com/package/mysql2)

chain. This manual flag plays a part in determining whether or not a course can be removed from the graph on the client side.

- `DegreeNodeData` objects contain information about an entire degree, including the title, miscellaneous requirements the system cannot enforce (e.g., "A student must take 9 additional credit hours of biology electives"), and a flag controlling the visibility of the node.
- `EdgeData` objects represent the relationship between `NodeData` objects. They include ids of the *source* and *target* nodes of an edge in the graph, a unique id for the edge, a flag indicating which line style to use (dashed or solid depending on the requisite type), and other data needed to represent more complicated "OR" relationships between nodes.

2) Course Requisite Chain Algorithms: The conversion process begins when the client requests the course requisite chain of a course listing or degree id. DegreeFlow queries the database for the `Coreq` or `Degree` associated with the given object and then runs an algorithm similar to a recursive preorder tree traversal with the `Coreq` or `Degree` as its root. If a course listing is provided, the process is as follows:

1) The server queries the database for course listing and receives a `Coreq` entity. This `Coreq` becomes the root of the tree-like structure, and its requisites become the branches.
2) DegreeFlow then creates an instance of `CoreqNodeData` from `Coreq` and appends it to a `CoreqNodeData` list.
3) DegreeFlow then iterates through each requisite of the current `Coreq`, creating and appending `EdgeData` to a list.
4) DegreeFlow recursively repeats steps 2 and 3 on every `prereq` or `precoreq` course of the current `Coreq`.

If a degree id is provided, the process is as follows:

1) The server queries the database for the `Degree` entity to which the given degree id belongs (This `Degree` becomes the root of the tree-like structure, and its requisites become the branches).
2) DegreeFlow then creates an instance of `DegreeNodeData` from `Degree` and appends it to a list of `DegreeNodeData` and `CoreqNodeData` list.
3) DegreeFlow then iterates through each requisite course of the degree, creating `EdgeData` and appending them to an `EdgeData` list (It is important to note that Degree nodes and edges are always invisible to keep the user output simple).
4) DegreeFlow recursively repeats steps 2 and 3 from the `Coreq` process on every `Coreq` that is a `req` of `Degree`.

The server does not know what is in the visualization and what is not. Therefore, the server will always return the same result—all nodes and edges within a course requisite chain. The client visualization determines which nodes and edges to render.

C. The Client

The client provides a graphical user interface that allows the user to visualize course requirements as a graph using node (course) and edge (requisite) data provided by the server. The client is written in Typescript and utilizes Vue.js[10] along with five application programming interfaces (APIs) to create a reactive, single-page user interface.

1) *Vue Flow*[11] is an interactive graph API made explicitly for Vue. The main part of DegreeFlow—the visualization—was

[10]A reactive web framework for building user interfaces (https://vuejs.org)
[11]https://vueflow.dev

created using Vue Flow interactions paired with custom nodes and edges.

2) *PrimeVue*[12] is a component library specifically made for Vue. All DegreeFlow's components came from PrimeVue; this includes the input fields, buttons, and sidebar menu. All icons also came from PrimeVue. Only pulling icons from this library did become a challenge due to its lack of variety.
3) *Pinia*[13] is a datastore library made explicitly for Vue. Pinia allowed for the creation of data stores that would persist data across components. This means that when the sidebar menu is closed, the Visualization still has access to the visualization name or other values.
4) *Dagre*[14] is a library that makes it easier to arrange graphs. Dagre was used within this system to implement the auto-arrange feature whenever a node is added or the user selects rearrange. While this is a good start for the auto-arranging of nodes, there is room for improvement as it leaves gaps for nodes that are present but not visible.
5) *Axios*[15] is an HTTP client for the browser that allows the client code to communicate with the server via HTTP requests.

The primary function of the client is to display a visualization of course and degree requisites. DegreeFlow does this by requesting `NodeData` and `EdgeData` from the server given a course listing or degree id. Once the server returns the included `NodeData` and `EdgeData`, the client visualizes this data by completing the following process:

1) Node and edge objects are created from the `NodeData` and `EdgeData`.
2) Nodes and edges already present in the visualization are removed from the set of new nodes and edges. If the manual flag on the new node or edge is set to true, the manual flag on the matching present node or edge is set to true.
3) All remaining nodes and edges are added to the graph. First, all the nodes are added, and then all the edges are added.
4) A flag on each edge determines visibility. This flag is toggled based on the user's selection within the alternatives menu (see section IV-D). If a node has no outgoing edges and was not manually added, its opacity is set to zero (not visible).
5) Nodes and edges are fed into the Dagre library, which returns suggested coordinate positions for the nodes in the flow graph canvas. These positions are extracted and used to set the positions on the Vue Flow nodes.

After nodes and edges are initialized, any change to a course's completion status triggers a change in the node fill colors, and any changes to the course's alternative requirements trigger a change in node opacities.

[12]https://primevue.org
[13]https://pinia.vuejs.org
[14]https://www.npmjs.com/package/dagre
[15]https://axios-http.com

IV. DegreeFlow Functionality

This section demonstrates the course-planning tasks supported by the DegreeFlow application. This functionality complements the current registration system and degree audit tools.

A. Search and Add to the Visualization

A user searches desired courses by typing the course subject and number into their respective fields (Ⓐ in Figure 9) within the *Edit* menu. The course subject is the first three letters in a course listing, such as ”CSC” for ”CSC 131”. Likewise, the course number is the last three digits of a course listing, such as the ”131” portion of ”CSC 131”. When the user selects the information button (Ⓑ in Figure 9), DegreeFlow loads the course’s name, description, and credit hours (Ⓒ in Figure 9). The user can search for a course already present in the visualization by selecting the information button with the course node they would like to search (Ⓓ in Figure 9).

A user can add courses to the visualization by typing the course subject and number into their respective input fields (Ⓐ in Figure 9) and selecting *Add Course* (Ⓔ in Figure 9). The input course, and all its requisites, will be added to the visualization. Similarly, a user can visualize a degree’s requisites by choosing a degree from the *Select a Degree* pick list (Ⓕ in Figure 9) and then selecting *Add Degree* (Ⓖ in Figure 9).

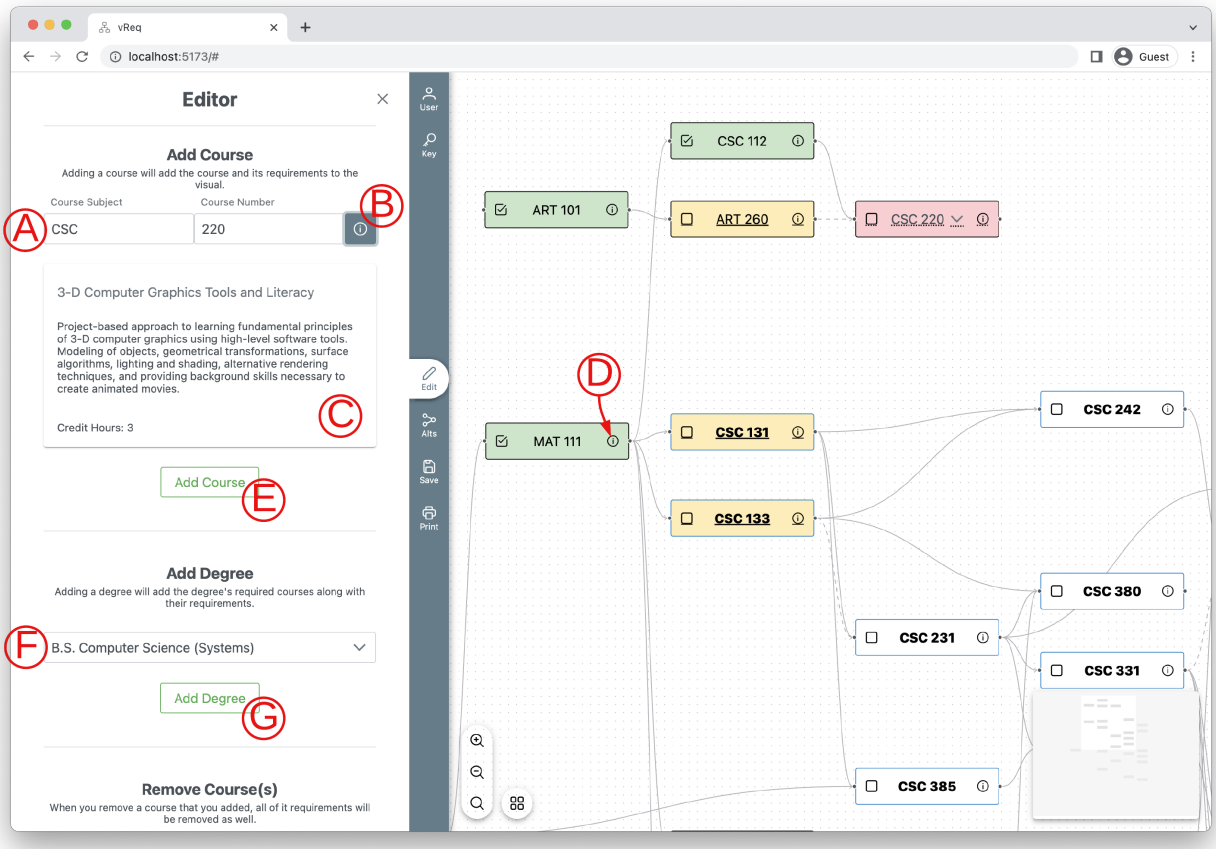

Fig. 9: Search and Add Functions

B. *Visualize Course/Degree Requisites*

With each course a user adds, DegreeFlow visualizes three types of course requisites if they are present:

1) *Prerequisite relationship:* Course A must be taken before Course B. This relationship is illustrated by a solid arrow coming from Course A pointing toward Course B (Ⓐ in Figure 10).
2) *Soft corequisite relationship:* Course A must be taken before or at the same time as Course B but not after. This relationship is illustrated by a dashed arrow coming from Course A pointing toward Course B (Ⓑ in Figure 10).
3) *Hard corequisite relationship:* Course A must be taken at the same time as Course B. This relationship often appears with labs required for a course but listed separately, e.g., BIO 201 + BIO 201L. This relationship is illustrated as two sub-nodes encapsulated within one parent node (Ⓒ in Figure 10).
4) *Degree Requirements:* Course A must be taken to receive a selected degree. This requirement is illustrated by a node with a blue outline and bold text (Ⓓ in Figure 10).

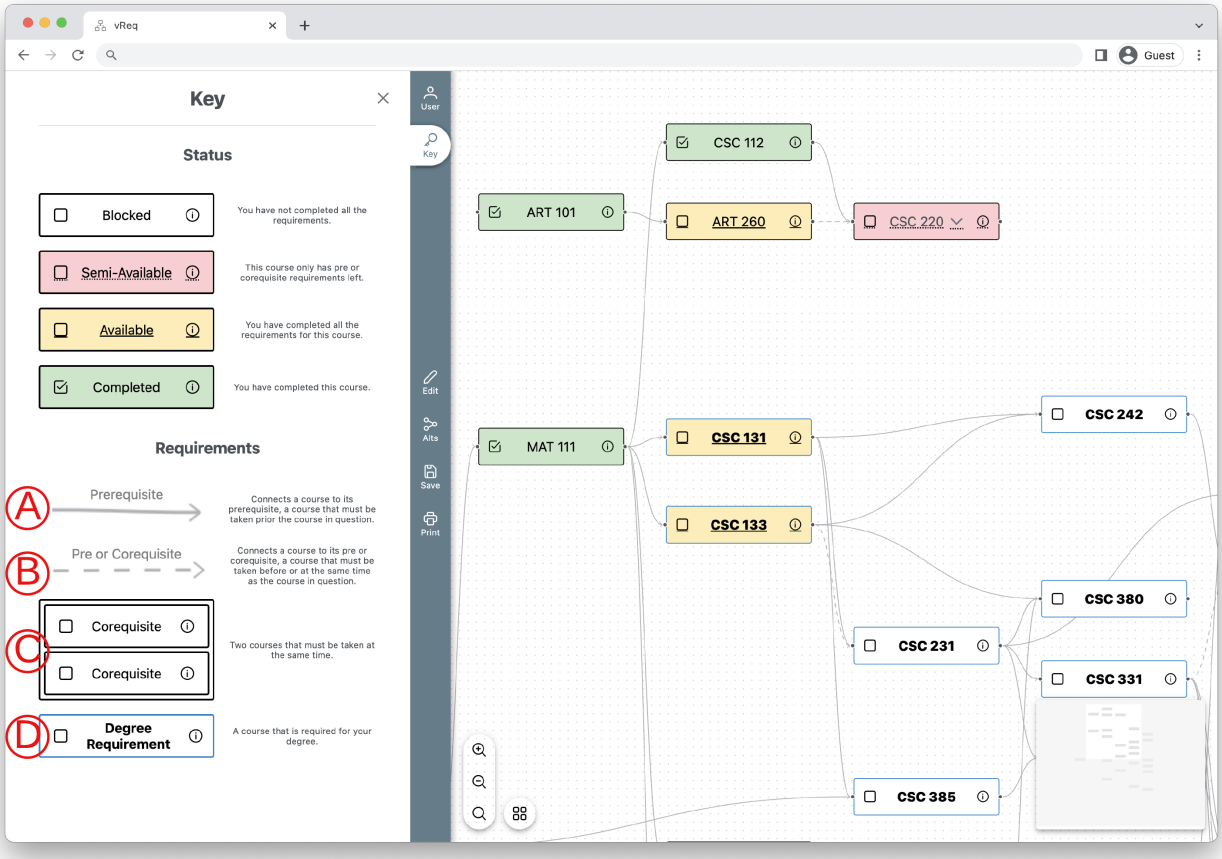

Fig. 10: Visualization of Course Relationships

C. Visualize Course Availability and Completion

After courses have been added to the visualization, students can mark courses as complete by selecting the check box within a node (Ⓔ in Figure 11). A course node's background color, check box, and text underline reflect its *availability*. *Availability* indicates whether a course can be taken based on the courses marked as complete by the student. There are four levels of availability:

1) *Blocked:* Courses with at least one incomplete prerequisite (Ⓐ in Figure 11). A student cannot register for these courses without an exemption.
2) *Semi-available:* Courses whose prerequisites are completed except for soft corequisites (Ⓑ in Figure 11). These are courses that students can register for if they also register for its soft corequisite.
3) *Available:* Courses with all prerequisites and corequisites completed (Ⓒ in Figure 11). Students can take these courses immediately.
4) *Completed:* Courses marked as completed (Ⓓ in Figure 11).

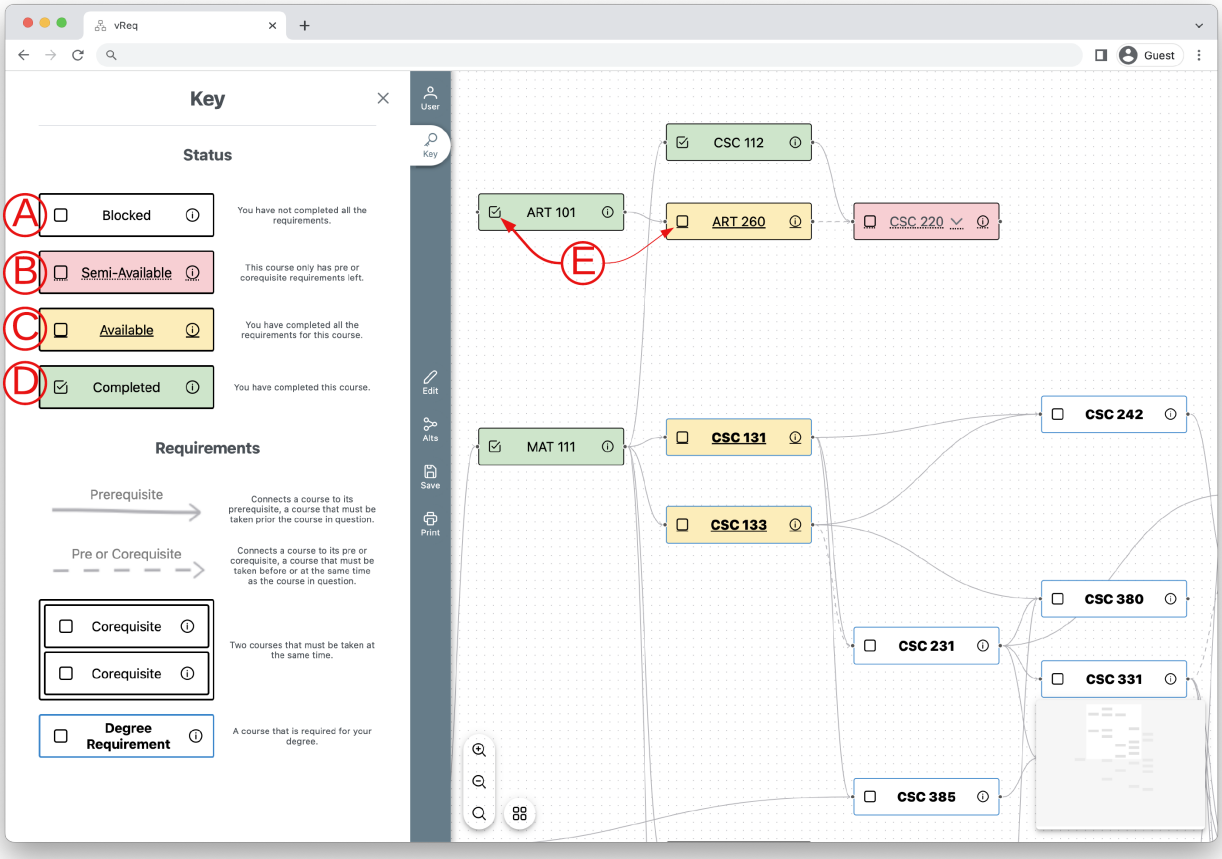

Fig. 11: Visualization of Course Availability/Completion Statuses

D. Select Alternative Requisites

Whenever a course or degree has alternatives to its requisites—in other words, its requisites contain an `or` condition—the user will find an entry in the *Alternatives (Alts)* menu. The alternative menu allows a user to view a course's alternative requirements by selecting a given course (Ⓐ in Figure 12). The user then selects which option (Ⓑ in Figure 12) they want to visualize within any of the listed requirements. This will change which edges and nodes are showing (Ⓓ in Figure 12). The visualization automatically hides alternative nodes that are *not* selected – they are technically still in the graph but invisible. This strategy for visualizing alternative requirements is limited to one level of `or` statements, e.g., `(A or B)and (C or D or E)`. While the server and database are capable of handling arbitrarily nested requisites, the client visualization currently cannot handle nested `or` statements such as `A OR (B AND (C OR D))`.

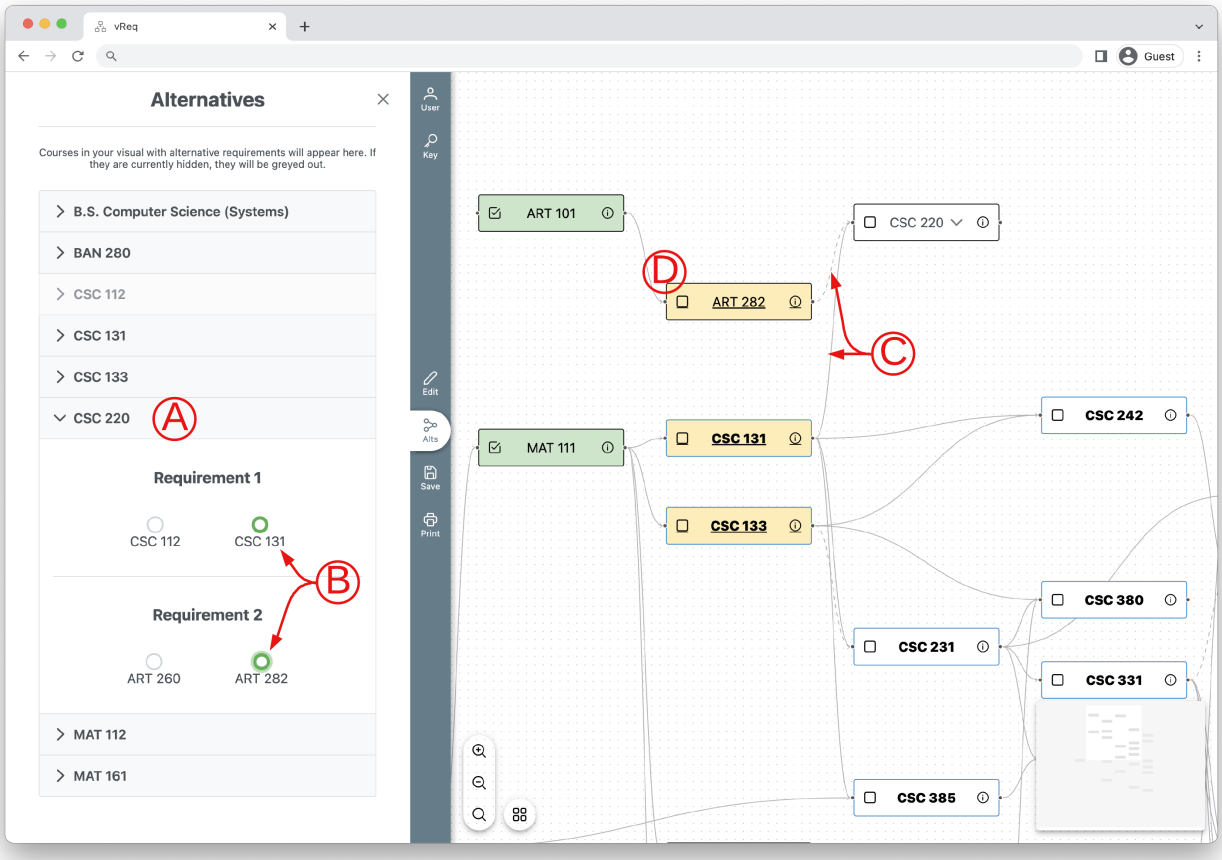

Fig. 12: Alternatives Menu

E. Arrange Nodes

A user can move course nodes and arrange them to create a multi-year course plan, as illustrated in Figure 13. To drag a course, a user clicks anywhere on the course node and drags it to where they want to place it. If the user would like to reset the node positions, they select the rearrange button (Ⓐ in Figure 13).

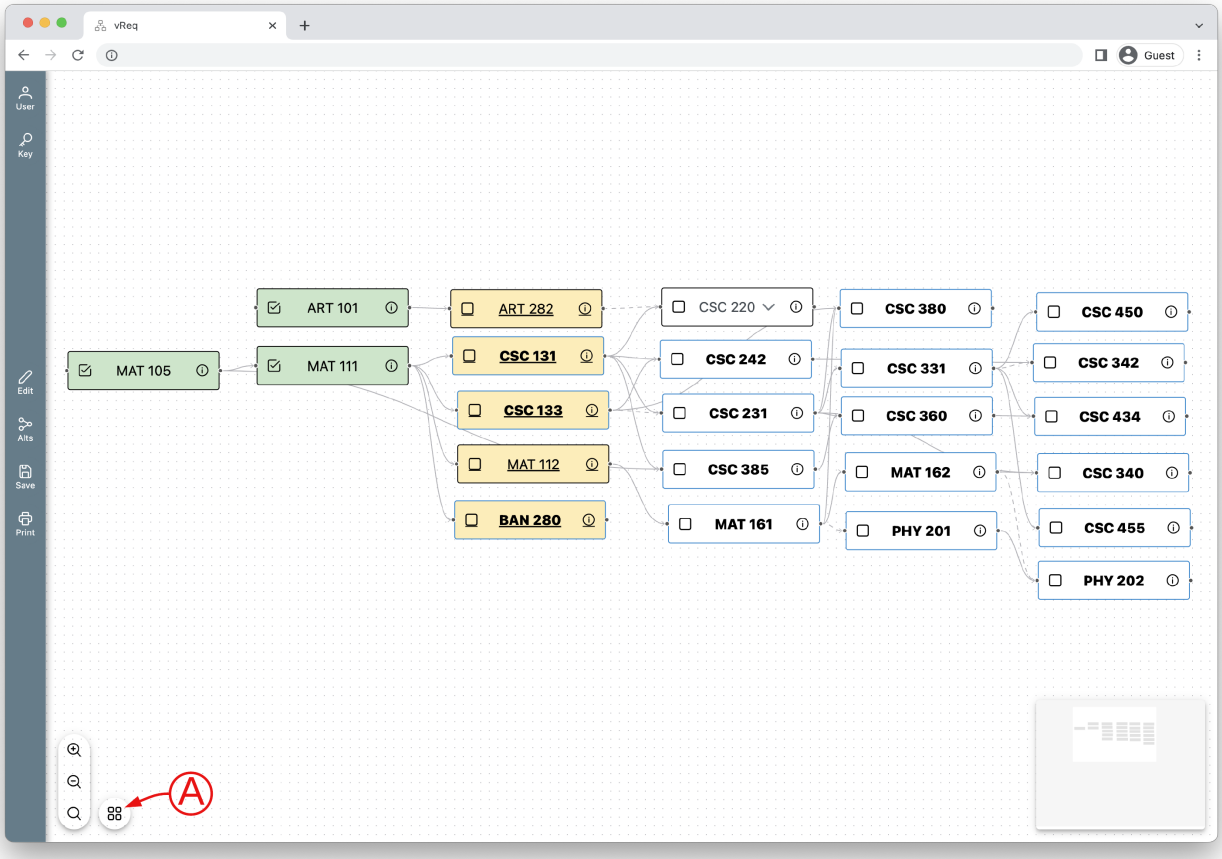

Fig. 13: Arranging Nodes

F. Remove Courses

A user can remove courses under the *Remove Course(s)* section of the *Edit* menu (Ⓐ in Figure 14) within the edit tab. The *Remove Courses* section lists all the courses and degrees a user has manually added to the visualization. To remove courses, the user selects the ⓧ next to the desired course or degree (Ⓑ in Figure 14). This will remove that course or degree and all of its requisites so long as it is not a requisite for another course or degree still present in the visual. To remove all the nodes and edges, a user can select the *Remove All* button (Ⓒ in Figure 14). A confirmation dialog is shown whenever a destructive action is about to occur, such as deleting or losing unsaved work.

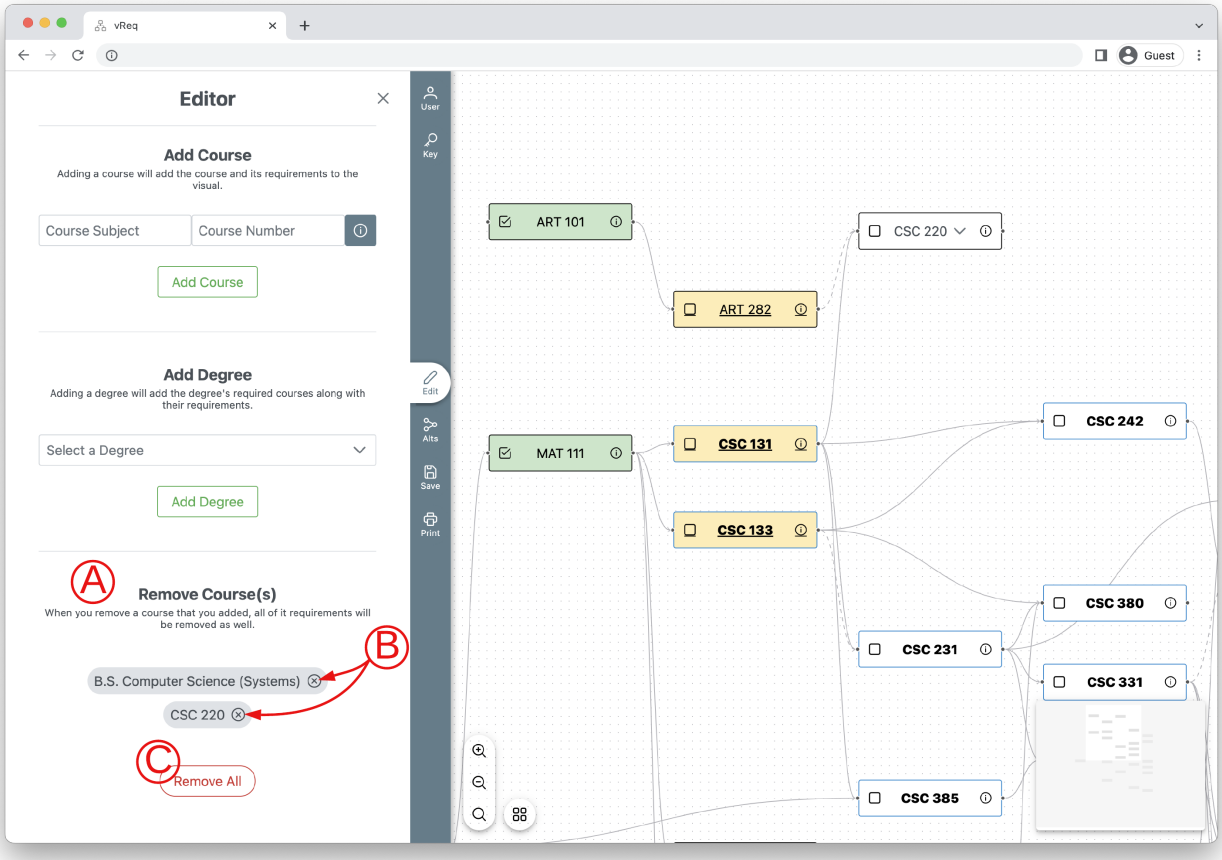

Fig. 14: Removing Nodes

G. *Manage the Visualization*

All visualization state management is handled under the *Save* tab. To create a new visual, a user selects *New Visual* (Ⓐ in Figure 15). This will close any currently opened visual and remove all nodes and edges from the visualization. As mentioned earlier, because this is a destructive action, a confirmation modal will appear for the user to confirm this action if there is unsaved progress on their current visual. To save a visualization in its current state, the user provides the visual with a title in the *Add Title* text box (Ⓑ in Figure 15) and selects *Save* (Ⓒ in Figure 15). To load a visual, a user selects the name of the visual they would like to load from the *Select a Visual* pick list (Ⓓ in Figure 15) and then selects *Open* (Ⓔ in Figure 15). Visuals are stored privately per user; thus, the user must be logged into their account to save or open visuals. To delete a visual, a user must open the visual they wish to delete. A delete Delete Visual button will appear at the bottom of the *Save* tab (near Ⓕ in Figure 15). Once again, selecting *Delete Visual* will open a confirmation modal asking the user to confirm their action.

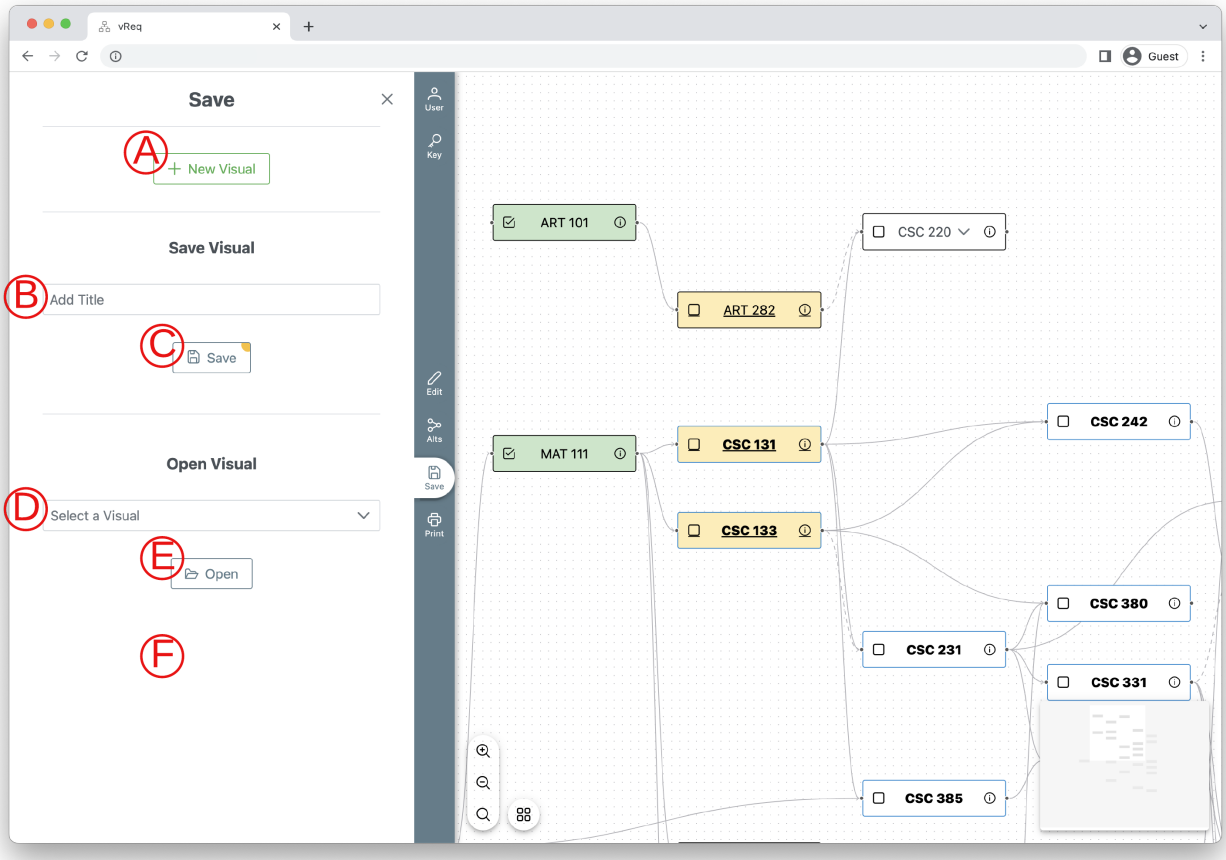

Fig. 15: Saving and Loading Visualizations

H. Print/Export the Visualization

To print or export the visual, a user navigates to the *Print* tab. This will preview the visual on the right-hand side (Ⓐ in Figure 16). The user has two options they can toggle. The first, *Print in Color* (Ⓑ in Figure 16), toggles the presence of color on the image. This was added so a user could print in black and white instead of grayscale. The second, *Include Course Descriptions* (Ⓒ in Figure 16), appends subsequent pages containing the course catalog entries for all visible courses in the visualization. Once a user selects their desired options, they select *Print* (Ⓓ in Figure 16) to open their web browser's print dialog. In the print dialog, the user can choose to print the visual or save it as a PDF.

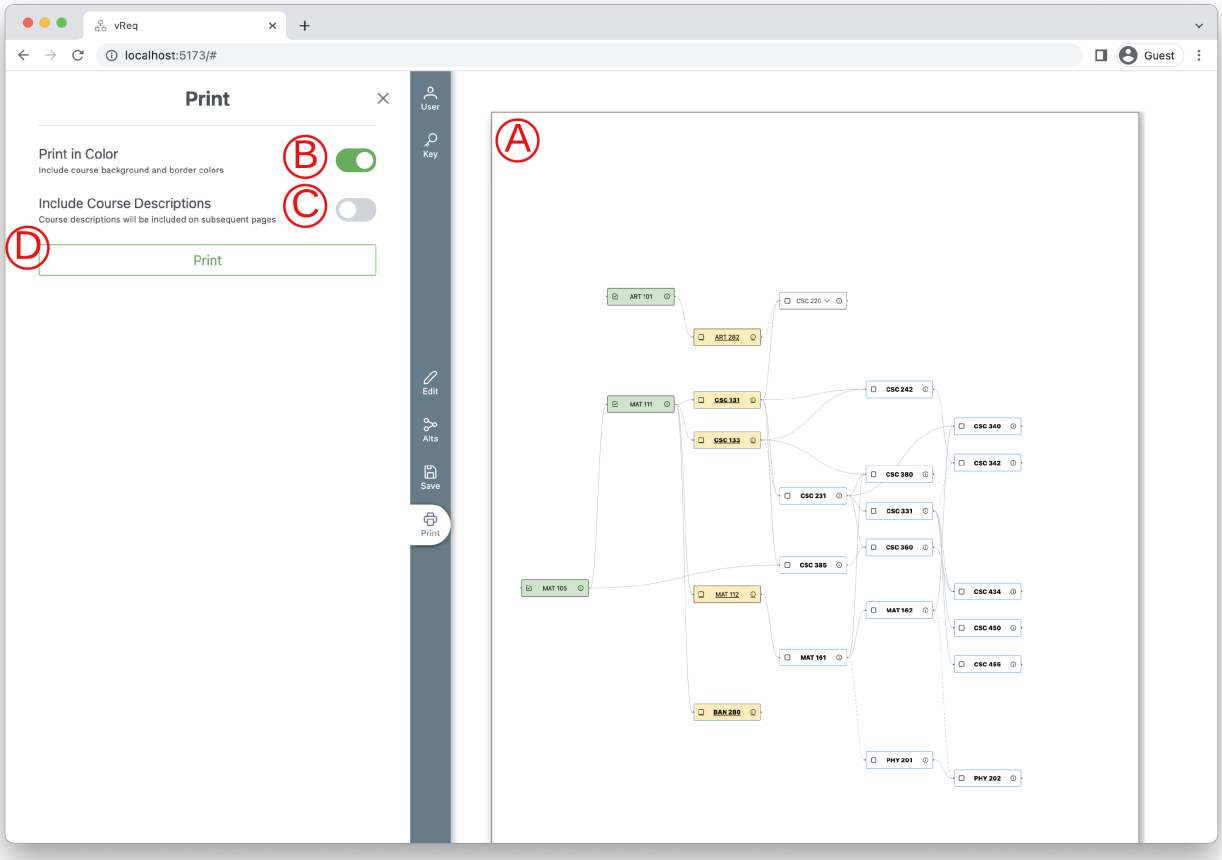

Fig. 16: Printing a Visualization

V. Evaluation

DegreeFlow was evaluated by volunteer students, academic advisors, and registrar officials (Table III), who provided both quantitative and qualitative feedback. The user evaluations were completed individually except for the representatives from the Registrar's office, who completed the evaluation as a group of three.

TABLE III: User Demographics

	Student	**Advisor**	**Registrar**
Computer Science Background	3	4	3[i]
Biology Background	1	1	0

[i] Three registrar officials tested DegreeFlow at the same time and submitted one usability survey

Each evaluation consisted of three stages. First, users completed a set of predefined tasks, including adding courses, adding majors, marking courses as complete, and selecting alternative requisites. The full user testing script, scenarios, and list of courses supported by DegreeFlow are provided in Appendix A. After completing the predefined tasks, users were given a chance to explore DegreeFlow in their own way and ask questions. The users then completed the System Usability Scale (SUS) [5] questionnaire. The SUS measures an application's ease of use by asking users to rate ten prompts using the Likert Scale; the SUS is provided in Appendix B. Their responses are then used to compute an overall score of 0–100. After completing the SUS survey, users were asked what their favorite feature was, what their least favorite feature was, and if they had any other comments. A consolidated list of this feedback can be found in Appendix C. The qualitative feedback was mainly positive, with some suggested changes or reoccurring pitfalls in DegreeFlow's usability.

Ten responses to the SUS questionnaire were collected (the Registrar representatives completed a collective response). The mean SUS score was 79.75, which is above the recommended score of 68 for a system to be deemed 'usable' [5]. The maximum score was 90, and the minimum score was 62.5. This means some users scored DegreeFlow as not usable. However, the majority of users scored DegreeFlow as usable. No average prompt score stood out among the rest, with values that ranged between 7.25 and 9. The complete response breakdown is provided in Appendix B.

Three features were repeatedly said to be users' favorite features:

1) *The visualization of course requirements as a flow graph:* Almost every user said this was one of their favorite features. Some would elaborate that they loved the visual simplicity of the flow graph and how it was not cluttered with excess information.
2) *The ability to print/export nodes in a custom arrangement:* Though not all users noticed the ability to arrange nodes in a custom format right away, once they did, they seemed to love it. Many claimed that because of this feature, DegreeFlow would be a much easier way to plan their courses instead of their current method, which usually involved a spreadsheet.
3) *The ability to mark nodes as complete and responsively see the availability of associated nodes:* Users liked this feature in tandem with the fact that you could save and return to the visualization each semester and track your progress.

Users stated they enjoyed using the tool and often emphasized most of their critiques were personal preferences. However, several reoccurring issues and suggestions emerged in their feedback:

1) *The lack of first-time user instructions:* Perhaps the most common critique was the first-time learning curve and how no instructions pointed first-time users where to go. Most users agreed once they learned DegreeFlow, it was not as hard to use, and upon returning to DegreeFlow, they would have no issue using it. However, since it was their first time, every user struggled with how they should get started. Most users agreed an interactive walkthrough, interactive tour, or video tour for first-time users would solve this issue.
2) *The alternatives menu:* Most users never even opened the alternatives menu to select between prerequisite options, and those that did open it did not understand its purpose completely. This issue is tied to the first one and could be solved in a similar manner. Some users also suggested adding an alternative symbol to those courses that could be exchanged.
3) *The course status colors:* Many users were confused by the red coloring of semi-available nodes. Most users said the color red made them think stop or error until they saw the key. That being said, everyone appeared to have a different color scheme in mind. A quick fix to this issue would be to make completed nodes grey, available nodes green, semi-available nodes yellow, and blocked nodes red. However, a better solution would be to make these colors customizable under user settings. This would not only fix this problem but also improve the usability of DegreeFlow for people who are color-blind.

A first-time user guide would be a top priority for the next version of DegreeFlow. Most users' concerns stemmed from not knowing the complete functionality of the tool. Adding something as simple as a video welcome could improve user feedback and the SUS score of DegreeFlow. A video to this effect has been created and is available online[16] for viewing.

[16]https://vimeo.com/langevinphotosandvideos/degreeflowtutorial

VI. Conclusion

Course planning is a difficult problem due to its many complex dimensions, such as changing requisites, choosing between alternative requisites, and conveying course catalog information to the user in a simple manner. DegreeFlow is a new system that addresses these issues with a focus on usability. The main challenges encountered in the implementation of DegreeFlow were:

- The extraction of UNCW's course data and transformation of the data into a normalized format due to the current data sources having inconsistent formats and lacking a normalized structure.
- The development of an algorithm that converts course catalog data into nodes and edges due to the algorithm's complexity.
- The determination of the best level of information abstraction within the visual due to the lack of a standardized metric.
- The construction of an intuitive system for selecting alternative requirements due to the lack of preexisting references.

DegreeFlow is a full-stack web application primarily written using the Vue.js and Express frameworks. DegreeFlow visualizes courses, degrees, and their requirements as an interactive flow graph. DegreeFlow was evaluated by twelve individuals, including students, academic advisors, and registrar officials, who completed fixed tasks and provided open-ended feedback. The evaluators scored DegreeFlow using the System Usability Scale (SUS) [5], and the average score of 79.75 surpassed the recommended score of 68 [5]. The feedback from these groups was overwhelmingly positive, with some common suggestions for improvement that can be corrected in future system iterations. Along with the improvements noted by users, future iterations of DegreeFlow could be improved by the addition of three features:

1) *Improving Server Course Requisite Chain Algorithms:* These algorithms are complex and reliant on network and database speeds. To prevent an unsustainable volume of requests from bogging down the system, the querying algorithm can be further improved, and its final result can be cached.
2) *Normalized Miscellaneous requirements:* As mentioned earlier, miscellaneous requirements for degrees are stored as strings but are not enforced in the system in any way. Miscellaneous requirements include minimum credit hour requirements, minimum grade requirements, courses requiring instructor approval, and more complicated scenarios such as "Take 3 additional credit hours in a CSC course above 300 that has not been counted towards your degree".
3) *Course Catalog Input:* The current system utilizes a database structure currently unused by the UNCW Registrar's Office. However, DegreeFlow only reads from the database, and all insertions or deletions must be done manually with SQL. A future system iteration could include a course catalog management page for registrar officials to create, update, and delete courses and degrees or further research a connection with UNCW's current catalog system.
4) *Automated Schedule Builder:* The current system has successfully displayed courses and their requisites as a graph for the user and other system components to interpret. Another feature a future system iteration could employ is auto-generating a schedule that prioritizes courses with more dependents and accounts for a user's preferred credit hours per semester.

This project successfully created a web application that visualizes course requisites as a graph and a database that stores these requisites in a normalized manner. DegreeFlow provides a user-friendly aid to simplify course planning for students and advisors.

REFERENCES

[1] T. Feghali, I. Zbib, and S. Hallal, "A Web-based Decision Support Tool for Academic Advising," *Educational Technology & Society*, vol. 14, pp. 82–94, Sep. 2011.

[2] "Unifying Campus Technology Solutions to Power Higher Ed — Ellucian." (), [Online]. Available: https://www.ellucian.com/.

[3] P. R. Aldrich, "The curriculum prerequisite Network: a tool for visualizing and analyzing academic curricula," Aug. 2014. DOI: 10.48550/arxiv.1408.5340. [Online]. Available: https://arxiv.org/abs/1408.5340v1.

[4] E. Hom-Nici, "Design and Construction of Visual Degree Audit Software: An Application of Visual Communication, Project Management, and Graph Theory," 2014. [Online]. Available: https://digital.library.txstate.edu/handle/10877/5053.

[5] J. Brooke, "Sus: A 'quick and dirty' usability scale," *Usability Evaluation In Industry*, pp. 207–212, Jun. 2020. DOI: 10.1201/9781498710411-35. [Online]. Available: https://www.taylorfrancis.com/chapters/edit/10.1201/9781498710411-35/sus-quick-dirty-usability-scale-john-brooke.

APPENDIX A

USER TESTING SCRIPT, SCENARIOS, AND SUPPORTED COURSES

Introduction

Today you will be testing out a web application that visualizes college course requirements. There are three phases to this session. First you will attempt to complete a series of predefined tasks on the application, then you will be given time to experiment with the application, and lastly you will be asked to complete a brief survey about the usability of the application and your experience with it. During all three phases please verbalize any questions, comments, or concerns you have about the application. Your name will not be attached to the feedback so please be as honest and open as possible. Do not worry about hurting anyone's feelings, we are here today to get feedback, both positive and negative, on this application.

Do you have any question before we begin phase one?

Phase 1

During phase one you will be given a scenario and a list of tasks to try and figure out. Please don't be afraid to make a mistake, the tasks are intentionally vague so as to not walk the user through the task. I cannot answer most questions in phase one as its purpose is to figure out where application's functionality is unclear. That being said please still ask all question you come across so I can make a note of them.

Do you have any questions before I hand you your scenario?

Phase 2

During phase 2 you will experiment with the application. I will provide you a list of courses that the application supports. Not all courses are currently supported since not all courses have been added to the database. That being said feel free try courses from off this list if you would like.

Do you have any questions before we begin phase two?

Phase 3

During phase three you will complete a brief survey reviewing the application's usability and features. Please let me know when you complete the first section of the survey, which reviews the application's usability, and I will ask you questions for the rest of the time.

Do you have any questions before we begin phase three?

Thank you!

Scenarios

Student – Computer Science

You are an undergraduate computer science major at UNCW and are preparing to sign up for courses. Your friend told you about this new web application that lets you visualize course requirements and decide to give it a shot. Another friend told you they are currently taking CSC 220 and they love it.

Attempt to answer each of the following questions using the application:

1. What is the title of CSC 220?
2. What requirements does CSC 220 have?
3. What is title **ART 260**?
4. Is CSC 220 a required course for Computer Science Majors with a Systems Concentration?
5. You have taken MAT 105, MAT 111, and CSC 131, can you take CSC 231?
6. Can PHY 201 be exchanged for another course?

Lastly complete the following tasks:

1. Delete CSC 220.
2. Mark 3 course you can take as complete.
3. Save the visualization for future editing.
4. Email a copy of the visualization to your advisor.

Student – Biology

You are an undergraduate biology major at UNCW and are preparing to sign up for courses. Your friend told you about this new web application that lets you visualize course requirements and decide to give it a shot. Another friend told you they are currently taking BIO 368 and they love it.

Attempt to answer each of the following questions using the application:

1. What is BIO 368?
2. What requirements does BIO 368 have?
3. What is **BIO 366**?
4. Is BIO 368 a required course for Biology Majors?
5. You have taken CHM 102, CHML 102, and BIO 201, can you take BIO 366?
6. Can BIO 325 be exchanged for another course?

Lastly complete the following tasks:

1. Delete BIO 368.
2. Mark 3 courses you can take as complete.
3. Save the visualization for future editing.
4. Email a copy of the visualization to your advisor.

Advisor – Computer Science

You are an advisor to an undergraduate computer science major at UNCW and are helping them answer questions prior to signing up for courses. Your colleague told you about this new web application that lets you visualize course requirements and decide to give it a shot. The student said they are interested in taking CSC 220.

Attempt to answer each of the following questions using the application:

1. What is CSC 220?
2. What requirements does CSC 220 have?
3. What is **ART 260**?
4. Is CSC 220 a required course for Computer Science Majors with a Systems Concentration?
5. You have taken MAT 105, MAT 111, and CSC 131, can you take CSC 231?
6. Can PHY 201 be exchanged for another course?

Lastly complete the following tasks:

1. Delete CSC 220.
2. Mark 3 course you can take as complete.
3. Save the visualization for future editing.
4. Email a copy of the visualization to your advisee.

Advisor – Biology

You are an undergraduate biology major at UNCW and are preparing to sign up for courses. Your friend told you about this new web application that lets you visualize course requirements and decide to give it a shot. The student said they are interested in taking BIO 368.

Attempt to answer each of the following questions using the application:

1. What is BIO 368?
2. What requirements does BIO 368 have?
3. What is **BIO 366**?
4. Is BIO 368 a required course for Biology Majors?
5. You have taken CHM 102, CHML 102, and BIO 201, can you take BIO 366?
6. Can BIO 325 be exchanged for another course?

Lastly complete the following tasks:

1. Delete BIO 368.
2. Mark 3 courses you can take as complete.
3. Save the visualization for future editing.
4. Email a copy of the visualization to your advisee.

Supported Courses

ACG 201	BIO 378	BIOL 314	CSC 133	CSC 465	ISE 410
ACGL 201	BIO 410	BIOL 319	CSC 201	CSC 466	ISE 444
ANT 430	BIO 411	BIOL 325	CSC 204	CSC 468	ISE 461
ART 101	BIO 412	BIOL 330	CSC 220	CSC 470	ISE 462
ART 111	BIO 420	BIOL 335	CSC 221	CSC 472	ISE 475
ART 220	BIO 428	BIOL 345	CSC 231	CSC 475	ISE 495
ART 260	BIO 430	BIOL 362	CSC 242	CSC 495	ISE 498
ART 282	BIO 434	BIOL 366	CSC 275	CSC 498	ISE 499
BAN 280	BIO 435	BIOL 440	CSC 302	CSC 499	MAT 105
BIO 105	BIO 438	BIOL 457	CSC 315	CYBR 201	MAT 111
BIO 140	BIO 439	BIOL 458	CSC 320	CYBR 202	MAT 112
BIO 150	BIO 440	BIOL 460	CSC 322	CYBR 245	MAT 115
BIO 160	BIO 443	BIOL 463	CSC 331	CYBR 251	MAT 150
BIO 170	BIO 445	BIOL 465	CSC 332	CYBR 325	MAT 151
BIO 180	BIO 448	BIOL 483	CSC 340	CYBR 340	MAT 152
BIO 190	BIO 452	BIOL 495	CSC 342	CYBR 343	MAT 160
BIO 201	BIO 455	CHM 101	CSC 344	CYBR 354	MAT 161
BIO 202	BIO 456	CHM 102	CSC 350	CYBR 358	MAT 162
BIO 240	BIO 457	CHM 211	CSC 351	CYBR 424	MAT 361
BIO 241	BIO 458	CHM 212	CSC 358	CYBR 431	MAT 381
BIO 246	BIO 459	CHML 101	CSC 360	CYBR 432	MIS 213
BIO 291	BIO 460	CHML 102	CSC 365	CYBR 435	MIS 310
BIO 308	BIO 462	CHML 211	CSC 368	CYBR 437	MIS 315
BIO 311	BIO 463	CHML 212	CSC 370	CYBR 440	MIS 320
BIO 313	BIO 465	CIT 110	CSC 380	CYBR 442	MIS 322
BIO 314	BIO 466	CIT 204	CSC 385	CYBR 444	MIS 323
BIO 315	BIO 470	CIT 213	CSC 402	CYBR 445	MIS 324
BIO 316	BIO 475	CIT 225	CSC 415	CYBR 446	MIS 352
BIO 317	BIO 480	CIT 301	CSC 421	CYBR 462	MIS 365
BIO 318	BIO 482	CIT 310	CSC 424	CYBR 475	MIS 366
BIO 322	BIO 483	CIT 320	CSC 430	CYBR 495	MIS 411
BIO 325	BIO 484	CIT 324	CSC 432	CYBR 498	OCN 458
BIO 330	BIO 485	CIT 352	CSC 433	CYBR 499	PHY 101
BIO 335	BIO 486	CIT 410	CSC 434	ECN 221	PHY 102
BIO 340	BIO 487	CIT 411	CSC 437	ECN 222	PHY 201
BIO 345	BIO 488	CIT 425	CSC 442	ENG 101	PHY 202
BIO 354	BIO 489	CIT 475	CSC 446	EVS 488	PHY 300
BIO 356	BIO 493	CIT 480	CSC 450	FIN 335	PSY 105
BIO 358	BIO 494	CIT 495	CSC 451	FST 220	PSY 256
BIO 360	BIO 495	CIT 498	CSC 452	ISE 102	STT 215
BIO 362	BIO 498	CIT 499	CSC 455	ISE 250	STT 315
BIO 366	BIO 499	CSC 100	CSC 457	ISE 333	
BIO 367	BIOL 140	CSC 105	CSC 458	ISE 335	
BIO 368	BIOL 240	CSC 112	CSC 461	ISE 340	
BIO 370	BIOL 241	CSC 121	CSC 462	ISE 350	
BIO 371	BIOL 246	CSC 131	CSC 464	ISE 360	

APPENDIX B

SYSTEM USABILITY SCALE DETAILED SURVEY RESULTS

TABLE IV: System Usability Scale Detailed Survey Results

Prompt	**User Ratings[i]**										**Avg.**
I think that I would like to use this system frequently.	5	4	2	5	4	4	4	4	4	5	4.1
I found the system unnecessarily complex.	1	3	2	1	3	2	2	2	1	1	1.8
I thought the system was easy to use.	4	3	3	5	5	4	4	4	5	5	4.2
I think that I would need the support of a technical person to be able to use this system.	2	2	1	1	2	2	1	4	1	1	1.7
I found the various functions in this system were well integrated.	4	3	4	5	4	4	3	5	4	5	4.1
I thought there was too much inconsistency in this system.	2	2	3	1	2	1	3	1	1	1	1.7
I would imagine that most people would learn to use this system very quickly.	5	4	4	5	5	4	3	4	4	5	4.3
I found the system very cumbersome to use.	1	3	2	1	2	1	1	1	1	1	1.4
I felt very confident using the system.	5	4	3	4	3	3	4	4	5	4	3.9
I needed to learn a lot of things before I could get going with this system.	1	3	3	1	4	1	3	3	1	1	2.1

[i] 1 = Strongly Disagree, 5 = Strongly Agree

TABLE V: System Usability Scale Detailed Computed Scores

Prompt	**SUS Scores[i]**										**Avg.**
I think that I would like to use this system frequently.	10	7.5	2.5	10	7.5	7.5	7.5	7.5	7.5	10	7.75
I found the system unnecessarily complex.	10	5	7.5	10	5	7.5	7.5	7.5	10	10	8
I thought the system was easy to use.	7.5	5	5	10	10	7.5	7.5	7.5	10	10	8
I think that I would need the support of a technical person to be able to use this system.	7.5	7.5	10	10	7.5	7.5	10	2.5	10	10	8.25
I found the various functions in this system were well integrated.	7.5	5	7.5	10	7.5	7.5	5	10	7.5	10	7.75
I thought there was too much inconsistency in this system.	7.5	7.5	5	10	7.5	10	5	10	10	10	8.25
I would imagine that most people would learn to use this system very quickly.	10	7.5	7.5	10	10	7.5	5	7.5	7.5	10	8.25
I found the system very cumbersome to use.	10	5	7.5	10	7.5	10	10	10	10	10	9
I felt very confident using the system.	10	7.5	5	7.5	5	5	7.5	7.5	10	7.5	7.25
I needed to learn a lot of things before I could get going with this system.	10	5	5	10	2.5	10	5	5	10	10	7.25
TOTAL	90	62.5	62.5	97.5	70	80	70	75	92.5	97.5	79.75

[i] Ratings are converted to scores according to John Brooke's *SUS: A quick and dirty usability scale [5]*

APPENDIX C

CONSOLIDATED QUALITATIVE USER FEEDBACK

Positives

- Visual simplicity
- The majority of people recognized Bold meant required
- Users liked the coreq dropdown
- Users liked that you can use cmd/ctrl + F to find a node
- Favorite Features:
    - Being able to move nodes
    - Checkbox functionality paired with responsive colors
    - Flow chart layout with requirements attached
    - Flow chart would replace handmade spreadsheet
    - Like arranging nodes and printing them that way

Negatives + Suggestions

- Editor:
    - Users would lose track of the remove courses section as it was pushed off the screen by the course description
        - Add bookmarks/ mini nav to the top of the editor
        - Make Remove and add two separate tabs
        - Allow the user to collapse the course description
    - Add section had an unclear workflow
        - Move add courses button to within the tile along with remove course button
    - They wish they could collapse the course description
    - Edit has the connotation of changing something, not adding something
        - Brake add and remove into separate tabs
    - Users couldn't always tell what course the description was for
        - Add course subject and number to course search results
    - i icon is not as clear as the search icon for the editor but not for the node
    - List direct requisites to the course description
- Cross-listing feature:
    - Unclear purpose at first
        - Add to the key/instructions
- Colors:
    - One user thought white meant available and yellow meant warning
    - Many users thought red meant stop/bad
    - Most users found yellow clear, but red less clear
    - Green → White → Yellow → Red
    - Grey → Green → Yellow → Red
    - Green → Yellow → Orange → Red
    - Green → Yellow → Orange → Grey
- Key:
    - Semi-Available needs more description to be clear
    - Some users didn't know what a hard coreq was
        - Add an example (BIO 101, BIOL 101)
    - When users started on the key, they weren't sure where to go to begin
    - Try to match state names to seanet terminology
    - When users started on the key, they were unsure what any of it meant since they didn't see an example of a course flow yet
    - When users didn't start on the key, they hardly ever opened the key
        - Change the title to directions/ instructions
    - Users thought the key might be interactive

- Course Flow:
    - No one knew what everything meant. Most figured out prereq, and the majority figured out the checkboxes and the colors (except for red)
    - Without instructions or looking at the key, some folks thought the dashed line meant OR, while others couldn't figure it out at all
    - A user clicked the i in hopes of deleting the node
    - Unclear zoom fit functionality, along with rearrange
        - Change the zoom fit icon
        - Add directions for both
    - Users could not always see the blue outline
        - Increase weight
        - Change color
- Alternatives:
    - Icon looks like the general share icon
    - Least intuitive feature. No one figured it out
        - Add more directions
- Nav/Menu bar:
    - Small
    - People did not look into any other tabs unless the instructions told them to
    - A user never thought to look for the key at the top of the nav as that is where settings usually are
    - Change edit to search
- Save:
    - Load being found in the save tab was not very intuitive for some users
- User:
    - Multiple people hit the logout shortcut in an attempt to close the sidebar
- Other:
    - Multiple users had trouble scrolling or didn't know there was more to a page
        - Add a scroll bar that is always visible
    - Requirements isn't a term often used
        - Use requisites
    - Multiple people would select the back arrow to reopen the sidebar
- Suggestions:
    - Add categorized printout feature (I.e., Taken, Can take, Can take*, Can't take)
    - Add symbol to nodes so users know if they can be exchanged or not
    - Add a tour, walkthrough, or video
    - Add scroll to pan in user settings
    - Add a nodes state for signed up for or plan to take
    - Hover over degree chips to highlight courses/ degree reqs
    - Center the nodes associated with an alternative change
    - As an advisor, I would like to save this to a student
    - Add advisor notes to the printout