جامعة الإمارات العربية المتحدة
United Arab Emirates University

MASTER THESIS NO. 2025: XX

College of Information Technology

Department of Information Systems and Security

ADVANCING ACADEMIC ADVISING WITH KNOWLEDGE GRAPHS: INTEGRATING MACHINE LEARNING AND LLMS FOR PERSONALIZED COURSE PLANNING

Sara Alshamsi

June 2025

United Arab Emirates University
College of Information Technology
Department of Science in Information Technology Management

ADVANCING ACADEMIC ADVISING WITH KNOWLEDGE GRAPHS: INTEGRATING MACHINE LEARNING AND LLMS FOR PERSONALIZED COURSE PLANNING

Sara Alshamsi

This thesis is submitted in partial fulfillment of the requirements for the degree of Master of Science in Information Technology Management

June 2025

United Arab Emirates University Master Thesis
2025: XX

Cover : Knowledge Graph
(Photo: By Sara Alshamsi)

Print: University Print Service, UAEU 2025

Declaration of Original Work

I, Sara Alshamsi, the undersigned, a graduate student at the United Arab Emirates University (UAEU), and the author of this thesis entitled *"Advancing Academic Advising with Knowledge Graphs: Integrating Machine Learning and LLMs for Personalized Course Planning"*, hereby, solemnly declare that this is the original research work done by me under the supervision of Prof. Nazar Zaki, in the College of Information Technology at UAEU. This work has not previously formed the basis for the award of any academic degree, diploma or a similar title at this or any other university. Any materials borrowed from other sources (whether published or unpublished) and relied upon or included in my thesis have been properly cited and acknowledged in accordance with appropriate academic conventions. I further declare that there is no potential conflict of interest with respect to the research, data collection, authorship, presentation and/or publication of this thesis.

Student's Signature: _____

Date: _____

Approval of the Master Thesis

This Master Thesis is approved by the following Examining Committee Members:

1) Advisor (Committee Chair): [advisor's name]
Title: [e.g. Associate Professor]
Department of . . .
College of . . .
Signature: _____ Date: _____

2) Member:
Title:
Department of . . .
College of . . .
Signature: _____ Date: _____

3) Member (External Examiner):
Title: Professor
Department of . . .
Institution: [e.g. XYZ University, USA]
Signature: _____ Date: _____

This Master Thesis is accepted by:

Dean of the College of Information Technology: Professor Fekri Kharbash

Signature: _____

Date: _____

Dean of the College of Graduate Studies: Professor Ali Al-Marzouqi

Signature: _____

Date: _____

Abstract

Academic advising plays a critical role in helping students make informed decisions, improve academic performance, and successfully navigate their university journey. However, with increasing university enrollment, traditional advising methods often struggle to scale, leading to student frustration and overburdened advisors. Additionally, designing course offerings that match student demand is a complex and error-prone process involving multiple stakeholders. To address these challenges, this thesis proposes an automated, data-driven system for generating personalized academic plans for students. The primary aim of this thesis is to develop a system that reduces students' dependency on advisors while simultaneously providing accurate estimates of course demand to assist in academic planning for upcoming semesters. The proposed system operates in two phases. In the first phase, Knowledge Graphs (KGs) are used to model relationships between courses, prerequisites, and student progress. In the second phase, Machine Learning (ML) techniques and Large Language Models (LLMs) are integrated to further personalize course recommendations. The system is designed to ensure logical course progression while adhering to university-specific academic policies, with a case study conducted at the United Arab Emirates University (UAEU). The generated academic plans demonstrate up to 70% similarity when compared to the generic degree plans provided by the university and show an average of 80% similarity when compared to actual plans followed by graduated students. This work introduces a hybrid system combining Knowledge Graph modeling with Machine Learning personalization for academic advising, offering a scalable and interpretable solution that aligns course planning with student needs and institutional constraints. This thesis addresses the scarcity of research applying Knowledge Graphs for personalized academic planning in universities, bridging the gap between traditional advising practices and automated, data-driven recommendation systems tailored to individual student backgrounds.

Keywords: Automated academic advising, KGs, ML, Personalized academic plans, Course demand prediction, LLMs.

Title and Abstract (in Arabic)

تعزيز الإرشاد الأكاديمي باستخدام الرسوم البيانية المعرفية: دمج تعلم الآلة والنماذج اللغوية الكبيرة لتخطيط المقررات الدراسية بشكل مخصص

الملخص

تلعب عملية الإرشاد الأكاديمي دورا حيويا في مساعدة الطلاب على اتخاذ قرارات تشكل مستقبلهم الأكاديمي، وتحسين أدائهم الأكاديمي، والتنقل بنجاح في رحلتهم الجامعية. ومع ذلك، ومع تزايد أعداد الطلاب المسجلين في الجامعات، غالبا ما تعجز الطرق التقليدية في الإرشاد الأكاديمي عن التكيف مع هذا النمو، مما يؤدي إلى إحباط الطلاب وزيادة الضغط على المرشدين الأكاديميين. بالإضافة إلى ذلك، يُعد تصميم خطط طرح المقررات بما يتماشى مع طلب الطلاب عملية معقدة وعرضة للأخطاء البشرية. تهدف هذه الرسالة إلى تطوير نظام مؤتمت يعتمد على البيانات لتوليد خطط أكاديمية شخصية للطلاب، مما يقلل من اعتمادهم على المرشدين الأكاديميين، ويوفر تقديرات دقيقة للطلب المتوقع على المقررات لدعم التخطيط الأكاديمي للفصول الدراسية القادمة. يعمل النظام المقترح على مرحلتين، حيث يتم في المرحلة الأولى استخدام الرسوم البيانية المعرفية لنمذجة العلاقات بين المقررات والمتطلبات السابقة وتقدم الطالب الأكاديمي، بينما تتضمن المرحلة الثانية دمج تقنيات تعلم الآلة مع نماذج اللغة الكبيرة لتعزيز التوصيات الأكاديمية المخصصة. وقد تم تصميم النظام لضمان تسلسل منطقي للمقررات بما يتوافق مع السياسات الأكاديمية الخاصة بالجامعة، وتم اختباره باستخدام جامعة الإمارات العربية المتحدة كدراسة حالة. أظهرت نتائج الدراسة أن الخطط الأكاديمية التي تم توليدها حققت نسبة تشابه تصل إلى سبعين بالمائة مقارنة بالخطط الدراسية العامة التي توفرها الجامعة، ومتوسط تشابه بنسبة ثمانين بالمائة عند مقارنتها بالخطط الفعلية التي اتبعها الطلاب الخريجون. يقدم هذا العمل مساهمة مهمة من خلال تطوير نظام هجين يجمع بين نمذجة الرسوم البيانية المعرفية وتقنيات التخصيص القائمة على تعلم الآلة، مما يوفر حلاً قابلاً للتوسع والتفسير يتماشى مع احتياجات الطلاب ومتطلبات المؤسسات الأكاديمية. كما تسد هذه الرسالة فجوة بحثية واضحة تتمثل في نقص الدراسات التي تطبق الرسوم البيانية المعرفية لتوليد

خطط دراسية مخصصة للطلاب الجامعيين، مما يربط بين أساليب الإرشاد التقليدية والأنظمة التوصية المؤتمتة والمعتمدة على البيانات.

مفاهيم البحث الرئيسية: الإرشاد الأكاديمي المؤتمت، الرسوم البيانية المعرفية، تعلم الآلة، الخطط الأكاديمية المخصصة، التنبؤ بطلب المقررات، النماذج اللغوية الكبيرة.

Author Profile

Sara AlShamsi is currently working as a Programmer Analyst at the Application Services Unit of the United Arab Emirates University (UAEU) in Al Ain. In this role, she contributes to the development and maintenance of mission-critical systems including the Banner ERP, PL/SQL-based applications, and .NET APIs that power UAEU's web and mobile platforms. She is also involved in business analysis, backend process automation, and technical support operations.

Prior to this, she gained industry experience as a UI/UX Designer at LuLu International Exchange and AD Ports Group, and completed internships and project roles in areas ranging from 3D design to educational game development with robotics. Sara also participated in prominent technology programs such as the Arab Youth Technology Fellowship and Microsoft's Customer Success initiative.

She holds a Bachelor's degree in Computer Science from UAEU and is currently pursuing a Master's degree in Information Technology Management at the same institution. Additionally, she has completed a UX Designer Nanodegree from Udacity.

Acknowledgements

My thanks go to all those who supported me throughout this journey.

I am deeply grateful to my dear parents for their quiet strength, endless love, and unwavering patience, which carried me through even the busiest and hardest days. I am forever indebted to them. To my grandparents, whose pride and memory I hold close to my heart, your spirit has been a constant source of inspiration.

I would like to thank my committee for their guidance, support and assistance throughout the preparation of this thesis, especially my advisor, Dr. Nazar, whose calmness, encouragement, and ability to make every challenge seem manageable have made all the difference.

Special thanks go to my amazing colleagues, Alyazia and Shamma, for their companionship and unwavering support during this journey. I truly could not have done it without you.

I am also grateful to my family, friends, and every kind soul who stood beside me. Your kindness, belief in me, and encouragement made a profound difference.

And finally, a quiet thanks to myself — once full of doubt, now full of hope — for believing enough to take the first step. This is just the beginning of everything I am meant to become.

Dedication

To my beloved parents, grandparents, colleagues, and all those who believed in me.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

CS	Computer Science
IS	Information Security
KG	Knowledge Graph
LLM	Large Language Model

Chapter 1: Introduction

1.1 Overview

The number of students pursuing higher education has been steadily increasing as obtaining an undergraduate degree or higher has become a standard expectation in recent years [1]. Higher education institutions provide academic advising services to guide students in selecting their courses while considering graduation requirements, prerequisites, and academic progress. Effective advising significantly contributes to student success; however, from the advisor's perspective, reviewing each student's academic history is time-consuming and increasingly difficult as enrollment grows [2].

Higher education institutions worldwide, including those in the United Arab Emirates (UAE), are experiencing rapid increases in student enrollment. According to recent reports, higher education enrollment in the UAE continues to grow as part of the country's national focus on expanding educational opportunities [3]. Estimates indicate that over 140,000 students are enrolled in UAE higher education institutions, reflecting this ongoing growth [4]. At the United Arab Emirates University (UAEU), the country's oldest and one of its largest universities, enrollment has steadily increased across all majors [5]. This growth places additional strain on traditional academic advising and registration systems.

Currently, most registration systems fall into two main categories [6]. The first category provides students with basic course information, requiring them to manually plan and register for their schedules. If a course is full, students must repeat the process, often leading to delays and frustration. UAEU's Banner system is a typical example of this approach. The second category features more advanced tools that help students search for courses and automatically generate all possible schedules, reducing manual effort. Examples include systems used at Rutgers University, the University of Technology Sydney, and AU SPARK.

Despite technological advances elsewhere, many institutions, including UAEU, still heavily rely on manual advising and scheduling processes. Students often face difficulties in

registering for needed courses, while advisors, overwhelmed by large caseloads, struggle to provide timely and personalized guidance. Furthermore, course offerings are often planned without precise knowledge of student demand, creating a gap between what students need and what is available. This mismatch can delay graduation and increase stress for both students and staff [6, 2].

1.2 Statement of the Problem

Manual advising and registration systems are becoming increasingly inefficient in the face of growing student populations. Traditional advising methods are not only time-intensive but also prone to human error, especially under pressure. Advisors may unintentionally overlook critical course recommendations, and students may miss essential courses, delaying their academic progress.

Moreover, course planning decisions are made before actual student demand is clear. Department chairs must predict which courses to offer, often with limited information. If students do not have accurate, timely course plans, their individual needs may not align with available offerings, leading to bottlenecks at critical stages in their academic journey.

There is a critical need for an intelligent, data-driven solution that supports both students and academic departments. Such a system would automate course planning for students, reduce advisor workload, and enable department chairs to better forecast course demand based on real, aggregated student data. This would ensure a smoother, more efficient advising and registration experience for everyone involved.

1.3 Research Objectives

The main objective of this research is to design and evaluate an intelligent recommendation system that generates personalized course plans for students in higher education. While the broader vision includes providing department chairs with insights for course scheduling, this research focuses specifically on the development and evaluation of the plan generation component.

To achieve this goal, the research will explore two approaches. The first approach

involves utilizing knowledge graphs (KGs) and their embeddings to represent course structures, prerequisites, and student academic progress, enabling a structured and dynamic understanding of academic pathways. The second approach applies machine learning (ML) techniques combined with large language models (LLMs) to analyze historical enrollment and academic records, generating course recommendations tailored to students' academic needs and goals.

The system seeks to automate and personalize the advising process, helping students navigate their academic journeys more efficiently. By addressing the gap between course offerings and actual student demand, the system aims to reduce manual workload, minimize registration bottlenecks, and optimize academic planning across institutions.

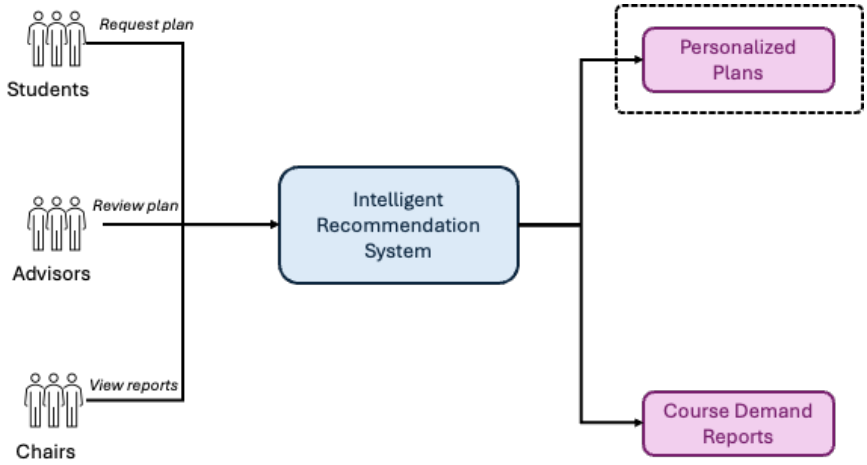


Figure 1.1: High-Level Overview of the Proposed Academic Advising System

An overview of the envisioned system is illustrated in Figure 1.1. Students interact with the Intelligent Recommendation System to generate course plans aligned with their academic records and program requirements. Academic advisors can review, validate, or refine these plans to ensure consistency with academic policies. Department chairs can access aggregated course demand reports, allowing them to make data-driven decisions regarding course offerings and section allocations. This integrated approach aims to streamline academic advising and improve the overall efficiency of the course planning process.

Chapter 2: Literature Review

Much research has been done on creating academic plans and recommending courses for students. This chapter explains what recommender systems are, compares them, and highlights gaps in the existing research.

2.1 Recommender Systems

A recommender system (RS) can be defined as a system that offers personalized suggestions for candidate items, tailored to meet user preferences [7]. Recommender systems are often integrated into other platforms to assist with recommendations. For instance, on Netflix, the latest movies can be suggested to users based on their viewing history and the type of content they generally prefer. This helps users discover new movies aligned with their interests without having to browse through extensive lists. As in the Netflix example, recommendations are based on data. The primary steps of a recommender system involve analyzing the collected data, extracting useful information from the analysis, and predicting what the user might find beneficial to recommend [8].

Recommender systems can be categorized into four main types: (1) collaborative filtering recommender systems, (2) content-based recommender systems, (3) knowledge-based recommender systems, and (4) hybrid recommender systems.

2.1.1 Collaborative Filtering Recommender Systems

Collaborative Filtering Recommender Systems (CFRS) are a type of recommender system that make recommendations based on similarities between users. In other words, they ignore the features of the items themselves and focus on user behavior with the assumption that similar users prefer similar items [9, 10, 11, 8, 12, 13, 7]. CFRS are considered the simplest [11] and most commonly used type of recommender systems [7].

The core process in CFRS involves analyzing data from all users, often grouping them into clusters [8], and predicting which items a user might be interested in based on their similarities with other users [10, 11]. These similarities are usually measured using

Euclidean distance, which is the simplest and most common similarity measure for such systems [12]. CFRS are more effective when there is a large amount of user data available for analysis.

One advantage of CFRS is that they do not require an understanding of the items themselves [10], as the focus is entirely on user data. However, they face several challenges, such as the cold start problem, where the system performs poorly when there is limited data on new users [10, 8, 14], sparsity, which occurs when there are not enough ratings to identify meaningful similarities between users [8], and the grey sheep problem, where users with unique or unusual preferences receive inaccurate recommendations [8].

In general, a CFRS can be classified into two categories: model-based and memory-based methods. Model-based methods create a model from the data and then use this model to make predictions, while memory-based methods store raw data to identify similar users or items and make predictions as needed [12].

Various techniques are employed to construct a CFRS, including matrix factorization [10, 15, 14, 7], correlation-based approaches, Bayesian networks, and association rules [8].

The paper [15] proposed an ML-based course enrollment RS that utilizes data such as enrollment history, prerequisite restrictions, meeting times, instructional methods, and instructors. The system uses matrix factorization to assist students in selecting the best courses to take and to save academic advisors' time. The model generates recommendations by analyzing registration history and student preferences, filtering out already completed courses and prioritizing failed ones.

Another study [8] proposed an automated RS that suggests elective courses to university students based on the affinity between courses taken by the target student and other students. This system employs a collaborative approach combined with association rule mining. The process begins by clustering students based on their grades using k-means clustering, and these clusters are then used to find the nearest neighbors. Association rule mining is applied to generate course recommendations. In addition to recommending courses, the system

predicts the expected grade, allowing students to choose courses in which they are likely to perform well.

The paper [16] utilized a collaborative filtering approach to recommend courses that students are more likely to pass among their eligible options. For each course a student is eligible for, the system recursively queries the database to find students who have completed the course, checking their grades to see if they are less than or equal to the student's grade in the prerequisite course. The system suggests the course to the student if the passing rate among these students is higher than 60%.

In another study [17], a method was outlined for predicting suitable courses for students based on their performance and peer correlations using the Pearson Correlation Coefficient. The process begins by gathering course scores and determining correlations between students to identify neighbors, specifically selecting 30 students who are most similar to the target student. Suitable courses are those taken by these neighbors but not yet completed by the target student. Then, a prediction formula estimates potential scores for these courses while applying priority rules, such as prioritizing failed courses, those with numerous prerequisites, and high-level curriculum courses.

The paper [6] proposed a recommendation system for students at Assumption University, allowing them to involve friends in the schedule selection process by considering social constraints, such as aligning schedules with friends. The system is built on a Constraint Satisfaction Problem (CSP) framework, with defined constraints and a backtracking approach for searching. To determine the search order, domain ordering is used to prioritize courses with fewer sections. By utilizing the Facebook API, which provides information about students' profiles and friends, the system calculates a schedule score. A schedule is considered more interesting when it includes sections favored by friends, as many students prefer to study together.

Lastly, the study [18] addressed the challenge of course enrollment recommendations without considering prerequisites or degree requirements. The authors proposed a Markov

Chain-based collaborative filtering model that predicts course enrollments based on past course sequences of similar students, identified using Jaccard set similarity.

2.1.2 Content-Based Recommender Systems

Content-Based Recommender Systems, or CBRS, differ from CFRS in that they focus on the features of items rather than considering other users. They recommend items based on the affinity between the items and the user's profile [10, 8], assuming that the user's behavior remains consistent over time, allowing for the prediction of desired content or items [12]. CBRS primarily considers two main components: (1) the user profile and (2) the description of items [10, 12, 7]. The user profile encompasses historical data, including user preferences, likes, ratings, search history, and purchase history [9, 10, 11, 8, 12, 7].

One advantage of CBRS is that it is not affected by the availability of data from other users [10]. This allows CBRS to perform more effectively in cold-start scenarios compared to CFRS, as it relies solely on data related to the target user.

However, there are several disadvantages. First, CBRS requires complex algorithms to analyze the user's profile and preferences [10]. Second, it typically works only for textual data [8]. Third, it cannot provide recommendations for users with empty profiles, meaning newcomers may not receive suggestions [8]. Fourth, data sparsity can be an issue, as most users interact with only a small fraction of the total items available [14].

Various methods and techniques can be employed in CBRS, including approximation theory, ML, nearest neighbor approaches, Bayesian classifiers, neural networks, and association rule mining [8].

2.1.3 Knowledge-Based Recommender Systems

KBRS are a type of RS that is used for complex domains when knowledge is needed to recognize solutions and justify them such as financial services [9, 8]. It makes recommendations according to deep knowledge it has about the features of items [11, 8]. It often adopts AI techniques to deduce a match between users and items [8].

2.1.4 Hybrid Recommender Systems

A hybrid RS combines multiple types of recommendation systems [9, 11, 8, 13], allowing for improved accuracy and performance while mitigating certain drawbacks associated with relying on a single type [8, 12]. Generally, there are three main designs for implementing a hybrid RS [12]. First, the monolithic design, in which multiple techniques are integrated into a single algorithm. Second, the parallelized design, where a minimum of two different models provide recommendations independently, which are then combined into one. Lastly, the pipelined design utilizes different models in such a way that the output of one model becomes part of the input for the subsequent model.

Paper [10] proposed a hybrid solution called CrsRecs that provides personalized course recommendations to students. This solution integrates several components: it combines students' preferences as input, conducts topic analysis of course descriptions using an ML algorithm to identify topic distributions and suggest clusters for courses, performs sentiment analysis of students' ratings on professors, and utilizes matrix factorization for rating predictions. The solution generates a ranking score for courses, which is then used to recommend the most suitable courses to students.

In another paper [12], a hybrid approach was employed to assist students in selecting elective subjects. This approach combined collaborative filtering and content-based methods, utilizing student enrollment data, grades, and subject content to generate a weighted ordered list of recommended elective subjects. The system consists of two models: the first model recommends the first elective option, while the second model suggests the second elective option. The study concluded that the hybrid approach outperformed the individual methods.

Table 2.1 summarizes the key characteristics, strengths, and limitations of the different four types of recommender systems.

2.1.5 Other Techniques for Recommendations

Other techniques used for recommendations include genetic algorithms [19], Natural Language Processing (NLP) [20], recurrent neural networks (RNNs) [21], and large language

Table 2.1: Comparison of Different Types of Recommender Systems

Type	**Description**	**Strengths**	**Weaknesses**
CFRS	Recommends items based on the preferences and behavior of similar users, ignoring item features.	Does not require item content knowledge; can discover complex user interests; widely used and mature.	Cold start problem (new users/items); sparsity issues; grey sheep problem with unique users.
CBRS	Recommends items by comparing item features with a user's past preferences or profile.	Works well for new users with sufficient profile information; independent of other users' data.	Limited to known user preferences; struggles with cold start for new users; usually restricted to textual or structured item features.
KBRS	Recommends items based on domain knowledge about items and user requirements, often using rule-based or AI techniques.	Effective in complex domains where deep understanding is needed; no need for historical user-item interaction data.	Requires rich knowledge bases and expert input; building and maintaining knowledge bases can be costly and complex.
Hybrid Systems	Combine two or more recommendation approaches (e.g., CF + CBF) to enhance recommendation quality.	Mitigates the weaknesses of individual methods; can achieve higher accuracy and broader coverage.	More complex to design, implement, and maintain; requires careful integration of models.

models (LLMs) combined with knowledge graphs [22, 23].

In [19], the authors developed a RS that utilizes students' profiles, including reviews and ratings assigned to courses, as well as course content and professors' information. This system combines a collaborative filtering model with a content-based filtering model, which is further enhanced by incorporating a genetic algorithm aimed at optimizing the recommendation process and identifying the most suitable options for each student.

An RNN was utilized in [21], where the authors employed its variant, Long Short-Term Memory (LSTM), to assist students in making informed decisions regarding their course selections for a semester. The recommendations are goal-based, in that the student sets a desired grade (A or B) for a target course, which is used to provide recommendations for the previous semester that maximize the predicted probability of the student attaining this goal (the desired grade).

In [22], the authors developed a chatbot to answer queries about learning material recommendations by leveraging a LLM to understand and respond to the questions. The LLM was guided by expert-defined rules regarding the type of information to be provided, and a KG was used as the main source of contextual information about learning materials

and their semantic relationships. The extracted information from the KG was then used to feed the LLM's prompt, enabling it to provide more accurate responses to the user. In the KG, learning materials were connected based on the user's context. For instance, courses related to data analysis for computer scientists were distinct from those tailored for health experts, and these differences were reflected in the graph.

Knowledge Graphs were also employed in [23], where the authors sought to enhance LLM recommendations and provide explanations for why specific items were recommended, particularly in the context of learning path recommendations. The KG was constructed using data such as titles and descriptions of educational materials, with semantic relationships between courses defined by experts. Text mining techniques were applied to extract the main topics from each learning object, and the resulting topics were compared to create relationships between objects with high semantic similarity.

2.2 Gap in Literature

Using knowledge graphs (KGs) in recommender systems has been shown to produce high-quality recommendations that are easier to interpret and explain [24, 25]. However, the application of KGs for recommending academic plans to university students remains relatively underexplored, with most existing research relying on traditional recommendation methods. Although recent studies have begun to investigate the use of KGs, these efforts have primarily focused on general learning contexts without considering the individual backgrounds and academic trajectories of university students.

A comparative summary of existing course recommendation studies is provided in Table 2.1, highlighting the features utilized, the scope of recommendations, and the techniques applied. An "x" in the table indicates the presence of a specific feature, scope, or technique in the corresponding study.

Despite these advancements, no existing work has specifically applied KGs to generate personalized study plans tailored to university students. Yet, KGs offer unique advantages by providing a human-readable structure that extends beyond simple tabular data, as well as

embeddings capable of capturing deep relational insights.

To address this gap, this thesis proposes the development of an academic plan recommendation system that leverages knowledge graphs to generate course suggestions personalized to each student's academic background and goals. By integrating KGs into the recommendation process, the system aims to enhance the relevance, interpretability, and overall effectiveness of academic planning support in a university setting.

Table 2.1: Comparison of Course Recommendation Systems

Criteria		[22]	[23]	[15]	[26]	[16]	[12]	[20]	[27]	[17]	[21]	[19]	[28]	[6]	[10]	[8]	[18]	[29]	[13]	[30]	[31]	Thesis
Features	grades				x	x	x		x	x	x	x				x						
	Course Prerequisites			x		x				x												x
	Course Content	x	x				x					x			x			x				x
	Enrollment History			x	x	x	x	x	x	x	x		x			x	x		x	x		x
	Meeting Times			x	x									x						x	x	
	Available Seats				x			x												x	x	
	Instructors			x								x			x					x	x	
	Course Sequences				x			x			x		x			x	x	x	x			
	Course Ratings											x			x							
	friends					x								x								
	Knowledge Graphs	x	x																			x
Scope	University			x	x	x	x	x	x	x	x	x	x	x	x	x	x		x	x	x	x
	General learning	x	x															x				
Techniques	ML			x											x							x
	Matrix Factorization			x											x							
	Sentiment Analysis														x							
	Mathematical Model				x															x	x	
	Scheduling Algorithm																					
	Collaborative Filtering			x		x	x			x		x	x	x		x	x		x			x
	Markov Chain												x				x					
	Content-Based RS						x					x										x
	NLP							x														
	RNNs								x		x											
	Genetic Optimization																					

Chapter 3: Research Methodology

This section outlines the research methodology employed in developing the academic plan recommendation model. Figure 3.1 shows an overview of the steps that were followed.

In general, a knowledge graph is constructed to represent university courses, incorporating course levels, topics from each syllabus, and their respective majors. To support the recommendation process, the Node2Vec algorithm generates embeddings for each course. These embeddings are then used to create study plans based on student progress. This approach provides a structured, data-driven foundation for personalized academic recommendations.

3.1 Tools

The course recommendation system was built using Neo4j for knowledge graph construction and Python for data processing and analysis. Various libraries supported different steps of the process. Py2neo enabled communication between Python and Neo4j, while Pandas and NumPy handled data manipulation. Scikit-learn was used to calculate course similarity scores, Pyvis for visualizing the knowledge graph, and Streamlit for creating an interactive interface to present the generated plans.

Two distinct approaches were developed for course plan generation. The first approach leveraged the KG in Neo4j. The second approach used, in addition to the KG, frequent pattern mining with Apriori and refined the recommendations using the Claude 3.5 Sonnet LLM via OpenRouter. Some tools were used in both approaches, while others were specific to a single method.

3.2 Data Collection

The data for this research was gathered from the College of IT at UAEU. The dataset consisted of three main objects: Courses, Students, and Majors. Figure 3.2 shows the datasets collected and their content.

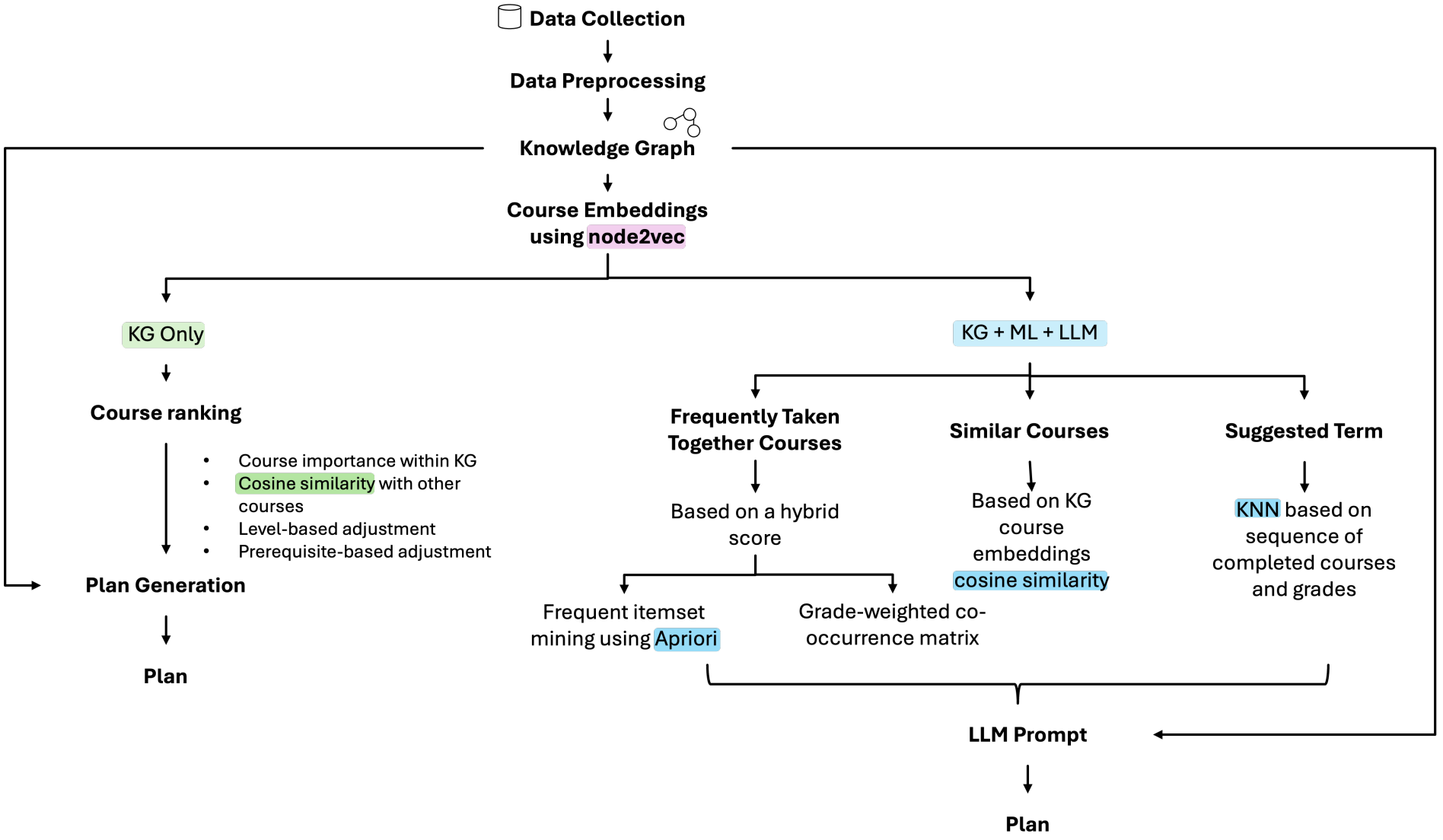


Figure 3.1: Overview of Methodology Steps

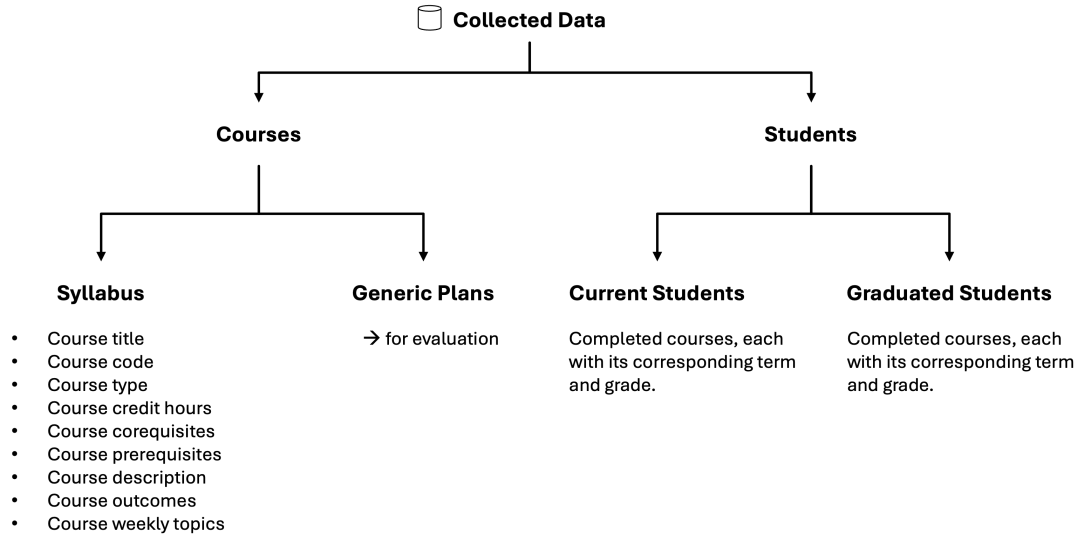


Figure 3.2: Datasets Collected and Their Content

3.2.1 Majors

The dataset includes three undergraduate programs: Bachelor of Computer Science (CS), Bachelor of Information Security (IS), and Bachelor of Science in Information Technology (IT). Each course is linked to one or more majors. A Computer Science student at UAEU must complete 43 courses, which include 37 mandatory courses, 4 major electives, and 2 cultural electives, while an Information Security student must complete 44 courses, comprising 40 mandatory courses, 2 major electives, and 2 cultural electives. Similarly, an Information Technology student must complete 43 courses, which include 36 mandatory courses, 3 major electives, 2 cultural electives, and 2 free electives.

3.2.2 Courses

The course details were extracted from the respective course syllabi and systematically organized into an Excel sheet for a structured format. Each record in the dataset includes key attributes such as the course ID or code, title, credit hours, type (mandatory, general elective, or major elective), description, learning outcomes, topics or chapters covered, prerequisites (indicating courses that must be completed prior to enrollment), and co-requisites (indicating courses that must be taken simultaneously within the same term).

3.2.3 Students

To provide personalized course recommendations, the system first identifies the courses a student has completed and those they are currently registered for. Therefore, the dataset includes student IDs, their majors, completed courses along with grades, and registered courses for the current semester.

Additionally, historical data from students who have completed all their courses and graduated with a GPA of 3.3 or higher was used. The dataset included structured records containing student identifiers, academic terms, course codes, course titles, and grades (in letters). The goal is to learn from high-performing students. However, due to the GPA restriction and the limited number of graduates meeting this criterion, the dataset includes only 119 IS students, 92 CS students, and 9 IT students. While this relatively small sample size may be considered a limitation, the results will determine whether it is sufficient for meaningful analysis.

3.3 Data Preprocessing

The gathered data was primarily structured; however, some preprocessing was necessary. The first step involved introducing a new object type: the course level, which was extracted from the course ID. For instance, a course with the ID ‘ISEC324‘ corresponds to level number 3, indicating that this course is mostly appropriate for third-year students.

The second step involved processing student historical records by converting letter grades into numerical weights for analysis. To standardize course representation, a unique course label was created by combining the subject code and course title. Additionally, the term values, which originally varied across students based on enrollment periods, were normalized within each student's record. This was done by ranking the terms sequentially for each student, ensuring that their first recorded term was assigned 1, the second was assigned 2, and so on. This normalization allowed for a consistent representation of course progression across all students, regardless of their actual enrollment years.

The third step introduced the final object type: topics. The collected data included course

descriptions, outcomes, and covered chapters, which were initially in raw format. In this step, ChatGPT was employed to extract the main key topics from each of the three inputs for every course. Consequently, each course X was associated with multiple topics, allowing for potential relationships with other courses Y that share similar topics. To enhance data uniformity and ensure that courses with identical topics could be effectively linked in the knowledge graph, all topics were converted to lowercase.

3.4 Knowledge Graph Construction

Based on the gathered data, a knowledge graph has been constructed using a tool called *Neo4j* to form the main source of information for the recommendation system. This tool allows building a knowledge graph and querying it using the Cypher language

What is a Knowledge Graph

A knowledge graph is a graph that represents relational facts about different entities, often in the form of a triplet. The triplet consists of three parts: a head entity, a relationship (or an edge), and a tail entity [32]. For example, in the context of university courses, we can express the relationship between two courses as follows: course A is a prerequisite for course B. In this example, course A serves as the head entity, *prerequisite* is the relationship, and course B is the tail entity. The edges can include properties, representing additional information about the nodes or their relationships.

Graph databases, which are optimized for storing and querying such structures, provide a flexible and efficient way to represent complex data. Their inherent structure, combined with fast data access, makes graph databases highly practical for managing interconnected data [33].

Knowledge Graph Structure

The structure used for the knowledge graph is based on five main entities: Course, Student, Major, Topic, and Level. Table 3.1 illustrates the different relationships that connect these entities. For example, the course entity has a relationship called *requires* with another course entity.

Table 3.1: KG Triplets

Head Entity	**Relationship**	**Tail Entity**
course	co_course	course
course	prerequisite_for	course
course	requires	course
student	completed	course
course	completed_by	student
student	registered_for	course
course	registered_by	student
major	has_student	student
major	has_course	course
course	has_level	level
course	description_topic	topic
course	outcome_topic	topic
course	material_topic	topic

As shown in Figure 3.3, the knowledge graph structure illustrates course relationships based on prerequisites. This figure represents only a subset of the full graph, highlighting how, for example, the Data Structures course serves as a prerequisite for multiple other courses, including the Object-Oriented Programming course. Similarly, Object-Oriented Programming is also a prerequisite for several courses. The prerequisite relationship is fundamental to structuring any academic plan.

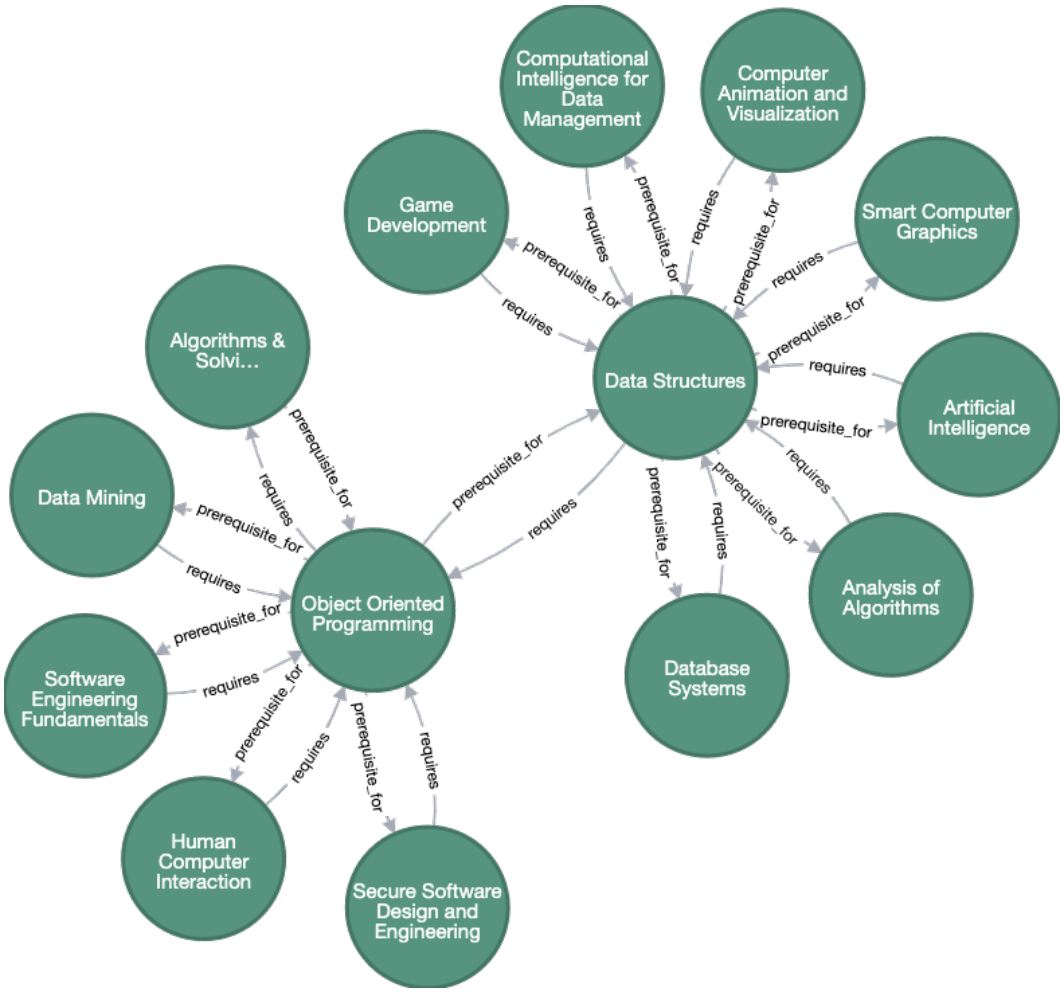


Figure 3.3: KG Highlighting `prerequisite_for`/`requires` Relationships

3.4.1 Course Embeddings

As a KG was employed in this system, an effective approach to leverage KGs in recommendation systems is to use an embedding-based method. This involves converting the KG into a knowledge graph embedding (KGE) [34], which can then be applied to various tasks, including prioritizing items for recommendation.

Node2Vec

In this study, the created graph is transformed into embeddings using the Node2Vec algorithm [35], which generates embeddings for courses within an in-memory graph representation. Node2Vec is effective for learning representations of nodes in a graph, as it captures both the relationships and structural properties of the nodes (in this case, the courses). Each course is represented as a vector in a multi-dimensional space, where the

embedding encapsulates all relationships associated with the courses, illustrating how they are interconnected. These embeddings enable comparisons between courses; for instance, courses with similar embeddings may indicate related topics or prerequisites.

The primary goal of the Node2Vec algorithm is to map nodes in a graph into vectors or an embedding space, such that nodes with similar contexts are positioned close to each other; in other words, their vectors form smaller angles with one another.

But what defines the context of a node? In Node2Vec, the context refers to the neighbors of a node, which can be determined in two main ways: community-based relationships, where the node has direct connections to others in the same community (using breadth-first search, or BFS), and structural equivalence, where context is based on the node's role in the graph from a broader perspective (using depth-first search, or DFS). For example, in Figure 3.4, node n shares a community with node c, but has a similar structural role to node e.

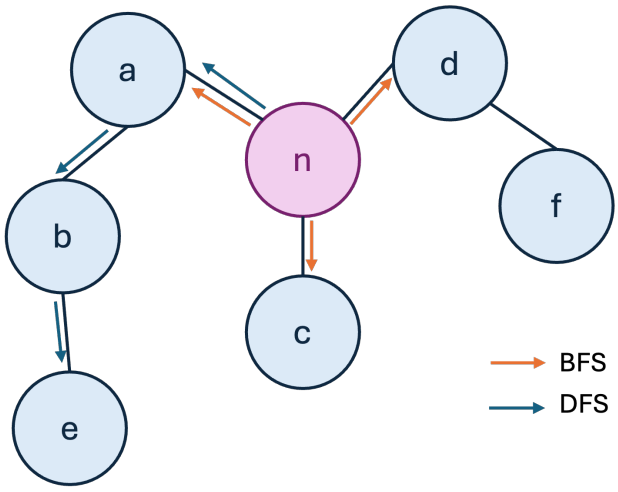


Figure 3.4: BFS and DFS Search Starting from Node n

To approximate the context of each node without traversing the entire graph, Node2Vec uses biased random walks, with the number of walks being a hyperparameter. These random walks are guided by two key hyperparameters: p (the return factor) and q (the in-out factor), which determine the edge transition probabilities. These probabilities control which node will be visited next. Specifically, the parameter p influences the likelihood of immediately returning to a node that was just visited in a walk, while q affects the likelihood of either staying in the current neighborhood or visiting nodes that are further away. For example, a

small value of q increases the likelihood of moving away from the starting node, leading to a more DFS-like traversal rather than BFS-like traversal. Similarly, A high value of q decreases the likelihood of moving away from the starting node, resulting in a more BFS-like traversal [36].

After determining the transition probabilities for each node, the walk sampling begins. For each edge, a biased coin flip is performed to decide which edge to follow next. The nodes that are visited during the walk represent the context for each node.

Next, Node2Vec aims to find embeddings such that nodes with similar contexts are close together in the embedding space.

The probability of observing a neighboring node v given the current node u is represented by $P(\vec{v} \mid \vec{u})$. This probability depends on the similarity between the embeddings of u and v, calculated as $f(u) \cdot f(v)$, where f is the embedding function. Higher similarity between embeddings results in a higher probability, meaning nodes with similar embeddings are more likely to appear together.

For each node u, the probability of observing all nodes in its context (denoted $N_s(u)$) is represented by $P(N_s(u) \mid u)$. Node2Vec assumes that the probability of each neighboring node is independent, so the overall probability for observing the context $N_s(u)$ is the product of the probabilities for each individual neighbor:

$$\prod_{v_i \in N_s(u)} P(v_i \mid u)$$

where v_i are individual neighbors of u.

To learn embeddings for the entire graph, we apply this process to all nodes u in the graph. The objective is to maximize the probability of observing actual neighbors for each node. This leads to the following objective function, where we sum the log probabilities of observing each neighborhood:

$$\sum_{u \in V} \log P(N_s(u) \mid u)$$

Here, V represents the set of all nodes in the graph.

The Node2Vec algorithm then seeks to optimize this objective function to maximize the logarithmic probability of observing a neighbor $N_s(u)$ in the embedded space of node u.

3.5 Plan Generation Using Knowledge Graphs

3.5.1 Course Ranking Mechanism

Once embeddings are generated for each course within the target student's major s, a course prioritization algorithm is applied. First, the graph is queried to retrieve the completed courses, registered courses, and remaining courses. The primary objective is to identify and rank the remaining courses based on the student's major, considering their importance within the knowledge graph and their relevance to the student's knowledge progression.

Each course's embedding is a high-dimensional vector within the graph space. The L2 norm of this vector (i.e., the magnitude or length of the vector) is used to reflect the centrality or importance of the course within the knowledge graph. A higher norm indicates a course that is more interconnected and central, signifying its pivotal role in the graph. Mathematically, for a vector $\mathbf{v} = [v_1, v_2, \ldots, v_n]$, the L2 norm is calculated as:

$$\|\mathbf{v}\|_2 = \sqrt{\sum_{i=1}^{n} v_i^2}$$

To calculate the importance of each course, a logarithmic transformation is applied to the L2 norm:

$$\text{importance} = \ln(1 + \|\mathbf{v}\|_2)$$

This transformation compresses the range of values, making differences in centrality among courses easier to compare. Courses with higher norms are assigned higher scores.

For each remaining course, the score is further adjusted based on its level: lower-level courses receive higher multipliers to emphasize their importance in being completed earlier in the student's academic plan.

Personalized Course Similarity

To personalize the recommendations, the cosine similarity between the embeddings of each course is computed and stored in a similarity matrix. The cosine similarity between two courses i and j with embeddings $\mathbf{v}_i$ and $\mathbf{v}_j$ is given by:

$$\text{cosine\_similarity}(i, j) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|}$$

where $\mathbf{v}_i \cdot \mathbf{v}_j$ is the dot product of the embeddings $\mathbf{v}_i$ and $\mathbf{v}_j$, and $\|\mathbf{v}_i\|$ and $\|\mathbf{v}_j\|$ are the magnitudes (or norms) of the respective embeddings. This metric evaluates the similarity between the content of two courses, where a value closer to 1 indicates high similarity and a value closer to 0 indicates low similarity.

The similarity matrix is used to enhance the scores of remaining courses based on their relationships with the completed courses. For each remaining course, its similarity to all completed courses is summed and added to its score. This ensures that courses closely related to the student's completed coursework are prioritized, promoting continuity and building upon existing knowledge.

Boosting Prerequisite Scores

Lastly, the ranking scores of courses are used to boost the scores of their prerequisites. Each prerequisite's score is increased proportionally to the ranking score of its associated course, weighted by a factor x. This adjustment ensures that prerequisites of high-ranking courses are prioritized.

3.5.2 Plan Generation

To generate a study plan for a student s, there is one main input: the maximum number of credit hours they wish to take each semester. Essential information is first retrieved from the knowledge graph. This includes the student's major, the list of courses they have already completed, and any courses currently registered (which are treated as completed). The list of remaining courses is then determined by identifying all required courses for the student's major m and excluding the completed and registered courses.

The completed courses are fed into the Course Ranking algorithm, which outputs a prioritized list of courses. This list is then normalized to a 0 to 1 range. Next, the list is dynamically balanced by forming groups of courses, with each group containing approximately equal-thirds of courses from the high-score, medium-score, and low-score set. This ensures a balanced distribution of scores across the list, which would prevent an overloaded semester with only high-score, high-effort courses, and therefore promoting a well-balanced course plan.

The course planning algorithm prioritizes courses based on a balanced ranked list while ensuring several constraints are met. The total credit hours per semester do not exceed the student's specified maximum. Courses with prerequisites are scheduled only after all prerequisites are completed, while co-requisites are scheduled in the same term as their corresponding courses. Elective courses are selected based on predefined limits; for example, a Computer Science student must complete two cultural electives (one from each group) and four major electives. Additionally, certain courses follow fixed scheduling rules, such as Senior Project 1 being scheduled two semesters before the final semester, Senior Project 2 one semester before the final semester, and the internship during the final semester.

3.6 Plan Generation Using ML and LLM

To enhance course plan recommendations using a KG, a more effective approach involves integrating multiple data sources, including the KG and historical enrollment data. Machine learning techniques can be applied to identify patterns in course selection and academic progression. These insights, along with structured KG data, are then incorporated into a single prompt for an LLM, which generates a personalized study plan tailored to each student's academic history and requirements.

3.6.1 LLM Prompt

To generate personalized course plans using LLMs, a structured prompt is designed to provide essential academic information. The prompt includes a student's completed courses, remaining credit hours, GPA, and categorized available courses. It ensures that prerequisites

and corequisites are met, balances credit hours per semester, and accounts for commonly taken courses, similar courses, and suggested enrollment terms. The LLM processes this data to generate an optimized study plan, aligning with graduation requirements. The following sections detail how commonly taken courses, similar courses, and suggested enrollment terms are determined.

3.6.2 Frequently Taken Together Courses

To determine which courses are frequently taken together, a hybrid approach was employed. This approach integrates frequent itemset mining using the Apriori algorithm with a grade-weighted co-occurrence matrix to enhance the robustness of course association discovery. The method follows four main steps: (1) transaction generation, (2) frequent itemset mining, (3) co-occurrence matrix computation, and (4) hybrid score computation.

(1) Transaction Generation

To identify patterns in course enrollment, student course records are transformed into transactions. Each transaction consists of the set of courses taken by a student within a specific term. Formally, for a given student S_i in term T_j, the enrolled courses are represented as:

$$C_{i,j} = \{c_1, c_2, \ldots, c_n\}$$

where c_k denotes a course taken in term T_j. Transactions are extracted by grouping student records based on student ID and term. As a result, the number of transactions for a student is equal to the number of terms they have completed.

(2) Frequent Itemset Mining

To identify commonly co-enrolled courses—i.e., courses that exhibit a strong association—the Apriori algorithm, a foundational technique for frequent itemset mining [25], is applied to the generated transactions. Apriori extracts frequent itemsets, where an itemset is defined as a subset of courses appearing together in the same term. The support of an

itemset X is computed as:

$$\text{Support}(X) = \frac{\text{Number of transactions containing } X}{\text{Total number of transactions}}$$

Only itemsets with support above a predefined threshold σ (i.e., minimum support) and containing at least two courses are retained.

(3) Co-Occurrence Matrix Computation

While frequent itemset mining captures common course groupings, it does not account for the strength of relationships between courses based on grades or performance. To address this, a co-occurrence matrix is constructed, where each entry $M(c_i, c_j)$ quantifies the weighted co-occurrence of two courses c_i and c_j. The weight is computed based on the average grade weight of students who took both courses:

$$M(c_i, c_j) = \sum_{S_k \in P_{i,j}} \frac{G_k(c_i) + G_k(c_j)}{2}$$

where $P_{i,j}$ represents the set of students who took both courses c_i and c_j, and $G_k(c)$ is the grade weight of student S_k in course c.

(4) Hybrid Score Computation

To combine the strengths of both methods, a hybrid score is computed for each frequent itemset. Given an itemset $X = \{c_1, c_2, ..., c_m\}$, its hybrid score is calculated as:

$$H(X) = \alpha \cdot \text{Support}(X) + (1 - \alpha) \cdot \text{AvgCo}(X)$$

where α is a weighting factor balancing support and $\text{AvgCo}(X)$ represents the average co-occurrence of all course pairs within X. If α = 1, the result is entirely based on the frequent itemset support, while if α =0, then the result is entirely based on the grade-weighted co-occurrence frequency.

3.6.3 Similar Courses

To identify similar courses within a given major, the KG was utilized (see Section 3.4). Let $G = (V, E)$ represent the constructed KG, where V is the set of nodes (courses). Each course $c_i \in V$ is associated with a high-dimensional embedding $\mathbf{e}_{c_i} \in \mathbb{R}^d$, as described in

Section 3.4.1.

To measure the similarity between courses, the cosine similarity is used, and it is computed as follows:

$$\text{sim}(c_i, c_j) = \frac{\mathbf{e}_{c_i} \cdot \mathbf{e}_{c_j}}{\|\mathbf{e}_{c_i}\| \|\mathbf{e}_{c_j}\|} \tag{3.1}$$

A similarity threshold τ is defined such that two courses c_i and c_j are considered similar if:

$$\text{sim}(c_i, c_j) \geq \tau. \tag{3.2}$$

3.6.4 Suggested Term

The K-Nearest Neighbor (KNN) algorithm is a classification method that assigns an unclassified sample point, in this case, a student, the classification of the nearest of a set of previously classified points, i.e., other students [37]. The similarity measure determines the "nearest" students.

To determine the most appropriate term for a course C in the academic plan of a student S, a KNN-based approach is used. If S is a new student, k is set to include all available students, whereas for continuing students, k is set to a fixed constant. Similarity in this context is defined on the basis of the order in which students complete the same courses and their corresponding grades. This is achieved in two main steps: (1) constructing student vectors, and (2) recommending the term.

Similarity Measure

Example: Consider two students, S_1 and S_2, who have completed the courses CSBP119, CSBP219, and CSBP319 in the same sequential order across terms and have achieved similar grades. These students are considered highly similar.

Formally, let the set of completed courses for student S_k be represented as:

$$C_k = [(c_1, g_1, t_1), (c_2, g_2, t_2), \ldots, (c_n, g_n, t_n)]$$

where c_i denotes a course, g_i represents the corresponding grade weight, and t_i indicates the term in which the course was completed.

If two students, S_1 and S_2, have:

$$C_1 = [(CSBP119, g_1, 1), (CSBP219, g_2, 2), (CSBP319, g_3, 3)]$$

$$C_2 = [(CSBP119, g_1', 1), (CSBP219, g_2', 2), (CSBP319, g_3', 3)]$$

such that their grades satisfy $|g_i - g_i'| \leq \epsilon$ for a small threshold ϵ, then their similarity score is high.

However, if another student S_3 has taken the same courses but in a different sequence of terms, e.g.,

$$C_3 = [(CSBP119, g_1'', 2), (CSBP219, g_2'', 3), (CSBP319, g_3'', 5)]$$

then their similarity score is reduced. In this case, the difference is not just in the order of the courses within a list but in the terms in which they were completed.

This formulation assumes that only the three courses are considered while ignoring the influence of other courses in the students' records.

(1) Student Vector Construction

In order to quantify the academic profile of a student and compare their similarity to other students, a student vector is constructed based on the courses they have completed, the grades they received, and the terms in which these courses were taken.

Each student's academic record consists of a set of courses they have completed, along with corresponding grades and the terms in which the courses were taken. The goal is to represent this information in a numerical vector format, which can be used for similarity analysis.

Let the set of completed courses for a student S_k be represented as:

$$C_k = \{(c_1, g_1, t_1), (c_2, g_2, t_2), \ldots, (c_n, g_n, t_n)\}$$

where c_i denotes a course, g_i represents the corresponding grade achieved in that course,

and t_i indicates the sequential order in which the course was completed (e.g., 1 for Term 1, 2 for Term 2, ...).

Given a target student, a vector is constructed for each student in the dataset, where each component corresponds to a course from the complete list of courses required for a given major. If a student has not completed a course, the corresponding vector component is set to 0. Otherwise, if the course was completed by the student, the corresponding component in the vector encodes both the normalized grade the student received for that course and the similarity between the term in which the student completed the course and the term in which the target student completed the same course. The final value for each course is computed as the product of the normalized grade and the term similarity weight.

For a given course, if the student has not completed it, the corresponding vector component is set to 0.

Formally, let $C = \{c_1, c_2, \ldots, c_n\}$ be the set of all required courses for the major. For each student S, a vector v_S is constructed such that:

$$v_S = [v_1, v_2, \ldots, v_n]$$

where each component v_i is given by:

$$v_i = \begin{cases} g_i \cdot w_i, & \text{if student } S \text{ has completed course } c_i \\ 0, & \text{otherwise} \end{cases}$$

Here, g_i is the normalized grade for course c_i, computed as:

$$g_i = \frac{\text{grade achieved}}{\text{maximum possible grade}}$$

which ensures that grades are mapped to the range $[0, 1]$.

The term similarity weight w_i is defined as:

$$w_i = \frac{1}{1 + |t_i - t_i^*|}$$

where t_i represents the term in which the student completed course c_i, and t_i^* is the term in which the target student completed the same course. This weight assigns higher values when the course is taken closer to the target term.

By constructing these vectors for all students, we obtain a structured representation that allows for similarity measurements between students based on their course completion sequences and grades.

Example: Consider two students, S_1 and S_2, who have completed the same set of courses in the same sequential order across terms and achieved similar grades. These students are considered highly similar.

Assume that the list of courses required for completion in the given major is restricted to the three courses $\{\text{CSBP119}, \text{CSBP219}, \text{CSBP319}\}$, and both students completed these courses. The completed courses for students S_1 and S_2 are represented as follows:

$$C_1 = \{(\text{CSBP119}, 3, 1), (\text{CSBP219}, 3.5, 2), (\text{CSBP319}, 4, 3)\}$$

$$C_2 = \{(\text{CSBP119}, 3, 1), (\text{CSBP219}, 3.5, 2), (\text{CSBP319}, 4, 3)\}$$

where each tuple represents a course, the grade achieved, and the term in which it was completed.

Now, the student vector for S_1 is computed as follows:

- For the course CSBP119, the grade is 3, normalized to $\frac{3}{4} = 0.75$. If the target student has completed the same course in term 1, the term similarity weight is calculated as:

  $$\frac{1}{1 + |1 - 1|} = 1$$

  Thus, the final value for CSBP119 is:

  $$0.75 * 1 = 0.75$$

- For the course CSBP219, the grade is 3.5, normalized to $\frac{3.5}{4} = 0.875$. If the target student has completed the same course in term 2, the term similarity weight is calculated as:

  $$\frac{1}{1 + |2 - 2|} = 1$$

  Hence, the final value for CSBP219 is:

  $$0.875 * 1 = 0.875$$

- For the course CSBP319, the grade is 4, normalized to $\frac{4}{4} = 1$. If the target student has completed the same course in term 3, the term similarity weight is calculated as:

$$\frac{1}{1+|3-3|} = 1$$

  Therefore, the final value for CSBP319 is:

$$1 * 1 = 1$$

Thus, the student vector for S_1 is:

$$\text{Vector}(S_1) = [0.75, 0.875, 1]$$

Since S_2 completed the same courses in the same order with the same grades, their vector will be identical:

$$\text{Vector}(S_2) = [0.75, 0.875, 1]$$

These student vectors can now be used to identify the k nearest students to the target student.

(2) Term Recommendation

The cosine similarity between the target student's vector and those of other students is then computed to identify the most similar students. For new students, all historical student data is considered, while for continuing students, the search is limited to the top k most similar students.

After identifying the similar students, the recommended term for course C is determined by aggregating the enrollment terms of C across these students. The final suggested term is computed as the average term of enrollment among the similar students, adjusted as follows:

$$T_C = \begin{cases} \lfloor \mu \rfloor, & \text{if } \mu \mod 1 \neq 0.5 \\ \lfloor \mu \rfloor + 1, & \text{otherwise} \end{cases} \tag{3.3}$$

where μ represents the mean term value across similar students. This ensures that term placement is rounded in a consistent manner.

3.6.5 Plan Generation

The final step involves integrating all the relevant data into a single LLM prompt to generate a structured academic plan for the student. The prompt is designed to guide the LLM in constructing a personalized course schedule that ensures timely graduation while adhering to university policies and academic constraints.

The prompt begins by defining the role of the model, stating: *"You are a university course advisor. Generate a full course schedule of X credit hours based on the student's progress."* Here, X represents the total number of credit hours required for the student's major M.

Following this introductory instruction, the prompt provides a detailed summary of the student's academic progress. This includes a list of completed courses along with their corresponding details such as course code, course name, and credit hours. Additionally, the student's total completed credit hours, the remaining credit hours required for graduation, the current GPA, and the term sequence are explicitly stated. The term sequence follows the convention of *Term 1* for new students, while for continuing students, it is represented as *Term (n+1)*, where *n* denotes the last completed term.

Next, the prompt enumerates the courses that are still available for enrollment. These courses are categorized based on their classification within the program, such as mandatory courses, general electives (subdivided into General Elective 1 and General Elective 2), and major electives. Courses that have already been completed are excluded from this list. For each available course, the following details are included:

- The course code and credit hours extracted from the KG.
- The course contact hours (a constant value of 3 hours).
- Prerequisites and corequisites extracted from the KG.
- Courses that are frequently taken with this course, as discussed in Section 3.6.2.
- Similar courses, as outlined in Section 3.6.3.
- The suggested term for enrollment, as determined in Section 3.6.4.

To ensure the generated plan adheres to institutional policies, a set of predefined

constraints is incorporated into the prompt. The following planning rules are included:

- For most semesters, the total credit hours should be between 15 and 16 credit hours.
- Each semester must include at least four courses and no less than 12 credit hours.
- No semester may exceed 18 credit hours.
- A semester should have a maximum of 18 contact hours.
- The academic workload should be balanced, maintaining an average of 15 to 16 credit hours per semester.
- The overall average contact hours should stay around 15 to 18 per semester to maintain a balanced workload.
- Prioritize courses that are prerequisites for many other courses.
- A student with zero completed courses must finish the degree within nine semesters.
- The entire degree plan should be structured to finish within nine semesters (or less), including any semesters the student has already completed. For example, if a student completed one semester, the plan should be finished in eight additional semesters.
- A course must not be scheduled in a semester unless all its prerequisite courses have already been completed in previous semesters.
- A course and its prerequisite(s) cannot be listed in the same semester.
- Corequisite courses must be taken together in the same semester.
- The total credit hours in the plan must equal 130 credit hours to meet graduation requirements.
- The Internship Course (*ITBP495*) must be taken alone in the final semester, with no other courses scheduled.
- The Senior Project Courses (*ITBP480* and *ITBP481*) must be taken in the two semesters immediately preceding the internship semester.
- *ITBP481* must be taken immediately before the internship semester.
- The classification of a course must not be altered; mandatory courses must remain mandatory, and electives must remain within their designated categories.

Finally, to ensure a structured output, the prompt specifies that the generated academic plan should be formatted as a JSON object. This facilitates seamless integration into web-based platforms for visualization. The JSON structure is defined as follows:

```
{
  "Spring 2025": [
    {"id": "CSBP219", "name": "Object Oriented Programming", "credit_hours": 3},
    {"id": "CSBP221", "name": "Programming Lab II", "credit_hours": 1}
  ],
  "Fall 2025": [
    {"id": "CSBP301", "name": "Database Systems", "credit_hours": 3}
  ]
}
```

3.7 Plan Evaluation

To effectively evaluate the resulting academic plan, it is essential to incorporate both qualitative and quantitative measures.

3.7.1 Qualitative Evaluation

To assess the quality of the course recommendation system and its knowledge progression, five domain experts, including academic advisors and professors, will evaluate academic plans for students at different stages in their degree programs. The evaluation includes three academic plans for students at varying levels.

Experts will assess the overall quality of the plan, its alignment with knowledge progression, and the balance of course difficulty and workload. They will also identify any missing considerations that should be incorporated. Their feedback will help refine the recommendation system by highlighting key areas for improvement. Additionally, evaluation scores will measure how well the recommendations align with academic progression and workload distribution, guiding further enhancements to the course planning framework.

3.7.2 Quantitative Evaluation

Along with the qualitative evaluation, a quantitative assessment will be carried out to measure how well the recommended plan matches the university's generic plan on a year-by-year basis. The generic plan will serve as a baseline for comparison, although it is not the final reference.

Another metric involves comparing the generated plans with those actually followed by graduated students. Since real student plans often differ from the generic plan, this comparison gives a more practical evaluation. The process involves generating a plan for a major m using the proposed model, then comparing it with the plans of 30 graduated students. The average similarity score for each year is calculated to assess how closely the generated plans align with actual student plans.

Chapter 4: Experimental Work

This chapter presents the results of the experimental work conducted to evaluate the course recommendation system described in the methodology chapter.

4.1 Knowledge Graph Construction

Let us consider the course CSBP219. The data in Table 4.1 presents information extracted from the syllabus of the Object Oriented Programming course and the generic plans for the CS and IS majors.

Table 4.1: Course Details for Object-Oriented Programming

Course Code	CSBP219
Course Title	Object Oriented Programming
Course Credit Hours	3
Course Type	Mandatory
Course Prerequisites	CSBP119
Course Corequisites	CSBP221
Course Major	CS, IS
Course Description	Object-oriented design, encapsulation and information hiding, separation of behavior and implementation, classes and subclasses, inheritance (overriding, dynamic dispatch), polymorphism (subtype polymorphism vs. inheritance), class hierarchies, collection classes and iteration, Primitive Data Structures and Application (Array, String, and String Manipulation), Programming Practice using an IDE (modularity, testing, and documentation).

Table 4.1 (continued)

Course Outcomes	• Implement classes to solve a given problem. • Test simple classes. • Design classes using existing classes and libraries. • Develop a class hierarchy using inheritance. • Develop classes for simple data structures.
Course Weekly Material	• User Defined Classes, Constructors, object instantiation, Encapsulation • Accessing private members, accessor and mutator methods • toString method • Assignment operator • Deep and shallow copy, copy constructor • . . . . .

Topics must be extracted from three data fields: course description, course outcomes, and course weekly material. The identified topics are presented in Table 4.2.

Table 4.2: Extracted Topics from the Object-Oriented Programming Course

Source	**Topic**
Outcomes	Classes
	Problem-solving
	Testing
	Class Design
	Inheritance
	Data Structures
Description	Encapsulation
	Classes and subclasses
	Inheritance
	Polymorphism
	Class hierarchies
	Arrays
	Strings
	IDE programming practice
	Testing
	Documentation
Weekly Material	Objects & Reference Variables
	File Input/Output
	Classes & Constructors
	Encapsulation
	Deep & Shallow Copy
	UML Diagrams
	Object Composition
	GUI
	Arrays
	ArrayLists
	Inheritance
	Polymorphism
	Abstract Classes & Interfaces

As shown in Figure 4.1, the essential relationships associated with the Object-Oriented Programming course are presented. This course requires the completion of the Algorithms & Problem Solving course as a prerequisite and mandates enrollment in Programming Lab II as a co-course. It is categorized as a level 2 course, indicated by the initial digit in its code, CSBP219. Additionally, it is part of both the Bachelor of Science in Computer Science and Information Systems programs. It is important to note that this figure presents a subset

of the overall graph; other relationships related to this course are not shown. Overall, the nodes presented in the figure are of three types: *Course*, *Major*, and *Level*.

Figure 4.1: Subset of the Graph Including Relationships of CSBP219

To introduce the fourth node type, *Topic*, the extracted topics are integrated into the graph, as they play an effective role in identifying courses that share similar topics, thus indicating course similarity. Figure 4.2 illustrates the topics associated with the course. Each relationship is labeled as *material_topic*, *description_topic*, or *outcome_topic* to indicate the source from which the topic was derived.

As for the last node type, *Student*, the students are added to the graph. Each student is connected by an edge to their enrolled major, zero or more edges to the registered courses, and zero or more edges to the completed courses.

These steps were applied to all 84 courses from the three majors. The resulting graph consists of 1,790 nodes of the following types: *Course*, *Major*, *Level*, *Topic*, and *Student*, with a total of 2,675 relationships.

Figure 4.2: Topics Related to the Course CSBP219 Represented in the KG

For the following sections in this chapter, consider Student S, who is enrolled in the CS program. This student has not yet completed any courses and is currently registered for five courses: *CSBP119* (Algorithms & Problem Solving), *CSBP121* (Programming Lab I), *MATH105* (Calculus I), *PHYS105* (Physics I), and *ARCH366* (History and Theories of Contemporary Architecture).

4.2 Course Embeddings

The embeddings of all courses were generated using Node2Vec with the following parameters:

- **embeddingDimension:** 256

- **walkLength:** 60
- **relationshipWeightProperty:** 'weight'
- **iterations:** 20
- **returnFactor (p):** 1.5
- **inOutFactor (q):** 0.3

During the experimental work, it was observed that running the Node2Vec algorithm without edge weights resulted in plans that did not prioritize low-level courses. To address this, a weight adjustment process was applied to the relationships in the knowledge graph. Initially, all relationship weights were reset to a default value of 1. Then, weights for course-level relationships were updated to reflect their significance, with foundational courses (level 1) assigned the highest weights and higher-level courses progressively lower weights. Additionally, both outgoing and incoming relationships for courses were adjusted based on course levels to emphasize foundational courses and their dependencies. These weights were used as the value for the *relationshipWeightProperty* parameter in Node2Vec.

For instance, the 256-dimensions embedding vector for *CSBP219* is as follows:

$$\text{embedding of CSBP219} = \begin{bmatrix} -0.01998 & 0.4209 & 0.2114 & 0.4082 & -0.0325 \\ -0.0385 & 0.1578 & -0.1598 & -0.0772 & -0.01498 \\ -0.3546 & 0.2379 & -0.2858 & -0.0376 & 0.5557 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

4.3 Plan Generation Using Knowledge Graphs

The first step in this approach involves identifying a ranked list of courses. This list includes all remaining courses in the CS program that the student has not yet completed.

Continuing with the same example, *CSBP219*, we first compute its L2 norm based on its embedding. The computed results are as follows:

- L2 norm for `CSBP219`: 4.2813

- Logarithm of the L2 norm: 1.6642

This logarithmic value serves as the initial score for CSBP219. The score is then adjusted based on the course's level. Since CSBP219 is a level 2 course, its score is boosted slightly less than that of level 1 courses.

Next, CSBP219 is compared to previously completed courses using cosine similarity, which measures the relationship between course embeddings. Similarity scores closer to 1 indicate a stronger relationship. The results of these comparisons are shown below:

- CSBP219 vs ARCH366: Similarity score = -0.0541
- CSBP219 vs PHYS105: Similarity score = 0.2198
- CSBP219 vs CSBP121: Similarity score = 0.8208
- CSBP219 vs MATH105: Similarity score = 0.2221
- CSBP219 vs CSBP119: Similarity score = 0.8169

The results show that CSBP219 is closely related to CSBP121 and CSBP119, as their similarity scores are close to 1. This can be clearly explained by looking into the KG shown in Figure 4.3, where CSBP219 shares common topics with CSBP121 and CSBP119, unlike the other completed courses. The score of CSBP219 is then adjusted by adding its similarity scores with all completed courses.

Since there are no remaining prerequisites for CSBP219 that the student needs to complete, no additional boost is applied to its score. If prerequisites existed, the score of CSBP219 would be further adjusted based on its role as a prerequisite for other incomplete courses. This adjustment ensures that the prerequisites of highly ranked courses receive higher scores, allowing them to be prioritized for completion.

This ranking process is repeated for all courses, producing a prioritized list of courses for the student S. Table 4.3 presents the ranked list, showing the normalized values obtained using Min-Max normalization.

After calculating the scores, the course planning algorithm generates a personalized plan for the student based on their desired maximum credit hours per semester. For example,

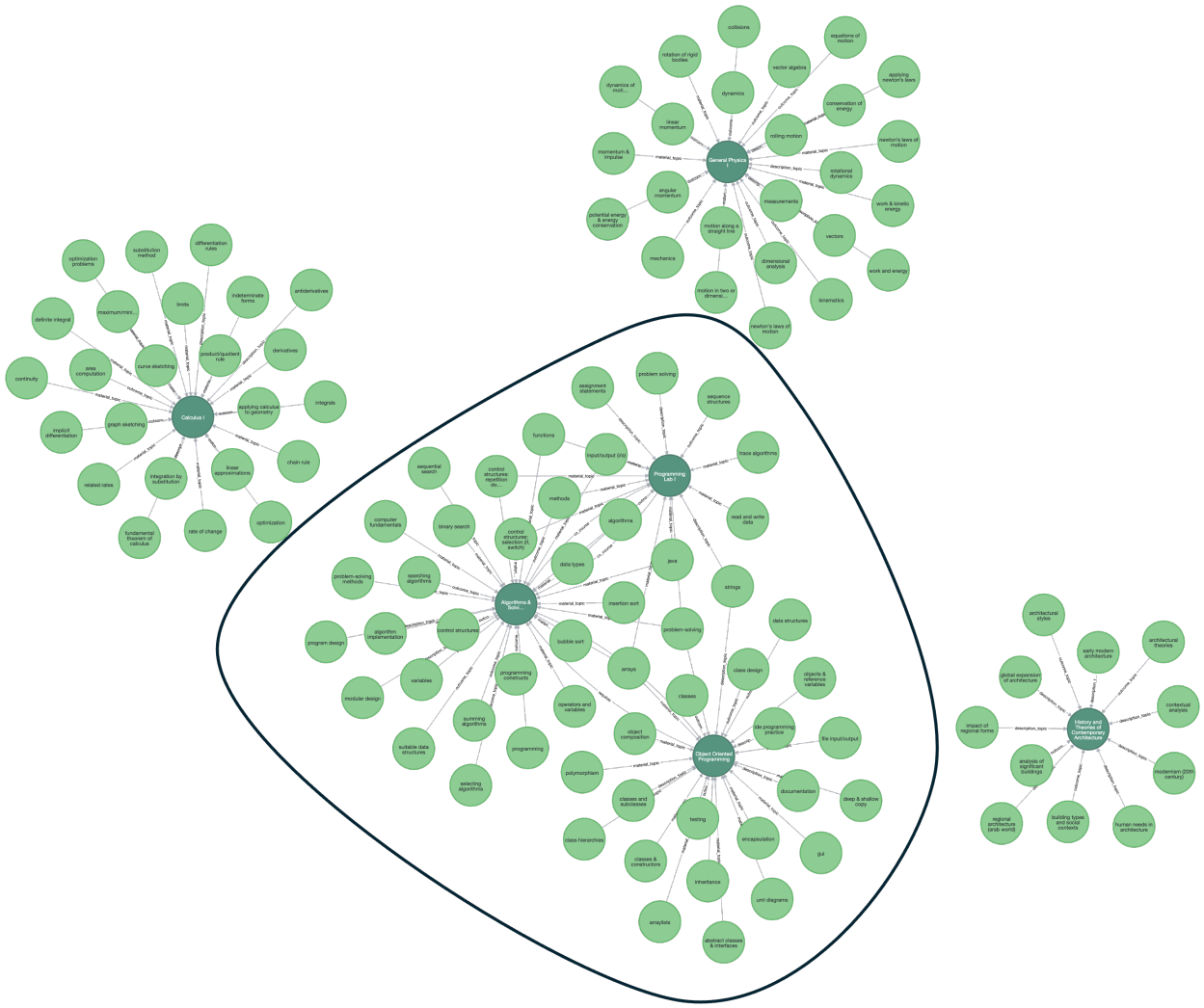


Figure 4.3: Connections Between CSBP219 and Completed Courses

consider student S, who prefers to take a maximum of 15 credit hours per semester.

The generated plan, as illustrated in Figure 4.4, assigns recommended courses to each semester, along with their normalized scores, ensuring that all 130 required credit hours are accounted for. It is important to note that the plan allows for a slight deviation from the maximum credit hour limit, permitting an additional 3 credit hours if co-requisite courses need to be taken together.

Table 4.3: Ranked List of Courses for Student S

Course	**Normalized Score**
MATH110	1.00
ESPU1081	0.95
CSBP219	0.75
PHYS135	0.75
CENG205	0.65
STAT210	0.63
CSBP319	0.55
CENG210	0.44
...	...
CENG202	0.37
CSBP340	0.33
...	...

Spring 2025

Score	Name	Hrs
1.00	MATH110-Calculus II	3
0.95	ESPU1081 - Intro to Academic English for IT 1	3
0.29	ECON110-Principles of Economics	3
0.29	CHEM111-General Chemistry 1	3
0.75	CSBP219-Object Oriented Programming	3
0.42	CSBP221 - Programming Lab II	1

Term Credit Hours: 16

Fall 2025

Score	Name	Hrs
0.75	PHYS135-General Physics Lab I	1
0.29	MATH140 - Linear Algebra I	3
0.65	CENG205 - Digital Design & Computer Org.	3
0.63	STAT210-Probability and Statistics	3
0.29	ISLM100 - Islamic Culture	3

Term Credit Hours: 13

Spring 2026

Score	Name	Hrs
0.29	GEIT112 - Fourth Industrial Revolution	3
0.55	CSBP319 - Data Structures	3
0.44	CENG210-Communication & Networks Fund.	3
0.06	CSBP400 - Modeling & Simulation	3
0.15	CSBP320 - Data Mining	3

Term Credit Hours: 15

Fall 2026

Score	Name	Hrs
0.18	ITBP301-Security Principles & Practice	3
0.41	CSBP301 - Artificial Intelligence	3
0.15	CSBP315-Operating Systems Fund.	3
0.37	CENG202 - Discrete Mathematics	3
0.33	CSBP340 - Database Systems	3

Term Credit Hours: 15

Spring 2027

Score	Name	Hrs
0.06	CSBP483 - Mobile Web Content & Development	3
0.15	CSBP316 - Human Computer Interaction	3
0.15	SWEB300-Software Engineering Fundamentals	3
0.06	CSBP421-Smart Computer Graphics	3
0.29	HSS105 - Emirates Studies	3

Term Credit Hours: 15

Fall 2027

Score	Name	Hrs
0.15	ITBP370-Professional Responsibility in IT	3
0.14	ITBP321 - Web Application Development Lab	1
0.06	CSBP461 - Internet Computing	3
0.29	GESU121-Sustainability	3
0.06	CSBP411-Machine Learning	3
0.10	ITBP480 - Senior Graduation Project I	3

Term Credit Hours: 16

Spring 2028

Score	Name	Hrs
0.10	SWEB450 - Analysis of Algorithms	3
0.06	ITBP418-Entrepreneurship in IT	3
0.06	SWEB451-Game Development	3
0.06	CSBP476-Robotics and Intelligent Systems	3
0.09	ITBP481 - Senior Graduation Project II	3

Term Credit Hours: 15

Last Semester

Score	Name	Hrs
0.05	ITBP495 - Internship	12

Term Credit Hours: 12

Total Credit Hours: 130

Figure 4.4: Academic Plan for Student S Using the First Approach

4.4 Plan Generation Using ML, KG, and LLM

This section presents the plan generation approach based on data-driven insights extracted from historical enrollment data of the CS student S.

4.4.1 Frequently Taken Together Courses

Transaction Generation

The first step involves transforming the historical course enrollment records of CS alumni students into a transaction dataset. Each student's completed courses per academic term are treated as individual transactions. Table 4.4 illustrates an example of a student's completed courses distributed over four academic terms. This process resulted in a total of 1003 transactions, each representing a unique academic term record.

Table 4.4: Example of a Student's Completed Courses Across Terms

Term	**Completed Courses**
1	{CSBP121, ISLM100, CHEM111, MATH105, CSBP119}
2	{CSBP221, CSBP219, ESPU1081, PHYS105}
3	{CENG202, CENG205, CSBP319, CENG210, MATH110}
4	{SWEB300, CSBP421, STAT210, CSBP340, CSBP301}

Frequent Itemset Mining

To extract patterns of commonly grouped courses, the Apriori algorithm was applied to the transaction dataset. The analysis produced 75 frequent itemsets, indicating sets of courses that consistently appeared together across different students' terms. A sample of these frequent itemsets is presented in Table 4.5.These frequent itemsets reveal strong academic relationships between certain courses, suggesting natural pairing or sequencing tendencies among students.

Grade-Weighted Co-Occurrence Matrix

To complement the support-based analysis, a grade-weighted co-occurrence matrix was developed. This matrix not only captures how frequently courses are taken together but also incorporates student academic performance, providing a more nuanced measure of

Table 4.5: Frequent Course Itemsets Identified by Apriori Algorithm

Itemset	Course Combination	Support (Number of Transactions)
1	CSBP219, CSBP221	74
2	CSBP119, CSBP121	71
3	CSBP421, SWEB450	53
4	CSBP301, CSBP340	44
5	CSBP119, CSBP121, ESPU1081	35

the strength of these course pairings. Higher scores indicate frequent pairing coupled with strong student performance. Table 4.6 presents selected course pairs along with their grade-weighted co-occurrence scores. These scores help identify course combinations where students tend to perform well when taken together.

Table 4.6: Grade-Weighted Co-occurrence Scores for Selected Course Pairs

Course Pairing	Co-occurrence Score
CSBP219, CSBP221	274.55
CSBP119, CSBP121	254.65
CSBP421, SWEB450	187.95
CSBP301, CSBP340	167.25

The matrix visualization in Figure 4.5 further illustrates the relationships between course pairs. Darker or redder cells represent stronger grade-weighted co-occurrence, highlighting pairs that are not only frequently enrolled together but also associated with better academic outcomes.

Hybrid Score Calculation

The final stage of the analysis involved computing a hybrid score for each course pair or group by combining the support from the frequent itemsets and the grade-weighted co-occurrence score. A weighting factor (α) of 0.8 was used, assigning 80% importance to frequent itemset support and 20% to the co-occurrence score.

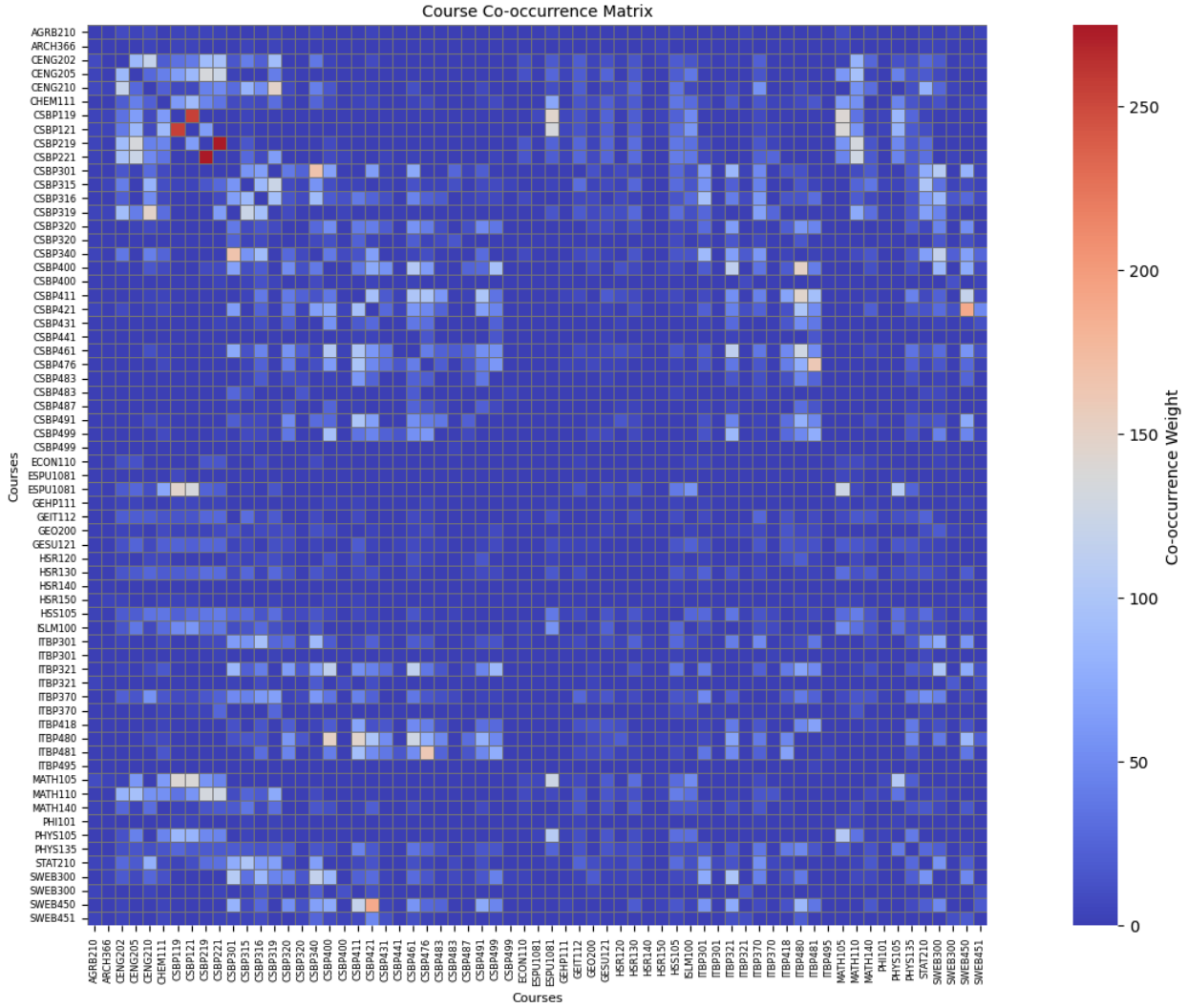


Figure 4.5: Heatmap of the Grade-Weighted Co-Occurrence Matrix for CS Courses

Table 4.7: Course Pairs with Support, Co-occurrence, and Hybrid Score

Itemset	Course Pair	Support	Avg. Co-occurrence	Hybrid Score
74	(CSBP219, CSBP221)	74	274.55	114.11
71	(CSBP119, CSBP121)	71	254.65	107.73
53	(CSBP421, SWEB450)	53	187.95	79.99
44	(CSBP301, CSBP340)	44	167.25	68.65
42	(CSBP476, ITBP481)	42	160.35	65.67
35	(CSBP119, CSBP121, ESPU1081)	35	178.37	63.67
34	(CSBP119, CSBP121, MATH105)	34	178.28	62.86

The pair *(CSBP219 - Object-Oriented Programming, CSBP221 - Programming Lab II)* exhibits the highest hybrid score of 114.11, indicating a strong correlation between frequent enrollment and high student performance. The co-occurrence value of 274.55

further suggests that students who take both courses tend to perform well. Similarly, the pair *(CSBP119 - Algorithms and Problem Solving, CSBP121 - Programming Lab I)* also demonstrates a high hybrid score of 107.73, highlighting it as a common course combination associated with academic success.

Some course pairs with lower support, such as *(CSBP476 - Robotics & Intelligent Systems, ITBP481 - Senior Graduation Project 2)*, still achieve competitive hybrid scores due to strong performance despite fewer students enrolling in these courses. On the other hand, pairs like *(CSBP421 - Smart Computer Graphics, SWEB450 - Analysis of Algorithms)* have a support value of 53, but their hybrid score is lower (79.99), indicating that while these courses are often taken together, the academic performance is not as strong compared to other pairs.

Overall, the hybrid score balances course frequency with academic performance, so pairs with lower support but high performance, like *(CSBP301 - Artificial Intelligence, CSBP340 - Database System)*, may still rank lower due to weaker academic outcomes despite being taken together frequently.

4.4.2 Similar Courses

To identify similar courses within the knowledge graph, cosine similarity was applied to the course embeddings. A similarity threshold of 0.7 was selected to filter out weak relationships, ensuring that only course pairs with a significant degree of similarity were retained. For each course, the most similar courses were then consolidated into a recommendation list based on this similarity measure.

The results revealed several meaningful relationships between courses. For example, *CSBP219 (Object-Oriented Programming)* was highly similar to *CENG205*, *MATH110*, *CENG202*, and *CSBP221*, indicating strong connections based on shared content or prerequisite knowledge. Similarly, *CSBP119 (Algorithms and Problem Solving)* was found to be closely related to courses such as *ESPU1081*, *MATH105*, *CSBP121*, and *PHYS105*.

Advanced courses also showed relevant similarities. For instance, *CSBP421 (Smart*

Computer Graphics) was closely linked to *SWEB450* and *ITBP480*. Likewise, *CSBP301 (Artificial Intelligence)* showed strong similarity to *CSBP340*, *STAT210*, and *ITBP321*.

These results demonstrate that the embedding-based similarity approach effectively captures meaningful relationships between courses.

4.4.3 Suggested Term

To determine the suggested term for each course, the K-Nearest Neighbors (KNN) algorithm was applied. In this method, each student is compared to their k most similar students based on vector similarity. For the purpose of this example, the target student S was compared to their 15 nearest neighbors ($k = 15$).

Student Vector Construction

The vector representing student S was constructed based on the courses they have completed, the grades they received, and the terms in which they completed these courses. Table 4.8 shows a sample of the course details for Term 1 of student S.

Table 4.8: Course Grades for Term 1 of Student S

Course	**Grade**	**Term**
ARCH366	B	1
PHYS105	A	1
CSBP121	A	1
MATH105	A	1
CSBP119	A	1

Each course is assigned a normalized grade weight based on the following scale:

$$\text{Grade Weights} = \left\{ \begin{array}{l} \text{A} : 4,\ \text{A-} : 3.7,\ \text{B+} : 3.3,\ \text{B} : 3,\ \text{B-} : 2.7, \\ \\ \text{C+} : 2.3,\ \text{C} : 2,\ \text{C-} : 1.7,\ \text{D+} : 1.2,\ \text{D} : 1, \\ \\ \text{P} : 1,\ \text{Others} : 0 \end{array} \right\}$$

Normalized by dividing by 4, the values become:

$$\text{Normalized Grade} = \left\{ \frac{3}{4}, \frac{4}{4}, \frac{4}{4}, \frac{4}{4}, \frac{4}{4} \right\}$$

Since all courses were taken in the same term as the target, the term weight is 1 for all:

$$\text{Term Weight} = \frac{1}{1 + |\text{term}_{\text{student}} - \text{term}_{\text{target}}|} = 1$$

Thus, the vector for Student S is populated as:

$$\text{Vector} = [0.75, 1.0, 1.0, 1.0, 1.0, 0, 0, \ldots, 0]$$

where values correspond to the courses completed, and zero elsewhere.

The vectors of all other students are computed similarly. Cosine similarity is calculated between Student S and each student. After calculation, the top 15 most similar students are selected based on the highest similarity scores, as shown in Figure 4.6.

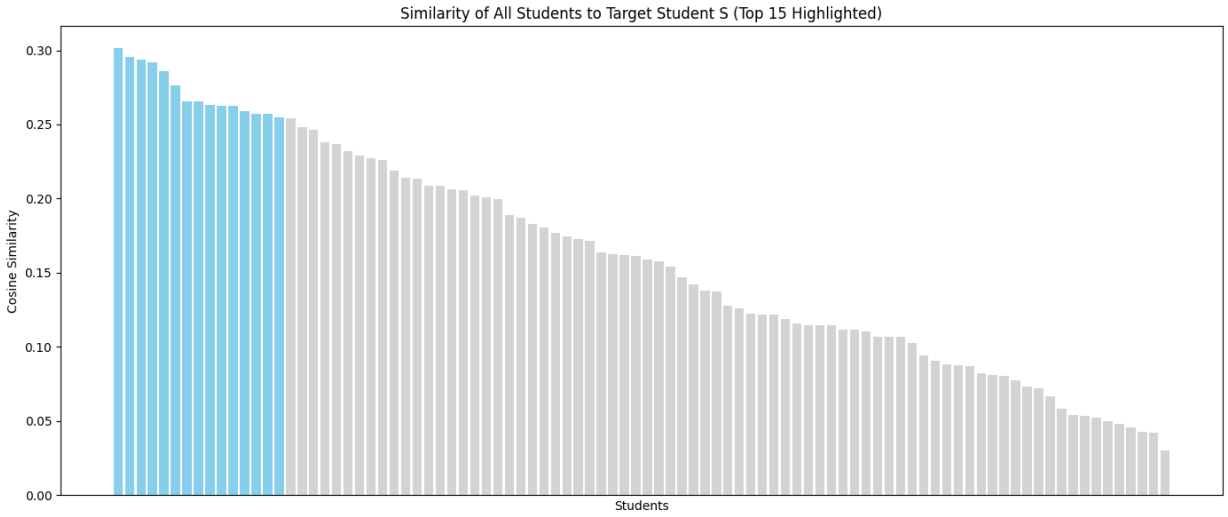


Figure 4.6: Similarity of All Students to Target Student

From the top 15 similar students, we extracted courses not taken by S. For each such course, we aggregated the terms in which similar students completed it. Table 4.9 shows examples of courses along with the terms in which these students completed each course.

Table 4.9: Example of Terms Distribution for Nearest 15 Students

Course	**Terms (per student)**
ITBP321 - Web App Development Lab	7, 7, 9, 7, 6, 8, 5, 8, 10, 6, 11, 5, 8, 10, 7
CSBP421 - Smart Computer Graphics	11, 5, 9, 10, 7, 9, 7, 9, 8, 6, 7, 7, 9, 8, 4
ITBP480 - Senior Graduation Project 1	10, 8, 9, 9, 9, 9, 8, 8, 8, 7, 10, 8, 9, 10, 9
CSBP411 - Machine Learning	9, 8, 8, 8, 7, 8, 10, 7, 8, 8, 7, 7, 8, 11, 10
STAT210 - Probability and Statistics	4, 4, 4, 4, 6, 5, 7, 4, 5, 4, 3, 3, 5, 7, 4

The terms represent the semester number in which each student completed the corresponding course. For each course, the average term was computed by summing the terms in which each of the 15 students completed the course and dividing by 15. For example:

- **ITBP321 - Web App Development Lab**: Average term = 8
- **CSBP421 - Smart Computer Graphics**: Average term = 8
- **ITBP480 - Senior Graduation Project 1**: Average term = 9
- **CSBP411 - Machine Learning**: Average term = 8
- **STAT210 - Probability and Statistics**: Average term = 5

It is important to highlight that the recommended term values are approximate. This is because our plan generation process does not account for the summer semester. In real student records, term numbering may include summer terms, where a student could have *Term 1* as the first Fall, *Term 2* as the first Spring, and *Term 3* as the first Summer. However, in our generated plans, we consider only the Fall and Spring semesters. The sequence is adjusted such that *Term 1* represents the first Fall (or Spring), *Term 2* the first Spring (or Fall), and *Term 3* the second Fall (or Spring), and so on. Therefore, the recommended term serves as an approximation based on the average completion time of previous students and might slightly differ from the actual sequence due to the exclusion of summer semesters.

4.4.4 Plan Generation

At the end, after consolidating all the gathered data, a prompt for the LLM is generated. It includes general information about the student and the remaining courses, categorized by their types. Each course is detailed with its code, description, credit hours, contact hours, prerequisites, corequisites, most frequently taken with, similar courses, and suggested term. Additionally, the general constraints and rules are provided. Then, using the Anthropic LLM *Claude 3.5 Sonnet*, the generated plan for the student would be as presented in Figure 4.7.

Prompt: University Course Advisor

Student Data

- Completed Courses: ARCH366, PHYS105, CSBP121, MATH105, CSBP119
- Completed Credit Hours: 13 credit hours
- Remaining Required Credit Hours: 117 credit hours
- GPA: 3.5
- Current Term Sequence: 2

Available Courses (Grouped by Category, Excluding Completed Courses)

Mandatory Courses (Select all of them)

- ESPU1081 (Intro to Academic English for IT 1): 3 credit hours, 3 contact hours. Prerequisites: None, Corequisites: None. Mostly taken with: CHEM111, CSBP121, CSBP119, PHYS105, MATH105. Similar Courses: ITBP370, ISEC416. Suggested Term: 1
- MATH110 (Calculus II): 3 credit hours, 3 contact hours. Prerequisites: MATH105, Corequisites: None. Mostly taken with: CSBP319, CENG205, CENG202, CSBP219, CSBP221. Similar Courses: MATH105, STAT210. Suggested Term: 2
- CSBP219 (Object Oriented Programming): 3 credit hours, 3 contact hours. Prerequisites: CSBP119, Corequisites: CSBP221. Mostly taken with: CENG205, CENG202, MATH110, CSBP221. Similar Courses: CSBP119, CSBP121. Suggested Term: 2
- CSBP221 (Programming Lab II): 1 credit hour, 3 contact hours. Prerequisites: None, Corequisites: CSBP219. Mostly taken with: CENG205, CENG202, MATH110, CSBP219. Similar Courses: N/A. Suggested Term: 2
- . . .

Major Electives (Select EXACTLY 4 for the entire plan)

- CSBP491 (Computational Intelligence for Data Management): 3 credit hours, 3 contact hours. Prerequisites: CSBP301, CSBP319, Corequisites: None. Mostly taken with: CSBP411. Similar Courses: CSBP476, CSBP487, CSBP441, CSBP477. Suggested Term: 7
- CSBP441 (Applied Computer Vision): 3 credit hours, 3 contact hours. Prerequisites: CSBP301, Corequisites: None. Mostly taken with: Not Enough Information. Similar Courses: CSBP411, CSBP421, CSBP476, CSBP487, CSBP491, CSBP477. Suggested Term: 8
- . . .

General Elective 1 (Select EXACTLY 1 for the entire plan)

- AGRB210 (Introduction to Agribusiness): 3 credit hours, 3 contact hours. Prerequisites: None, Corequisites: None. Mostly taken with: Not Enough Information. Similar Courses: N/A. Suggested Term: 2
- . . . (other courses)

General Constraints and Rules

- For most semesters, the total credit hours should be between 15 and 16 credit hours.
- . . .

Spring 2025

Index	Name	Hrs
1	ESPU1081 - Intro to Academic English for IT 1	3
2	MATH110 - Calculus II	3
3	CSBP219-Object Oriented Programming	3
4	CSBP221 - Programming Lab II	1
5	CHEM111-General Chemistry 1	3
6	ISLM100 - Islamic Culture	3

Term Credit Hours: 16

Fall 2025

Index	Name	Hrs
1	CENG205 - Digital Design & Computer Org.	3
2	CENG202 - Discrete Mathematics	3
3	GEIT112 - Fourth Industrial Revolution	3
4	HSS105 - Emirates Studies	3
5	ECON110-Principles of Economics	3

Term Credit Hours: 15

Spring 2026

Index	Name	Hrs
1	CSBP319 - Data Structures	3
2	CENG210 - Communication & Networks Fund.	3
3	MATH140-Linear Algebra I	3
4	STAT210-Probability and Statistics	3
5	GESU121-Sustainability	3

Term Credit Hours: 15

Fall 2026

Index	Name	Hrs
1	CSBP315-Operating Systems Fund.	3
2	CSBP340 - Database Systems	3
3	ITBP301 - Security Principles & Practice	3
4	CSBP316-Human Computer Interaction	3
5	PHYS135-General Physics Lab I	1
6	CSBP301 - Artificial Intelligence	3

Term Credit Hours: 16

Spring 2027

Index	Name	Hrs
1	ITBP370 - Professional Responsibility in IT	3
2	SWEB300 - Software Engineering Fundamentals	3
3	SWEBA50 - Analysis of Algorithms	3
4	CSBP491- Computational Intelligence for Data Management	3
5	CSBP441 - Applied Computer Vision	3

Term Credit Hours: 15

Fall 2027

Index	Name	Hrs
1	CSBP400-Modeling & Simulation	3
2	CSBP461-Internet Computing	3
3	ITBP321 - Web Application Development Lab	1
4	CSBP411-Machine Learning	3
5	CSBP421-Smart Computer Graphics	3
6	ITBP480 - Senior Graduation Project I	3

Term Credit Hours: 16

Spring 2028

Index	Name	Hrs
1	ITBP418 - Entrepreneurship in IT	3
2	ITBP481 - Senior Graduation Project II	3
3	CSBP483 - Mobile Web Content & Development	3
4	SWEB451 - Game Development	3

Term Credit Hours: 12

Fall 2028

Index	Name	Hrs
1	ITBP495- Internship	12

Term Credit Hours: 12

Total Credit Hours: 130

Figure 4.7: Academic Plan for Student S Using the Second Approach

Chapter 5: Results and Evaluation

Evaluating a course plan recommendation system is not a straightforward task because there are no standard or universally accepted measures to define what makes a "good" course plan. The quality of a plan can vary depending on several factors, such as the student's academic progress, personal preferences, and future goals. Some students may prefer a lighter workload, while others might want to finish early or prioritize certain subjects. Additionally, many courses can be taken in different semesters without affecting the student's graduation timeline. This flexibility makes it difficult to judge if one plan is truly better than another.

5.1 Case Studies

To effectively evaluate the system, we examined seven real student cases across three majors: CS, IS, and IT. Most of the cases focused on CS students. The aim of selecting a diverse set of cases was to assess the recommendations under different academic situations. For example, a newly admitted student receives a basic plan, while students who have progressed in their degrees receive tailored plans based on their completed and registered courses.

Table 5.1 summarizes the details of the seven student cases, including each student's academic status, completed courses, and currently registered courses. These cases cover students at various academic stages, including newly admitted students, those in their first year, students halfway through their studies, and those in their final year.

For each case, the course plans generated using both methods: (1) planning based on KG and (2) the enhanced planning approach based on ML, KG, and LLM are presented together to allow for a clear comparison.

Table 5.1: Summary of Student Cases

Case	Student Description	Courses Taken (Completed + Registered)	Figures
1	New CS Student	None	Figure 5.1a, Figure 5.1b
2	First-Year CS Student	ARCH366, PHYS105, CSBP121, MATH105, CSBP119	Figure 5.2a, Figure 5.2b
3	2nd-3rd Year CS Student	GESU121, ENGL101, MATH105, PHYS105, CSBP119, CSBP121, CSBP222, CSBP215, CSBP214, CSBP213, STAT201, MATH202, CSBP216, CSBP311, CSBP319, CENG210, ENGL251, ITBP211, ITBP212, CSBP314, ITBP370, CSBP316, CSBP319, ITBP301	Figure 5.3a, Figure 5.3b
4	3rd Year CS Student	GESU121, ENGL101, MATH105, PHYS105, CSBP119, CSBP121, CSBP222, CSBP215, CSBP214, CSBP213, STAT201, MATH202, CSBP216, CSBP311, CSBP319, CENG210, ENGL251, ITBP211, ITBP212, CSBP314, GEIT112, GEO200, CENG202, CENG205	Figure 5.4a, Figure 5.4b
5	4th Year CS Student	GESU121, ENGL101, MATH105, PHYS105, CSBP119, CSBP121, CSBP222, CSBP215, CSBP214, CSBP213, STAT201, MATH202, CSBP216, CSBP311, CSBP319, CENG210, ENGL251, ITBP211, ITBP212, CSBP314, GEIT112, GEO200, CENG202, CENG205, ITBP370, CSBP316, CSBP461, ITBP301, CENG210, CSBP421, ITBP321	Figure 5.5a, Figure 5.5b
6	New IS Student	None	Figure 5.6a, Figure 5.6b
7	New IT Student	None	Figure 5.7a, Figure 5.7b

Spring 2025

Score	Name	Hrs
1.00	PHYS105-General Physics I	3
0.82	MATH105-Calculus I	3
0.58	CSBP119-Algorithms & Problem Solving	3
0.35	CSBP121-Programming Lab I	1
0.51	ESPU1081 - Intro to Academic English for IT 1	3
0.41	PHI101 - Introduction to Philosophy	3

Term Credit Hours: 16

Fall 2025

Score	Name	Hrs
0.87	CSBP219- Object Oriented Programming	3
0.27	CSBP221-Programming Lab III	1
0.59	CENG205 - Digital Design & Computer Org.	3
0.58	MATH110-Calculus II	3
0.41	PSYC100-Introduction to Psychology	3
0.40	PHYS135-General Physics Labl	1

Term Credit Hours: 14

Spring 2026

Score	↓ Name	Hrs
0.70	STAT210 - Probability and Statistics	3
0.38	ISLM100 - Islamic Culture	3
0.39	HSS105-Emirates Studies	3
0.87	CSBP319 - Data Structures	3
0.37	CHEM111 - General Chemistry 1	3

Term Credit Hours: 15

Fall 2026

Score	Name	Hrs
0.72	CSBP301 - Artificial Intelligence	3
0.41	CSBP340 - Database Systems	3
0.37	MATH140-Linear Algebra I	3
0.34	GEIT112 - Fourth Industrial Revolution	3
0.34	CENG210 - Communication & Networks Fund.	3

Term Credit Hours: 15

Spring 2027

Score	Name	Hrs
0.28	CENG202 - Discrete Mathematics	3
0.27	GESU121-Sustainability	3
0.21	ITBP301 - Security Principles & Practice	3
0.20	CSBP316-Human Computer Interaction	3
0.19	CSBP315-Operating Systems Fund.	3
0.12	ITBP321 - Web Application Development Lab	1

Term Credit Hours: 16

Fall 2027

Score	Name	Hrs
0.19	CSBP320 - Data Mining	3
0.18	ITBP370-Professional Responsibility in IT	3
0.18	SWEB300 - Software Engineering Fundamentals	3
0.17	SWEB450-Analysis of Algorithms	3
0.09	CSBP476 - Robotics and Intelligent Systems	3

Term Credit Hours: 15

Spring 2028

Score	Name	Hrs
0.09	SWEB451-Game Development	3
0.09	CSBP483-Mobile Web Content & Development	3
0.08	ITBP418-Entrepreneurship in IT	3
0.08	CSBP411-Machine Learning	3
0.14	ITBP480 - Senior Graduation Project I	3

Term Credit Hours: 15

Fall 2028

Score	Name	Hrs
0.07	CSBP400-Modeling & Simulation	3
0.05	CSBP421-Smart Computer Graphics	3
0.05	CSBP461 - Internet Computing	3
0.12	ITBP481 - Senior Graduation Project II	3

Term Credit Hours: 12

Last Semester

Score	Name	Hrs
0.04	ITBP495-Internship	12

Term Credit Hours: 12

Total Credit Hours: 130

(a) KG-based course plan

Spring 2025

Index	Name	Hrs
1	MATH105-Calculus I	3
2	CSBP119 - Algorithms & Problem Solving	3
3	CSBP121-Programming Lab I	1
4	ESPU1081-Intro to Academic English for IT 1	3
5	PHYS105-General Physics I	3
6	ISLM100 - Islamic Culture	3

Term Credit Hours: 16

Fall 2025

Index	Name	Hrs
1	CSBP219 - Object Oriented Programming	3
2	CSBP221-Programming Lab II	1
3	CENG205 - Digital Design & Computer Org.	3
4	MATH110-Calculus II	3
5	CENG202 - Discrete Mathematics	3
6	PH1101-Introduction to Philosophy	3

Term Credit Hours: 14

Spring 2026

Index	Name	Hrs
1	CSBP319 - Data Structures	3
2	MATH140 - Linear Algebra I	3
3	CENG210 - Communication & Networks Fund.	3
4	CHEM111-General Chemistry 1	3
5	ECON110 - Principles of Economics	3

Term Credit Hours: 15

Fall 2026

Index	Name	Hrs
1	STAT210-Probability and Statistics	3
2	CSBP315-Operating Systems Fund.	3
3	CSBP340 - Database Systems	3
4	SWEB300 - Software Engineering fundamentals	3
5	HSS105 - Emirates Studies	3
6	Phrs135-General Physics Lab I	1

Term Credit Hours: 16

Spring 2027

Index	Name	Hrs
1	ITBP301-Security Principles & Practice	3
2	CSBP316 - Human Computer Interaction	3
3	CSBP301-Artificial Intelligence	3
4	ITBP370 - Professional Responsibility in IT	3
5	GESU121-Sustainability	3

Term Credit Hours: 15

Fall 2027

Index		Name	Hrs
	1	CSBP400 - Modeling & Simulation	3
	2	SWEB450 - Analysis of Algorithms	3
	3	CSBP421 - Smart Computer Graphics	3
	4	SWEB451-Game Development	3
	5	GEIT112 - Fourth Industrial Revolution	3

Term Credit Hours: 15

Spring 2028

Index	Name	Hrs
1	TPPP418 - Entrepreneurship in IT	3
2	CSBP461 - Internet Computing	3
3	TBP321 - Web Application Development Lab	1
4	CSBP411 - Machine Leaming	3
5	ITBP430-Senior Graduation Project I	3
6	CSBP441 - Applied Computer Vision	3

Term Credit Hours: 16

Fall 2028

Index	Name	Hrs
1	ITBP481- Senior Graduation Project II	3
2	CSBP476-Robotics and Intelligent Systems	3
3	CSBP487 - Computer Animation and Visualization	3

Term Credit Hours: 9

Spring 2029

Index	Name	Hrs
1	ITBP495- Internship	12

Term Credit Hours: 12

Total Credit Hours: 130

(b) (ML, KG, LLM)-enhanced course plan

Figure 5.1: Generated Plans for Student 1

Spring 2025

Score	Name	Hrs
1.00	MATH110-Calculus II	3
0.95	ESPU1081 - Intro to Academic English for IT 1	3
0.29	ECON110-Principles of Economics	3
0.29	CHEM111-General Chemistry 1	3
0.75	CSBP219-Object Oriented Programming	3
0.42	CSBP221 - Programming Lab II	1

Term Credit Hours: 16

Fall 2025

Score	Name	Hrs
0.75	PHYS135-General Physics Lab I	1
0.29	MATH140 - Linear Algebra I	3
0.65	CENG205 - Digital Design & Computer Org.	3
0.63	STAT210-Probability and Statistics	3
0.29	ISLM100 - Islamic Culture	3

Term Credit Hours: 13

Spring 2026

Score	Name	Hrs
0.29	GEIT112 - Fourth Industrial Revolution	3
0.55	CSBP319 - Data Structures	3
0.44	CENG210-Communication & Networks Fund.	3
0.06	CSBP400 - Modeling & Simulation	3
0.15	CSBP320 - Data Mining	3

Term Credit Hours: 15

Fall 2026

Score	Name	Hrs
0.18	ITBP301-Security Principles & Practice	3
0.41	CSBP301 - Artificial Intelligence	3
0.15	CSBP315-Operating Systems Fund.	3
0.37	CENG202 - Discrete Mathematics	3
0.33	CSBP340 - Database Systems	3

Term Credit Hours: 15

Spring 2027

Score	Name	Hrs
0.06	CSBP483 - Mobile Web Content & Development	3
0.15	CSBP316 - Human Computer Interaction	3
0.15	SWEB300-Software Engineering Fundamentals	3
0.06	CSBP421-Smart Computer Graphics	3
0.29	HSS105 - Emirates Studies	3

Term Credit Hours: 15

Fall 2027

Score	Name	Hrs
0.15	ITBP370-Professional Responsibility in IT	3
0.14	ITBP321 - Web Application Development Lab	1
0.06	CSBP461 - Internet Computing	3
0.29	GESU121-Sustainability	3
0.06	CSBP411-Machine Learning	3
0.10	ITBP480 - Senior Graduation Project I	3

Term Credit Hours: 16

Spring 2028

Score	Name	Hrs
0.10	SWEB450 - Analysis of Algorithms	3
0.06	ITBP418-Entrepreneurship in IT	3
0.06	SWEB451-Game Development	3
0.06	CSBP476-Robotics and Intelligent Systems	3
0.09	ITBP481 - Senior Graduation Project II	3

Term Credit Hours: 15

Last Semester

Score	Name	Hrs
0.05	ITBP495 - Internship	12

Term Credit Hours: 12

Total Credit Hours: 130

(a) KG-based course plan

Spring 2025

Index	Name	Hrs
1	ESPU1081 - Intro to Academic English for IT 1	3
2	MATH110 - Calculus II	3
3	CSBP219-Object Oriented Programming	3
4	CSBP221 - Programming Lab II	1
5	CHEM111-General Chemistry 1	3
6	ISLM100 - Islamic Culture	3

Term Credit Hours: 16

Fall 2025

Index	Name	Hrs
1	CENG205 - Digital Design & Computer Org.	3
2	CENG202 - Discrete Mathematics	3
3	GEIT112 - Fourth Industrial Revolution	3
4	HSS105 - Emirates Studies	3
5	ECON110-Principles of Economics	3

Term Credit Hours: 15

Spring 2026

Index	Name	Hrs
1	CSBP319 - Data Structures	3
2	CENG210 - Communication & Networks Fund.	3
3	MATH140-Linear Algebra I	3
4	STAT210-Probability and Statistics	3
5	GESU121-Sustainability	3

Term Credit Hours: 15

Fall 2026

Index	Name	Hrs
1	CSBP315-Operating Systems Fund.	3
2	CSBP340 - Database Systems	3
3	ITBP301 - Security Principles & Practice	3
4	CSBP316-Human Computer Interaction	3
5	PHYS135-General Physics Lab I	1
6	CSBP301 - Artificial Intelligence	3

Term Credit Hours: 16

Spring 2027

Index	Name	Hrs
1	ITBP370 - Professional Responsibility in IT	3
2	SWEB300 - Software Engineering Fundamentals	3
3	SWEBA50 - Analysis of Algorithms	3
4	CSBP491- Computational Intelligence for Data Management	3
5	CSBP441 - Applied Computer Vision	3

Term Credit Hours: 15

Fall 2027

Index	Name	Hrs
1	CSBP400-Modeling & Simulation	3
2	CSBP461-Internet Computing	3
3	ITBP321 - Web Application Development Lab	1
4	CSBP411-Machine Learning	3
5	CSBP421-Smart Computer Graphics	3
6	ITBP480 - Senior Graduation Project I	3

Term Credit Hours: 16

Spring 2028

Index	Name	Hrs
1	ITBP418 - Entrepreneurship in IT	3
2	ITBP481 - Senior Graduation Project II	3
3	CSBP483 - Mobile Web Content & Development	3
4	SWEB451 - Game Development	3

Term Credit Hours: 12

Fall 2028

Index	Name	Hrs
1	ITBP495- Internship	12

Term Credit Hours: 12

Total Credit Hours: 130

(b) (ML, KG, LLM)-enhanced course plan

Figure 5.2: Generated Plans for Student 2

Spring 2025

Score	Name	Hrs
1.00	CSBP301 - Artificial Intelligence	3
0.80	CSBP340 - Database Systems	3
0.31	AGRB210- Introduction to Agribusiness	3
0.14	ITBP418 - Entrepreneurship in IT	3
0.25	SWEB450 - Analysis of Algorithms	3

Term Credit Hours: 15

Fall 2025

Score	Name	Hrs
0.14	SWEB451-Game Development	3
0.14	CSBP400 - Modeling & Simulation	3
0.14	CSBP476-Robotics and Intelligent Systems	3
0.14	CSBP461-Internet Computing	3
0.35	ITBP321 - Web Application Development Lab	1

Term Credit Hours: 13

Spring 2026

Score	Name	Hrs
0.14	CSBP483-Mobile Web Content & Development	3
0.13	CSBP441 - Applied Computer Vision	3
0.36	SWEB300 - Software Engineering Fundamentals	3
0.14	CSBP411-Machine Learning	3
0.24	ITBP480-Senior Graduation Project I	3

Term Credit Hours: 15

Fall 2026

Score	Name	Hrs
0.14	CSBP421 - Smart Computer Graphics	3
0.22	ITBP481 - Senior Graduation Project II	3

Term Credit Hours: 6

Last Semester

Score	Name	Hrs
0.11	ITBP495 - Internship	12

Term Credit Hours: 12

Total Credit Hours: 130

(a) KG-based course plan

Spring 2025

Index	Name	Hrs
1	CSBP340 - Database Systems	3
2	SWEB300 - Software Engineering Fundamentals	3
3	CSBP301-Artificial Intelligence	3
4	CSBP483 - Mobile Web Content & Development	3
5	GEO200 - World Regional Geography	3

Term Credit Hours: 15

Fall 2025

Index	Name	Hrs
1	CSBP400-Modeling & Simulation	3
2	CSBP421-Smart Computer Graphics	3
3	CSBP499-Special Topics in Computer Science	3
4	SWEB451 - Game Development	3
5	ITBP418 - Entrepreneurship in IT	3

Term Credit Hours: 15

Spring 2026

Index	Name	Hrs
1	SWEB450 - Analysis of Algorithms	3
2	ITBP321 - Web Application Development Lab	1
3	CSBP461 - Internet Computing	3
4	CSBP411- Machine Learning	3
5	ITBP480 - Senior Graduation Project I	3
6	CSBP476 - Robotics and Intelligent Systems	3

Term Credit Hours: 16

Fall 2026

Index	Name	Hrs
1	ITBP481 - Senior Graduation Project II	3

Term Credit Hours: 3

Spring 2027

Index	Name	Hrs
1	ITBP495 - Internship	12

Term Credit Hours: 12

Total Credit Hours: 130

(b) (ML, KG, LLM)-enhanced course plan

Figure 5.3: Generated Plans for Student 3

Spring 2025

Score	Name	Hrs
1.00	CSBP340 - Database Systems	3
0.39	HSR120 - Intro to Heritage & Culture	3
0.17	CSBP421-Smart Computer Graphics	3
0.88	GESU121-Sustainability	3
0.31	SWEB450 - Analysis of Algorithms	3

Term Credit Hours: 15

Fall 2025

Score	Name	Hrs
0.43	ITBP321-Web Application Development Lab	1
0.18	CSBP461 - Internet Computing	3
0.88	MATH140-Linear Algebra I	3
0.17	SWEB451-Game Development	3
0.18	CSBP400 - Modeling & Simulation	3

Term Credit Hours: 13

Spring 2026

Score	Name	Hrs
0.17	CSBP476-Robotics and Intelligent Systems	3
0.55	ITBP301 - Security Principles & Practice	3
0.45	CSBP320-Data Mining	3
0.18	CSBP483-Mobile Web Content & Development	3
0.30	ITBP480 - Senior Graduation Project I	3

Term Credit Hours: 15

Fall 2026

Score	Name	Hrs
0.17	ITBP418 - Entrepreneurship in IT	3
0.17	CSBP411-Machine Learning	3
0.44	CSBP316 - Human Computer Interaction	3
0.28	ITBP481 - Senior Graduation Project II	3

Term Credit Hours: 12

Last Semester

Score	Name	Hrs
0.14	ITBP495- Internship	12

Term Credit Hours: 12

Total Credit Hours: 130

(a) KG-based course plan

Spring 2025

Index	Name	Hrs
1	MATH140-Linear Algebra I	3
2	GESU121-Sustainability	3
3	CSBP340 - Database Systems	3
4	PHYS135-General Physics Lab I	1
5	ITBP301 - Security Principles & Practice	3
6	CSBP316 - Human Computer Interaction	3

Term Credit Hours: 16

Fall 2025

Index	Name	Hrs
1	SWEB450 - Analysis of Algorithms	3
2	CSBP421-Smart Computer Graphics	3
3	CSBP491- Computational Intelligence for Data Management	3
4	CSBP320 - Data Mining	3
5	HSR120 - Intro to Heritage & Culture	3

Term Credit Hours: 15

Spring 2026

Index	Name	Hrs
1	CSBP461 - Intemet Computing	3
2	ITBP321 - Web Application Development Lab	1
3	CSBP400 - Modeling & Simulation	3
4	CSBP411-Machine Learning	3
5	ITBP418-Entrepreneurship in IT	3
6	ITBP480 - Senior Graduation Project I	3

Term Credit Hours: 16

Fall 2026

Index	Name	Hrs
1	CSBP483-Mobile Web Content & Development	3
2	CSBP441 - Applied Computer Vision	3
3	ITBP481 - Senior Graduation Project II	3

Term Credit Hours: 9

Spring 2027

Index	Name	Hrs
1	ITBP495 - Internship	12

Term Credit Hours: 12

Total Credit Hours: 130

(b) (ML, KG, LLM)-enhanced course plan

Figure 5.4: Generated Plans for Student 4

Spring 2025

Score	Name	Hrs
1.00	PHYS135-General Physics Lab I	1
0.39	GEHP111 - Happiness and Wellbeing	3
0.39	GEIT112 - Fourth Industrial Revolution	3
0.39	GESU121-Sustainability	3
0.13	ITBP480 - Senior Graduation Project I	3

Term Credit Hours: 13

Fall 2025

Score	Name	Hrs
0.39	MATH140-Linear Algebra I	3
0.08	ITBP418-Entrepreneurship in IT	3
0.12	ITBP481 - Senior Graduation Project II	3

Term Credit Hours: 9

Last Semester

Score	Name	Hrs
0.06	ITBP495 - Internship	12

Term Credit Hours: 12

Total Credit Hours: 130

(a) KG-based course plan

Spring 2025

Index	Name	Hrs
1	PHYS135-General Physics Lab I	1
2	CENG202 - Discrete Mathematics	3
3	MATH140 - Linear Algebra I	3
4	GEIT112 - Fourth Industrial Revolution	3
5	GESU121 - Sustainability	3

Term Credit Hours: 13

Fall 2025

Index	Name	Hrs
1	SWEB300 - Software Engineering Fundamentals	3
2	ITBP418 - Entrepreneurship in IT	3
3	GEO200 - World Regional Geography	3
4	ITBP480 - Senior Graduation Project I	3

Term Credit Hours: 12

Spring 2026

Index	Name	Hirs
1	ITBP481 - Senior Graduation Project II	3

Term Credit Hours: 3

Fall 2026

Index	Name	Hrs
1	ITBP495 - Internship	12

Term Credit Hours: 12

Total Credit Hours: 130

(b) (ML, KG, LLM)-enhanced course plan

Figure 5.5: Generated Plans for Student 5

Spring 2025

Score	Name	Hrs
1.00	MATH105-Calculus I	3
0.82	PHYS105-General Physics I	3
0.21	PSYC100 - Introduction to Psychology	3
0.21	GESU121-Sustainability	3
0.71	CSBP119-Algorithms & Problem Solving	3
0.56	CSBP121-Programming Lab I	1

Term Credit Hours: 16

Fall 2025

Score	Name	Hrs
0.72	MATH110-Calculus II	3
0.68	ESPU1081 - Intro to Academic English for IT 1	3
0.47	CSBP219-Object Oriented Programming	3
0.30	CSBP221-Programming Lab II	1
0.46	CENG205 - Digital Design & Computer Org.	3
0.33	CENG202 - Discrete Mathematics	3

Term Credit Hours: 16

Spring 2026

Score	Name	Hrs
0.17	CSBP319 - Data Structures	3
0.42	STAT210 - Probability and Statistics	3
0.17	ITBP370 - Professional Responsibility in IT	3
0.32	CENG210-Communication & Networks Fund.	3
0.21	HSS105 - Emirates Studies	3

Term Credit Hours: 15

Fall 2026

Score	Name	Hrs
0.47	ITBP301-Security Principles & Practice	3
0.17	CSBP320 - Data Mining	3
0.14	CSBP315-Operating Systems Fund.	3
0.21	PHI101 - Introduction to Philosophy	3
0.21	GEIT112 - Fourth Industrial Revolution	3

Term Credit Hours: 15

Spring 2027

Score	Name	Hrs
0.17	ISEC311-Network Security I	3
0.07	ISEC421-Risk Analysis and Management	2
0.16	ISEC312-Cryptography	3
0.13	ISEC323-Secure Software Design and Engineering	3
0.11	CSBP340 - Database Systems	3

Term Credit Hours: 14

Fall 2027

Score	Name	Hrs
0.16	ISEC322 - Design and Analysis of Security Protocols	3
0.10	ISEC324-Cryptography Lab	1
0.14	ISEC321 - Network Security II	3
0.04	ISEC414-Network Security Lab	1
0.04	ITBP418 - Entrepreneurship in IT	3
0.21	CHEM111-General Chemistry 1	3

Term Credit Hours: 14

Spring 2028

Score	Name	Hrs
0.07	ISEC413-Security Architecture and Mechanisms	3
0.04	ISEC423-Systems Security Lab	1
0.21	ISLM100 - Islamic Culture	3
0.04	ISEC412 - Digital Forensics	3
0.04	ISEC416 - Information Security Management	3
0.07	ITBP480 - Senior Graduation Project I	3

Term Credit Hours: 16

Fall 2028

Score	Name	Hrs
0.04	ISEC424-Hardware-Oriented Security and Trust	3
0.04	ISEC411 - Privacy and Anonymity	3
0.04	ISEC422 - Security Policy, Laws, and Governance	3
0.07	ITBP481 - Senior Graduation Project II	3

Term Credit Hours: 12

Last Semester

Score	Name	Hrs
0.03	ITBP495- Internship	12

Term Credit Hours: 12

Total Credit Hours: 130

(a) KG-based course plan

Spring 2025

Index	Name	Hrs
1	ESPU1081 - Intro to Academic English for IT 1	3
2	MATH105 - Calculus I	3
3	PHYS105-General Physics I	3
4	ISLM100-Islamic Culture	3
5	HSS105 - Emirates Studies	3

Term Credit Hours: 15

Fall 2025

Index	Name	Hrs
1	CSBP119 - Algorithms & Problem Solving	3
2	CSBP121-Programming Lab I	1
3	CHEM111-General Chemistry 1	3
4	MATH110-Calculus II	3
5	GESU121-Sustainability	3
6	GEHP111 - Happiness and Wellbeing	3

Term Credit Hours: 16

Spring 2026

Index	Name	Hrs
1	CSBP219-Object Oriented Programming	3
2	CSBP221-Programming Lab II	1
3	CENG205 - Digital Design & Computer Org.	3
4	ITBP370 - Professional Responsibility in IT	3
5	HSR130-Introduction to Language & Communication	3
6	GEIT112 - Fourth Industrial Revolution	3

Term Credit Hours: 16

Fall 2026

Index	Name	Hrs
1	CENG202 - Discrete Mathematics	3
2	CENG210 - Communication & Networks Fund.	3
3	CSBP319 - Data Structures	3
4	ITBP301- Security Principles & Practice	3
5	STAT210 - Probability and Statistics	3

Term Credit Hours: 15

Spring 2027

Index	Name	Hrs
1	CSBP315-Operating Systems Fund.	3
2	ISEC311 - Network Security I	3
3	ISEC312-Cryptography	3
4	CSBP340 - Database Systems	3
5	CSBP320 - Data Mining	3

Term Credit Hours: 15

Fall 2027

Index	Name	Hrs
1	ISEC322 - Design and Analysis of Security Protocols	3
2	ISEC324-Cryptography Lab	1
3	ISEC321 - Network Security II	3
4	ISEC414 - Network Security Lab	1
5	ISEC323-Secure Software Design and Engineering	3
6	ISEC421-Risk Analysis and Management	2
7	ISEC428-Special Topics in Information Security	3

Term Credit Hours: 16

Spring 2028

Index	Name	Hrs
1	ISEC411 - Privacy and Anonymity	3
2	ITBP418 - Entrepreneurship in IT	3
3	ITBP480-Senior Graduation Project I	3
4	ISEC422- Security Policy, Laws, and Governance	3
5	ISEC416 - Information Security Management	3

Term Credit Hours: 15

Fall 2028

Index	Name	Hrs
1	ISEC412 - Digital Forensics	3
2	ITBP481 - Senior Graduation Project II	3
3	ISEC413-Security Architecture and Mechanisms	3
4	ISEC423 - Systems Security Lab	1

Term Credit Hours: 10

Spring 2029

Index	Name	Hrs
1	ITBP495 - Internship	12

Term Credit Hours: 12

Total Credit Hours: 130

(b) (ML, KG, LLM)-enhanced course plan

Figure 5.6: Generated Plans for Student 6

Spring 2025

Score	Name	Hrs
1.00	MATH105-Calculus I	3
0.82	PHYS105-General Physics	3
0.22	ECON110 - Principles of Economics	3
0.22	ELEC100-Free Elective	3
0.72	CSBP119-Algorithms & Problem Solving	3
0.57	CSBP121-Programming Lab I	1

Term Credit Hours: 16

Fall 2025

Score	Name	Hrs
0.74	CSBP219-Object Oriented Programming	3
0.31	CSBP221 - Programming Lab II	1
0.73	MATH110-Calculus II	3
0.69	ESPU1081 - Intro to Academic English for IT 1	3
0.22	GEIT112 - Fourth Industrial Revolution	3

Term Credit Hours: 13

Spring 2026

Score	Name	Hrs
0.22	ISLM100-Islamic Culture	3
0.10	ELEC200 - Free Elective	3
0.47	CENG205 - Digital Design & Computer Org.	3
0.22	CHEM111-General Chemistry 1	3
0.10	HSR120 - Intro to Heritage & Culture	3

Term Credit Hours: 15

Fall 2026

Score	Name	Hrs
0.42	CENG210-Communication & Networks Fund.	3
0.40	STAT210-Probability and Statistics	3
0.28	CENG202- Discrete Mathematics	3
0.25	ITBP280 - IT Project Management Exhibition	3
0.25	CSBP319 - Data Structures	3

Term Credit Hours: 15

Spring 2027

Score	Name	Hrs
0.21	CSBP340 - Database Systems	3
0.22	GESU121 - Sustainability	3
0.18	CSBP320-Data Mining	3
0.18	CSBP315-Operating Systems Fund.	3
0.22	HSS105 - Emirates Studies	3

Term Credit Hours: 15

Fall 2027

Score	Name	Hrs
0.08	ITBP420 - Data Analytics	3
0.17	ITBP301 - Security Principles & Practice	3
0.15	ITBP322 - Web and Mobile Systems	3
0.12	CSBP301-Artificial Intelligence	3
0.12	CSBP316 - Human Computer Interaction	3

Term Credit Hours: 15

Spring 2028

Score	Name	Hrs
0.05	CSBP483 - Mobile Web Content & Development	3
0.12	ITBP370 - Professional Responsibility in IT	3
0.11	ITBP321-Web Application Development Lab	1
0.05	ITBP418 - Entrepreneurship in IT	3
0.02	CENG529 - Networking Lab	1
0.05	CENG530-Computer Network Protocols	3
0.08	ITBP480 - Senior Graduation Project I	3

Term Credit Hours: 17

Fall 2028

Score	Name	Hrs
0.11	ITBP323-Systems Integration and Administration	3
0.11	ITBP324-Cloud Computing Fundamentals	3
0.05	ISEC411 - Privacy and Anonymity	3
0.08	ITBP481 - Senior Graduation Project II	3

Term Credit Hours: 12

Last Semester

Score	Name	Hrs
0.04	ITBP495 - Internship	12

Term Credit Hours: 12

Total Credit Hours: 130

(a) KG-based course plan

Spring 2025

Index	Name	Hrs
1	ESPU1081 - Intro to Academic English for IT 1	3
2	MATH105-Calculus I	3
3	CSBP119-Algorithms & Problem Solving	3
4	CSBP121-Programming Lab I	1
5	CHEM111-General Chemistry 1	3
6	ISLM100 - Islamic Culture	3

Term Credit Hours: 16

Fall 2025

Index	Name	Hrs
1	PHYS105-General Physics I	3
2	MATH110-Calculus II	3
3	CSBP219-Object Oriented Programming	3
4	CSBP221-Programming Lab II	1
5	HSR130-Introduction to Language & Communication	3
6	HSR150 - Introduction to Government Policy & Urban Structu	3

Term Credit Hours: 16

Spring 2026

Index	Name	Hrs
1	CENG205 - Digital Design & Computer Org.	3
2	STAT210 - Probability and Statistics	3
3	CENG202- Discrete Mathematics	3
4	CSBP319 - Data Structures	3
5	HSS105 - Emirates Studies	3

Term Credit Hours: 15

Fall 2026

Index	Name	Hrs
1	CENG210-Communication & Networks Fund.	3
2	ITBP370 - Professional Responsibility in IT	3
3	ITBP280 - IT Project Management Exhibition	3
4	CSBP315-Operating Systems Fund.	3
5	GESU121 - Sustainability	3

Term Credit Hours: 15

Spring 2027

Index	Name	Hrs
1	ITBP301-Security Principles & Practice	3
2	CSBP340 - Database Systems	3
3	CSBP316 - Human Computer Interaction	3
4	CSBP301 - Artificial Intelligence	3
5	GEIT112 - Fourth Industrial Revolution	3

Term Credit Hours: 15

Fall 2027

Index	Name	Hrs
1	CENG530-Computer Network Protocols	3
2	CENG529 - Networking Lab	1
3	ITBP324-Cloud Computing Fundamentals	3
4	CSBP320-Data Mining	3
5	ITBP322-Web and Mobile Systems	3
6	ELEC100 - Free Elective	3

Term Credit Hours: 16

Spring 2028

Index	Name	Hrs
1	ITBP480-Senior Graduation Project I	3
2	ITBP418-Entrepreneurship in IT	3
3	ITBP321 - Web Application Development Lab	1
4	ITBP323-Systems Integration and Administration	3
5	CSBP483 - Mobile Web Content & Development	3
6	ELEC200 - Free Elective	3

Term Credit Hours: 16

Fall 2028

Index	Name	Hrs
1	ITBP481 - Senior Graduation Project II	3
2	ISEC411 - Privacy and Anonymity	3
3	ITBP410 - The Internet of Things	3

Term Credit Hours: 9

Spring 2029

Index	Name	Hrs
1	ITBP495-Internship	12

Term Credit Hours: 12

Total Credit Hours: 130

(b) (ML, KG, LLM)-enhanced course plan

Figure 5.7: Generated Plans for Student 8

5.2 Quantitative Evaluation

5.2.1 Comparison to Generic Plan

KG-Based Plan

From a quantitative perspective, the KG-Based Plan recommended for new students in CS, IS, and IT was compared to the generic plans provided by the university. While the generic plans are not considered the ultimate reference, these comparisons serve as a baseline for future evaluations. For a new CS student, the recommended plan, as shown in Figure 5.1a, exhibited a 53% similarity with the generic plan. Specifically, 23 out of 43 courses appeared in the same academic year as in the generic plan. Similarly, the recommended plan for a new IS student, as in Figure 5.6a, showed a 73% similarity, in which 32 out of 44 courses were assigned to the same academic year as in the generic plan. Finally, the recommended plan for a new IT student, shown in Figure 5.7a, revealed a 56% similarity, with 24 out of 43 courses aligning with the same academic year as the generic plan.

(ML, KG, LLM)-Enhanced Plan

From a quantitative perspective, the enhanced Plan recommended for new students in CS, IS, and IT was also compared to the generic plans provided by the university. For a new CS student, the recommended plan exhibited a 67% similarity with the generic plan. Specifically, 29 out of 43 courses appeared in the same academic year as in the generic plan. Similarly, the recommended plan for a new IS student showed a 70% similarity, in which 31 out of 44 courses were assigned to the same academic year as in the generic plan. Finally, the recommended plan for a new IT student revealed a 63% similarity, with 27 out of 43 courses aligning with the same academic year as the generic plan.

Table 5.2: Comparison of Recommended Plans to Generic Plans for New Students

Major	Approach	Courses in Same Year	Total Courses	Similarity (%)
CS	KG Only	23	43	53%
	ML, KG, LLM	29	43	67%
IS	KG Only	32	44	73%
	ML, KG, LLM	31	44	70%
IT	KG Only	24	43	56%
	ML, KG, LLM	27	43	63%

As seen in Table 5.2, the similarity between the recommended plans and the generic plans varies across different majors and approaches. From the results, it is observed that the second approach generally yields better results than the first approach, particularly for CS and IT students. The CS plan's similarity improves from 53% (first approach) to 67% (second approach), indicating a more accurate alignment of courses with the generic plan when enhanced methods are applied. Similarly, for the IT plan, the similarity rises from 56% (first approach) to 63% (second approach), showing that the enhanced plan brings a noticeable improvement in course alignment. On the other hand, the IS plan shows a smaller difference between the two approaches, with the first approach achieving a 73% similarity, while the second approach slightly reduces the similarity to 70%.

Overall, the second approach shows clear advantages for CS and IT students, while the first approach performs better for IS students. However, it is important to note that overall similarity may not be the optimal metric for evaluating the quality of these plans. While the similarity score provides some insight into how well the generated plans align with the university's generic plans, it doesn't capture other factors such as course workload distribution or the adaptability of the plan to individual student needs.

As one of the advisors involved in the evaluation pointed out, the generic plan for CS students, for example, contains too many 4xx-level major electives in the last year, which is not ideal. These advanced courses often require a significant amount of work and can create an imbalanced workload, which might not be the most suitable for students in their final year. This feedback suggests that a more balanced workload could be a better metric for

evaluating the quality of the plans, rather than purely focusing on the alignment with the generic plan.

5.2.2 Comparison to Real Students' Plan

Another comparison was conducted by evaluating the generated academic plans for new students, which cover all semesters until graduation, against the actual plans followed by students who graduated from the studied programs. A sample of 30 graduates from the CS major, 30 graduates from the IS major, and 5 graduates from the IT major was used in the analysis.

Comparisons were performed year by year using three similarity measures: Jaccard similarity, Cosine similarity, and Dice similarity coefficient. Table 5.3 presents the average similarity results across years for each major, comparing the KG-based plan and the enhanced plan that incorporates ML, KG, and LLM.

The results show that the enhanced method consistently improves similarity scores across most years and measures when compared to the KG-based plan. For the CS major, the average Cosine similarity increased from 65.3% in the KG-based plan to 67.4%. Similarly, the IS major saw an increase from 75.2% to 80.8%, and the IT major improved from 64.4% to 68.5%.

Table 5.3: Comparison of Average Similarity Measures Across Years

Major	Year	KG-based			ML+KG+LLM		
		Jaccard	Cosine	Dice	Jaccard	Cosine	Dice
CS	1	0.474	0.665	0.632	0.501	0.690	0.658
	2	0.298	0.462	0.448	0.384	0.562	0.545
	3	0.294	0.445	0.444	0.324	0.483	0.482
	4	0.578	0.726	0.721	0.508	0.667	0.662
	5	0.967	0.967	0.967	0.967	0.967	0.967
	Avg	**0.522**	**0.653**	**0.642**	**0.537**	**0.674**	**0.663**
IS	1	0.475	0.682	0.639	0.525	0.758	0.683
	2	0.477	0.641	0.640	0.583	0.731	0.729
	3	0.468	0.655	0.625	0.572	0.724	0.718
	4	0.652	0.794	0.784	0.716	0.838	0.828
	5	0.983	0.990	0.989	0.983	0.990	0.989
	Avg	**0.611**	**0.752**	**0.735**	**0.676**	**0.808**	**0.789**
IT	1	0.417	0.628	0.585	0.557	0.759	0.714
	2	0.353	0.518	0.517	0.341	0.512	0.498
	3	0.328	0.506	0.485	0.426	0.592	0.587
	4	0.528	0.688	0.687	0.454	0.681	0.621
	5	0.800	0.883	0.867	0.800	0.883	0.867
	Avg	**0.485**	**0.644**	**0.628**	**0.516**	**0.685**	**0.658**

It was observed that the similarity in year 5 was the highest in all majors, followed by year 4, while the similarity in the first three years was relatively lower. This pattern highlights the gradual alignment of students' actual plans with the generated plans as they progress through their academic journey. The similarity measures for both approaches follow similar trends, with year 5 achieving the highest scores for both methods. To further analyze the differences for each student, Table 5.4 illustrates the year-by-year similarity for each student for the CS, IS, and IT majors, respectively. The figures show an upward trend in similarity scores for both approaches as later years are compared.

In particular, there were some exceptional cases, such as one student in the CS group, who completed all courses within four years. As a result, the similarity score for year 5 was zero, as year 5 typically consists only of the internship course. Another example is one student in the IS group, who completed the internship and an additional course in year 5, resulting in a cosine similarity of 0.7 for that year.

The discrepancies between the actual academic plans of students and the generated plans can be attributed to several key factors. One significant factor is that many students enroll in summer semesters, which were not accounted for in the current implementation. Additionally, some students fail courses and must retake them in subsequent semesters, disrupting the recommended sequence of courses. Furthermore, some students start their studies later than expected due to university or college requirements. As a result, they do not take foundational courses in their first semester, which are critical for shaping their academic plans. Another contributing factor is that some students do not follow the recommended course sequence from the beginning, leading to greater deviations from the generated plans. During the first two years of study, students are often presented with a wide range of options. Their choices during this period significantly influence the structure and progression of their academic plans in later years.

Despite these variations, the overall similarity of 80% observed in the IS plan using the second approach is considered high, particularly given the inherent diversity and flexibility in academic plans.

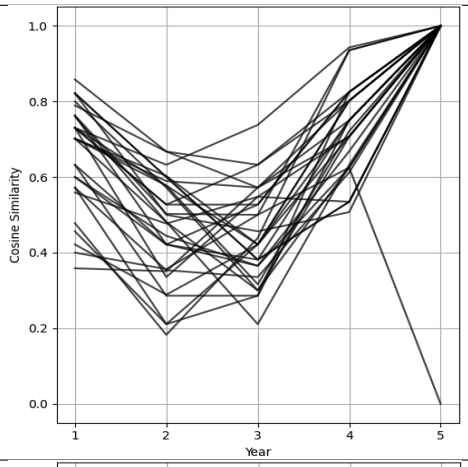
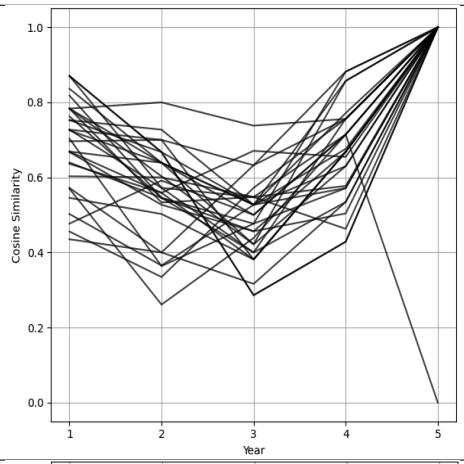
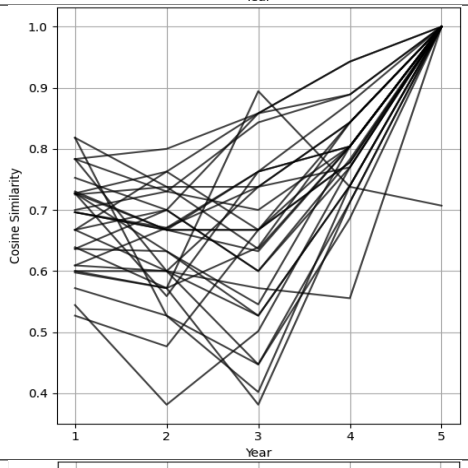
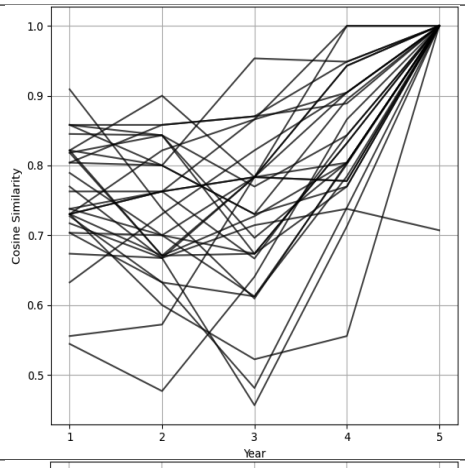
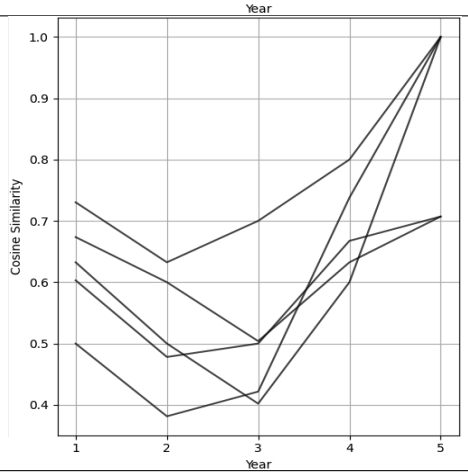
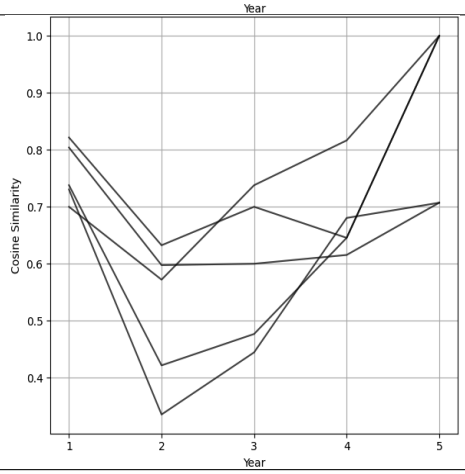
5.3 Qualitative Evaluation

To qualitatively assess the system, academic advisors with experience in course planning reviewed seven real cases. The advisors consisted of five experts from various departments at UAEU's CIT, including the department chair for CS. The evaluation was conducted in two phases: first, reviewing the plans generated using the initial method, and second, evaluating plans produced after enhancements were introduced. Advisors provided valuable feedback highlighting strengths, weaknesses, and areas for improvement. Overall, the advisors found that the plans respected prerequisite and corequisite constraints while aiming to maintain a manageable workload for students.

5.3.1 Evaluation of the Initial Method

In the first evaluation phase, the advisors reviewed the plans generated using the initial method. Generally, the advisors stressed the importance of proper course sequencing and

Table 5.4: Comparison of Similarity Measure Figures

Major	KG-based	ML+KG+LLM-based
CS		
IS		
IT		

workload balancing. They recommended including easier courses in the initial semesters to help students build a strong GPA foundation, while reserving more complex courses for later semesters. Specifically, they suggested assigning three core courses, one math course, and one elective per semester during the first two years. Additionally, the number of courses or overall course load should be reduced during semesters containing the senior project to allow students to focus on their graduation requirement.

One point that raised was the placement of electives. Advisors suggested that elective courses should not be scheduled too early, especially when students may not yet have a clear understanding of their academic interests. For instance, in Case Study 7, free electives were scheduled in the third semester, which was considered inappropriate. The advisors recommended deferring electives to later semesters or allowing students to define their interests so the system can tailor recommendations accordingly. They also proposed that instead of automatically selecting electives, the system could display recommendation scores for all elective options, allowing students to make more informed decisions.

5.3.2 Evaluation of the Enhanced Method and Further Recommendations

Following improvements to the system, the advisors reviewed the newly generated plans. While many earlier issues were addressed, several new observations and future enhancement suggestions were identified.

One of the recurring suggestions was the need for more flexible elective recommendations. Advisors emphasized that students should be shown the best elective options, along with scores or rankings, so they can make informed choices rather than having the system auto-select courses. This enhancement would improve user control and personalization.

The advisors also identified cases where the senior project semesters were underloaded. For example, in Case 3, the last semester contained only *Senior Project II*. If the student were in good standing, they might prefer to shift *Senior Project II* earlier. The same issue appeared in Case 5. Although this was not critical, the advisors suggested that the system should aim to distribute the load more evenly and allow flexibility for students in good

academic standing.

To further enhance workload balancing, the advisors proposed ranking courses based on difficulty levels, either using expert-defined values, average student grades, or a combination of both. This difficulty ranking could then be used to calculate a semester load indicator, helping to avoid overloading students in any given term.

Specific cases highlighted the need for better balancing. In Case 4, the Fall 2025 semester contained many heavy courses, which could have been distributed more evenly by shifting one course to Fall 2026. Similarly, in Case 7, the Fall 2026 semester appeared too heavy.

Advisors also recommended leveraging student academic profiles, including high school performance and grades in completed courses, to personalize recommendations. For example, students who perform well in programming but struggle with other subjects could benefit from plans that balance difficult programming courses with easier non-programming courses.

Additionally, there were some observations regarding prerequisite placements. For instance, in Case 1, the course *Security Principles*, which is a prerequisite for many other courses, should ideally have been placed in the fourth semester. However, it must be noted that as long as the overall plan remained feasible within 4.5 years, minor deviations should be acceptable.

5.4 Strengths

5.4.1 Dynamic Personalized Planning

A key strength of the recommendation system is its ability to dynamically generate personalized academic plans using KG embeddings and students' academic history. This allows the system to tailor recommendations based on each student's unique progression, ensuring a customized learning path. The system automatically generates a plan that aligns with the courses a student has already completed. Additionally, by incorporating LLMs, students can express their preferences in natural language, making the process more intuitive. These preferences are then considered by the model when generating the optimal academic

plan.

5.4.2 Flexibility

The system is versatile and adaptable to any academic program, not just IT-related disciplines. By relying on course data such as descriptions, learning outcomes, and prerequisites, and history of students, the system can integrate seamlessly with diverse curricula. Beyond plan generation, the KG can be used for other tasks, such as identifying overlaps between programs or comparing the similarities between majors. In its current implementation, certain rules, such as the placement of Senior Project and internship courses, are hardcoded. However, these rules can be integrated into the KG by updating corresponding nodes, enhancing flexibility and supporting broader program integration.

5.4.3 Demand-Driven Offerings

As the studied cases are based on the UAEU curriculum, the university's registration system allows students to create academic plans and register for courses accordingly. However, students have the flexibility to register for all planned courses or only a subset and can modify their plans during the add/drop period, typically at the beginning of the semester. This period enables students to adjust their schedules, often due to conflicts or closed course sections that have reached maximum capacity.

One of the primary challenges students face is the lack of effective course planning, leading to insufficient availability of required courses. The proposed recommendation system addresses this issue by automatically generating academic plans for all students, allowing universities to anticipate course demand in advance. As a result, students are less likely to encounter closed sections or scheduling conflicts, while colleges can optimize course offerings for the upcoming semester and beyond.

5.5 Limitations

Despite its strengths, the recommendation system has several limitations that warrant future improvements.

5.5.1 Dependency on Data

The system relies heavily on the quality of course materials, descriptions, and learning outcomes to determine course similarity. Variability in the accuracy and completeness of these descriptions may impact the effectiveness of the recommendations.

Additionally, differences between the alumni data used for training and the current curriculum present a challenge. For instance, in the old curriculum, *Programming Lab 2* was paired with *Data Structures*, whereas in the current curriculum, it is paired with *Object-Oriented Programming*. This change affected the system's term suggestions, requiring manual rule adjustments to maintain consistency with the current study plan.

5.5.2 Hardcoded Elements

Certain aspects of the system, such as the placement of internship and senior project courses, were hardcoded. Furthermore, adjusting the weights of level edges to prioritize lower-level courses did not yield the expected results, necessitating manual score adjustments to achieve the desired prioritization.

5.5.3 Exclusion of Certain Student Circumstances

The current algorithm does not consider summer semesters or students pursuing minors, limiting its adaptability to unique academic paths.

5.5.4 Graph-Based Limitations

The KG approach primarily captures relationships (edges) between nodes rather than the semantic meaning of the nodes themselves. As a result, it may overlook connections between related topics. For example, terms such as "Array" and "ArrayLists" are treated as distinct entities despite their conceptual similarity. Incorporating semantic analysis could help address this limitation.

Additionally, course prioritization could be improved by considering factors such as course difficulty derived from historical grade data. However, determining course difficulty based solely on grades is challenging, as grading standards vary across instructors and

student cohorts. A more robust approach would require additional contextual data to ensure balanced and personalized study plans.

Chapter 6: Conclusion

Academic advising is often a time-intensive task for both students and advisors. This thesis aimed to develop and evaluate an intelligent recommendation system for academic advising in higher education institutions. By integrating knowledge graphs, machine learning, and large language models, the system provides personalized course recommendations that streamline the advising process for both students and advisors.

The system was evaluated by five academic advisors with expertise in reviewing study plans. The results indicate that the generated academic plans adhere to prerequisite and corequisite constraints while maintaining a manageable workload, demonstrating the effectiveness of the recommendation system. KG-based plans were observed to provide a reasonable approximation of a viable academic path, but the incorporation of ML and LLM led to more logical and refined recommendations.

Despite these successes, the evaluation highlighted areas for improvement. Assuming that the LLM does not possess inherent knowledge of course content, enhancing the system's understanding of semantic similarities between courses—particularly in terms of their topics—could further improve recommendation quality. Currently, the system primarily relies on course embeddings derived from relationships within the knowledge graph, which may not fully capture the nuances of course content and student preferences. Another direction for future research involves assessing the system's applicability across different academic programs and universities. Extending the evaluation to diverse educational institutions would help determine the system's adaptability and effectiveness in various academic contexts.

Based on the feedback received, several future enhancements were identified. The system could benefit from increased transparency in elective course recommendations by displaying scores or rankings, allowing students to make more informed decisions. Additionally, implementing course difficulty rankings would help balance workloads more effectively,

while leveraging student academic profiles could further tailor the recommendation process to individual needs. Improvements in semester workload distribution, particularly in the allocation of major electives, could prevent students from encountering overloaded semesters.

Overall, while the enhanced recommendation approach addressed several initial challenges, the insights provided by academic advisors offer a clear roadmap for further refinements. Future enhancements will continue to improve the system's recommendations, ultimately enhancing the academic advising experience for students and advisors alike.

References

[1] S. Marginson, "The worldwide trend to high participation higher education: Dynamics of social stratification in inclusive systems," *Higher Education*, vol. 72, no. 4, pp. 413–434, 2016.

[2] A. Assiri, A. A.-M. AL-Ghamdi, and H. Brdesee, "From traditional to intelligent academic advising: A systematic literature review of e-academic advising," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 4, 2020. [Online]. Available: http://dx.doi.org/10.14569/IJACSA.2020.0110467

[3] Embassy of the United Arab Emirates, Washington, D.C., "Education in the uae," https://www.uae-embassy.org/discover-uae/society/education-in-the-uae, 2025, accessed: 2025-04-19.

[4] Program for Research on Private Higher Education (PROPHE), "Global enrollment by region and country," https://www.prophe.org/en/global-data/global-data/global-enrollment-by-region-and-country/, 2025, accessed: 2025-04-19.

[5] United Arab Emirates University, "Annual reports," https://www.uaeu.ac.ae/en/about/annual-report.shtml, 2020, accessed: 2025-04-19.

[6] S. Channarukul, N. Saejiem, K. Bhumichitr, R. Jiamthapthaksin, V. Nicklamai, and K. Terdvikran, "Social-aware automated course planner: An integrated recommender system for university registration system," in *Proceedings of the 2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, Phuket, Thailand, 2017, pp. 545–548.

[7] Z. Zhao, W. Fan, J. Li, Y. Liu, X. Mei, Y. Wang, Z. Wen, F. Wang, X. Zhao, J. Tang, and Q. Li, "Recommender systems in the era of large language models (llms)," *arXiv preprint*, vol. arXiv:2307.02046, 2024. [Online]. Available: https://arxiv.org/abs/2307.02046

[8] A. Al-Badarenah and J. Alsakran, "An automated recommender system for course selection," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 3, 2016.

[9] N. Lynn and A. Emanuel, "A review on recommender systems for course selection in higher education," 2020.

[10] Y.-K. Ng and J. Linn, "Crsrecs: A personalized course recommendation system for college students," in *2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA)*, Larnaca, Cyprus, 2017, pp. 1–6.

[11] M. C. Urdaneta-Ponte, A. Mendez-Zorrilla, and I. Oleagordia-Ruiz, "Recommendation systems for education: Systematic review," *Electronics*, vol. 10, no. 14, p. 1611, 2021.

[12] J. Rivera, "A hybrid recommender system to enrollment for elective subjects in engineering students using classification algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 11, 2020.

[13] N. Bendakir and P. Andre-Aisenstadt, "Using association rules for course recommendation," in *AAAI Workshop - Technical Report*, 2006.

[14] J. Ji, Z. Li, S. Xu, W. Hua, Y. Ge, J. Tan, and Y. Zhang, "Genrec: Large language model for generative recommendation," *arXiv preprint*, vol. arXiv:2307.00457, 2023. [Online]. Available: https://arxiv.org/abs/2307.00457

[15] X. Wang, L. Cui, M. Bangash, M. Bilal, L. Rosales, and W. Chaudhry, "A machine learning-based course enrollment recommender system," in *Proceedings of the Conference*, 2022, pp. 436–443.

[16] B. Nadir and A. Ali, "Online course registration and advisory systems based on students' personal and social constraints," *Kurdistan Journal of Applied Research*, vol. 6, p. 11, 2021.

[17] N. Salehudin, H. Kahtan, M. Abdulhak, and H. Al-bashiri, "A proposed course recommender model based on collaborative filtering for course registration," *International Journal of Advanced Computer Science and Applications*, vol. 10, 2019.

[18] E. S. Khorasani, Z. Zhenge, and J. Champaign, "A markov chain collaborative filtering model for course enrollment recommendations," in *2016 IEEE International Conference on Big Data (Big Data)*, Washington, DC, USA, 2016, pp. 3484–3490.

[19] A. Esteban, A. Zafra, and C. Romero, "Helping university students to choose elective courses by using a hybrid multi-criteria recommendation system with genetic optimization," *Knowledge-Based Systems*, vol. 194, p. 105385, 2020.

[20] Z. A. Pardos and W. Jiang, "Designing for serendipity in a university course recommendation system," in *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge (LAK '20)*. Association for Computing Machinery, 2020, pp. 350–359.

[21] W. Jiang, Z. A. Pardos, and Q. Wei, "Goal-based course recommendation," in *Proceedings of the 9th International Conference on Learning Analytics & Knowledge (LAK19)*. Association for Computing Machinery, 2019, pp. 36–45.

[22] H. Abu-Rasheed, M. H. Abdulsalam, C. Weber, and M. Fathi, "Supporting student decisions on learning recommendations: An llm-based chatbot with knowledge graph contextualization for conversational explainability and mentoring," *arXiv preprint*, vol. arXiv:2401.08517, 2024. [Online]. Available: https://arxiv.org/abs/2401.08517

[23] H. Abu-Rasheed, C. Weber, and M. Fathi, "Knowledge graphs as context sources for llm-based explanations of learning recommendations," *arXiv preprint*, vol. arXiv:2403.03008, 2024. [Online]. Available: https://arxiv.org/abs/2403.03008

[24] E. Palumbo, G. Rizzo, and R. Troncy, "entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, ser. RecSys '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 32–36. [Online]. Available: https://doi.org/10.1145/3109859.3109889

[25] R. Kanjirathinkal, "Explainable recommendations," Ph.D. dissertation, Stanford University, 2017.

[26] M. Chen, X. Huang, H. Chen, X. Su, and J. Yur-Austin, "Data driven course scheduling to ensure timely graduation," *International Journal of Production Research*, vol. 61, no. 1, pp. 336–361, 2021.

[27] N. A. Volk, G. Rojas, and M. V. Vitali, "Uninet: Next term course recommendation using deep learning," in *2020 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*. IEEE, 2020, pp. 377–380.

[28] A. Polyzou, A. N. Nikolakopoulos, and G. Karypis, "Scholars walk: A markov chain framework for course recommendation," *Proceedings of The 12th International Conference on Educational Data Mining (EDM 2019)*, 2019.

[29] K. H. Tsai, T. C. Hsieh, T. K. Chiu, M. C. Lee, and T. I. Wang, "Automated course composition and recommendation based on a learner intention," in *Seventh IEEE International Conference on Advanced Learning Technologies (ICALT 2007)*, Niigata, Japan, 2007, pp. 274–278.

[30] G. M. Thompson, "Using information on unconstrained student demand to improve university course schedules," *Journal of Operations Management*, vol. 23, no. 2, pp. 197–208, 2005.

[31] M. Carter, "A comprehensive course timetabling and student scheduling system at the university of waterloo," in *Proceedings*, 2000, pp. 64–84.

[32] N. Guan, D. Song, and L. Liao, "Knowledge graph embedding with concepts," *Knowledge-Based Systems*, vol. 164, pp. 38–44, 2019.

[33] J. Guia, V. G. Soares, and J. Bernardino, "Graph databases: Neo4j analysis." in *ICEIS (1)*, 2017, pp. 351–356.

[34] X. Zou, "A survey on application of knowledge graph," *Journal of Physics: Conference Series*, vol. 1487, p. 012016, 2020.

[35] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," https://arxiv.org/abs/1607.00653, 2016.

[36] E. Palumbo, D. Monti, G. Rizzo, R. Troncy, and E. Baralis, "entity2rec: Property-specific knowledge graph embeddings for item recommendation," *Expert Systems with Applications*, vol. 151, pp. 113 235 –, Aug. 2020. [Online]. Available: https://hal.science/hal-03489886

[37] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

UAEU

جامعة الإمارات العربية المتحدة
United Arab Emirates University

UAEU MASTER THESIS NO. 2025: XX

The main goal of this thesis is to . . . [A small introduction/summary of your thesis]

Fareeha Jamal received her PhD in Mathematics from the Department of Mathematical Sciences, College of Science at the United Arab Emirates University, UAE. She received her Master and Bachelor of Science in Mathematics from National University of Sciences and Technology, Pakistan.

www.uaeu.ac.ae

Online publication of thesis:
https://scholarworks.uaeu.ac.ae/etds/