Reimagining Graduate Academic Planning: A User-Centered Web Application for the Plan of Study Process

PeiQing Guo

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science and Application

Dwayne C. Brown, Chair

Kurt Luther

Onyeka Emebo

May 14, 2025

Blacksburg, Virginia

Keywords: Some Keywords, Subject matter, etc.

Reimagining Graduate Academic Planning: A User-Centered Web Application for the Plan of Study Process

PeiQing Guo

(ABSTRACT)

Graduate academic planning is a complex and often frustrating process, particularly when students must rely on outdated, manual processes to complete essential requirements like the Plan of Study. At Virginia Tech, Computer Science (CS) graduate students are required to submit this form to outline their coursework; however, the current process involves PDF downloads, Excel spreadsheets, and fragmented information sources. To address these inefficiencies, this study presents the design, development, and evaluation of an interactive web-based Plan of Study application tailored to the needs of graduate students in Computer Science at Virginia Tech. The system dynamically generates degree requirements, supports drag-and-drop semester planning, and performs real-time validation based on user input. It integrates a Django/PostgreSQL backend with a React-based interface, offering a user-centered solution that improves both the accuracy and usability of course planning. Findings from a needs-finding survey (n=68) informed the core features, while a post-deployment evaluation (n=20) demonstrated improvements in user confidence, planning efficiency, and overall preference over manual methods. This research highlights how dynamic and interactive digital tools can modernize academic advising and improve student experience in higher education.

Reimagining Graduate Academic Planning: A User-Centered Web Application for the Plan of Study Process

PeiQing Guo

(GENERAL AUDIENCE ABSTRACT)

Many graduate students find the process of planning their courses confusing and time-consuming. At Virginia Tech, Computer Science students are required to fill out a Plan of Study form, which often involves juggling multiple websites, checking complex requirements, and submitting Excel and PDF files. To make this easier, we built a user-friendly web application that helps students build and adjust their academic plans in one place. The tool allows users to drag and drop courses into different semesters, and checks if their plan meets the degree specific requirements as they build it. We asked 68 students what features they needed, and then built the system based on that feedback. After launching it, 20 students tested the tool, and most said it saved them time, increased their confidence, and was easier to use than the old way. This project shows how modern, interactive tools can help students plan their education more easily and accurately.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, Dr. Chris Brown, for his unwavering support, insightful guidance, and consistent encouragement throughout every stage of this research. His mentorship—and the generosity of his family—have been pivotal in shaping both this project and my academic journey. I am also sincerely thankful to my committee members. Dr. Kurt Luther, whose course first introduced me to Django and PgAdmin—skills that became foundational to the development of this system. It is no exaggeration to say that without his instruction, much of this project would not have been possible. I also deeply appreciate Dr. Onyeka Emebo for graciously joining my committee on short notice and offering thoughtful feedback. The development of the Plan of Study application was made possible by the collaboration of several talented individuals. Yasmin Tanyu contributed significantly to the layout and structure of the Home page and played a key role in the qualitative analysis of user survey responses, helping prioritize features and shape the early design direction. Huayu Liang refined the user interface on both the Dashboard and Plan pages, improving visual consistency and component design. Shreyas Pawar developed the dynamic search bar functionality, enabling flexible attribute-based filtering across the course catalog and administrative views. Additionally, Elena Hall provided valuable design insights on layout and color choices, enhancing the system's usability and visual appeal. For writing support, I am grateful to XiaoXiao Gan, Huayu Liang, Tianjia Wang, Dibyendu Bose, Minhyuk Ko, and Yoseph Alebachew, who reviewed various sections of this thesis, and to Victor Lopez for his assistance with the final draft of the literature review. I also wish to thank Steven Van Zandt for his invaluable moral and spiritual counsel during the final stretch of this journey. Finally, to my family—thank you for your constant

love, encouragement, and support, both emotional and financial, without which none of this would have been possible. To everyone who contributed, inspired, or challenged me along the way—thank you.

Contents

List of Figures

CS Computer Science MS Master of Science MEng Master of Engineering VM Virtual Machine ER Diagram Entity Relationship Diagram PoS Plan of Study

Chapter 1

Introduction

Academic planning plays a critical role in helping graduate students navigate complex degree requirements and achieve long-term academic and career goals [3, 4]. This is particularly true in high-demand fields like Computer Science, where curricular structures are dense and varied [10]. At Virginia Tech, one of the most important academic planning documents is the Plan of Study—a formal outline of a student's intended coursework that must comply with departmental and institutional guidelines [11].

Despite its significance, the current Plan of Study process remains largely manual and disjointed. Graduate students are instructed to complete a downloadable PDF and Excel spreadsheet, while separately consulting departmental websites, graduate school policies, and course listings. There is no unified platform that allows students to verify requirements, track progress, or receive real-time feedback. This fragmented workflow often leads to planning errors, increased advisor workload, and a poor student experience.

To address these limitations, this research introduces an interactive, web-based Plan of Study (PoS) system specifically designed for Computer Science graduate students at Virginia Tech. The system consolidates planning tasks into a centralized platform, providing drag-and-drop course organization, automatic requirement validation, semester scheduling, and progress tracking. Built using a Django/PostgreSQL backend and React front-end, the system supports modular degree templates, nested prerequisite structures, and dynamic visual planning features.

This study is guided by two research questions:

- **RQ1:** What features do students consider essential for an effective academic planning tool?
- **RQ2:** How do students perceive a web-based Plan of Study system compared to the current manual process?

To answer these questions, the study was conducted in two phases. First, an exploratory survey involving 68 Virginia Tech Computer Science graduate students was distributed to identify key pain points in the current process and to gather feature preferences for a future tool. These insights directly shaped the system's core functionality and user interface design.

After development, the system was evaluated through a structured user study involving 20 graduate students. Participants engaged in a cognitive walkthrough — they were given high-level academic planning tasks and encouraged to freely explore the platform's capabilities. Upon completing the tasks, they filled out a post-use survey that assessed usability, perceived effectiveness, confidence, and system stability. The study protocol and survey instrument were reviewed and approved by the Institutional Review Board (IRB).

Compared to prior academic planning tools—such as Feghali et al.'s "Online Advisor" [2] and Samaranayake et al.'s interactive planner [8]—this system emphasizes student autonomy, flexibility, and dynamic validation. Where earlier solutions often relied on static data or administrator-centric workflows, the PoS tool developed in this study supports individualized, real-time planning with embedded academic logic and visual clarity.

The contributions of this work include:

- Empirical insights into the challenges and needs of graduate students navigating academic planning workflows.

- A feature-rich planning tool co-designed with students and validated through real-world usage and survey feedback.
- Recommendations for scaling dynamic planning systems and integrating them into university advising infrastructure.

Chapter 2

Related Work

2.1 Challenges in Existing Academic Planning Systems

Academic planning—especially at the graduate level—involves balancing complex prerequisite structures, departmental constraints, and evolving program requirements. However, current tools often fall short in offering centralized, personalized, and interactive planning experiences.

A primary challenge is information fragmentation. Feghali et al.[2] highlighted how students and advisors at the American University of Beirut depended on static transcripts and forms, making real-time tracking of progress difficult. Though institutional systems like AUBsis housed academic data, they suffered from clunky interfaces and lacked meaningful integration for effective advising.

Another shortcoming is the lack of automation and interactivity. Samaranayake et al.[8] observed that many university systems still present degree requirements via static formats (e.g., PDFs, non-editable web pages), offering no intelligent planning support. These tools typically do not account for a student's completed coursework dynamically, leaving users to manually determine if they fulfill program constraints.

Scalability issues in manual advising are also prevalent. Shatnawi et al.[9] noted that with increasing student-to-advisor ratios, advisors tend to provide generic recommendations due

to time and tool limitations, overlooking individual student contexts.

Further, some platforms like that of Rajesh et al.[5] prioritize administrative workflows—tracking enrollment, attendance, or exam scores—but don't empower students with course planning functionalities that adapt to requirements dynamically.

2.2 Gaps and Solutions Proposed in Prior Research

To bridge these gaps, researchers have introduced various technological approaches:

Feghali et al.[2] developed the Online Advisor, a visual system overlaying institutional data to present course history and degree rules in one place. While helpful, it lacked real-time validation and required extensive backend integration, limiting scalability.

Samaranayake et al.[8] designed a rule-based system that could model degree requirements as logical constraints. Though powerful, it remained primarily a backend tool and wasn't deployed for direct student use.

Shatnawi et al.[9] proposed an association-rule-based advisor to recommend courses based on student history. Their system helped reduce advising load but didn't provide students with a planning interface or interactivity.

Rajesh et al.[5] built a Django-based admin platform that consolidated institutional data but focused more on administrative control than student planning empowerment.

Most recently, Roy et al.[7] proposed a deep learning-based model that predicts students' academic performance and recommends personalized study plans. While primarily targeted at undergraduate learners, their model incorporates fMRI-based data to assess cognitive status and personalize planning based on predicted performance. The novelty lies in inte-

grating predictive analytics with planning—a direction this thesis builds upon with a more interactive, student-driven approach.

2.3 Addressing Gaps in Prior Research

Unlike previous systems that emphasize administrative utility, static planning, or complex data modeling, our web-based Plan of Study (PoS) tool is explicitly:

Designed for graduate-level planning where program rules are more flexible and nuanced.

Built with drag-and-drop interactivity, nested prerequisite validation, and real-time feedback.

Developed using user-centered design and grounded in student feedback, enabling a planning experience that reflects real student needs.

Supports dynamic templates that align with Virginia Tech's graduate level program policies, with potential adaptability for other departments.

By combining the strengths of rule-based validation, user experience design, and modular system architecture, this work advances both the practical deployment of academic tools and the broader research agenda on intelligent educational technologies.

Chapter 3

Needs Finding

3.1 Initial Survey

Given the gaps identified in Chapter 2, our goal was to validate the demand for a modernized planning system and identify the features students would value most in a web-based Plan of Study tool. To achieve this, we conducted an exploratory survey targeting graduate students in the Virginia Tech Computer Science department.

The survey was reviewed and approved by the Institutional Review Board (IRB) at Virginia Tech and distributed via course mailing lists and personal outreach. In total, 68 CS undergraduate and graduate students responded. All participants were either currently enrolled or had recently completed the Plan of Study process.

The survey consisted of seven questions and took approximately 5–7 minutes to complete. It included a mix of multiple-choice and open-ended questions. These questions were designed to:

- Understand how students currently plan their academic coursework.
- Identify limitations in existing planning workflows and tools (e.g., PDF/Excel, DARS, HokieGPS).
- Elicit feature suggestions for a new digital planning platform.

For example, one open-ended prompt asked: **"If you are going to use a Plan of Study web application, what is one feature you would love to see on there?"** This question served as the primary source for identifying user needs relevant to **RQ1**.

Responses were analyzed using thematic analysis in collaboration with a fellow researcher. This qualitative approach involved coding and categorizing common feature requests and recurring user frustrations. Initial codes were clustered into themes such as automated requirement validation, visual course planning, integrated course information, and support for alternative planning paths. These themes were then used to inform and prioritize the functional requirements of the Plan of Study system.

3.2 RQ1 Results

Students reported that their current planning methods often involved manually tracking courses in Excel, cross-referencing DARS reports, and consulting multiple disconnected university websites. Common complaints included fragmented information, uncertainty about requirement fulfillment, and lack of interactivity. These findings provided a strong foundation for designing a more centralized and dynamic solution that directly addressed the planning pain points voiced by students.

From the responses we got from this initial survey, even though it was focused more on the HokieGPS website, the frustrations student displayed on various of features on existing application like HokieGPS for those who has used HokieGPS indicates that the frustrations we had against the current process is not an assumption made by us as researchers, but a frustration shared by students in Computer Science. This includes the responses from the survey question "If you have used HokieGPS, what did you dislike about it?" like **"I did not like the fact that it wasn't aesthetically pleasing."**, **"A LOT of quality of life**

features are done poorly/ missing", and **"It was a little difficult to follow and find everything since there was so much information."**.

To address RQ1 — **What characteristics do students consider essential for an effective academic planning tool?** — we analyzed open-ended responses from the 68 survey participants. Several recurring themes emerged from the analysis.

- **Automated Requirement Validation (24/68):** A significant number of students emphasized the need for a system that can automatically verify whether selected courses fulfill degree requirements. This feature was seen as essential for minimizing errors and reducing the need for manual auditing.

- **Multiple Planning Scenarios (12/68):** A few students highlighted the importance of being able to experiment with alternate course paths, such as simulating different academic tracks or exploring "what-if" scenarios for degree completion.

- **Interactive Visual Planning (7/68):** Students expressed a clear preference for an intuitive drag-and-drop interface that allows them to organize courses by semester and visualize prerequisites or course dependencies within a structured layout.

- **In-Context Course Information (5/68):** Respondents valued having immediate access to essential course details—such as descriptions, prerequisites, and term availability—directly within the planning interface, without needing to consult external resources.

- **Credit and Summary Views (4/68):** Several students requested a concise overview of credit-related information, including totals for completed, in-progress, and remaining credits—preferably presented in a dashboard-style format for easy tracking.

Several feature requests, such as GPA tracking for individual courses (3/68), professor-specific ratings (4/68), difficulty ratings (7/68) and test reviews (2/68), are not feasible to implement within the scope of this project. Although these suggestions reflect genuine student interests, they face significant practical limitations. For example, instructor assignments are often not confirmed until shortly before the semester begins, making it impossible to provide reliable data for long-term planning that may span to 3 or 4 years. Given that this tool is designed to support multisemester academic planning, incorporating such dynamic, instructor-specific information is not technically viable. That is why some of the features we are implementing based on the survey results were requested by only a small number of students. For instance, as few as 4 out of 68, since more complex or personalized requests could not be supported within the system's intended design.

Feedback made it clear that students are seeking not just a course checklist, but a comprehensive, dynamic planning environment. These insights guided the design and functionality of our proposed web-based system, ensuring that it is in line with actual student needs and expectations of the students.

Chapter 4

Web Application Development

4.1 Overview

This web application integrates the requirements tracking capabilities of Feghali and Samaranayake mentioned in Chapter 2 along with some of the needs mentioned in the previous survey mentioned in Section 3.2 with a modern mainly student-facing interface powered by React, Django, and PostgreSQL. Unlike prior work that emphasized back-end data models or administrator workflows, our tool prioritizes student experience, offering:

- Drag-and-drop visual planning
- Real-time requirement validation
- Centralized course and requirement visibility
- Iterative feedback integration from real users

This system bridges the usability gap identified in prior literature by combining structured academic rules with an interface designed to empower student decision making.

4.2 Back-End

The back-end of the Plan of Study (PoS) system is developed using the Django web framework, a high-level Python framework that encourages rapid development and clean design[1]. This section details the architectural structure of the back-end and the relational database schema that supports key planning and validation functionalities. The system follows a modular multiapp architecture, where each Django app represents a discrete domain area. This modularization aligns with Django's 'app-per-function' paradigm and improves the separation of concerns and maintainability. The backend exposes its functionality through a RESTful API, implemented using the Django REST Framework (DRF), allowing the front-end client to dynamically interact with back-end services. Key Django applications include the following:

User Model

The user model is extended via the Details object in `users.models`, which augments Django's default User model through a OneToOneField association. The Details model includes a role attribute (admin or regular), implemented as a character field, and a list of associated plan identifiers, enabling each user to manage multiple Plan instances. The role attribute, in combination with the user's authentication status, is used to enforce access control throughout the backend. These attributes are checked during request handling to determine authorization for various actions, such as course creation, modification, and plan management, ensuring secure and role-aware interactions between front-end and back-end components.

Course Model

The Course model constitutes the main data structure within the system's academic representation layer. It encapsulates all the key properties required to describe and operate on course entities within the Plan of Study (PoS) environment. Each instance of the Course model includes the following fields: major, class number, title, description, credits, credit type, elective field, seasons, and link. Additional boolean and structured fields such as editable credits, prereq groups, and coreq groups, capture mutable credit settings and hierarchical prerequisite/corequisite structures, respectively.

The prerequisite and corequisite attributes are stored as nested lists in a JSONField, allowing for expressive logical groupings (e.g., conjunctive and disjunctive requirements). This data model supports flexible requirement validation while maintaining normalized relational integrity through foreign key constraints on supporting taxonomies such as ElectiveField, CreditType, and Major.

This model serves as the foundational schema for academic content in the application. All users have read access to course data, which is surfaced uniformly across views and planning tools. However, write access is restricted to users with administrative roles, enforced through a custom permission system integrated into the *ClassViewSet* and validated at the view layer using the *check_admin_permission()* method.

A supplementary field, course id list, was appended to the model to support batch-level operations within plan manipulation workflows. This field facilitates intermediate mappings between semester blocks and plan validation routines and is further discussed in the plan model section.

Requirement Model

The Requirement model is designed to provide a flexible, scalable mechanism for representing degree-specific academic requirements across programs such as the MS and PhD in Computer Science. During schema design, we observed that the majority of curriculum constraints can be expressed in a normalized format: Courses with a specific attribute must satisfy a quantitative comparison against a required credit threshold, as illustrated in Figure 4.1.

This insight enabled the definition of requirement objects using a compact yet expressive set of fields. Each Requirement instance includes an attribute (e.g., elective field, major), an attribute_value, and a comparison operator drawn from a predefined set (e.g., $\leq$, $\geq$, $\equiv$). The comparison is applied to a requirement_size, representing the minimum or exact number of credits to be fulfilled by courses matching the given attribute. In addition, each requirement is bound to a specific credit_type and optionally scoped to a major. The include boolean field further supports inclusion/exclusion logic, allowing requirements to enforce both positive and negative filters over course selection.

All fields are fully configurable, which allows administrators to update or add new degree requirements without modifying application logic. This avoids the brittleness associated with hardcoded requirement rules and ensures forward compatibility across evolving academic policies.

Access to modify Requirement objects is restricted to users with administrative privileges, enforced at the view layer using custom permission checks. Read access, however, is universally permitted to support validation workflows and user feedback in the planning interface.

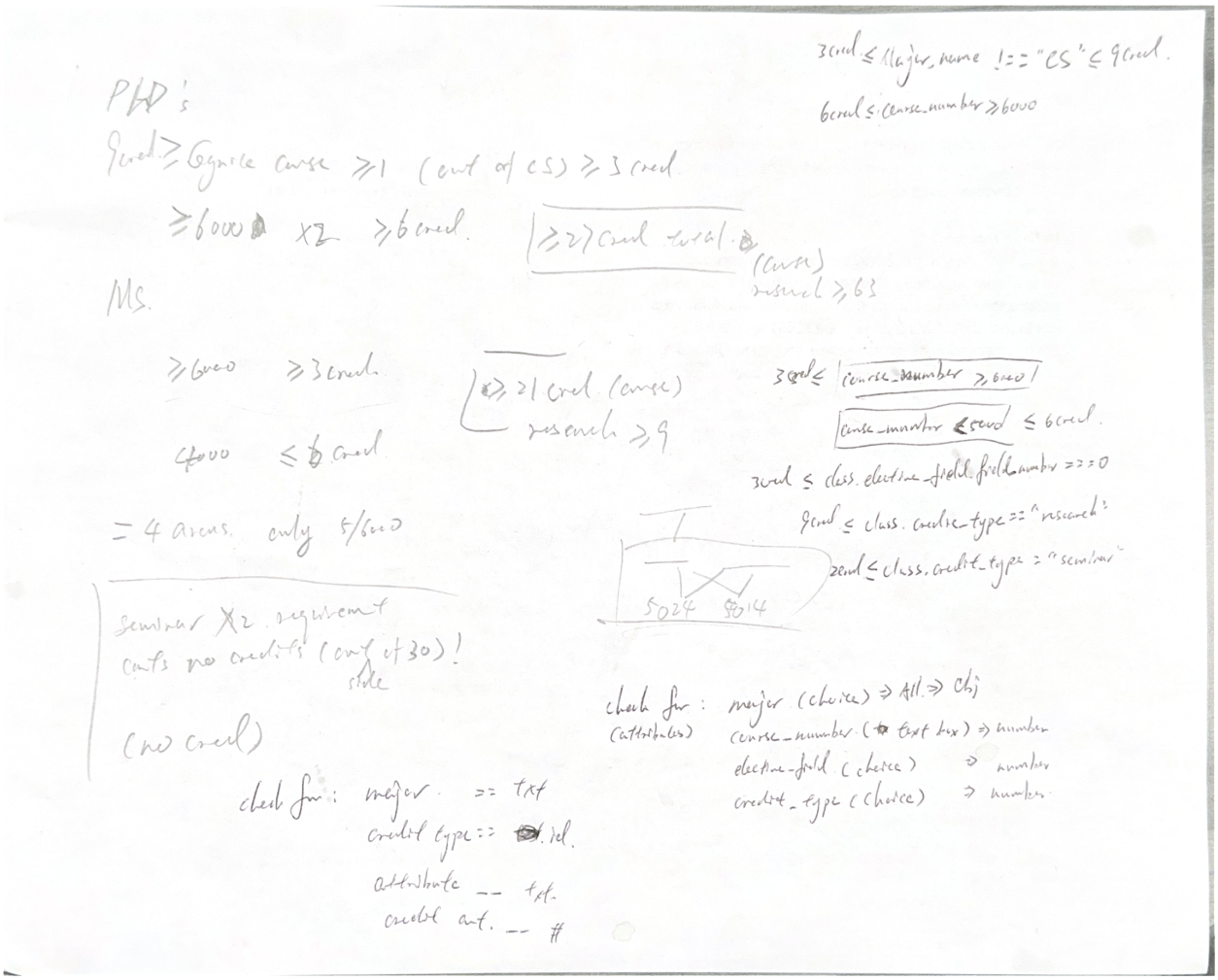


Figure 4.1: Initial design sketch illustrating the process of translating graduate program requirements (for both MS and PhD in Computer Science) into quantifiable comparison-based conditions. Each academic rule (e.g., 'at least 3 credits from 6000-level courses'[11]) is broken down into components such as attribute, comparison operator, and target value to enable programmatic validation.

Template Model

The Template model serves as a flexible blueprint for defining degree requirements across different academic programs. It was designed with extensibility in mind, allowing the system to accommodate not only graduate-level Computer Science students at Virginia Tech but also undergraduate programs, other departments, or institutions with distinct academic policies.

Each Template instance encapsulates a collection of associated Requirement objects, which together define the core set of constraints necessary for program completion. In addition to these atomic requirements, the Template model also supports broader curriculum structures, such as breadth requirements, through dedicated fields. For example, the Master of Science in Computer Science program at Virginia Tech requires students to complete at least 3 credits in each of 4 distinct elective fields (termed "Areas")[11]. These breadth constraints are represented using fields such as min_elective_fields and min_each_elective.

Templates also account for program-specific criteria such as the minimum credit load required for full-time status, represented by the min_credits field. These values differ between degree levels and institutions, and their inclusion as configurable fields ensures adaptability without the need for code-level changes.

Due to their high-level academic impact, Template objects are immutable by regular users. Write access is restricted to administrators, enforced through back-end permission logic. However, read access is universally permitted, allowing students to browse applicable degree structures and validate their academic plans accordingly.

Plan Model

The Plan model serves as the central organizing structure for the application's academic planning functionality. Each Plan instance aggregates the user-defined selection of courses, organized by semester, and links together all relevant requirement and template data. This model is fundamental to the system's core functionality, enabling users to build, manage, and validate personalized academic trajectories in accordance with institutional policies.

During the design phase, it became evident that a single course could appear multiple times within a given plan—across different semesters—and that each instance might require distinct configurations. A common use case arises in the context of research credit requirements for MS students. These students often enroll in a research-oriented course offered every semester, but must accumulate a specific number of total research credits. Importantly, the credit value for such a course may vary by term and is editable by the student.

This requirement made it impractical to directly associate Course instances from the centralized database with a plan in a one-to-one fashion. To address this, the implementation introduces a duplication mechanism whereby a shallow copy of the original Course object is created when it is added to a plan. Each duplicated course includes an additional attribute, course_id, which is a randomly generated five-digit identifier unique to that instance. This enables multiple instances of the same course to coexist within a plan while retaining independent credit values and editing capabilities.

Access to each Plan object is restricted to its creator. Permissions are enforced such that only the user who generated the plan has read and write access to it. This ensures user-specific data privacy and prevents unauthorized access or modification of individual academic plans.

Semester Model

The Semester model represents a single academic term within the context of a specific Plan. Each Semester instance is linked to its parent Plan via a foreign key relationship and encapsulates the set of courses assigned to that term. In addition to storing references to these course instances, the model tracks metadata including the academic year, the corresponding term (e.g., Fall, Spring), and configurable credit constraints—specifically, the minimum required credits and the maximum allowable credits for the semester.

This structure enables users to distribute their academic workload across multiple terms while adhering to program-specific credit policies and constraints. Because each Semester is intrinsically tied to its parent Plan, its access permissions are inherited from the Plan model. As such, only the creator of the associated Plan has read and write access to the Semester, ensuring consistent enforcement of user-level access control throughout the planning hierarchy.

Supporting Models

The supporting models refer to a group of foundational data structures that serve auxiliary roles within the academic planning system. These models are designed to enhance modularity, reusability, and maintainability by centralizing shared attributes that are referenced across higher-level entities such as courses, templates, and semesters.

The major model stores metadata on academic programs, including the full name and a standardized abbreviation. The credit_type model associates credit classifications (e.g., research, core, elective) with their respective majors, providing structure for requirement filtering and validation. The elective_field model encodes elective categories by storing a type name, associated major, and a numerical identifier that facilitates sorting and mapping.

Finally, the season model defines academic terms (e.g., Fall, Spring) using a simple label-based structure.

Each of these models is implemented independently to accommodate potential variations across majors and institutions. For example, the attributes required to define an elective field can differ between undergraduate and graduate programs, justifying its separation from other course-related models.

From a permissions perspective, write access to the major, credit_type, and elective_field models is restricted to administrative users. The Season model is treated as immutable within the system: even administrators are not permitted to modify season entries, ensuring consistent scheduling logic. All supporting models are globally readable, since they form the base taxonomy required to render courses, requirements, and planning interfaces. Therefore, no permission checks are enforced for read operations on these models. A visual overview of the full back-end schema is shown in Figure 4.2

4.3 Database Schema

The database schema underlying the Plan of Study (PoS) system is designed to balance relational integrity, extensibility, and ease of querying. Built using Django's ORM on top of a relational database (e.g., SQLite or PostgreSQL during development), the schema adheres to normalized design principles and facilitates dynamic academic planning, validation, and role-based access control.

At its core, the schema reflects a hierarchical data model, with users at the top level and academic planning elements cascading beneath. Each user (extended via the Details model) can create multiple Plan instances, each of which contains multiple Semesters, which in turn

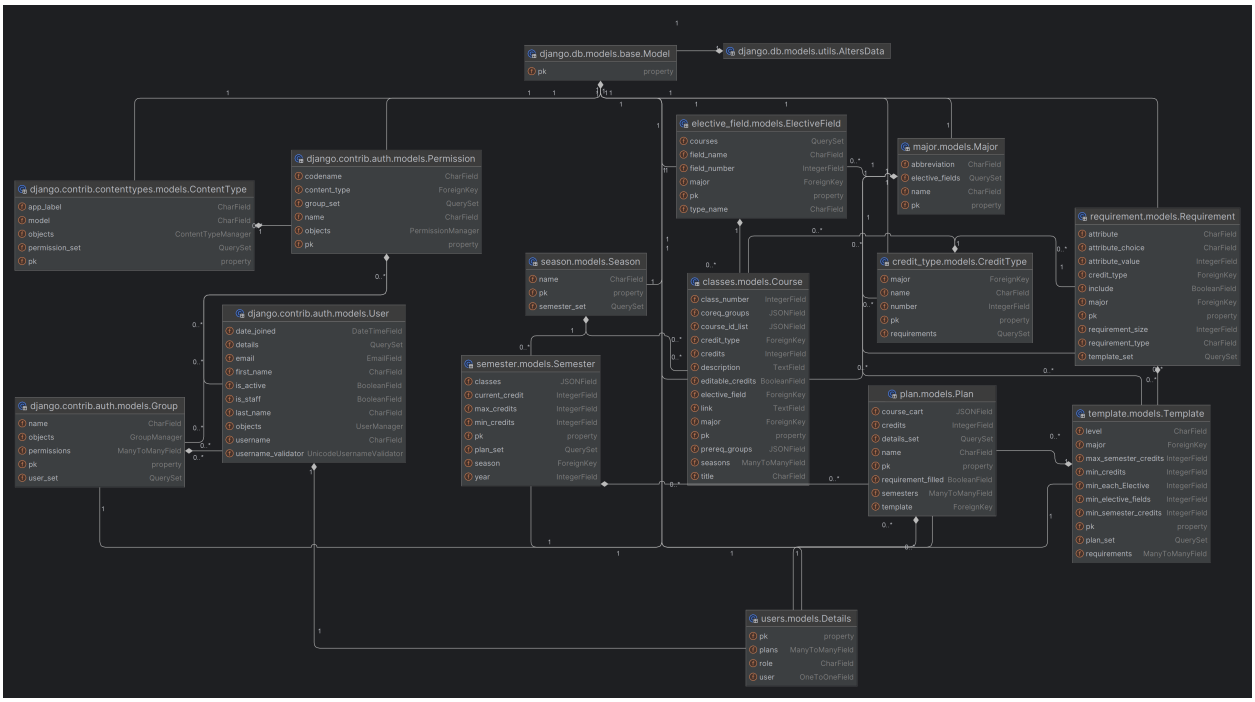


Figure 4.2: Class Diagram for the backend

hold Course copies. This nesting allows for granular customization, such as variable credits or duplicated course use across terms, without mutating the global course catalog.

Key model interrelations include:

- One-to-many relationships between User → Plan, Plan → Semester, and Semester → Course.
- Many-to-many relationships between Course and Season, supporting flexible scheduling.
- One-to-one links such as User → Details, ensuring single-profile extensibility.
- Foreign keys for taxonomy models (e.g., Major, CreditType, ElectiveField), enabling sorting and filtering without hardcoding definitions.

The Requirement and Template models collectively implement a rules engine for degree

validation. Requirements are expressed through a comparison-based logic structure (e.g., "at least 3 credits of elective field X"), while templates bundle multiple such rules to define full program outlines. This layered design allows academic administrators to adapt templates as policies evolve, without affecting existing user plans.

All models are scoped by fine-grained permission policies:

- Data creation and mutation are restricted to authenticated users, with additional privileges (e.g., modifying templates or global models) limited to administrators.
- Read access is broadly permitted for foundational taxonomy models and templates to facilitate user planning.

Overall, the schema is engineered to support the following:

- Flexibility: via JSON fields, user-specific course instances, and dynamic rule linking
- Scalability: by separating concerns across domain-specific apps
- Maintainability: through clear relational boundaries and DRF-based data exposure

This foundation ensures that the system can grow beyond its current domain to support future features such as GPA tracking, cross-department integration, and audit reporting. A visual overview of the full database schema is shown in Figure 4.3, which presented using enhanced entity relation diagram. It illustrates the different entities, relationships between them, the types of attributes and the constraints such as foreign keys.

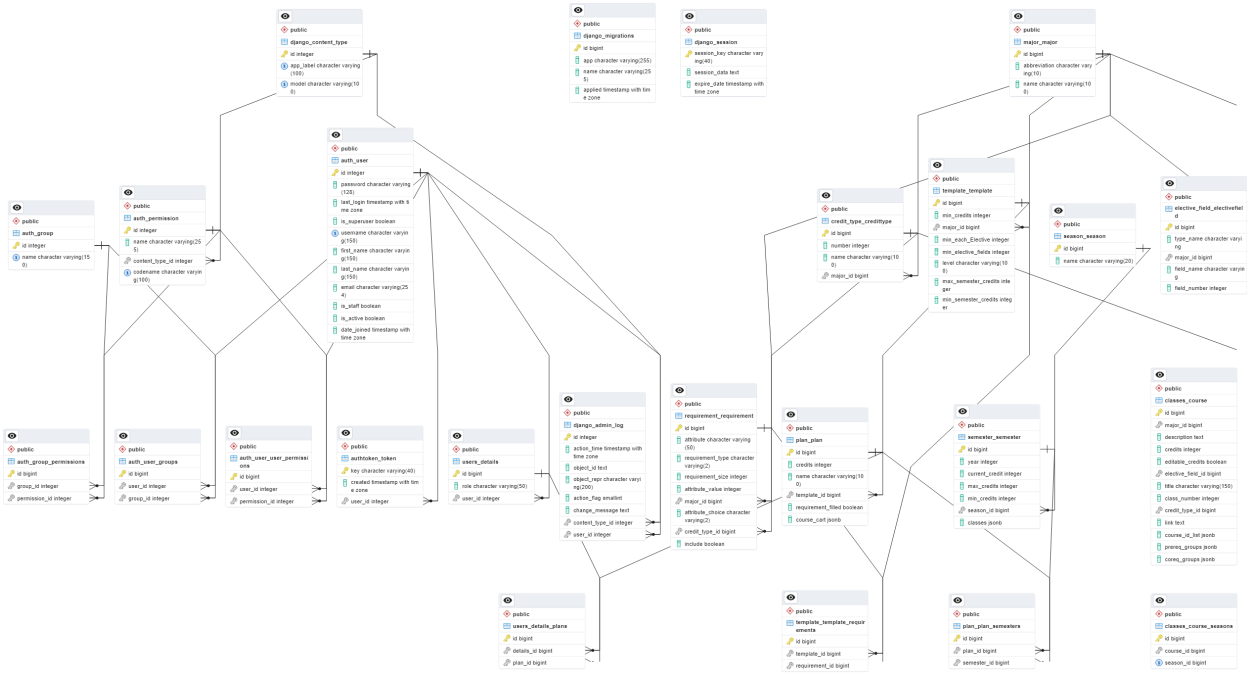


Figure 4.3: ER Diagram for the database

4.4 Front-End

The front-end of the Plan of Study (PoS) system is implemented using React, a widely adopted JavaScript library for building dynamic user interfaces. React's component-based architecture enables the application to be modular, reusable, and highly maintainable—qualities that are critical for managing complex user interactions and data-driven updates in a planning tool.

The front-end serves as the primary interface through which users interact with the system. It communicates with the Django REST API to fetch, update, and display data related to courses, semesters, academic plans, templates, and requirements. Each major data entity is encapsulated within a dedicated component or set of components that manage rendering logic, local state, and interaction behavior.

Routing within the application is handled using React Router, allowing for seamless transi-

tions between views such as the course catalog, semester planning interface, template viewer, and user-specific dashboards. The application maintains responsiveness across screen sizes and devices using modern layout techniques such as CSS Grid and Flexbox, along with conditional rendering patterns in React.

State management is primarily performed using built-in React hooks such as useState, useEffect, and useContext, allowing for localized as well as shared state between components. In more complex workflows—such as course validation, plan editing, or semester updates—state is derived from back-end API responses and updated reactively based on user input.

User permissions and roles are enforced client-side based on authentication tokens and metadata returned by the backend. The interface dynamically adjusts to provide or restrict access to administrative features (e.g., course creation, requirement editing) based on the authenticated user's role.

While authentication and permission checks are enforced in the back-end, an additional layer of validation is also implemented on the front-end. The React application incorporates role- and permission-based access controls that dynamically tailor the user interface to each user's privileges. For instance, administrative functionalities such as adding or editing courses are only rendered for users with the appropriate role, based on their authentication token and user metadata. This dual-layered security model reduces the likelihood of unauthorized operations reaching the backend, resulting in fewer permission-related errors during API calls. It also improves the overall user experience by proactively preventing disallowed actions from being initiated on the client side.

Overall, the React-based front-end provides a responsive, interactive, and role-aware user experience, tightly integrated with the underlying RESTful backend to support real-time academic planning and validation.

4.4.1 Courses Page

The Courses page is one of the two primary interfaces with which users interact when building their Plan of Study. It is designed to provide both a searchable catalog of all available courses and a streamlined mechanism for selecting and adding courses to personalized plans.

At the top of the page is a search bar that allows users to filter and query courses based on any attribute within the course JSON object. This includes fields such as title, class number, elective field, credit type, and associated major. The results are rendered in a scrollable table view, with each row displaying key metadata about a course (e.g., title, class number, credit type, and elective field).(Figure 4.4)

Given the number of attributes stored in each Course object, displaying all relevant information within a single row would overwhelm the user interface. To address this, a dedicated information button is placed within the "Action" column of each course row. When clicked, it expands a secondary row beneath the course to reveal additional details such as the course description, external resource link, and its prerequisite/corequisite relationships. The display for all the course details was implemented to fulfill the fourth feature need mentioned in Section 3.2 - In Context Course Information. This expanded row also includes an 'Add to Plan' button that allows users to assign the selected course to one of their existing plans via a dropdown menu.(Figure 4.4)

For users with administrative privileges, the page provides three additional actions: adding, editing, and deleting courses. An admin-only "Add Course" button is located in the bottom-right corner of the page (Figure 4.4). Clicking this opens a modal dialog containing a form where administrators can input the necessary attributes to create a new Course object. Notably, the elective field dropdown is dynamically populated based on the selected major, reflecting the hierarchical relationship between majors and elective fields in the underlying

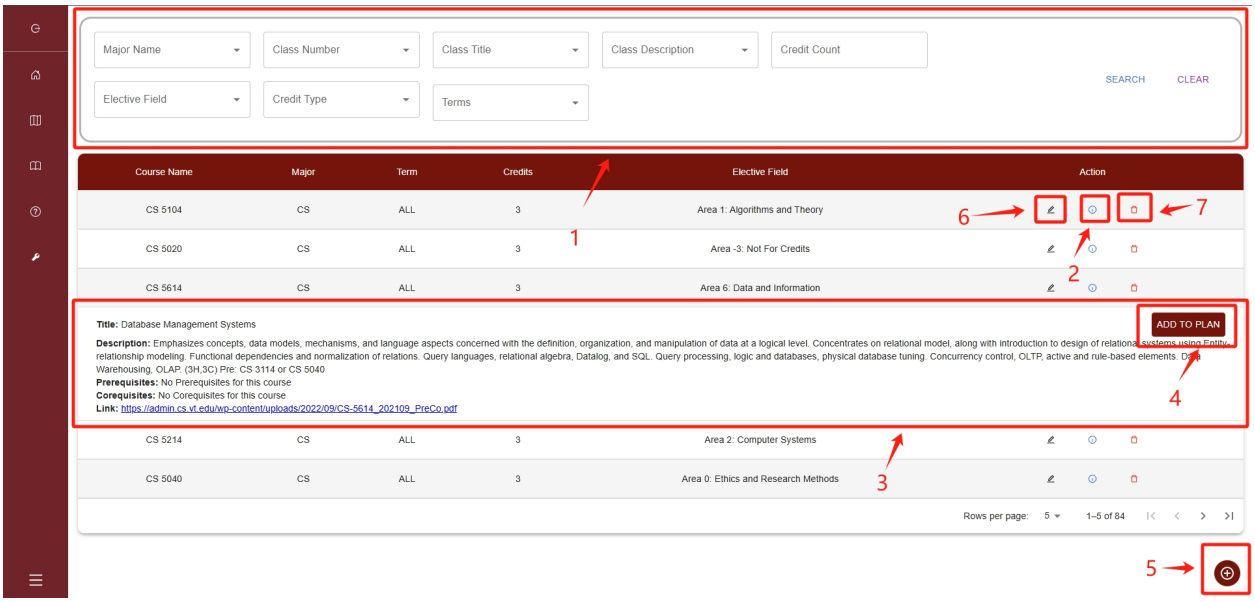


Figure 4.4: Full interface layout of the Courses page. (1) Search bar enabling attribute-based filtering across the course catalog; (2) Information toggle button for expanding course details; (3) Expanded detail view displaying additional metadata including description, prerequisites, and links; (4) Add to Plan button, allowing users to assign the course to a selected academic plan; (5) Admin-only control to open the Add Course dialog for course creation; (6) Edit Course button that opens a pre-populated form for modifying existing course data; (7) Delete Course button triggering a confirmation dialog to prevent accidental deletion.

schema. (Figure 4.5)

The edit functionality mirrors the add process. Admin users have access to edit buttons located alongside the information and delete buttons in the Action column (Figure 4.4). When the edit button is clicked, the same dialog used for course creation is opened, but this time it is pre-populated with the selected course's data to assist in updating (Figure 4.6).

To prevent accidental deletions due to the proximity of action buttons, the delete functionality is gated by a confirmation dialog. When an admin user selects the delete icon, a modal prompts the user to confirm the deletion. The course is only removed from the database after explicit confirmation.

This modular and role-sensitive interface design supports both regular users and admin-

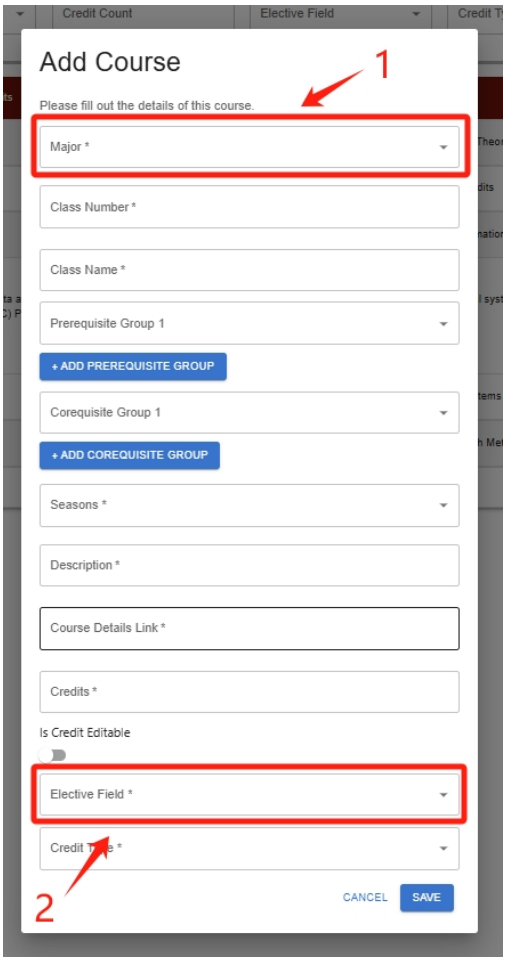


Figure 4.5: Interface layout of the Add Course dialog. (1) Dropdown menu for selecting the course's associated major; (2) Elective field selector dynamically populated based on the chosen major, ensuring consistency with the underlying program structure and elective classification.

istrative workflows while maintaining clarity, preventing data loss through misclicks, and ensuring the integrity of the course catalog.

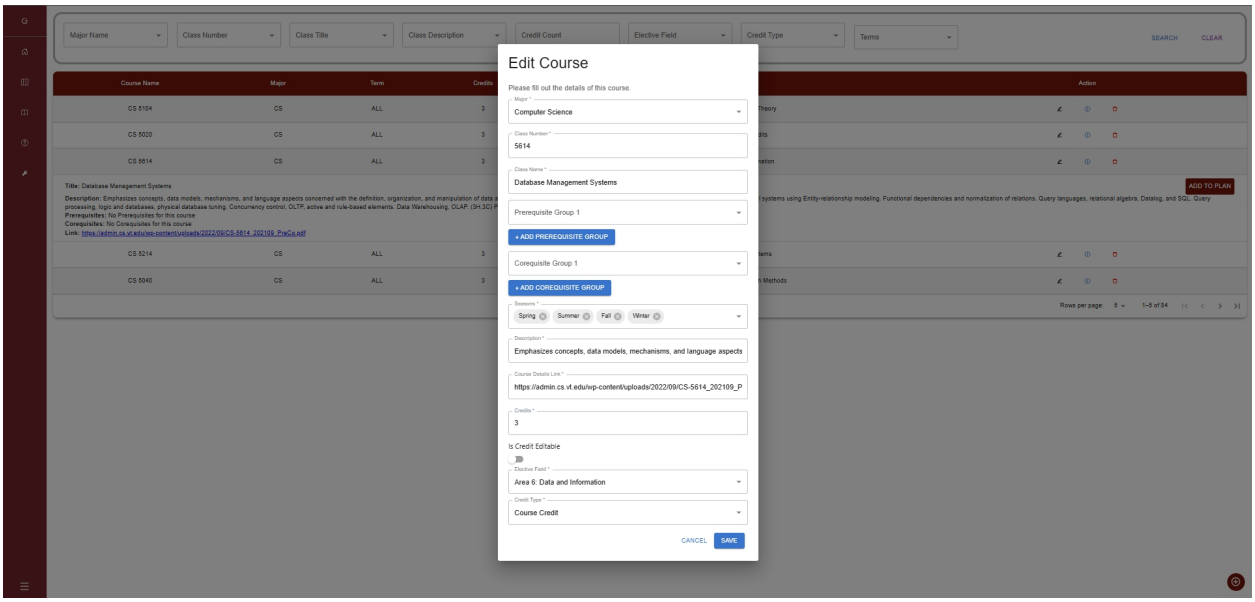


Figure 4.6: Interface layout of the Edit Course Dialog, which is a prepopulated version of the Add Course dialog.

4.4.2 Plan and Dashboard Pages

Dashboard Page

The Dashboard page serves as the central hub for managing user-created academic plans. The implementation for this dashboard page that allows for users to experiment with multiple different plans in multiple different levels was aimed to fulfill the fifth feature that was mentioned in the Section 3.2 - Multiple Planning Scenarios. It provides a clean, minimal interface where users can view, create, and delete plans. When creating a new plan, users are required to select both a major and a degree level (e.g., MS or PhD), which enables the system to associate the plan with the correct set of academic requirements. These parameters are later used for validating degree completion as courses are added.

The design of this page prioritizes usability and clarity, with a deliberate focus on eliminating visual clutter. Its goal is to provide users with a straightforward, distraction-free environment

where the available actions—such as starting a new plan or removing an existing one—are clearly presented and easily accessible. (Figure 4.7) This feature of allowing students to create as many plans in any degree as they want to fulfills the "Scenario Planning" feature request from RQ1 results.

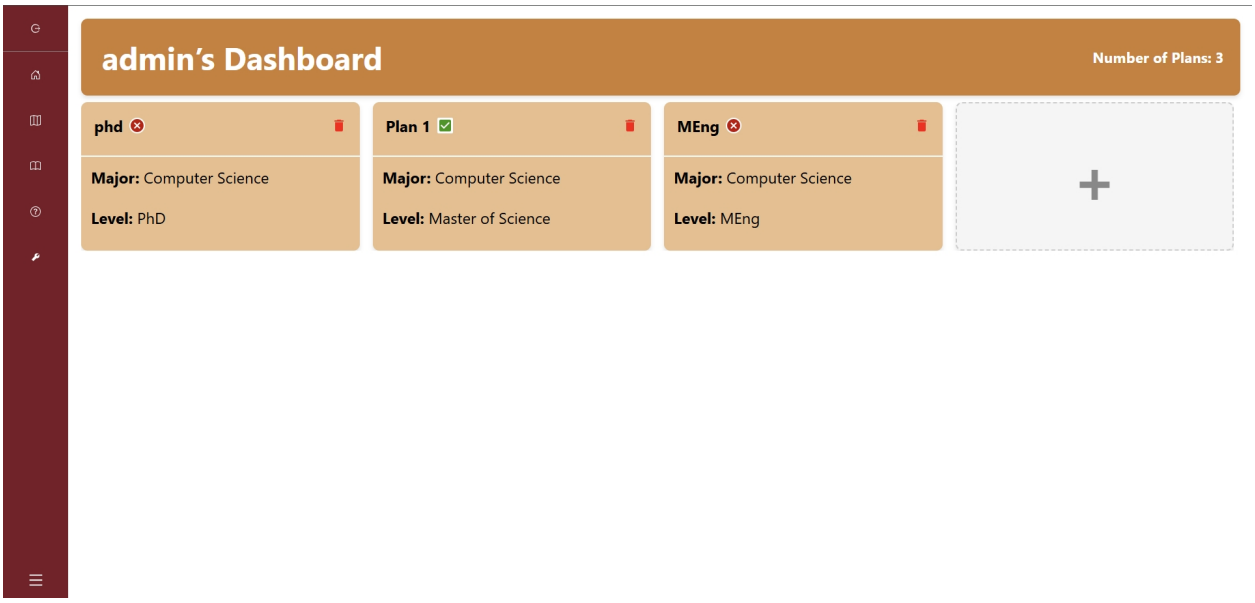


Figure 4.7: The full interface for user Dashboard Page

Plan Page

The Plan page is one of the core components of the application, where users actively build and refine their academic trajectories. On the right side of the interface is the course cart, which temporarily stores courses that have been added from the Courses page. These courses that are in the course cart are not yet associated with any specific semester and therefore are not included in requirement validation until they are formally placed into a semester.

Users can assign courses to semesters via a drag-and-drop interface. Courses dragged from the course cart into a semester become part of the plan's validation workflow. The interface also supports intra-semester transfers, allowing users to move courses between semesters as

needed(Figure 4.8). This drag-and-drop functionality implemented here is to fulfill the third feature need mentioned in Section 3.2 - Interactive Visual Planning.

Before finalizing any drag-and-drop action, the system performs prerequisite and corequisite validation. If a course's dependency constraints are not met in the destination semester (e.g., missing a prerequisite that should be in a prior term), a warning notification is triggered, and the action is reverted. While this feature has been fully implemented, it is currently inactive in the live system due to the policy at Virginia Tech allowing override-based course additions. As such, prerequisite and corequisite data for graduate-level Computer Science courses have been temporarily removed from the database.

To guide new users, the semester list always ends with a "visual prompt": a grey rectangle resembling a semester container with a plus sign in the center, signaling the option to create a new semester. Edit and delete actions for semesters are intentionally de-emphasized and grouped within a dropdown menu to reduce visual noise and prioritize core functionality(Figure 4.9a).

The requirement check interface is designed to be non-intrusive yet informative. Instead of persistently displaying all individual requirement rules, the system shows a summary indicator at the top-right corner of the page—either "All Requirements Fulfilled" or "Not Fulfilled Yet." Clicking on this status indicator reveals a dropdown with the full list of requirements and their fulfillment statuses, allowing users to examine validation results only when needed. This display not only includes the requirements that are stored in the requirement list in the template object associated with the plan, but also the requirements that are stored directly in the template object itself, which was mentioned in Chapter 4.2(Figure 4.9b). This requirement checking functionality is implemented to fulfill both the first and the second feature need mentioned in Section 3.2 - Automated Requirement Validation, and Credit Summary Views.

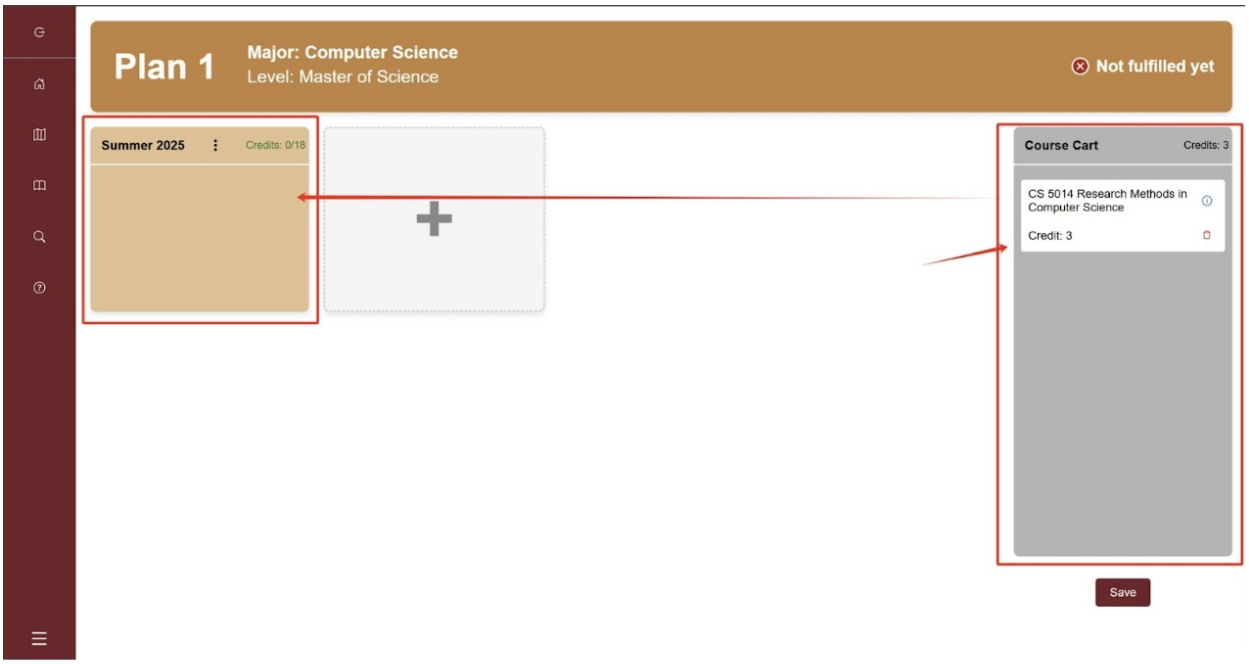


Figure 4.8: The drag and drop process in Plan Page

4.4.3 Admin-Specific Pages

The application includes four admin-only pages: the Elective Field page, Template page, Major page, and Credit Type page. These pages are accessible via routes under the /admin URL path and are displayed exclusively to users with administrative privileges. Navigation to these pages is provided through the "Admin" tab in the sidebar, which is displayed conditionally only after an authenticated admin user logs in.

To enforce secure access, each of these admin pages is protected by a client-side and server-side permission check. If an unauthenticated user attempts to access one of these routes directly through a URL, the system triggers a modal dialog that prompts the user to log in. If a logged-in user without administrative privileges tries to access the same route, a different dialog appears, indicating that administrative access is required. In both cases, a yellow "OK" button is displayed in the bottom right corner of the dialog, which redirects

the user to the home page that provides general information about the application.

This dual-layer permission model ensures that only authorized users can access or modify core configuration data. All four admin pages serve as control panels to maintain the foundational structures of the system. For example, if a specific major or degree level updates its curriculum requirements, an administrator can make the necessary changes directly within the Template page through an intuitive interface.

In terms of layout and interaction design, these pages follow a consistent structure based on the Course page interface. Each page features a search bar to filter items, an "Add" button in the bottom right corner to create new entries, and edit/delete controls located within an "Actions" column of a tabular view (see Figure 4.4). However, unlike the Course page, these admin pages do not include an information dropdown. This is because the data models for Elective Fields, Templates, Majors, and Credit Types are relatively simple, and all necessary attributes can be displayed directly within a single row.

4.4.4 User Help and FAQ Resource

To help users to navigate and use the Plan of Study system, a dedicated FAQ page is hosted separately on a public Google site. This page addresses frequently asked questions and provides step-by-step guidance for common user actions, including

- How do I register?
- How do I create a semester to add my classes in?
- How do I create a plan?
- How do I add a class into the semesters?

- All I see is “Not fulfilled yet”—how do I check which requirements my plan needs to fulfill?

This resource complements the interface design by offering real-time support for new users and reduces the learning curve when interacting with the planning tool. The FAQ page is accessible through a link on the home screen of the application as well as the sidebar and is publicly viewable without authentication.

4.4.5 Summary of Features in Relations to RQ1

To summarize, the features requested by users of RQ1 are all fulfilled in the UI design of the front-end of this web application. The features “Automated Requirement Validation”, “Interactive Visual Planning”, and “Credit and Summary View” were fulfilled on the Plan page. The “Multiple Planning Scenarios” feature was fulfilled in the implementation of the user dashboard page. Lastly, the “In-Context Course Information” feature was fulfilled in both the Course page and the Plan page so that the detailed information for each individual course is easily accessible to users in multiple pages.

4.5 Deployment

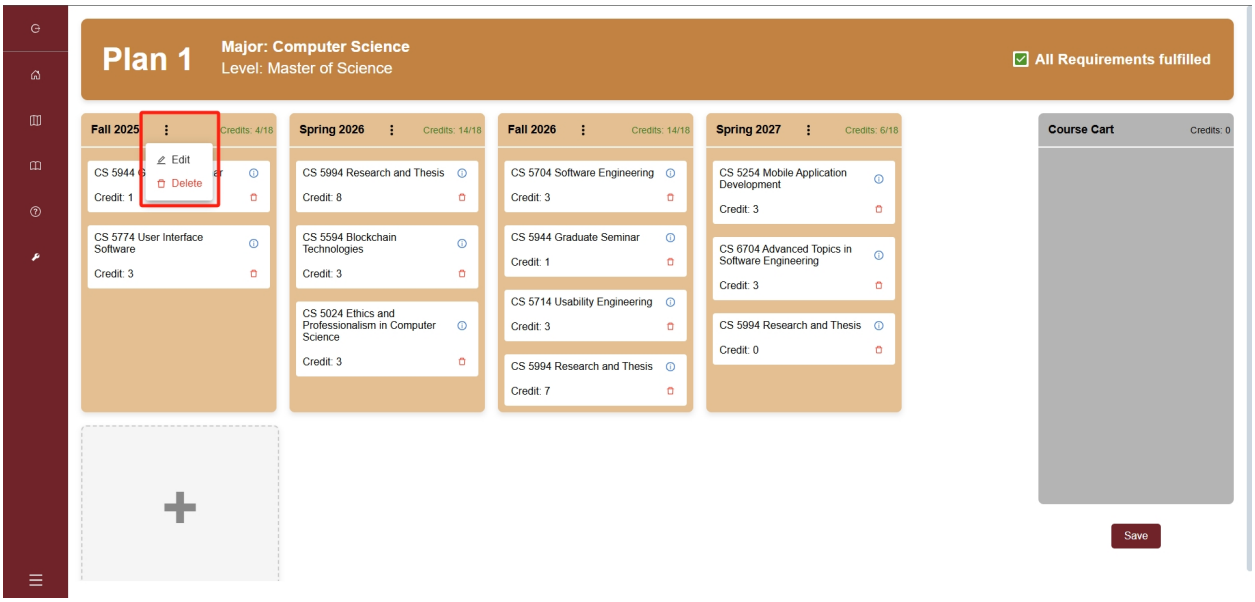
After completion of the local development of the system, the application was deployed to a publicly accessible server hosted on a university-provided virtual machine (VM). Upon approval, the VM was provisioned and configured with all necessary software dependencies to support the full stack: the front-end (React), back-end (Django), and PostgreSQL database.

To deploy the front-end, the React application was first compiled using the npm run build command, which generates an optimized production-ready set of static files. Then these build

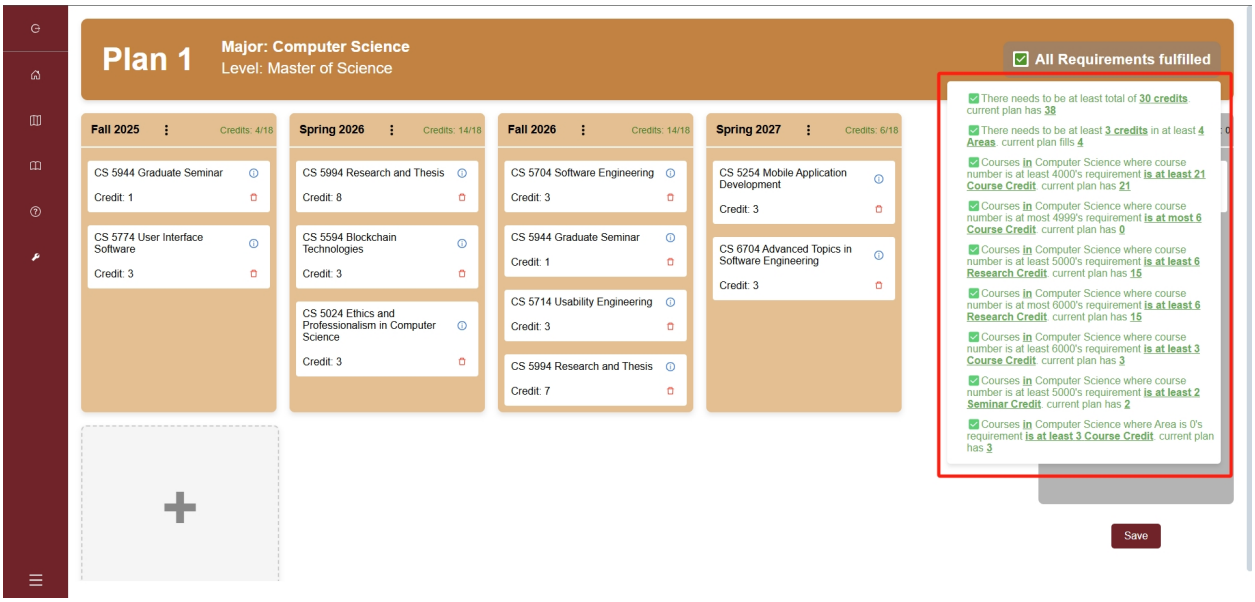
artifacts were transferred to the default web directory of the server located at /var/www/html/. This directory is used by the Apache HTTP Server to serve web content.

A critical configuration step involved the creation of an .htaccess file within the same directory. The purpose of this file was to handle client-side routing properly. By default, React applications using the browser history API fail to resolve subpage routes when the user refreshes the page directly or attempts to navigate via a bookmarked subpath. Without server-side route handling, this results in a "404 Not Found" error.

To mitigate this, the .htaccess file was configured to redirect all unmatched requests to the application's index.html file, allowing React Router to handle the path resolution client-side. This approach ensures a seamless and intuitive navigation experience across all pages of the application, regardless of whether they are accessed through direct URLs or internal routing. Currently this web application is posted at the URL `plan-of-study.cs.vt.edu`.



(a) Interface layout for the plain Plan page. [red block area] The display for semester objects edit and delete functionality.



(b) Interface layout for the Plan page after the requirement check is clicked. [red block area] The display of all requirements.

Figure 4.9: Comparison of the Plan Page before and after revealing the requirement check details.

Chapter 5

User Evaluation and System Impact

5.1 Post-Deployment Study Overview

To evaluate the effectiveness of the developed Plan of Study (PoS) system, we conducted a post-deployment user study involving graduate students in the Virginia Tech Computer Science department. Using **purposive sampling**, an email invitation was distributed department-wide to all enrolled MEng, MS, and Ph.D. students. A total of 20 participants voluntarily responded to the survey after interacting with the tool. Among the respondents, 5 were MEng students, 7 were MS students, and 8 were Ph.D. students.

Before completing the survey, participants were asked to conduct a **cognitive walkthrough** [6] of the system. Specifically, they were instructed to use the application to create or revise a plan of study that fulfilled their degree requirements, test features such as drag-and-drop planning, requirement validation, and course filtering, and explore semester management and progress tracking functionalities.

The study and its instruments were reviewed and approved by the Institutional Review Board (IRB) at Virginia Tech. All participants were 18 years or older, and their participation was voluntary, with data collected anonymously and reported in aggregate.

The post-deployment survey consisted of both Likert-scale and open-ended questions, totaling 20 items. The Likert-scale questions measured students' perception of usability, time

savings, confidence, technical stability, and satisfaction with the tool. Open-ended items captured qualitative feedback on the most helpful features, areas needing improvement, and any frustrations encountered.

Key results from the Likert-scale responses are summarized below:

- **Opinion of current manual method:** 72.2% of respondents rated the current manual process 3 or below out of 5, with a mean rating of 2.72/.
- **Preference over manual method:** 82.5% of respondents rated the tool 4 or 5 out of 5 in comparison to the current PDF/Excel-based method, with a mean rating of 4.15.
- **Perceived time savings:** 95% agreed the tool saved them time during planning, with a mean rating of 3.95.
- **Increased planning confidence:** Participants' self-reported confidence increased from a pre-use mean of 3.75 to a post-use mean of 4.30.
- **Ease of navigation:** 90% gave a 4 or 5 rating to the statement "The tool was easy to navigate" (mean = 3.80).
- **System stability:** The statement "I encountered no major technical issues" received a mean rating of 4.25.
- **Recommendation rate:** The mean agreement score for "I would recommend this tool to other students" was 4.05.

These quantitative results indicate the system was well received from both functional and usability perspectives. Out of all of the data from the post deployment survey, it is worth noting the 3.75 as the mean of confidence score for students before using this tool. This

number echoes the findings from the needs finding survey. Since our target audience and user base is mainly graduate level students, we expected the confidence scores to have a mean that is greater than 3. Considering how complicated the current manual is, we expected the mean to be less than 4. the 3.75 is a mean that we expected and this further showed that students holds a level of frustrations toward the current manual process for building their plan of studies.

5.2 RQ2 Results

This section addresses **Research Question 2 (RQ2)**: **"How do students perceive a web-based Plan of Study system compared to the current manual process?"**

The post-deployment survey data clearly indicates that students overwhelmingly prefer the web-based planning tool. Participants cited it as easier to use, more efficient, and confidence-building compared to the department's existing workflow involving downloadable PDFs and Excel files.

Three specific metrics reinforce this conclusion:

- **Preference:** 82.5% of respondents rated the web tool favorably (4 or 5 out of 5) when asked how it compares to the manual process.
- **Time savings:** 95% agreed or strongly agreed that the system helped them complete their plan more quickly.
- **Confidence gain:** Students were asked, "How confident are you in your ability to complete a valid Plan of Study?" before and after using the system. The average self-reported confidence score increased from 3.75 (pre-use) to 4.30 (post-use) on a 5-point Likert scale.

- **Ease of use:** 90% of respondents found the system easy to navigate (Q6), and 85% agreed that it was easy to understand what steps to take (Q7).
- **Recommendation:** 95% said they would recommend the tool to other graduate students in the department (Q12).

When asked which features were most helpful (Q13, Q15), students frequently mentioned the automatic requirement validation, real-time feedback, and drag-and-drop course assignment as standout elements. Several participants noted that the ability to visualize and edit their plan interactively gave them greater clarity and control over their academic progress.

These findings confirm that the system met its intended goal: to enhance the academic planning experience. Moreover, the user feedback affirms that the features prioritized during the needs-finding phase—such as automated requirement validation, visual planning, and real-time feedback—were not only implemented but also effective.

The alignment between early user needs and final system performance provides strong evidence of the system's success in addressing the planning challenges faced by graduate students. These results also offer a foundation for broader deployment and continued refinement of academic planning tools across departments and institutions.

Chapter 6

Discussion

6.1 Overview

The journey from scattered PDFs and error-prone spreadsheets to an integrated planning tool reveals a fundamental truth: what students need most is not technological complexity, but clarity. This chapter synthesizes insights from the needs-finding surveys and post-deployment feedback, which shows how simple, targeted features, such as automated requirement checking, drag-and-drop planning, and centralized course information, can effectively address long-standing frustrations with the traditional process. In addition to measurable gains, including a 15% increase in student confidence, this chapter also examines the remaining challenges students reported and offers guidance on how future improvements, both to this tool and to similar systems, can further enhance the academic planning experience.

6.2 Summary of Key Findings

The study was guided by two research questions:

- **RQ1:** What features do students consider essential for an effective academic planning tool?
- **RQ2:** How do students perceive a web-based Plan of Study system compared to the

current manual process?

For **RQ1**, analysis of 68 student responses identified five essential features for an effective planning tool:

- Automated requirement validation (35% of respondents)
- Clear credit tracking (6%)
- Interactive visual planning (10%)
- Integrated course information (7%)
- Scenario planning (12%)

The **RQ2** evaluation with 20 users demonstrated strong preference for our system over manual methods:

- 82.5% preferred the web tool
- 95% reported time savings
- Confidence scores increased from 3.75 to 4.30 (5-point scale)
- 90% found the interface intuitive
- 95% would recommend to peers

These results confirm that the implemented features directly addressed students' most pressing needs while significantly improving the planning experience. The alignment between identified requirements and post-deployment satisfaction validates our user-centered design approach.

6.3 Interpretation of Results

The results suggest that the most valued features—automated validation, visual planning, and centralized information—directly respond to the frustrations caused by the manual process. The increase in post-use confidence scores (from 3.75 to 4.30) indicates that real-time feedback mechanisms provided reassurance and clarity during plan creation.

The drag-and-drop interface enabled students to experiment with different course arrangements, facilitating a better understanding of semester balance and prerequisite fulfillment. Though prerequisite validation was disabled in the live version due to its functionality clash with the force add process and how frequent force add has been used for graduate students, students still reported increased satisfaction with how the system represented course relationships and availability.

These findings support the conclusion that usability improvements, such as intuitive interfaces and real-time checks, can significantly impact how students engage with academic planning tools.

To take this thesis a step further, we took a deeper look at the post deployment survey data to find connections between the increase in the users' confidence level on creating their own PoS and their positive thoughts on specific features. From the multiple-choice responses to the question on the survey **"Which features did you find most helpful?"**, we found that 11 out of the 20 respondents selected Visual layout of semesters, 12 out of 20 respondents selected Drag-and-drop planning, and 15 out of 20 selected Automatic requirement validation. The high score in the first two, visual layout of semesters, and drag-and-drop planning indicated that having features that help users to visualize the task they are trying to complete helps with the boost of confidence. The high score in the last one, Automatic requirement validation, indicates that having a feature that the application does

a specific task where otherwise it would require users to take extra time to do can lead to a boost in quality of user experience as well.

6.4 Contributions and Recommendations

6.4.1 Academic and Practical Contributions

This work advances the field of academic planning systems by introducing a student-centered, web-based tool specifically designed for graduate-level programs. Unlike previous systems, such as Feghali et al.'s "Online Advisor" [2] and Samaranayake et al.'s planner [8], which often relied on static data or administrator-focused workflows, this system takes a different approach. It combines interactive planning, nested prerequisite logic, modular requirement templates, and real-time validation. These are features that are rarely integrated together in graduate-level planning tools.

From a research perspective, the project demonstrates how student input can be translated into actionable design decisions through thematic analysis and iterative development. It shows that involving users in the designing process can lead to meaningful improvements in usability and user confidence.

On a practical level, the system addresses widespread frustrations in academic planning by unifying scattered tools into a single interface. Through features like automated validation, drag-and-drop scheduling, and centralized course visibility, the platform greatly improved the quality and efficiency for graduate students in CS at VT. Survey results confirm its effectiveness: 95% of participants reported time savings, and 82.5% preferred it over the traditional PDF/Excel method.

6.4.2 Recommendations for Future Implementation

Based on these findings, the following forward-looking guidelines are recommended for academic institutions and developers aiming to modernize their planning systems:

- **Design with students at the center.** Many students expressed frustration with existing planning processes, which are often not designed with their needs in mind. Current tools frequently prioritize administrative workflows over usability, leaving students to piece together information from multiple disconnected sources. In contrast, early survey responses in this study made it clear that students want systems that help them monitor progress and organize their degree plans in a clear and accessible way. One student wrote: **"Checklist of things already completed/pending."** Another emphasized the need for high-level visibility: **"I would love to see the requirements of my degree briefly there like a quick overview..."** This kind of clarity was a common request, reinforcing the importance of designing academic tools around how students actually plan and make decisions.

- **Prioritize features students find most helpful.** After using the system, students highlighted specific features that improved their planning experience. One participant wrote: **"The courses filter section also helps me skim through information more efficiently."** Another noted: **"I liked that now it's easy to visualize and do the plan of study."** These quotes show the value of providing intuitive, searchable interfaces and a visual planning structure.

- **Address frustrations and unmet expectations.** Despite positive feedback, users also described several challenges. One participant noted: **"There are some bugs in the course filtering feature: choosing between majors doesn't update; I think the course number and course title can be together, when select some**

filter options, the results didn't come up correctly." Another said: **"maybe add a column that says the exact name of the course instead of me clicking the 'action' everytime to see it."** A third user suggested: **"I wish it was directly implemented with canvas."** These comments highlight the importance of continuous improvement, better integration, and fine-tuning the interface for smoother use.

These recommendations aim to guide future development and institutional adoption of academic planning systems that are scalable, maintainable, and grounded in user experience. The success of this study underscores the importance of co-designing educational tools with those they are intended to serve.

Chapter 7

Limitations and Future Works

7.1 Limitations

Despite the promising outcomes, the study has several limitations:

- **Needs-Finding:** The initial exploratory survey included responses from 68 graduate students and provided valuable insights for system design. However, the sample may not fully capture the diversity of academic planning challenges across different departments or institutions, limiting the generalizability of the findings. Also, when doing the needs finding for this project, we did not research and check if other universities currently has something that was designed for students. This could potentially give us a fuller view on the different features we could have implemented that complements the results from the initial needs finding survey.
- **Post-Deployment Sample Size:** The post-deployment evaluation involved only 20 participants, which restricts the ability to draw broad conclusions from the results.
- **Prerequisite Validation Disabled:** While prerequisite and corequisite validation was implemented, it was disabled during user testing due to the department's frequent reliance on force-add procedures, which bypass standard prerequisite enforcement. This limitation prevented full assessment of the system's validation features in real-world use.

- **Department-Specific Scope:** The system was evaluated exclusively within the context of Virginia Tech's Computer Science graduate program. Its applicability and effectiveness in other academic disciplines or institutional settings remain untested.
- **Lack of Advisor Feedback:** The study did not include formal input from faculty advisors or Graduate Program Directors, whose perspectives could provide important insights into administrative workflows and system adoption from a non-student viewpoint.

These limitations highlight how academic planning tools must navigate not just technical challenges, but institutional realities. Our next steps will focus on addressing these gaps to make the system more adaptable across different academic contexts.

7.2 Future Work

The success and limitations of the current system point to several opportunities for expanding its capabilities and enhancing its impact. Future development can focus not only on adding functionality but also on improving usability, integration with institutional systems, and adaptability to broader academic contexts.

- **Advisor Interfaces:** While the current platform focuses on student use, a logical next step is the development of advisor-facing tools. A dedicated dashboard for advisors and Graduate Program Directors could facilitate the review and approval of student plans, enable real-time commenting, and allow for collaborative plan building. This would streamline communication between students and advisors, reduce email-based feedback cycles, and embed advising into the same interface students use to create their plans.

- **Expanded Integration with Institutional Systems:** Future iterations of the system could benefit greatly from tighter integration with university-wide infrastructure such as Banner, DARS, or Canvas. This would allow for real-time syncing of course enrollments, automatic verification of prerequisites and completed credits, and seamless audit generation. Such integration could reduce administrative redundancy, improve data accuracy, and extend the platform's utility from planning into registration and graduation auditing.

- **Unified Course and Planning Interface:** Currently, the system separates course browsing (Course Page) and academic planning (Plan Page) into two distinct interfaces. While this promotes organization, it introduces friction by requiring users to navigate back and forth. A future enhancement could involve combining these two functionalities into a single, unified page. For example, users could browse available courses and directly drag them into semester containers without changing views. This would improve task flow, reduce cognitive load, and provide a more intuitive, seamless user experience.

- **Cross-Departmental and Undergraduate Support:** The system was designed specifically for graduate Computer Science programs at Virginia Tech, but many of its core architectural decisions (e.g., modular requirement models, flexible templates, JSON-based prerequisite encoding) lend themselves to broader application. Expanding the platform to support undergraduate students or other departments would involve creating a centralized administrative backend where academic programs can define their own templates, requirement types, and elective structures. This would enable multi-program deployment while preserving program-specific logic and policies.

- **Mobile and Responsive Design Considerations:** Although not a focus of the initial prototype, future development could improve support for mobile devices and

tablets. Enabling students to review or modify their academic plans from any device could increase accessibility and encourage ongoing engagement with the tool throughout their academic journey.

- **Integration of Predictive and Personalized Recommendations:** Inspired by work such as Roy et al.[7], future versions of the system could incorporate basic predictive analytics—such as course difficulty indicators, average GPA, or student performance trends—to recommend optimal course selections. These features could help students make more informed decisions and better balance workload across semesters.

- **UI Improvements:** Several participants noted that the user interface of the plan page, while functional, appeared overly simplistic and lacked visual polish. Feedback suggested improving the color scheme and adding visual enhancements such as animations during drag-and-drop interactions and a loading indicator. Instead of redirecting users to a separate FAQ page, a built-in tutorial or guided walkthrough could better support onboarding. Additionally, enhancing the search bar to support more advanced filtering options could improve usability and reduce unnecessary backend calls.

- **Functionality Improvements:** The current course page retrieves all course data from the backend upon each load or filter reset, which negatively impacts performance. To improve speed and responsiveness, the system should implement lazy loading or pagination—initially fetching only the subset of courses needed to populate the visible view and loading more on demand.

Incorporating these enhancements would extend the system's value to a broader range of stakeholders and help embed it more deeply into institutional workflows. These improvements would not only support individual academic planning but also contribute to broader goals of advising efficiency, student retention, and curriculum optimization.

Chapter 8

Conclusions and Summary

8.1 Conclusions

This thesis presented the design, development, and evaluation of a web-based Plan of Study (PoS) application designed for graduate students in the Virginia Tech Computer Science department. The work was driven by two core research questions: (1) What features do students consider essential for an effective academic planning tool? and (2) How do students perceive a web-based system compared to the current manual process?

The project followed a user-centered design approach, beginning with a needs-finding survey of 68 students to identify pain points and desired features. Key needs included automated requirement validation, credit and summary views, in-context course information, and multiple planning scenarios. These insights shaped the architecture, interface design, and feature prioritization of the final PoS system.

Following development, a post-deployment study involving 20 graduate students evaluated the system's usability, efficiency, and user satisfaction. The results demonstrated a clear preference for the new tool over the traditional PDF and spreadsheet-based process. Students reported increased confidence in building their plans, time savings, and an overall improvement in experience. These outcomes affirm the system's effectiveness in addressing both functional challenges and user experience gaps in the academic planning process.

Importantly, the system moves beyond static advising models by offering real-time feedback, modular academic templates, and flexible plan customization. Unlike earlier tools, it empowers students to take ownership of their academic trajectories through an interactive, dynamic planning interface.

This study highlights the value of applying student-informed design to a high-stakes academic workflow. By grounding development in real user needs and validating outcomes through empirical feedback, the PoS system not only provides a working solution to a local institutional challenge but also serves as a scalable model for improving academic planning tools in higher education more broadly.

8.2 Summary of Contributions

This research makes the following key contributions:

- **User-Informed Feature Set:** Identified and prioritized essential academic planning features based on survey data from real graduate students.
- **End-to-End System Implementation:** Developed a full-stack application using Django, PostgreSQL, and React that integrates visual planning with real-time requirement validation.
- **Post-Deployment Evaluation:** Conducted a user study confirming that the system improves usability, efficiency, and user confidence over existing methods.
- **Scalable Academic Architecture:** Designed a flexible and extensible requirement model capable of supporting diverse academic policies and future expansion.

References

[1] Django Software Foundation. *Django Documentation*. Django Software Foundation, Lawrence, Kansas, 2023. https://docs.djangoproject.com/.

[2] Tony Feghali, Imad Zbib, and Sophia Hallal. A web-based decision support tool for academic advising. *Journal of Educational Technology & Society*, 14(1):82–94, 2011.

[3] Doreen H. Kinkel and Scott E. Henke. Impact of undergraduate research on academic performance, educational planning, and career development. *Journal of Natural Resources and Life Sciences Education*, 35(1):194–201, 2006. doi: https://doi.org/10.2134/jnrlse2006.0194. URL https://acsess.onlinelibrary.wiley.com/doi/abs/10.2134/jnrlse2006.0194.

[4] Ingrid Moses. Planning for quality in graduate studies. In *Quality in postgraduate education*, pages 4–13. Routledge, 2017.

[5] B. Rajesh, G. Prasanthi, Y. V. S. Varsha, P. Ajay Kumar, S. B. N. D. Poojitha, and P. Tapasvi. Innovative django powered next-gen advanced academic planning management system for streamlining college operations. *International Journal of Novel Research and Development*, 9(3):g23–g29, 2024. URL https://www.ijnrd.org/papers/IJNRD2403604.pdf. ISSN: 2456-4184.

[6] John Rieman, Marita Franzke, and David Redmiles. Usability evaluation with the cognitive walkthrough. In *Conference companion on Human factors in computing systems*, pages 387–388, 1995.

[7] Ayon Roy, Deepayan Bhowmik, Sandip Dey, Arpan Sengupta, Sayantan Bera, and

Suvankar Dutta. A deep learning approach to predict academic result and recommend study plan for improving student's academic performance. In Raja Sengupta, Nikhil R. Dey, Siddhartha Bhattacharyya, and Aboul Ella Hassanien, editors, *Ubiquitous Intelligent Systems*, volume 283 of *Lecture Notes in Networks and Systems*, pages 435–448. Springer, Singapore, 2021. doi: 10.1007/978-981-16-3676-9_34. URL https://doi.org/10.1007/978-981-16-3676-9_34.

[8] Sobitha Samaranayake, Athula DA Gunawardena, and Robert R Meyer. An interactive decision support system for college degree planning. *Athens Journal of Education*, 10 (1):101–115, 2023.

[9] Raed Shatnawi, Qutaibah Althebyan, Baraq Ghalib, and Mohammed Al-Maolegi. Building a smart academic advising system using association rule mining. *arXiv preprint arXiv:1407.1807*, 2014.

[10] Shorelight. Should you get a master's degree in computer science? https://shorelight.com/student-stories/should-you-get-a-masters-degree-in-computer-science/, 2024. Accessed: 2024-04-28.

[11] Virginia Tech Department of Computer Science. Master's degree with thesis in computer science, 2024. URL https://cs.vt.edu/Graduate/Degrees/MSThesis.html. Accessed: 2025-04-24.

Appendices

Plan of Study Site Survey

- You are invited to participate in our survey about the plan of study. This survey seeks to collect information to determine whether or not you would like to take part in an interview concerning how we can optimize the process of planning out classes for your degree. It will take approximately 5-7 minutes to complete. You must be 18 years or older to participate, and your participation in this study is completely voluntary. Please answer the following questions honestly. However, if you feel uncomfortable answering any questions, you can withdraw from the survey at any point. Your survey responses will be strictly confidential and data from this research will only be reported in aggregate. Your responses will be coded and presented anonymously. If you have questions at any time about the survey or procedures, please contact Dr. Chris Brown (dcbrown@vt.edu) or PeiQing Guo (guo1340@vt.edu). Thank you very much for your time and support, we greatly appreciate it!

* Indicates required question

1. Email *

2. Name *

3. How did you plan out your classes for your degree? *

Mark only one oval.

- ◯ The PDF my academic advisor sent
- ◯ HokieGPS
- ◯ Scroll through VT Time Table
- ◯ Other: _____

4. If you are going to use a plan of study website, what is one feature you would love to see on there? *

5. Have you ever used HokieGPS? *

Mark only one oval.

- Yes
- No

Hokie GPS unofficial evaluation

6. If you have used Hokie GPS, what did you like about it? *

7. What did you dislike about it? *

Plan of Study Post-Survey

This study aims to evaluate a plan of study website to support academic planning for Computer Science graduate students at Virginia Tech. We are interested in understanding how users perceive the website, suggestions for improvement, and its impacts on academic planning. Your participation will help us design motivations for the integration of future web-based planning tools into academic workflows.

You must be 18 years or older to participate and your participation in this survey is voluntary. If at any point, you feel uncomfortable answering any of the questions, you may withdraw from the survey. Your responses will be kept condential and data from this survey will be reported in aggregate. If you have any question about the survey, you may contact Peiqing Guo (guo1340@vt.edu) or Dr. Chris Brown (dcbrown@vt.edu). Thank you for your time. We appreciate your help and support.

* Indicates required question

1. Email: *

2. What is the degree you are pursuing? *

3. How many years are you in the program? *

4. How do you feel about the current plan of study that the department is using (the pdf and excel) *

Mark only one oval.

	1	2	3	4	5	
I dol	◯	◯	◯	◯	◯	I like it

5. What is the biggest challenge you have been facing with the current process of building your plan of study using the process that the department is using (the pdf and excel) *

6. I found the Plan of Study tool easy to navigate. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

7. It was easy to understand which courses fulfilled which requirements. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

8. The layout of the site helped me build my plan more efficiently. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

9. I encountered no major technical issues while using the tool. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

10. I prefer using this system over the existing PDF/Excel method. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

11. I feel more confident about my Plan of Study after using this tool. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

12. I would recommend this tool to other students. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

13. Which features did you find most helpful? *

Check all that apply.

- [ ] Visual layout of semesters
- [ ] Automatic requirement validation
- [ ] Course filtering
- [ ] Drag-and-drop planning
- [ ] Other: _____

14. Which features do you think need improvement? *

Check all that apply.

- [ ] Visual layout of semesters
- [ ] Automatic requirement validation
- [ ] Course filtering
- [ ] Drag-and-drop planning
- [ ] Other: _____

15. What did you like most about using the Plan of Study Assistant? *

_____

_____

_____

_____

_____

16. What challenges or frustrations did you encounter while using it? *

_____

_____

_____

_____

_____

17. Are there any features you wish the tool had? *

18. Compared to the original process, this tool would saved me a lot of time planning out my plan of study. *

Mark only one oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

19. Before using this tool, how confident were you in building a valid Plan of Study by yourself? *

Mark only one oval.

	1	2	3	4	5	
Not	◯	◯	◯	◯	◯	Very confident

20. After using the tool, how confident are you in building a valid Plan of Study by yourself? *

Mark only one oval.

	1	2	3	4	5	
Not	◯	◯	◯	◯	◯	Very confident