NittanyPath Project Phase I Report

Feb 18, 2020

Eric Zhewen Li, zxl163@psu.edu

Instrcutor: Dr. Wang-Chien Lee

Collaborators: None

Table of Contents

List of Diagrams

List of Tables

0. Introduction

NittanyPath is a startup project which aims to help Nittany State University (NSU) maintain their software system for course information management[1]. In Phase I, Eric Li, one of the students enrolled in CMPSC 431W, is going to systematically design and implement prototypes for validation of feasibility[1]. *NittanyPath* will be a database-backed web application and is intended to manage course-related information, including courses, students, faculty members, etc.

There will be two phases of the project. Phase I is to, "based on the provided project description, analyze the requirements of *NittanyPath* in order to specify its system functionality and to identify data needed for the system functions as well as business rules (integrity constraints) to be imposed upon the needed data[1]". In addition, it includes a conceptual database design with the *entity-relationship* model to express the data and constraints identified. Phase I also includes a technology survey to research on the current web/database application technologies, and logical database design and schema normalization.

Phase II, in general, is system prototype, and it includes Database Population, Functionality Implementation, Progress Review, and Reflections and Final Deliverables[1]. In detail, the system is supposed to reorganize the raw data in accordance with the schema and populate the database. In addition, an implementation of the web application for *NittanyPath* should be built to demonstrate its functionality.

1. Requirement Analysis

NittanyPath aims to replace the current course management system. To be more specific, the system manages course-related information, including courses, section offerings, enrolled students, teaching faculty members, and etc. The system should also automate the enrollment process, including checking pre-requisite requirements, restricting enrollment numbers, maintaining waitlist, and etc. The system should also restrict certain access and give different response based on different types of the current user. In this section, a detailed analysis on the client's requirement for *NittanyPath* is discussed, including system functionalities, types of data that must be stored, and how the data is accessed in support of the system functionality.

1.1. *NittanyPath* Users

This section analyzes the end users to better understand the purposes and goals of *NittanyPath*, so that we can understand the client's requirement from users' perspective, and then implement corresponding functions with respect to their needs. For *NittanyPath*, the primary users considered are <u>students</u>, <u>faculty members</u>, and <u>teaching assistants</u>. For the scope of this project, we do not concern about the design of administrators. We also assume all users can log in with their email ID and the login password[2], and new users should be able to login with their initial password and setup their own password upon the first time they login *NittanyPath*. Among all *NittanyPath* users, they share some common attributes, and their basic information will be captured in the University Member Table. And these common attributes include Email ID, Legal Name, Date of Birth (Age), Legal Gender, University ID (9-digit), Home Address, and Initial Login Password. In addition, address is a composite entity set, with its attributes of Street, City, State, and Zip Code that require to be captured.

In addition to these common attributes, University Members also share some features, where they can login with their initial password and after they logged in, they will be able to change this basic information (including login password) as they wish, except the Email and University ID.

1.1.1. Students

Students are one of the key users of *NittanyPath*. The system needs to be able to maintain detailed information about students enrolled at Nittany State University. Students are one kind of University Members, and the Student Table inherits from the University Member Table. In addition to the common attributes, we also need to capture Major ID, Academic Program, Advisor ID, Minor ID, and Status.

For a student to be a valid user for the system, s/he must take at least one class. And the attribute Status is keeping track of the students' current status – if a student graduates, s/he will be removed from the system six months after their graduation.

1.1.2. Faculty Members

Faculty Members are also one kind of University Members and hence they inherit those common attributes. Additional information includes Department ID, Title, Office Phone, and Office Location.

1.1.3. Teaching Assistants

Teaching Assistants are one kind of Students and hence one kind of University Members. Teaching Assistant Table inherit attributes from both University Member and Student Table. Additional information includes Office Hours and Office Location.

1.2. Departments

At Nittany State University, departments are home of majored students and affiliated faculty members. As required by the user, every faculty member belongs to exactly one department and all students must major in at least one department. Information that needs to be captured is Department Name and Building Name.

1.3. Courses

In *NittanyPath*, a course can only be offered by one department. The system needs to store information about which department is the course affiliated with, course number and name of the course, course description, and its own dropping deadline. Information that needs to be captured include Department ID, Course Number, Course Description, and Drop Deadline.

1.4. Sections

A section is the lower-level unit of a course, and it is represented by an entity set affiliated with Course. And when one course is removed from the system, all its affiliated section offerings should be removed too. Sections are denoted by section numbers, and given the combination of course subject, course number, and section numbers, we should be able to uniquely identity all sections from each other stored in the system. Each section is taught by a teaching team of professors and teaching assistant(s). We also need to store information about each section's capacity limit, exams, homework assignments.

1.5. Views and Functionalities

Views, or External Schemas, according to our textbook[3], are defined as models that allow "data access to be customized and authorized at the level of groups of users". And some views need to be defined in *NittanyPath*, with respective to different types of users, to enable the implementation of some features, and to increase security level of the system.

1.5.1. Students' Views

According to the university's requirements, students should be able to browse and search for course information, including course's subject and number, course description, enrollment limit, current enrollment status, meeting time and location, and instructor's name, using NittanyPath. In addition, students should be able to see the details about courses they are currently enrolled in, including exams and homework that have been assigned, and their grades when they are graded by the teaching team. There will be a forum for each class, where students can create posts, view and comment on posts sent from other members of the class (including students and the teaching team). For privacy purposes, students should also be restricted so that they can only view their own grades, and they cannot delete a post that is not sent from themselves.

1.5.2. Faculty Members' Views

For faculty members, they should be able to see a list of all the courses they are currently teaching or have taught in the past. And when they click into a specific class, they should then be able to see the class roster, where a list of all members related to the class is shown, including their names, email ID and grades. Functionalities should be made available to faculty members so that they can create entries for homework, assignments, and exams for the courses they are currently teaching. They should also be able to submit assignment grades, and a final grade for individual students. With the course forum, faculty members should be able to make announcements that can be seen by all students and teaching assistant(s) related to this course. They should be able to view, post, and delete any forum entries as well. However, faculty members' privileges should be restricted within the courses they are currently teaching only.

1.5.3. Teaching Assistants' Views

For teaching assistants, they will see a list of all the courses they are currently taking (as they are also students) and teaching at the same time. And the system should be smart enough to distinguish between these and show them different views and assign different privileges accordingly. For any course they are currently teaching, teaching assistants should be able to create and grade homework assignments, make announcements, and see the forum entries. Within one semester, each teaching assistant can only teach one course section. In addition, the teaching assistant cannot teach a course that s/he is currently enrolled in.

1.5.4. Course Search and Enrollments

Students should be able to search for a class and its detailed information by providing adequate information, including the course subject, number, and section number. *NittanyPath* should show a list of all course (or section) offerings that meet the search requirements and students should be able to choose a certain course section from the list to enroll into. And when students are attempting to enroll, NittanyPath must check if the student has met the prerequisite requirements and section's current enrollment status (open, full, or waitlist) – a student can only successfully enroll when all conditions are met; otherwise, system should reject and present a warning message. Additionally, a student cannot enroll into a different section of the course s/he is already enrolled into, nor retake a course s/he has passed in a previous semester.

1.5.5. Dropping from a Class

Students should be able to drop from any class they are currently enrolled into. And *NittanyPath* should show a list of the student's current enrollments to choose. And before a student is successfully removed from the course roster, the system should check if the drop deadline associated with the course has passed, or if this is the only course s/he is currently enrolled in – system should reject and present a warning message if either condition is not met.

2. Conceptual Database Design

2.1. Relationship between Users

The class hierarchies and inheritance relationships between the three types of users in *NittanyPath* are illustrated by the following Entity-Relationship (ER) Diagram below in *Diagram 2.1*. A weak entity set called ‘address’ is introduced to capture separate information about Street, City, State, and Zip Code. Upon departure of a university member, his/her affiliated address information will be removed from the system. Total Participation and Key Constraints are identified between University Member and Address as each member must have one address stored in the system.

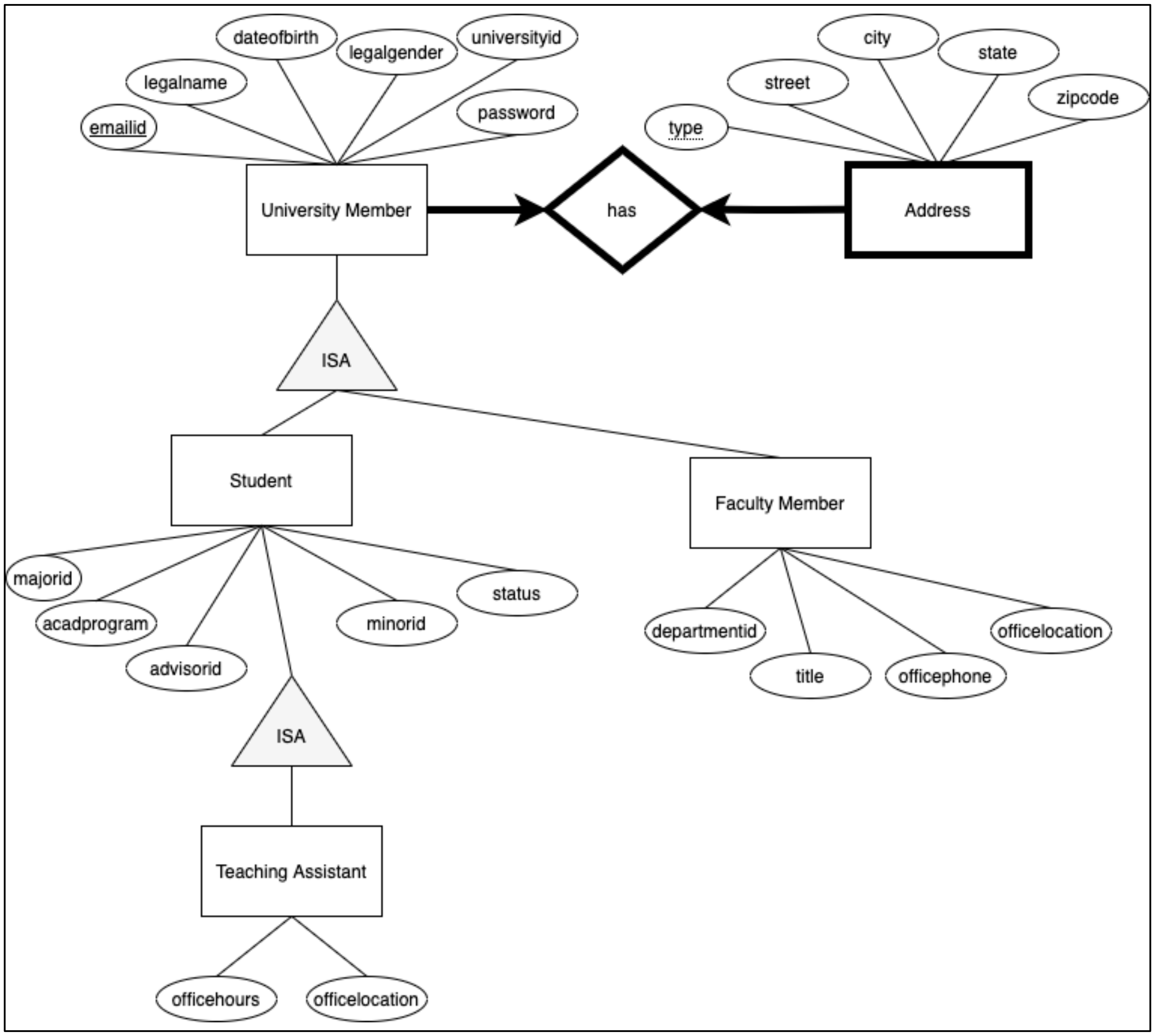

Diagram 2.1. ER Diagram between Different Types of Users

2.2. Department

To translate the requirements for Department table into ER model, we need to introduce two relationships – ‘majors / minors in’ and ‘affiliated with’ between Student and Department and Faculty Member and Department, respectively. Assuming we only capture the last time each faculty member joined the university (department), we only need a ‘since’ attribute for the ‘affiliated with’ relationship. Total Participation and Key Constraints are identified on Faculty Member as they must have one and exactly one department. A new entity set called ‘Declared Program’ is introduced between Student and Department, as each student is required to have at least one major, and optionally one or more minor. Total Participation Constraint is identified as each student must have at least one major. And these requirements can be translated into the following ER Diagram in *Diagram 2.2.* Please note in the diagram, ‘ISA’ Inheritance was omitted for clear illustration purposes.

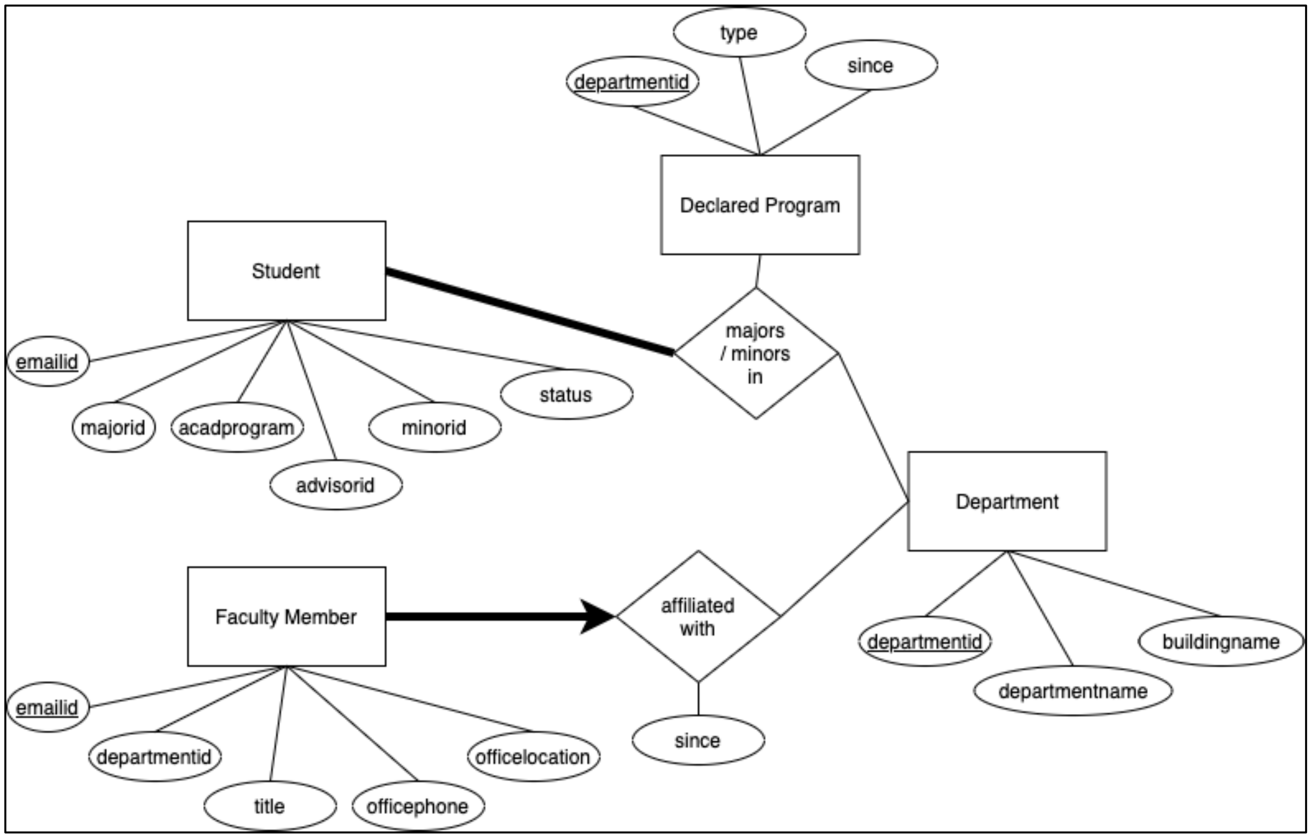

Diagram 2.2. ER Diagram for Student, Faculty Member, and Department

2.3. Course and Section Offerings

To translate course and section offerings into ER Diagram, we first need to see that sections are a weak entity set associated with courses, and when a course entry is removed from the system, all of its affiliated section offerings should also be removed. And a course can be uniquely identified by using a superkey that is composed of its subject and course number. And any section offering can be uniquely identified by a partial key, which is a composition of its semester and section number. Shown in the ER *Diagram 2.3.* below is one potential way to translate these requirements. Total Participation and Key Constraints are identified from section to teaching team, as all sections are required to be taught by exactly one team. Similarly, Total Participation Constraint is identified from teaching team to section because intuitively, all teams have to teach some section of some course. Please note that details about teaching team is omitted for now for better illustration and will be further discussed later in *Section 2.4*.

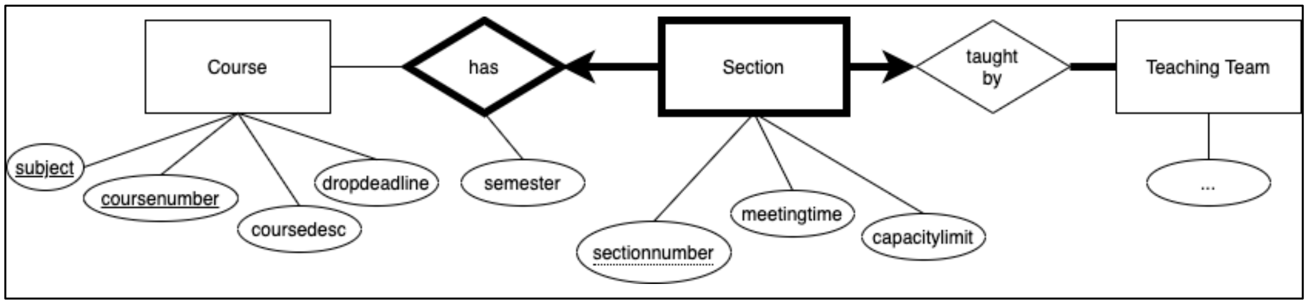

Diagram 2.3. ER Diagram for Course and Section

2.4. Teaching Team

A teaching team is consisted of at least one professor and one or more teaching assistant. Each section of the course should be taught by one and exactly one teaching team. And each section should have its own meeting time, capacity limit, and homework and exam assignment. A thorough discussion about assignments will be made later in *Section 2.5*. Shown next page in *Diagram 2.4.* is the ER Diagram translated for the teaching team and its relationship with faculty member and teaching assistant, respectively. Total Participation Constraint is identified between teaching team and faculty member as every teaching team is required to have one faculty member teaching. Each teaching team can be uniquely identified using its course subject, course number, semester, and section number altogether.

2.5. Class Enrollment

Students can enroll into classes by themselves through *NittanyPath*. One student can only enroll into one section of a certain course during the same semester, but s/he can enroll into multiple sections that are affiliated with different courses. Thus, given the course subject, course number, semester, and

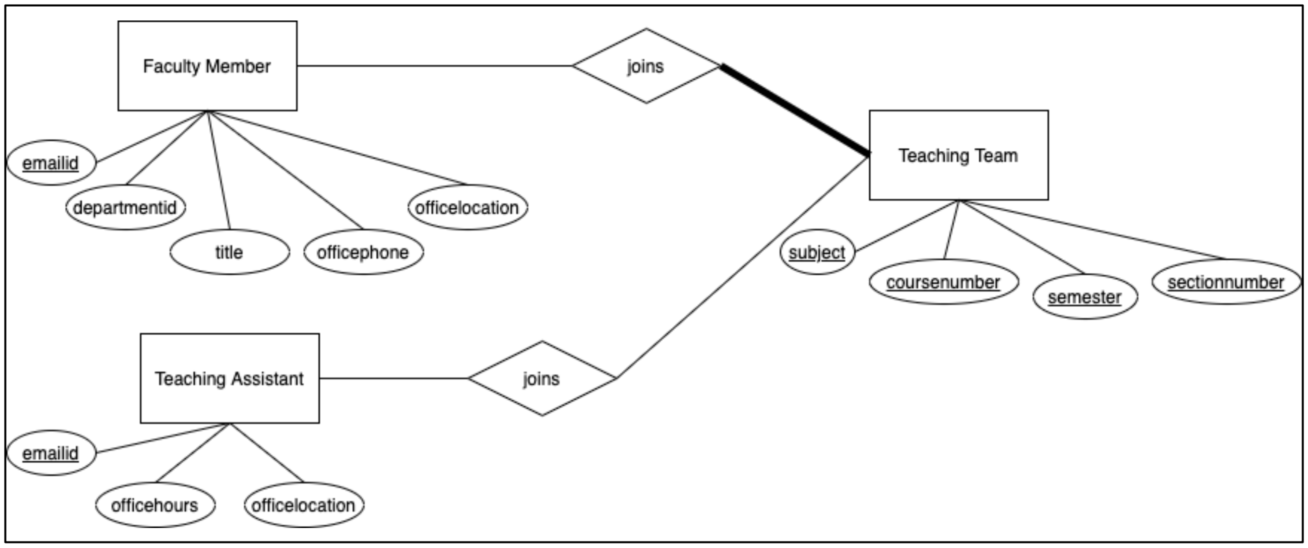

Diagram 2.4. ER Diagram for Teaching Team

student's information, we can uniquely identify the enrollment within the system. However, such constraints cannot be captured by the ER Diagram and requires assistance from programming language. Shown below in *Diagram 2.5.* is the ER Diagram translated for enrollment and its relationship with student. No Integrity Constraint is identified here, and users' requirements will be implemented using programming language's help. Certain details about section is omitted here for clearer illustartions.

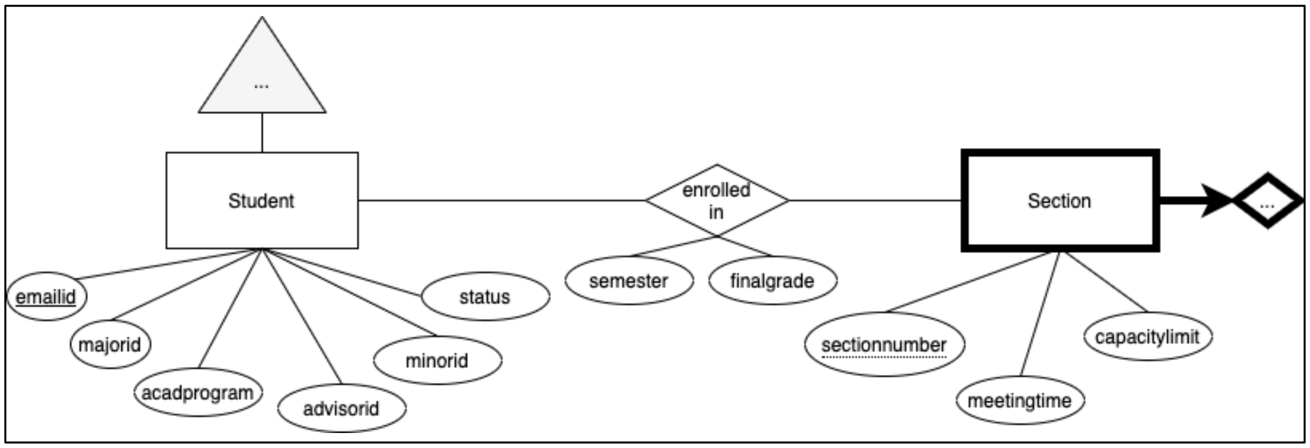

Diagram 2.5. ER Diagram for Student Enrolled in Section

2.6. Assignments and Grades

Assignments are students' graded work that is associated with the section, including homework, reports, exams and etc. Each section should have its own assignments and faculty member teaching the section should have access to create or delete any assignment. Only students currently enrolled in the section can see all its affiliated assignments and they can only see their own grades, and hence we are introducing an aggregation here – assignment is affiliated with a certain section and its grade is affiliated with a composition of student, section, and assignment altogether. Shown below in *Diagram 2.6.* is the ER Diagram translated for assignment and grade. Assignment is a weak entity set as it is affiliated with section. Key Constraint is identified as given the combination of student, section (of a course), and assignment, we should be able to identify its associated grade. Details about section is omitted here for better illustration purposes.

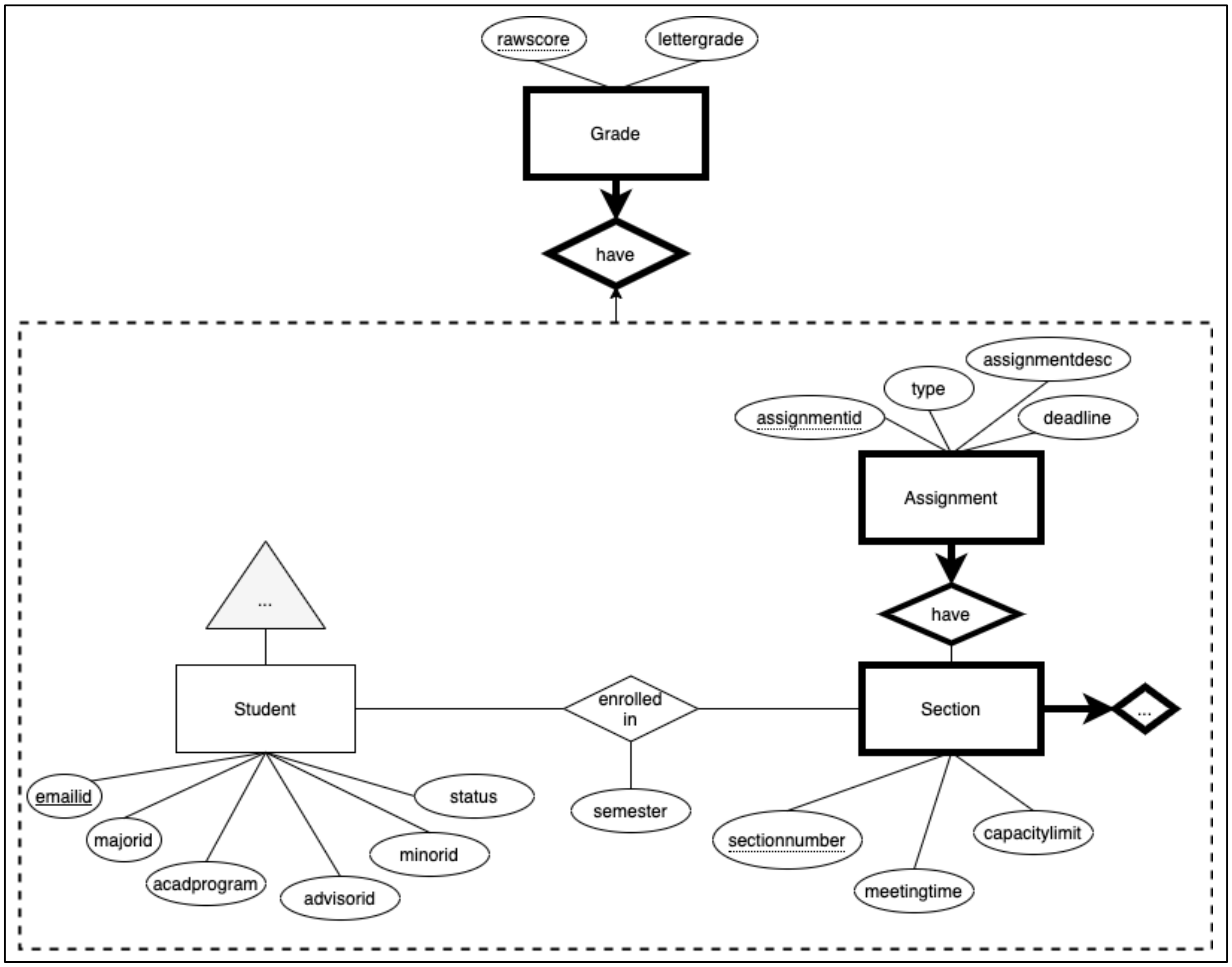

Diagram 2.6. ER Diagram for Assignment and Grade

2.7. Forum

Forum is a place where students can post questions about the coursework for other students, faculty members, and teaching assistant to reply to, and it is affiliated with each course section as well. In the forum, any class member can reply to an existing post while only teaching team members or the owner student of the post can delete it. Only students currently enrolled in the section and its teaching team can see the affiliated forum. Shown below in *Diagram 2.7.* is the ER Diagram translated for forum, post, and reply, all introduced as weak entity sets.

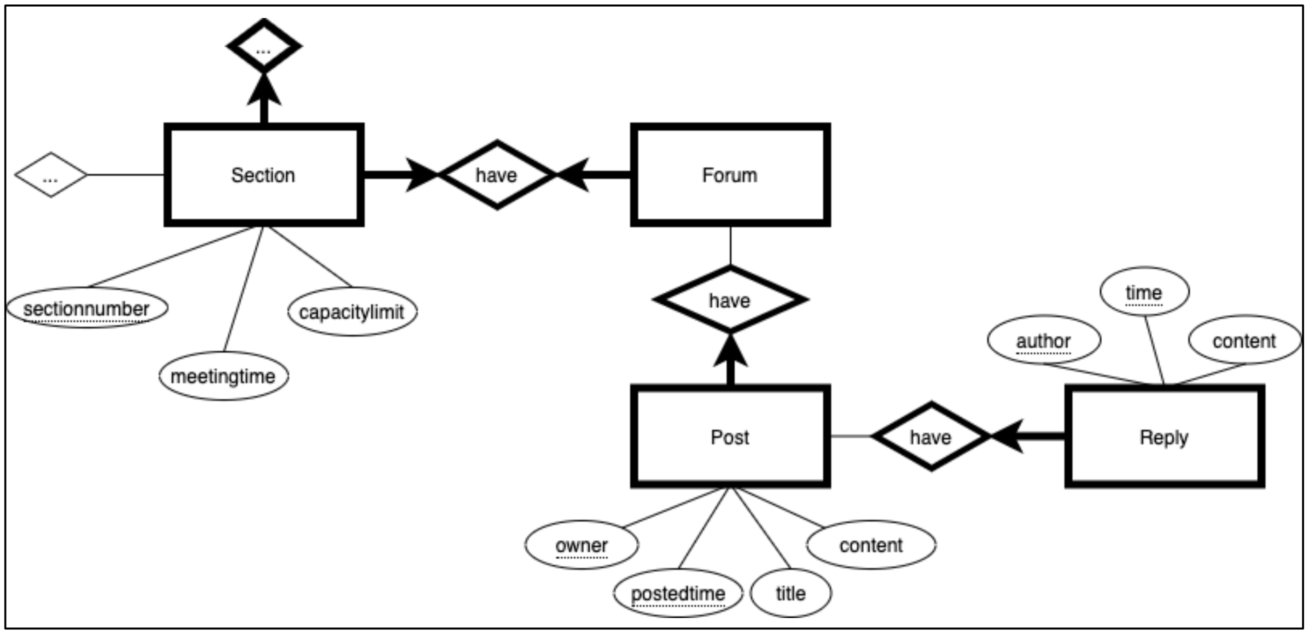

Diagram 2.7. ER Diagram for Forum, Post, and Reply

3. Technology Survey

As required by the Nittany State University, *NittanyPath* will be a system that lives online and to be accessed using a modern web browser (Firefox, Google Chrome, Safari etc.) Hence, this project will utilize web programming and database technologies in correspondence with current market trends and technology innovations. A web-application is consisted of front-end, framework, and back-end and in this section of the report, a thorough and detailed analysis is conducted, and a choice will be made on the most appropriate programming languages and tools for *NittanyPath* project.

3.1. Front-End Framework and User Interface (UI)

Due to the nature of this project being database-heavy and data-oriented, less efforts will be made on designing a fancy user interface with advanced front-end framework. The goal here is to make sure *NittanyPath* is clean and easy to navigate, while the functionalities are implemented through the back-end database and web framework of our choice. As a result, Hypertext Markup Language (HTML) has been identified as our front-end choice as it provides all basic functions and is well-adopted and supported. To be more specific, HTML5, the latest and recommended version by World Wide Web Consortium (W3C) will be used to develop the front-end of *NittanyPath*.

In addition, Bootstrap, a free and open-sourced CSS framework will also be used to increase the cleanness and consistence of the system. By coupling HTML5 and Bootstrap, it would be easy to both construct and maintain the front-end structure and user-interface for *NittanyPath*.

3.2. Web Framework and Programming Language

HTML5 and Bootstrap provides the interface of *NittanyPath*. And for web applications, we need to support various kinds of interactions between the user and the web application. According to Wikipedia[4], web frameworks "provide a standard way to build and deploy web applications on the World Wide Web". And it is preferred that we adopt an existing solution rather than reinvent the wheels. And that is when Web Frameworks come into play.

Popular web frameworks include React, AngularJS, Django, and Flask. To choose between these frameworks, we need to come up with several rubrics and by using a comparison and decision matrix, we can then analyze the pros and cons among these frameworks. The aspects concerning web frameworks are the easiness of its programming language, functionality counts, and supporting resources. Shown below in *Table 3.2.a* is a needs weighting matrix, where a weight of programmer's needs is decided for later comparisons.

		Ease of Pgm.	**Ease of Mtn.**	**Cost**	**Eff.**	**Func.**	**Total**	**Weight**
1	Ease of Programming	1.00	1.00	1.50	1.00	0.80	5.30	**0.25**
2	Ease of Maintenance	0.80	1.00	1.00	1.50	0.80	5.10	**0.25**
3	Cost	0.80	0.80	1.00	0.80	1.00	4.40	**0.10**
4	Efficient	0.80	0.80	1.00	1.00	0.80	4.40	**0.10**
5	Functionality	1.50	2.00	1.20	1.20	1.00	6.90	**0.30**

Table 3.2.a. Needs Weighting Matrix

Now with the calculated weightings of each need, we can then conduct a comparison between these four frameworks. Detailed comparisons and results are shown below in *Table 3.2.b*. And as shown in the diagram, the primary choice for web framework is Django as it is easy to program (with Python), has a lot of support (190,000+ posts on Stack-Overflow) and is more flexible and offers more functionalities than Flask. However, the learning curve for Django is higher than that of Flask. As a result, Django is chosen as the primary framework and Flask as the alternative.

		Web Frameworks							
		React		AngularJS		Django		Flask	
Selection Criteria	Weight	Rating	Wgtd. Score	Rating	Wgtd. Score	Rating	Wgtd. Score	Rating	Wgtd. Score
Ease of Prog.	0.25	JS 1	0.25	JS 1	0.25	Python 5	**1.25**	Python 5	1.25
Ease of Maint.	0.25	1	0.25	1	0.25	4	**1.00**	5	1.25
Cost	0.10	5	0.50	5	0.50	5	**0.50**	5	0.50
Efficient	0.10	3	0.30	3	0.30	4	**0.40**	4	0.40
Function.	0.30	2	0.60	3	0.90	5	**1.50**	4	1.20
	Total Score		1.90		2.20		**4.65**		4.60
	Rank		4		3		**1**		2
	Choice	No		No		Yes – Prim.		Yes – Alter.	

Table 3.2.b. Web Framework Scoring

3.3. Programming Tool and Integrated Development Environment (IDE)

Since we chose Django as our web framework, all the programming will be in Python. To achieve the best programming results, we need to choose an IDE with good support in web programming and database management system. According to guru99.com[5], the most popular Python IDEs in 2020 are PyCharm, IDLE, and Sublime Text. However, IDLE has poor support for web programming and does not provide adequate debugging tools, it was not an ideal choice for *NittanyPath*. And Sublime Text, although being lightweight and easy-to-use, does not have too many functionalities out-of-the-box. It also requires extra plug-ins to be installed in order to work with database systems. As a result, PyCharm will be used to develop *NittanyPath*.

3.4. Back-End Database System

According to db-engines.com[6], currently the most popular database systems are Oracle, MySQL, and Microsoft SQL. However, considering the scope of this project, both Oracle and Microsoft SQL are not a good fit here due to the high cost and learning curve of system set-up. As a result, MySQL will be the back-end database system used for *NittanyPath* project. In addition, due to the built-in feature of PyCharm, SQLite has been identified as an alternative solution due to its light weight and ease of set-up on local machines.

4. Logical Database Design and Normalization

4.1. Conceptual Database Design

Based on the Requirement Analysis and ER Diagrams illustrated in *Section 2*, we can first finalize the relational schema for the *NittanyPath* database.

4.1.1 *NittanyPath* Users

Among all *NittanyPath* users, they share some common attributes, and their basic information will be captured in the University Member Table. Shown below in *Table 4.1.1.a* is these common attributes that need to be captured and their corresponding domain and validation requirements.

ATTRIBUTE		DOMAIN	VALIDATION
Email ID (*Primary Key*)		`CHAR(7)`	'abc1234'
Legal Name		`CHAR(50)`	
Date of Birth (Age)		`DATE`	
Legal Gender		`CHAR(9)`	Male / Female
University ID		`CHAR(9)`	Digit Only
(Entity Set) Home Address	Street	`CHAR(50)`	
	City	`CHAR(20)`	
	State	`CHAR(2)`	Legal State Abbreviation
	Zip Code	`CHAR(5)`	Digit Only
Initial Login Password		`CHAR(50)`	

Table 4.1.1.a. Specifications for University Member Table

Inherited from University Member Table, there will be three addition tables – Student, Faculty Member, and Teaching Assistant (further inherit from Student). And shown below in *Table 4.1.1.b – 4.1.1.d* are their attributes, domains, and validations.

ATTRIBUTE	DOMAIN	VALIDATION
Email ID (*Primary Key*)	`CHAR(7)`	*Foreign Key*, Ref. University Member
Advisor ID		
Major (Department) ID	`CHAR(5)`	*Foreign Key*, Ref. Department
Minor ID		
Academic Program	`CHAR(5)`	Bachelor / Master / PhD
Status	`CHAR(10)`	Active / Graduated

Table 4.1.1.b. Specifications for Student Table

ATTRIBUTE	DOMAIN	VALIDATION
Email ID (*Primary Key*)	`CHAR(7)`	*Foreign Key*, Ref. University Member
Department ID	`CHAR(5)`	*Foreign Key*, Ref. Department
Title	`CHAR(20)`	
Office Phone	`CHAR(10)`	Digit Only
Office Location	`CHAR(50)`	

Table 4.1.1.c. Specifications for Faculty Member Table

ATTRIBUTE	DOMAIN	VALIDATION
Email ID (*Primary Key*)	`CHAR(7)`	*Foreign Key*, Ref. University Member
Office Hours	`CHAR(50)`	
Office Location	`CHAR(50)`	

Table 4.1.1.d. Specifications for Teaching Assistant Table

4.1.2. Department

To capture necessary information about departments, we propose the following conceptual database shown below in *Table 4.1.2*.

ATTRIBUTE	DOMAIN	VALIDATION
Department ID (*Primary Key*)	`CHAR(5)`	Letter Only
Department Name	`CHAR(50)`	
Building Name	`CHAR(20)`	

Table 4.1.2. Specifications for Department Table

4.1.3. Course and Section Offerings

Courses are associated with departments at Nittany State University. As a result, entries stored in the table should have its affiliated department. And to further distinguish between courses within the same department, an attribute called course number is introduced and it can only be a three-digit number (100-400 for freshmen-senior and 500-800 for graduate courses). Department ID has been identified as a foreign key and the course is required to be affiliated with a valid department.

Section offerings are different sections within the same course. Section is a weak entity set associated with Course and each section will have its own meeting time, teaching team, seat capacity. Two attributes (Section Number and Semester) are used to uniquely identify different sections under the same course. And by combining the Primary Key of Course with Partial Key of Section, we can uniquely identify every single section stored in *NittanyPath*. *Table 4.1.3.a – Table 4.1.3.c* below shows specifications for Course, Section, and Semester table.

ATTRIBUTE		DOMAIN	VALIDATION
(Superkey)	Department ID	`CHAR(5)`	*Foreign Key*, Ref. Department
	Course Number	`CHAR(3)`	Digit Only
Course Description		`CHAR(100)`	
Drop Deadline		`DATE`	

Table 4.1.3.a. Specifications for Course Table

ATTRIBUTE		DOMAIN	VALIDATION
(Superkey)	Department ID, Course Number	`CHAR(5),CHAR(4)`	*Foreign Key*, Ref. Course
	Semester	`CHAR(4)`	*Foreign Key*, Ref. Semester
	Section Number	`INTEGER`	Positive
Teaching Team		`INTEGER`	*Foreign Key*, Ref. Teaching Team
Meeting Time		`DATE`	
Capacity Limit		`INTEGER`	Positive

Table 4.1.3.b. Specifications for Section Table

ATTRIBUTE	DOMAIN	VALIDATION
Semester ID	`CHAR(4)`	e.g. 'SU19'
Start Date	`DATE`	
End Date	`DATE`	

Table 4.1.3.c. Specifications for Semester Table

4.1.4. Teaching Team

Given a course and section, we should be able to uniquely identify its corresponding teaching team. Teaching Teams are associated with Faculty Members and Teaching Assistants and each teaching team should have at least one faculty member and optionally one or more teaching assistant.

To comply with the First-Normal-Form (1NF) requirement, only atomic values can be stored. In order to enable *NittanyPath* store arbitrary numbered additional faculty member, a new table called Additional Teaching Team is introduced to store information about secondary teaching faculty and teaching assistant. Shown below in *Table 4.1.4.a-b.* are both tables' design specifications.

ATTRIBUTE		DOMAIN	VALIDATION
(Superkey)	<u>Department ID, Course Number</u>	`CHAR(5),CHAR(4)`	*Foreign Key*, Ref. Course
	<u>Semester</u>	`CHAR(4)`	*Foreign Key*, Ref. Semester
	<u>Section Number</u>	`INTEGER`	*Foreign Key*, Ref. Section
Primary Teaching Faculty		`CHAR(7)`	*Foreign Key*, Ref. Faculty Mbr.

Table 4.1.4.a. Specifications for Teaching Team

ATTRIBUTE		DOMAIN	VALIDATION
(Superkey)	<u>Department ID, Course Number</u>	`CHAR(5),CHAR(4)`	*Foreign Key*, Ref. Course
	<u>Semester</u>	`CHAR(4)`	*Foreign Key*, Ref. Semester
	<u>Section Number</u>	`INTEGER`	*Foreign Key*, Ref. Section
	<u>Member ID</u>	`CHAR(7)`	*Foreign Key*, Ref. University Mbr.
Type		`CHAR(2)`	Faculty / TA

Table 4.1.4.b. Specifications for Additional Teaching Team Members

4.1.5. Class Enrollment

Class Enrollment Table keeps track of all enrollment information in *NittanyPath*. And each enrollment can be uniquely identified given a student, a section of the course, and semester. Hence, the primary key needed here is a superkey by combining student's email ID, course subject and number, section number, and semester. Refer to Table *4.1.5.* on next page for more details about this table.

4.1.6. Assignment

Assignment includes all work that needs to be graded and is associated with each section. Each section can have several different assignments, of various types (homework, attendance, exam etc.) Hence, a new table is needed here, and additional information needs to be captured include description and deadline. Shown next page in *Table 4.1.6.* is the specification for assignment table.

ATTRIBUTE		DOMAIN	VALIDATION
(Superkey)	<u>Department ID, Course Number</u>	`CHAR(5),CHAR(4)`	*Foreign Key*, Ref. Course
	<u>Semester</u>	`CHAR(4)`	*Foreign Key*, Ref. Semester
	<u>Section Number</u>	`INTEGER`	*Foreign Key*, Ref. Section
	<u>Student ID</u>	`CHAR(7)`	*Foreign Key*, Ref. Student
Enrollment Time		`TIMESTAMP`	
Final Grade		`CAHR(2)`	A-F, IP, LD

Table 4.1.5. Specifications for Enrollment

ATTRIBUTE	DOMAIN	VALIDATION
<u>Assignment ID</u> (*Primary Key*)	`INTEGER`	
Department ID, Course Number	`CHAR(5),CHAR(4)`	*Foreign Key*, Ref. Course
Semester	`CHAR(4)`	*Foreign Key*, Ref. Semester
Section Number	`INTEGER`	*Foreign Key*, Ref. Section
Type	`CHAR(4)`	HW, Exam, Attn
Description	`CAHR(100)`	
Deadline	`TIMESTAMP`	

Table 4.1.6. Specifications for Assignment

4.1.7. Assignment Grade

For every assignment, each student has their own grade. And an aggregation relationship was identified between enrollment and grade. Information that needs to be captured here includes raw score and letter grade. Shown below in *Table 4.1.7.* is the table's specifications.

ATTRIBUTE	DOMAIN	VALIDATION

(Superkey)	Assignment ID (*Primary Key*)	`INTEGER`	*Foreign Key*, Ref. Assnmt.
	Student ID (*Primary Key*)	`CHAR(7)`	*Foreign Key*, Ref. Student
	Raw Score	`REAL`	
	Letter Grade	`CHAR(2)`	A-F, IP

Table 4.1.7. Specifications for Assignment's Grade

4.1.8. Forum, Post, and Reply

Each course has its own forum, across its different affiliated sections. And each forum has its affiliated posts and each post has its affiliated reply. Given a specific section of the course, there is one and only one forum, so we are adding one field into the Section table to reflect that (*Table 4.1.8.a*). Additional information about Post includes owner ID, post time, title, and content. And additional information about Reply includes author ID, time, and content. Refer to *Table 4.1.8.b.-c.* for their design specifications.

ATTRIBUTE		DOMAIN	VALIDATION
(Superkey)	Department ID	`CHAR(5)`	*Foreign Key*, Ref. Department
	Course Number	`CHAR(3)`	Digit Only
Course Description		`CHAR(100)`	
Drop Deadline		`DATE`	
Forum ID		`INTEGER`	Positive

Table 4.1.8.a. Updated Specifications for Course Table

ATTRIBUTE		DOMAIN	VALIDATION
(Superkey)	Forum ID	`INTEGER`	*Foreign Key*, Ref. Course
	Owner	`CHAR(7)`	*Foreign Key*, Ref. University Mbr.
	Posted Time	`TIMESTAMP`	
Title		`CHAR(50)`	
Content		`CHAR(500)`	

Table 4.1.8.b. Specifications for Post Table

ATTRIBUTE		DOMAIN	VALIDATION
(Superkey)	Forum ID, Owner, Posted Time	`INTEGER`	*Foreign Key*, Ref. Post
	Author	`CHAR(7)`	*Foreign Key*, Ref. University Mbr.

	<u>Time</u>	`TIMESTAMP`	
Content		`CHAR(500)`	

Table 4.1.8.c. Specifications for Reply Table

4.2. Functional Dependencies (FDs) and Normalizations

4.2.1. NittanyPath Users

In addition to the specifications explained in Section 4.1, shown below is the schema of NittanyPath of all users, including Faculty Member, Student, and Teaching Assistant.

$$\begin{aligned}
\text{UniversityMember}(&\underline{emailid}, legalname, dob, legalgender, universityid, street, city, state, zipcode, password)\\
\text{Student}(&\underline{emailid}, advisorid, majorid, minorid, acadprogram, status)\\
\text{FacultyMember}(&\underline{emailid}, departmentid, title, officephone, officelocation)\\
\text{TeachingAssistant}(&\underline{emailid}, officehours, officelocation)
\end{aligned}$$

Diagram 4.2.1.a. Schema of *NittanyPath* Users

And with this schema, we can identify three FDs shown below:

$$zipcode \rightarrow (state, city)$$

$$emailid \rightarrow (password, universityid)$$

And we can apply schema normalizations by decomposing University Member table into three new tables and the new schema is shown below in *Diagram 4.2.1.b*, and these six tables are all in Boyce-Codd-Normal-Form (BCNF) as every non-key attribute is determined by the key attribute.

$$\begin{aligned}
\text{UniversityMember}(&\underline{emailid}, legalname, dob, legalgender, street, zipcode)\\
\text{Address}(&\underline{zipcode}, state, city)\\
\text{LoginCredential}(&\underline{emailid}, password, universityid)\\
\text{Student}(&\underline{emailid}, advisorid, majorid, minorid, acadprogram, status)\\
\text{FacultyMember}(&\underline{emailid}, departmentid, title, officephone, officelocation)\\
\text{TeachingAssistant}(&\underline{emailid}, officehours, officelocation)
\end{aligned}$$

Diagram 4.2.1.b. Normalized Schema of *NittanyPath* Users

4.2.2. Department

The following schema shown in *Diagram 4.2.2.* is already in BCNF and does not need further normalization.

$$\text{Department}(\underline{deptid}, deptname, deptbuildingname)$$

Diagram 4.2.2. Schema of Department

4.2.3. Course, Section, and Forum

All three schema shown on next page are already in BCNF and do not need further normalization.

$$\begin{aligned} \text{Course}&(\underline{subject, coursenumber}, coursedesc, dropddl, forumid) \\ \text{Section}&(\underline{subject, coursenumber, semester, sectionnnumber}, teachingteam, meetingtime, capacitylimit) \\ \text{Semester}&(\underline{semesterid}, startdate, enddate) \end{aligned}$$

Diagram 4.2.3. Schema of Course, Section, Semester, and Forum

4.2.4. Teaching Team

Both schemas shown below are already in BCNF and do not need further normalization.

$$\begin{aligned} \text{TeachingTeam}&(\underline{subject, coursenumber, semester, sectionnnumber}, facultyid) \\ \text{AdditionalTeachingTeam}&(\underline{subject, coursenumber, semester, sectionnnumber, memberid}, type) \end{aligned}$$

Diagram 4.2.4. Schema of Primary- and Additional-Teaching-Team

4.2.5. Enrollment, Assignment, and Assignment Grade

All three schema below are already in BCNF and do not need further normalization.

$$\begin{aligned} \text{Enrollment}&(\underline{subject, coursenumber, semester, sectionnnumber, studentid}, enrollmenttime, finalgrade) \\ \text{Assignment}&(\underline{assignmentid,}subject, coursenumber, semester, sectionnnumber, type, description, assignmentddl) \\ \text{AssignmentGrade}&(\underline{assignmentid,}rawscore, lettergrade) \end{aligned}$$

Diagram 4.2.5. Schema of Enrollment, Assignment, and Assignment Grade

4.2.6. Forum Post and Reply

Both schemas shown below are already in BCNF and do not need further normalization.

$$\begin{aligned} \text{Post}&(\underline{forumid, ownerid, posttime}, title, content) \\ \text{Reply}&(\underline{forumid, ownerid, posttime, replyauthorid, replytime}, content) \end{aligned}$$

Diagram 4.2.6. Schema of Forum Post and Reply

Appendix A. References

1. Lee, W. (2020). Overall Requirements. *SP2020 CMPSC 431W Database Management Systems.* 2.
2. Lee, W. (2020). Phase I – Database Design and Technology Survey. *SP2020 CMPSC 431W Database Management Systems*. 3-7.
3. Ramakrishnan, R. and Gehrke, J. (2003). *Database management systems*. Boston: McGraw-Hill. pp. 14.
4. *Web framework*. En.wikipedia.org. (2020). Retrieved 5 March 2020, from https://en.wikipedia.org/wiki/Web_framework.
5. *11 BEST Python IDEs in 2020*. Guru99.com. (2020). Retrieved 6 March 2020, from https://www.guru99.com/python-ide-code-editor.html.
6. *DB-Engines Ranking*. DB-Engines. (2020). Retrieved 6 March 2020, from https://db-engines.com/en/ranking.