Final Document

SCIS Curriculum Management

by

Gabrielle Moestar and Oscar Aparicio

Course Instructor: Masoud Sadjadi

Mentor: Tim Downey

Client: Eric Johnson

CIS 4911 – Senior Project

12/04/2012

COPYRIGHT AND TRADEMARK NOTICES

Contact Information

Florida International University - Modesto Madique Campus

School of Computer and Information Sciences

11200 SW 8th Street, Room ECS-258

Miami, FL 33199

SUMMARY

SCIS Curriculum Management System intends to serve as a centralized location where course information and requirements to the curriculum can be applied, sophisticated enough for the experts, but user friendly enough for the non tech savant users.

This final document describes the aspects of the project from the beginning to the end.

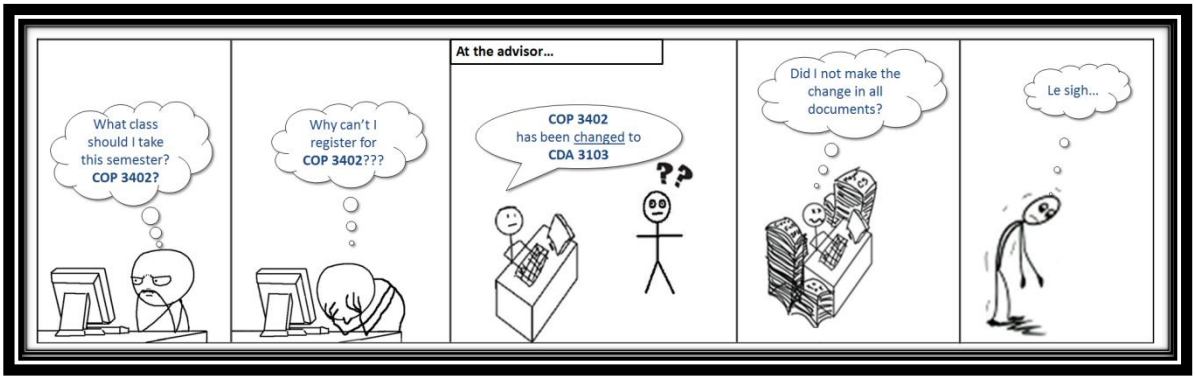
TABLE OF CONTENTS

1. Introduction

The SCIS Management system is a content management system designed to easily create and manipulate curriculum information with the help of a user-friendly web based application that will retrieve data from a database containing all course information and course requirements.

1.1 Problem Definition

The problem defined for this current system is the inability to easily create and maintain course information and requirements of the curriculum across documents consistent, which allows confusion for students.



This information can be obtained through different sources, an advisor, the departmental website, the catalog, and a major map. The data distribution through these documents appears to be inconsistent, and not in sync. (Undergraduate Programs)

The reason for this unfortunate inconvenience is initiated at the creation and maintenance of the curriculum by the curriculum committee.

1.2 Scope of System

The SCIS Curriculum Management system will be developed to serve as a central repository and main source of data to generate all the documents that needed for the documents.

The new system will have a graphical user interface that allows creation and modification to the curriculum. It will allow regular users, students, to view the curriculum, and allow special users with editor privileges, to create and modify the curriculum.

The system will permit flexibility for additional features in the future.

1.3 Overall Development Methodology

The SCIS Management system has been implemented with a relatively new methodology software development known as Agile Software Development.

Agile Software Development is based on iterative and incremental development to perform in a highly collaborative manner. It is a conceptual framework that encourages rapid and flexible response to change. This method promotes iteration which may not be functional enough for production, but the goal is to have an available release with minimal bugs at the end of each iteration.

In other words, a feature is developed and implemented until it's functional enough to present to the client, who in turn has to approve its current state. The project team proceeds to the next phase, and repeat the process until the product is finalized.

1.4 Definitions, Acronyms, and Abbreviations

SCIS – School of Information and Computing and Information Sciences

Major Map – A program of study designed to help you graduate in a timely manner

Catalog – A complete enumeration of major's offered by the department with it associated courses.

Students – Users with read only privileges

Advisors – Users with read and write privileges

CRUD – Create, Retrieve, Update, Delete

DGU – Degree Granted United: A college/school offering majors.

Subsystem - A group or library of well-related classes in a project

MVC – Model View Controller

1.5 Overview of Document

Chapter 2 – Feasibility Study describes solutions imagined for the given problem definition, considerations, and reasoning for the selected solution.

Chapter 3 – Project Plan lists the organization for the project and describes the tasks, milestones, and deliverables.

Chapter 4 – System Requirements defines both functional and nonfunctional requirements.

Chapter 5 – System Design provides high level description about the overall design of the system, its decomposed subsystems, hardware to software mapping, data management, and finally address some security and privacy issues.

Chapter 6 – Detailed design provides detailed information about the project and visually depicts the system with models.

Chapter 7 – System Validations is not presented in this project due to time constraint.

Chapter 8 – Glossary and Chapter 9 – Appendix allow additional and visual representations of the content that has been displayed in the entire document.

2. Feasibility Study

2.1 Current System

The current system yields towards old file system maintenance. Changes to the curriculum are approved by the faculty and sent to the university curriculum committee to be reviewed and approved.

Varied documents are created that are based on this information: catalogs, pamphlets, fliers, web sites, plans of study, major maps.

When changes are made to the curriculum, someone has to be sure that all the documents are updated.

Over time, the contents of these documents diverge and contain outdated information.

2.2 Alternative Solutions

2.2.1 Description of Alternatives

There are different alternatives of the database design that would meet client's requirements. Discussed below are the three most contrasting and controversially alternatives.

The first option was a straight forward design where maintaining and displaying information about the curriculum and classes provided now to students and faculty uses the following approach. Every semester, a new table with all possible information about the classes offered by the school will be added to the database. Note that no other table is created.

A second approach for the design of the database cuts down data redundancy, and brings relational structure to the data stored. This approach will bring a relational database design, which normalized up to the fourth normal form that features a history set of tables. This set of tables is an exact copy of the core entities where new records will be produced every single time a new catalog (version) is created. To avoid increasing data and slow down database queries, only activated records will be on the curriculum set of tables and proposed/prospective records will remain in the history set of tables.

Finally, we envisioned a more sophisticated approach, where we would treat the changes in data as version. This approach would help manage creating and updating multiple releases of all entities needed to keep track of changes. This approach consists on a 2-tuple entity divided on the identifier table and the data table. The identifier would be a unique record that will differentiate the entity. The second element of the 2-tuple entity is the data table. This table is where the different versions of the data will remain to keep track of all changes generated in every new catalog year.

2.2.2 Selection Criteria

Decision making at all levels is about information collection, evaluation and tradeoffs to analyze complex problems. For this project, we will take the following criteria in to account:

- Data redundancy
- Usability
- Expandability
- Maintainability

2.2.3 Analysis of Alternatives

Based on the criteria described in the previous section, we are going to analyzed all three proposed approaches to select the better approach and the one that fits best for the system needs.

The first approach will bring a lot of data redundancy to the database, becoming a giant set of tables that contains the same information with, possibly, minor changes. Data will grow up exponentially saturating the database server at some point.

This approach would not completely complain with the usability and maintainability we want for the system, since we should change the source code every semester to reflect the changes of the tables.

The second approach complies more with our needs for maintain little data redundancy, and there would not be changes to the database at all. Since the database would be

fully normalized, data redundancy will be reduce significantly, but not enough considering all the data needed to change every term.

This approach would be fine for our purposes of the system, however at the time of implementation we discover the inability to completely implement this the way we expected. This was, since there are going to be two tables of the same information and we chose to InnoDb as our database engine, foreign key relationships where just broken. Therefore, we decided to move to the next approach.

The last approach meets our selection criteria. Formalized database with entities that can be versioned will bring a very low data redundancy allowing a better way of managing this information. The expandability still exists, since the overall design was kept as it was thought, and maintainability will increased due to integrity relationships between entities. Therefore, this approach is the one we decided to implement.

3. Project Plan

3.1 Project Organization

This section describes in detail the roles of all group members throughout the major phases of the project, as well as the resources that will be needed.

3.1.1 Project Personnel Organization

Roles are not expected to change throughout the different phases of the project. Since this is a 2 person group, team members were forced function on multiple roles.

Name	Project Role
Gabrielle Moestar	Project Manager | Documenter | Front-End Developer | DBA | Tester
Oscar Aparicio	Project Manager | Version Control | Back-End Developer | DBA | Tester

The code that is to be written will be divided in front-end design and back-end work.

The front-end design is

3.1.2 Hardware and Software Resources

This section covers both the hardware and software resources needed to complete this project and develop the requested system.

Hardware	Software
Server to Host Database	Ubuntu Server 12.10 LTS, 64 bit
Computer	Netbeans IDE 7.2.1
	Java SE Runtime Environment 1.7
	L.A.M.P.
	phpMyAdmin
	Microsoft Word 2007-2010
	Microsoft PowerPoint 2007-2010
	Visual Paradigm 8.3
	MacOSX MySQL Workbench 5.2

3.2 Tasks, Milestones and Deliverables (work breakdown)

	Tasks	Task Dependencies
1	Review current system	x
2	Problem Definition	1
3	Obtain High Level User Requirements	2
4	Identify Alternative Solutions	3
5	Determine Solution and Recommendation	4
6	Hardware / Software Identification and Requests	3,5
7	Requirement Analysis	2
	Milestone: System Analysis	
8	Create Database Architecture	7
9	Populate Database	8
10	Create System Architecture	7
11	Tentative UI Design	7,8,10
12	Implement Back-end	8,10,11
13	Initial Testing	8,10, 11, 12
	Milestone: Tentative System	
14	Finalize System Implementation	13
15	Complete Functional Testing	12, 13
16	Evaluate Test Results	15
17	Arrange Final Document and Presentation	14, 15, 16
	Tasks	Task Dependencies

4. System Requirements

4.1 Functional and Nonfunctional Requirements

The intended system shall accomplish the following for a student i.e. a regular user. The system shall:

- Allow the user to choose a major, track, and year he got accepted to the university.
- Allow the user to view the curriculum information.

The intended system shall accomplish the following for an editor user i.e. an advisor. The system shall:

- Allow the user to login and logout.
- Allow the user to create, manipulate, and modify curriculum information and course requirements.
- Allow the user to create, manipulate, and modify majors and tracks.

The intended system shall accomplish the following seen as a whole. The system shall:

- Provide a database stored on SCIS network servers.

4.2 Nonfunctional Requirements

The non-functional requirements for the system are as follows:

Usability

The user interface should be understandable to non-technical users, allowing them to navigate to view the curriculum information. The font of the graphical user interface of the system should be clear and easy to read. Color scheme should make use of FIU's color schemes, yet maximize contrast i.e. light background with dark foreground.

Reliability

The system should be highly available, with 99% up time. Maintenance of the system should not be required more than once in a quarter year.

Performance

The system should respond within less than two seconds for any user action, including curriculum information retrieval, update submission, and any other user interaction with the system. The system should be available 24/7, with downtime allowed as specified in the above.

Supportability

The system will not interfere with existing curriculum data provided by the SCIS department nor its current services. The system will only be available in English. The system is web-based, therefore, compatible with any operating system that can run a supported web-browser mentioned below and connect to the internet.

Implementation

The system will be a web-based application supported in Internet Explorer 7+, Firefox 5+. The system should support cross-platform compatibility without the need to change ports across platforms.

5. System Design

This chapter provides a high-level description of the system design. First, the decomposition of the system with its associated subsystems will be presented, followed by mapping hardware to each software. The third section excessively elaborates on the data management and its structure. Finally, user security and privacy are addressed.

5.1 Overview

SCIS Curriculum Management was build up on two architectural patterns: The MVC patterns, and the Data Repository pattern.

The MVC pattern separates the representation of data from the user's interaction with it. MVC enforces better organization and code reuse.

The Data Repository pattern is used since all the course information and requirements pertain to the central database created for SCIS Curriculum Management system. For relationships and data management refer to **Chapter 5.4.**

5.2 Subsystem Decomposition

The following section provides the subsystem decomposition with a brief description.

- Framework: provided by Yii Framework[1]
- Controller: This part of the system handles all entity controls
- Components: This subsystem deals with the versioned entities and its concrete clases
- Models: This subsystem handels all database features to connect to the system.

5.3 Hardware and Software Mapping

[1] http://www.yiiframework.com/

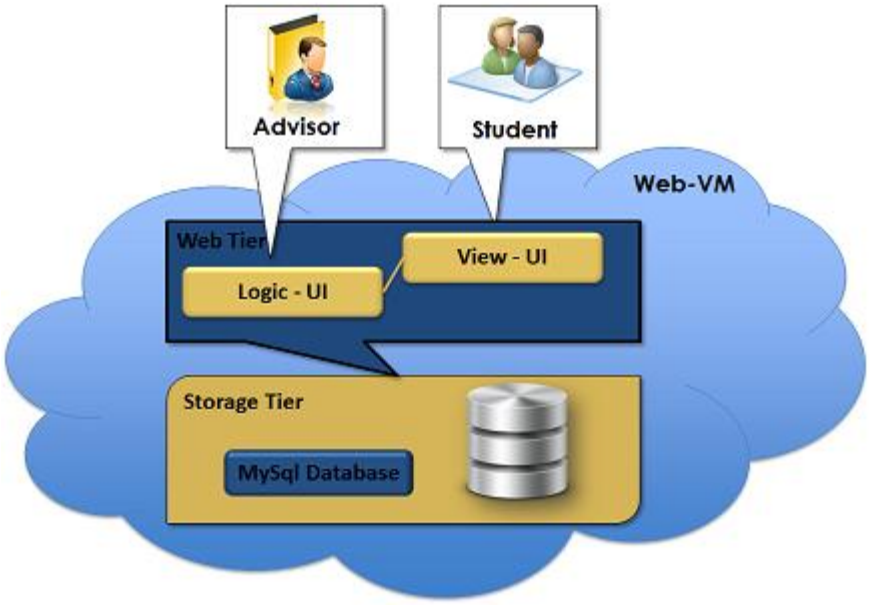


Figure 5-1 Hardware to Software tier mapping

5.4 Persistent Data Management

Appendix F visually depicts the relationships between the data and entities.

5.5 Security/Privacy

The database will be hosted on SCIS department servers, and therefore will be protected by the integrated firewalls on the network.

6. Detailed Design

This chapter will present the organization for the project as a whole, describing roles of all team members throughout the various phases of development. The chapter closes with a description of all major tasks, milestones, and deliverables.

6.1 Overview

This section describes the behavior and structure of curriculum subsystem.

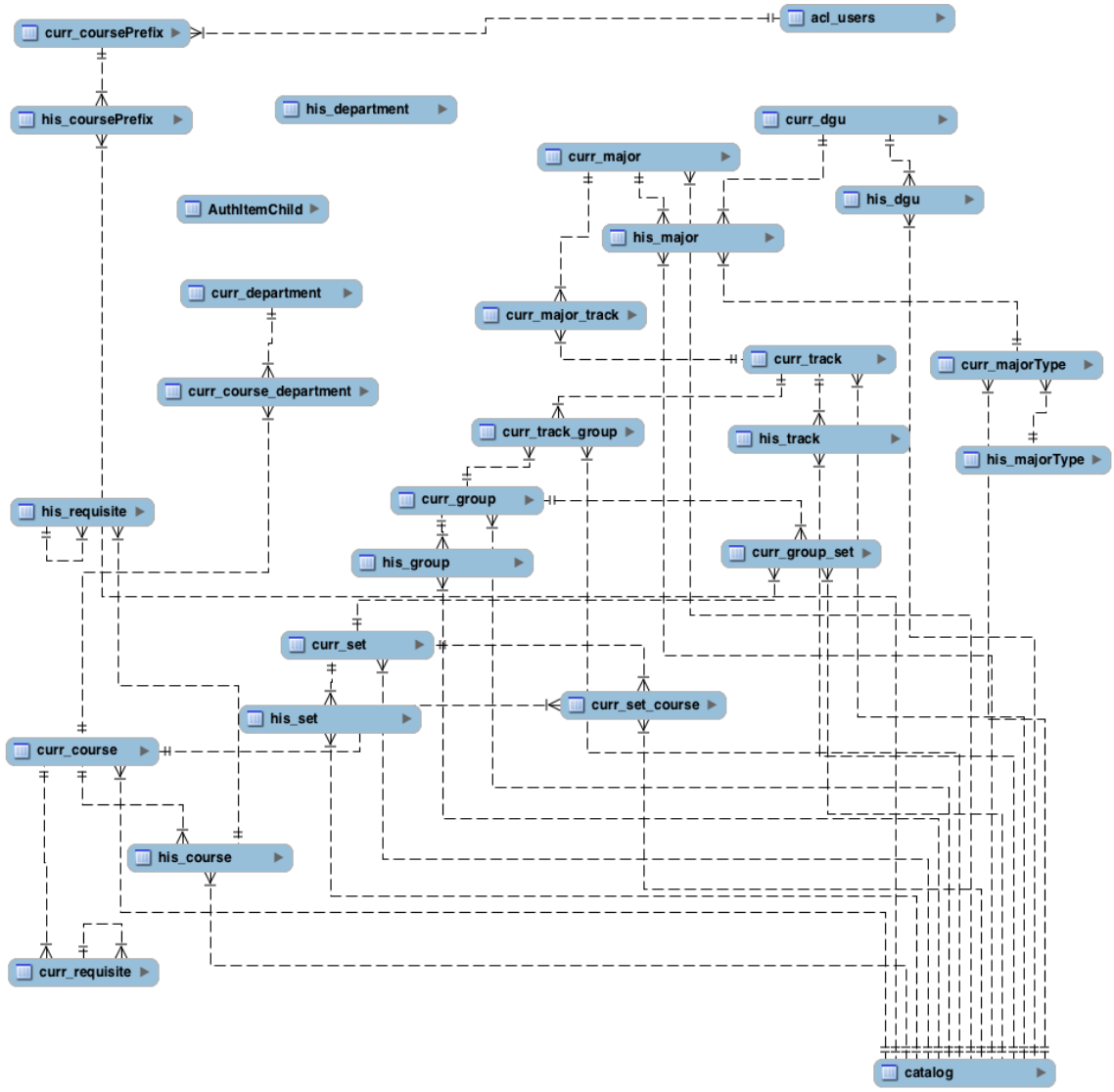


Figure 6-1 Database system structure

6.2 Static Model

The following is a detailed description of the curriculum components in the main subsystem

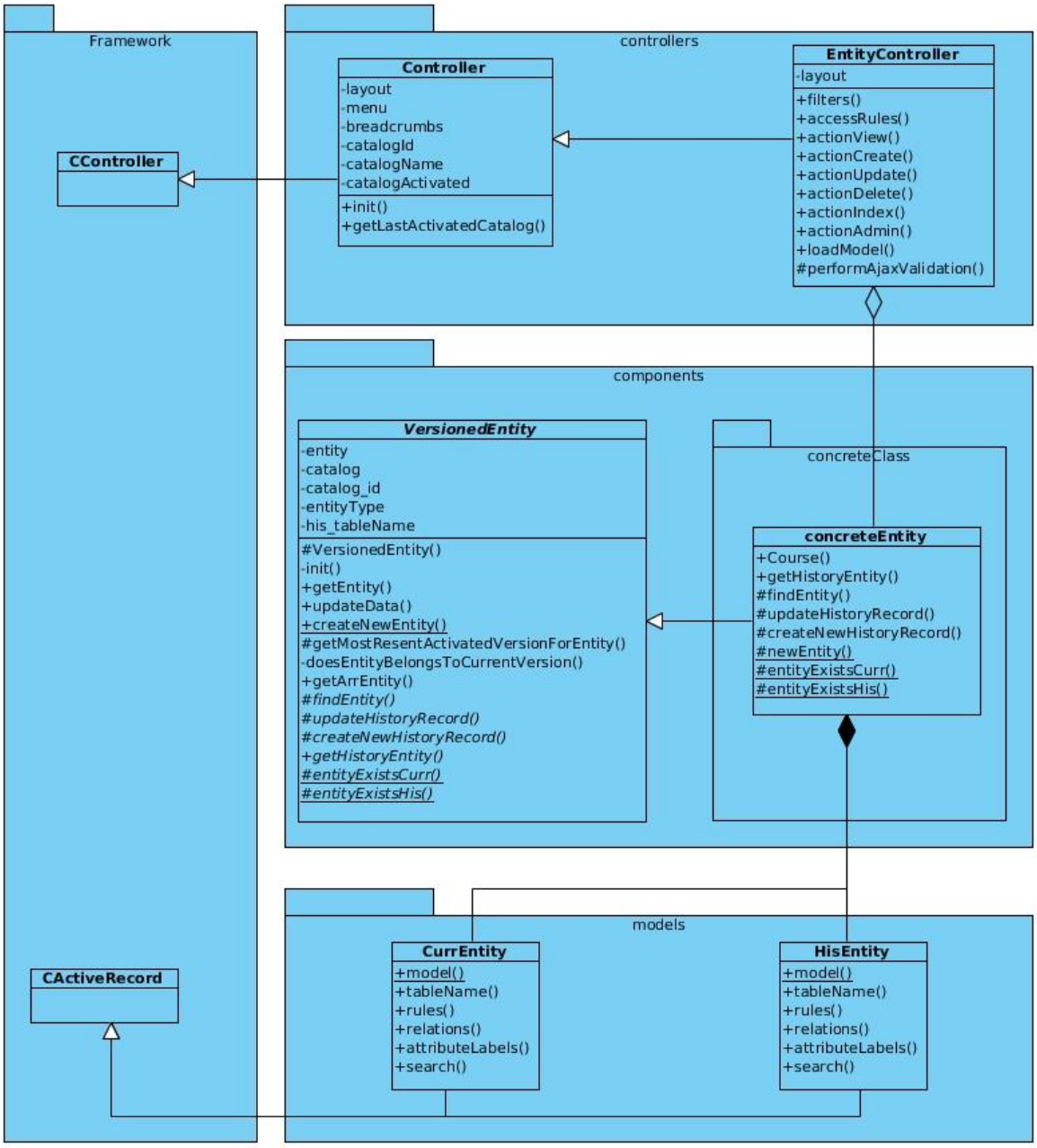


Figure 6-2 Overall Class Diagram

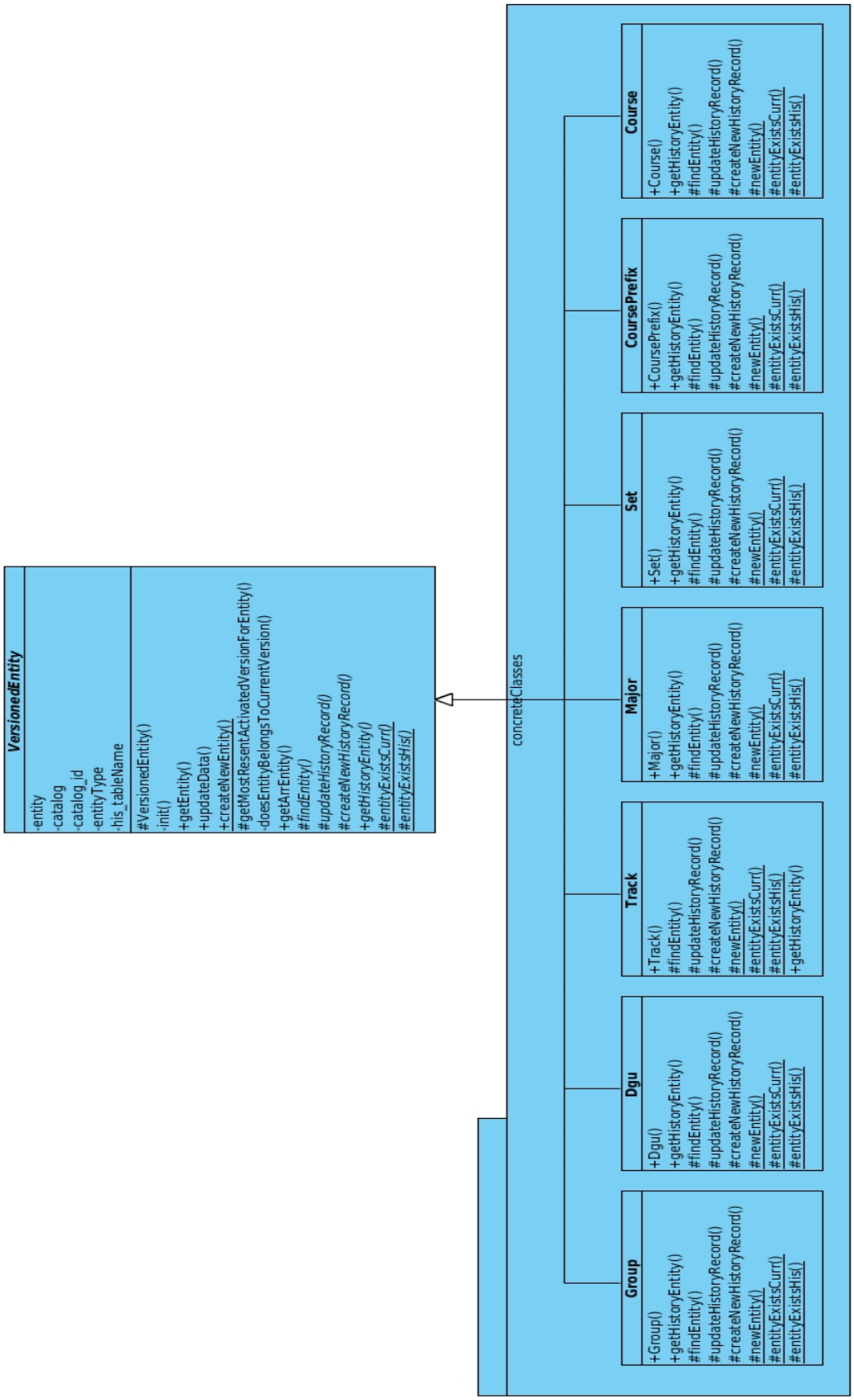


Figure 6-3 Detailed class diagram

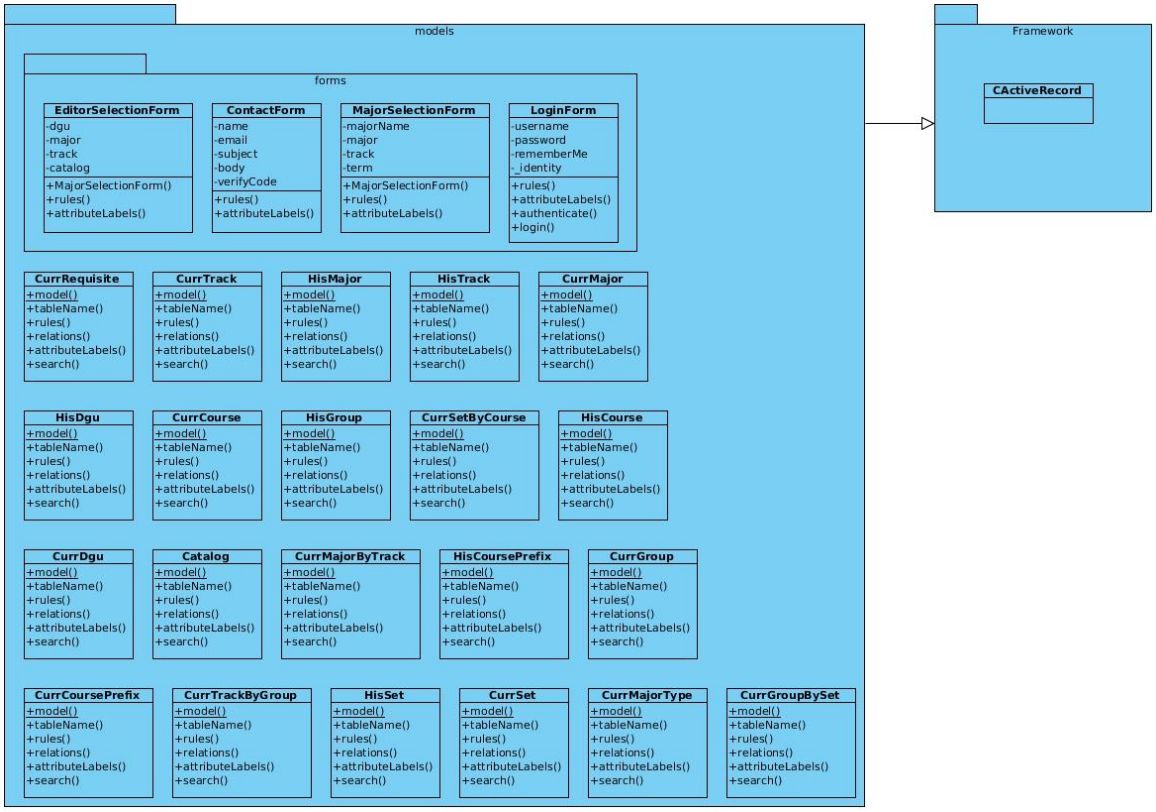


Figure 6-4 Model class diagram

6.3 Dynamic Model

This section presents state machine diagrams for the SCIS Curriculum Management's database behavior.

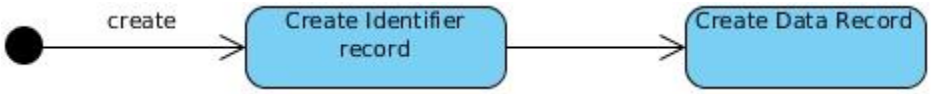


Figure 6-5 CreateEntity

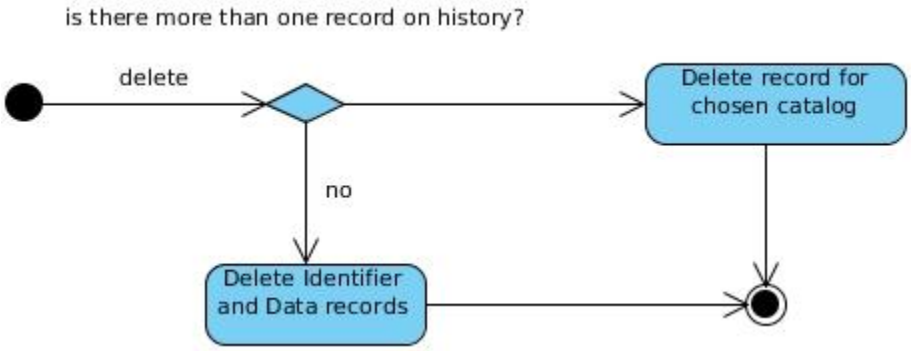


Figure 6-6 UpdateEntity

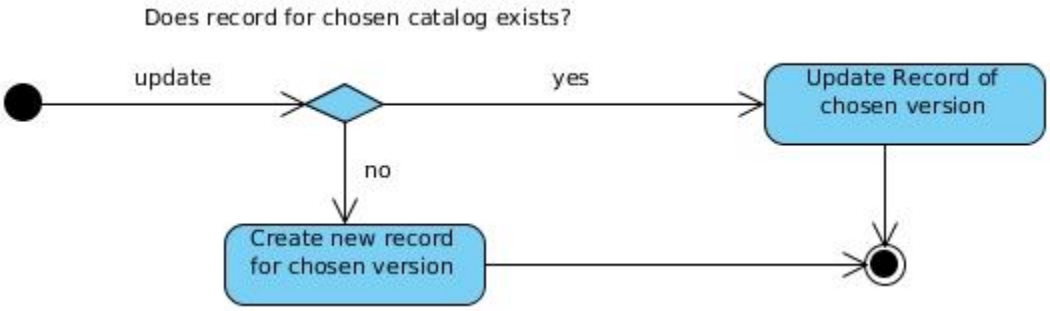


Figure 6-6-7 DeleteEntity

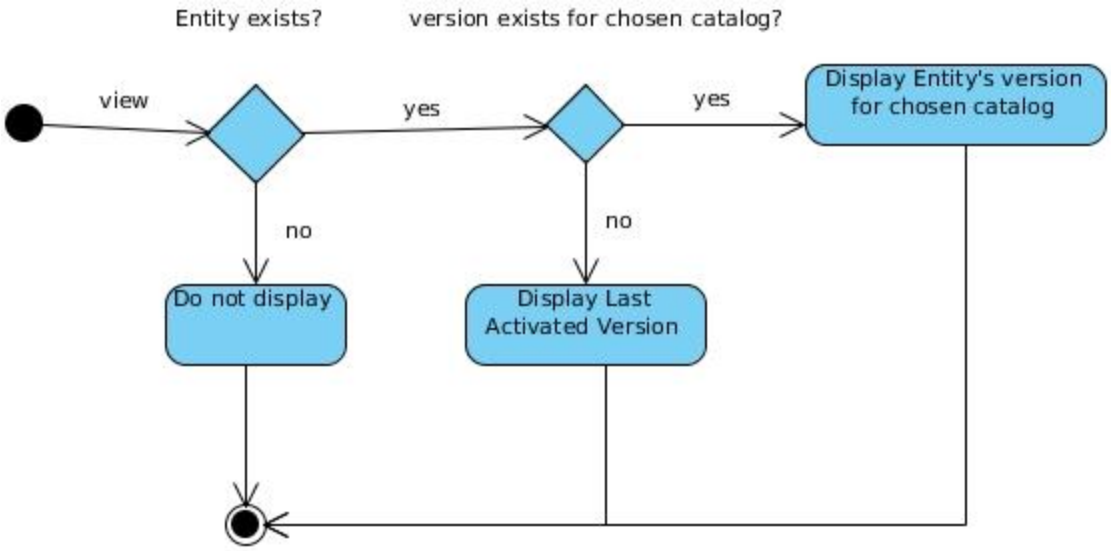


Figure 6-8 ViewEntity

6.4 Code Specification

Disclaimer: Refer to comments throughout the code as well as its attached document.

7. System Validation

Due to time constraint, we had to change the project schedule around. This did not allow us for an exhaustive test and stage environment. Therefore, the testing phase has been postponed until production.

8. Glossary

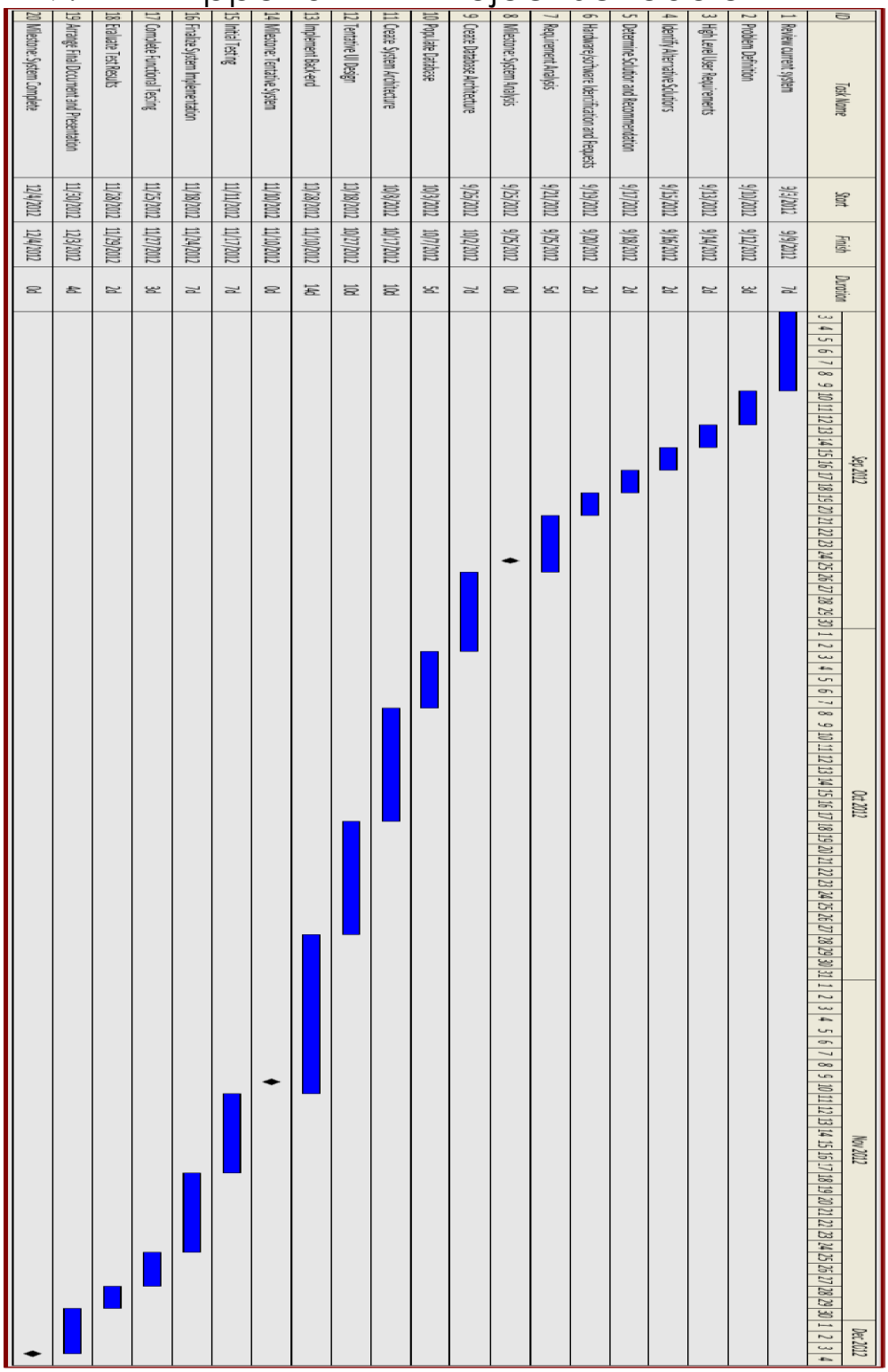
- **Functional requirement**: A description of a set of inputs, behavior, and output of a software system.
- **Non-functional requirements**: A requirement that specifies criteria used to judge the system.

  Class Diagram: A diagram that describes the structure of a system by showing the system's classes, attributes, and their relationship to other classes.

- **Dynamic Model**: A diagram containing a state machine depicting the main control object in a given subsystem.
- **Sequence Diagram**: A diagram that shows how objects operate with one another and their workflow.
- **Static Model**: A diagram providing a detailed description of the structure of a given subsystem.
- **Subsystem**: A group or library of well-related classes in a project.
- **Use Case**: A description of a potential series of interactions between software and an actor.

9. Appendix

9.1 Appendix A – Project Schedule



9.2 Appendix B – Use Cases

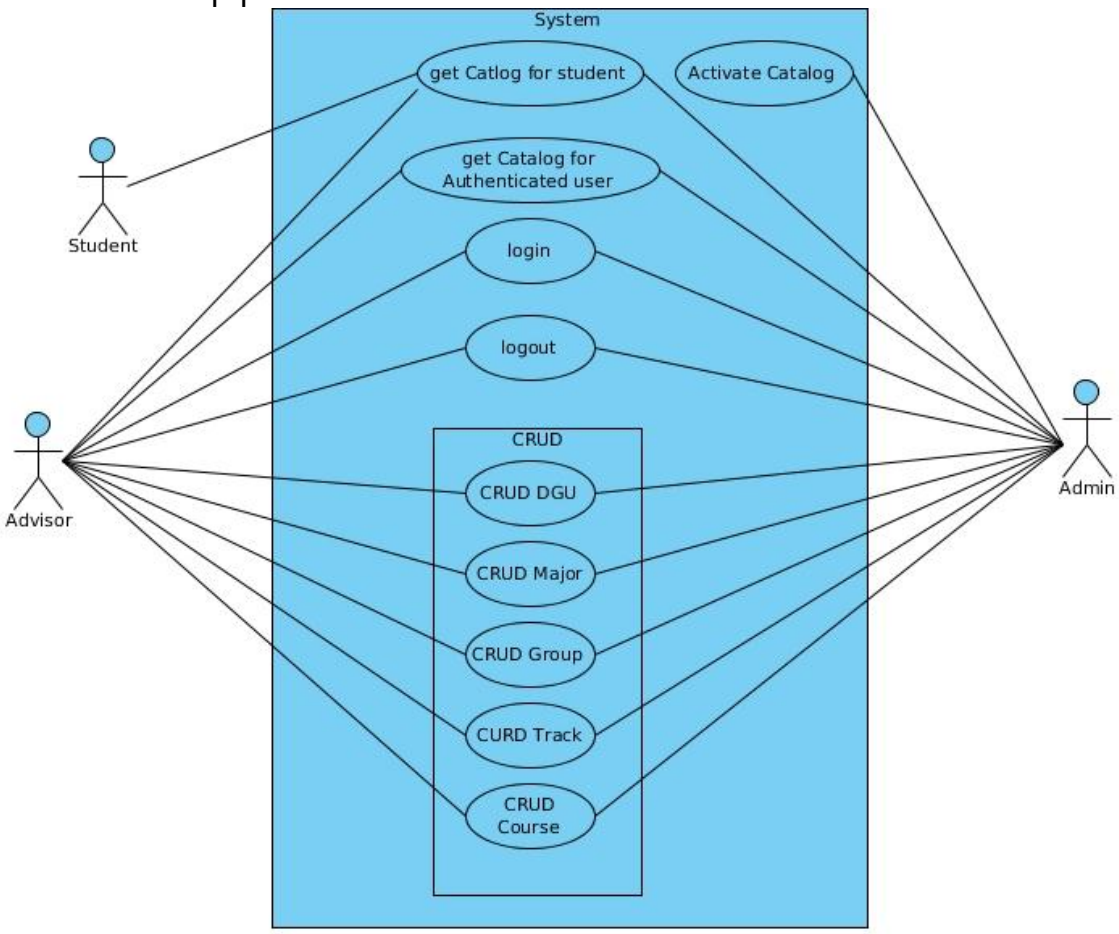


Figure 9-1 Use Case diagram

9.3 Appendix C – User Interface Design

The following view will be the interfaces depicting the process to update a course. In the example below, we are updating information on COP 3402.

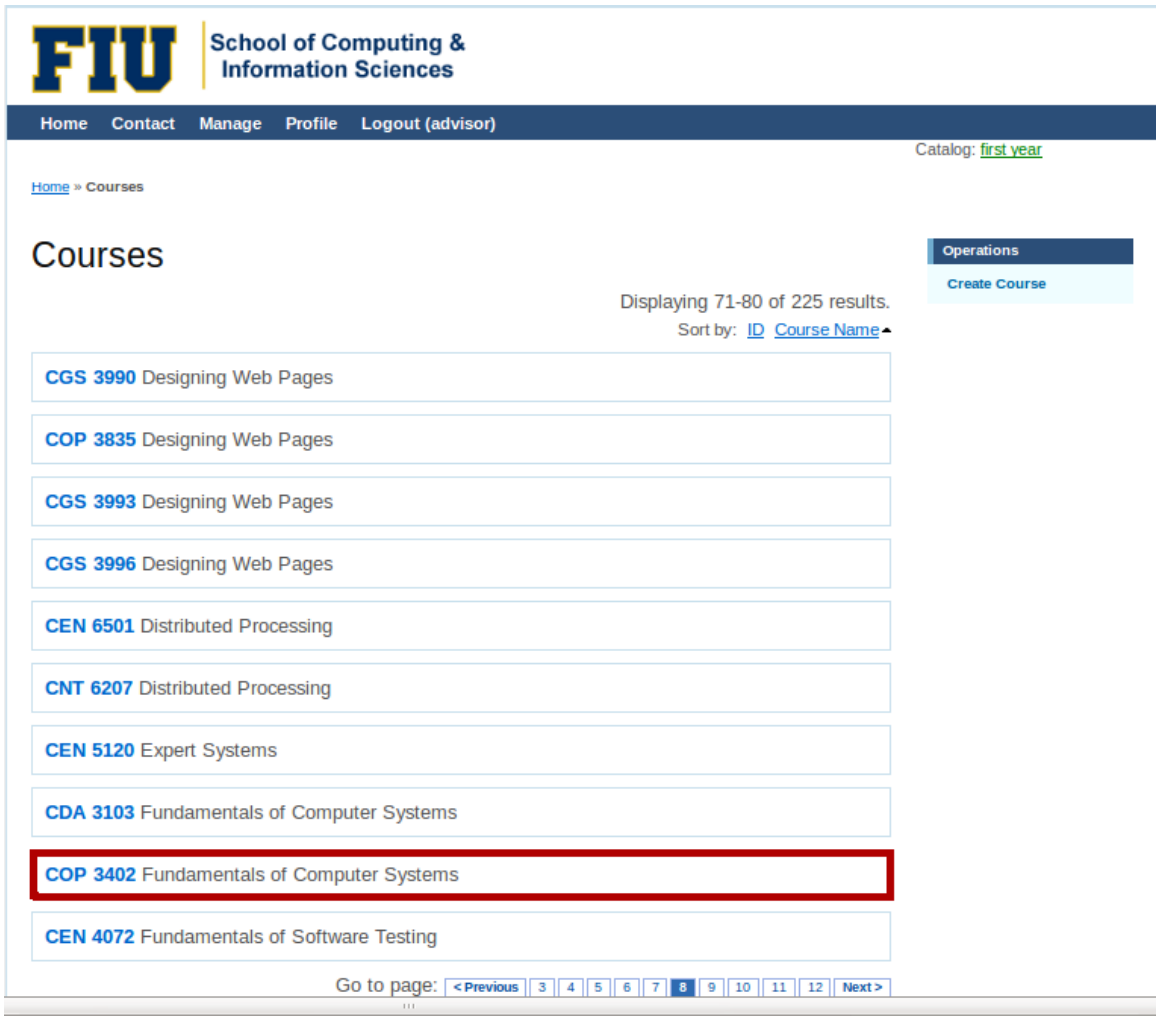


Figure 9-2 Course List

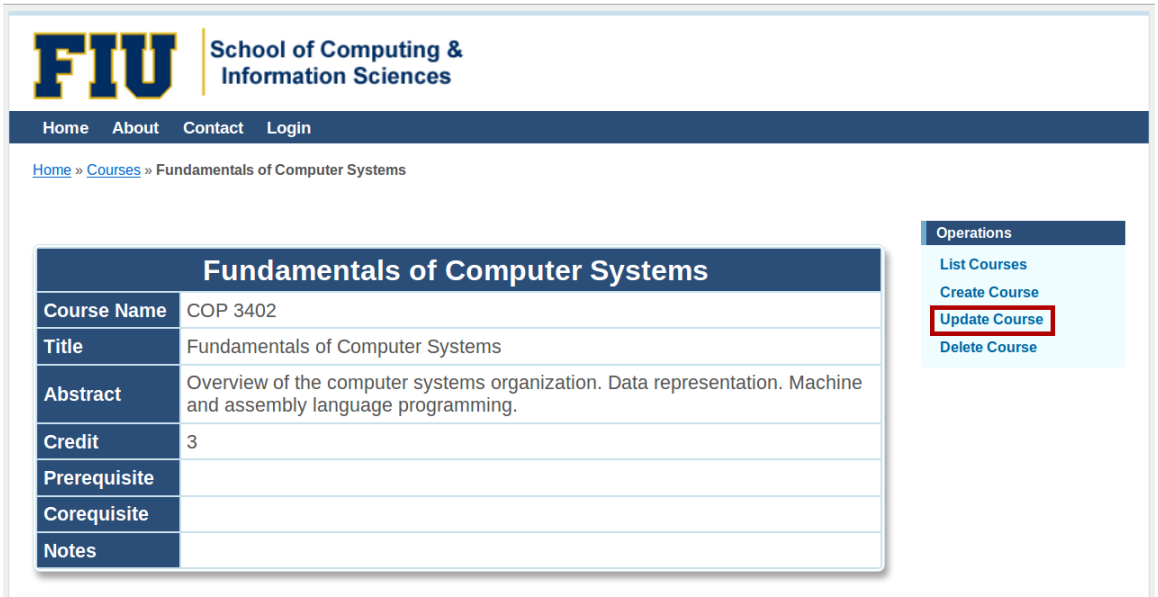


Figure 9-3 Course information

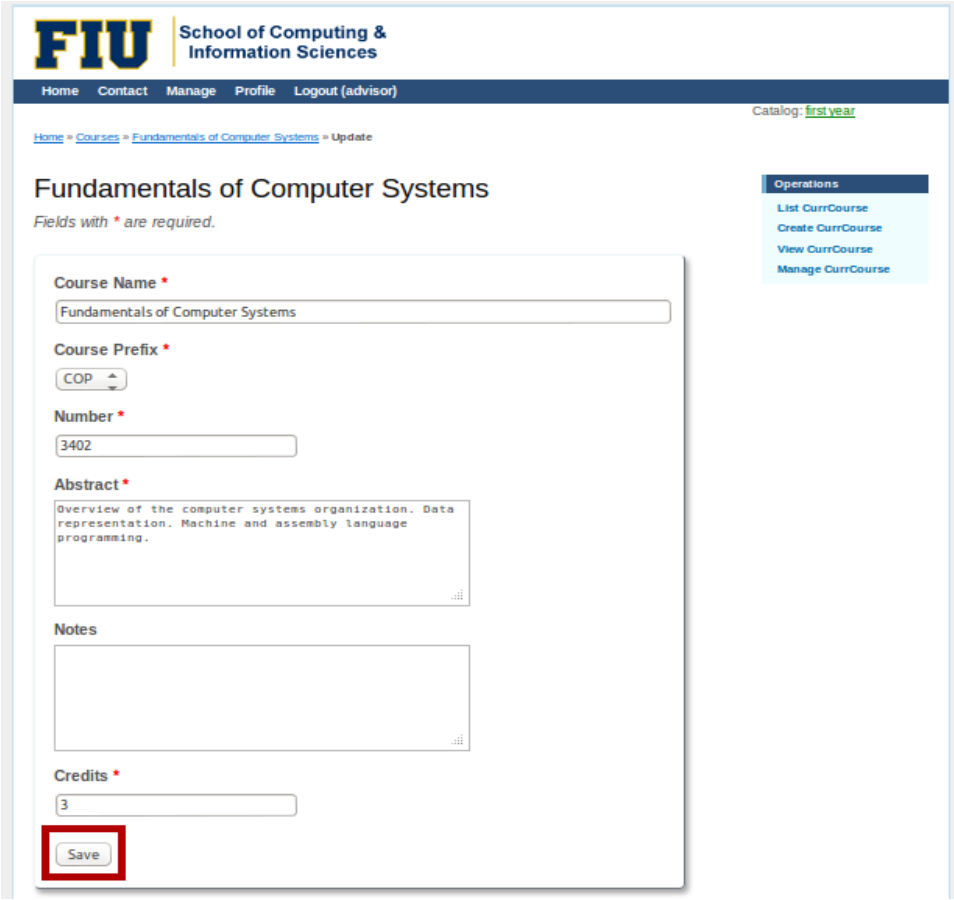


Figure 9-4 Modify information and save

The following view will display the interface of use case ID 3 – Generate catalog for students.

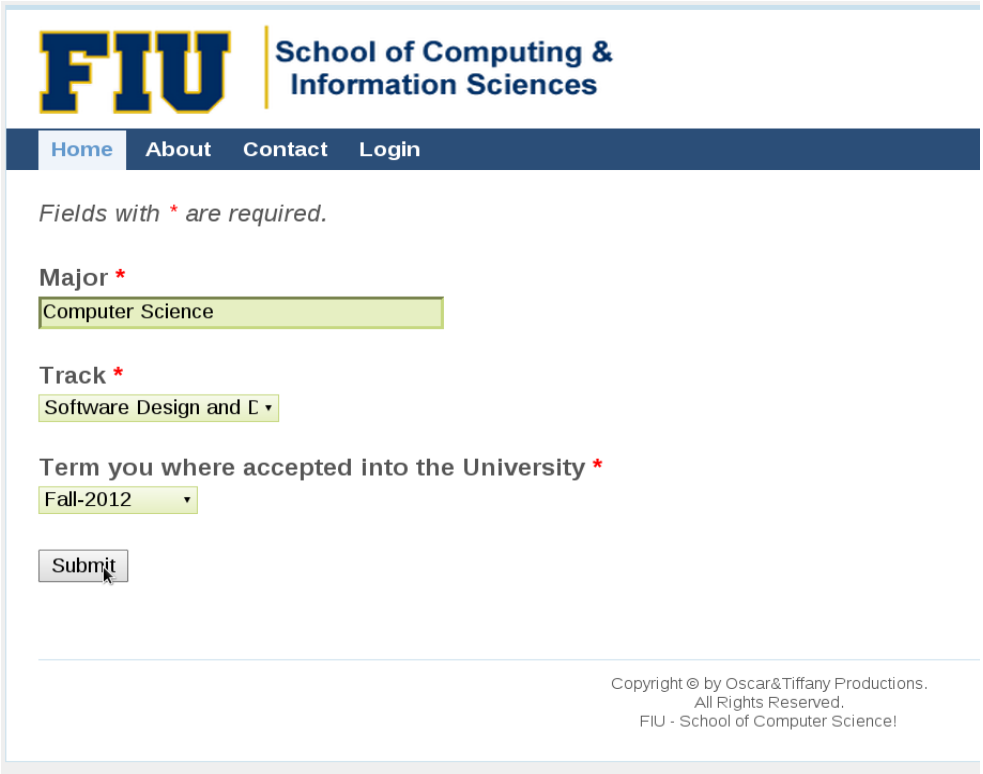


Figure 9-5 Input and Select Major, Track, and Term

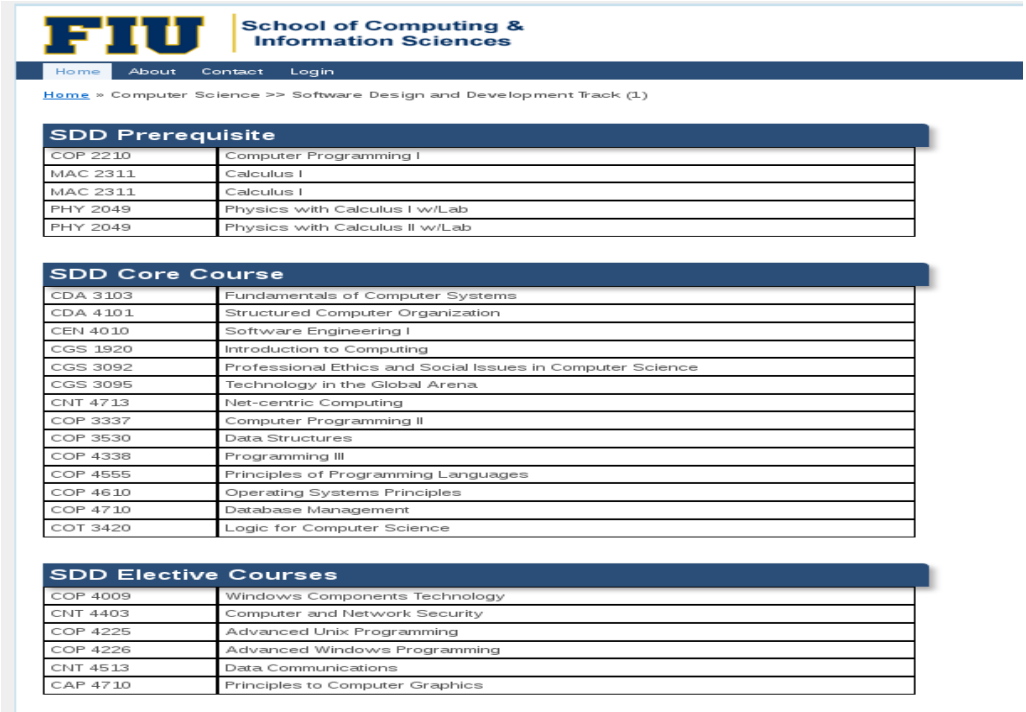
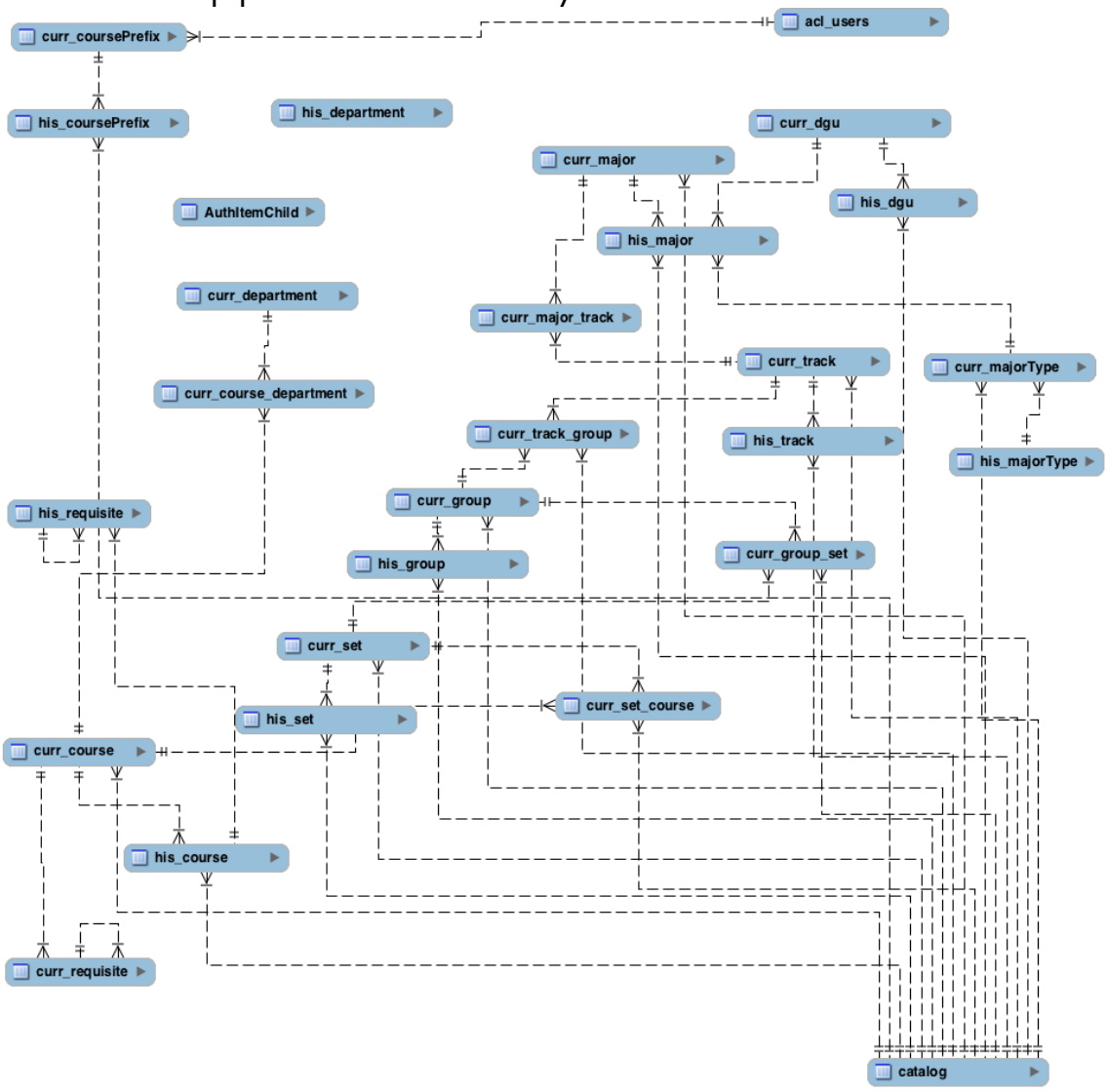


Figure 9-6 Display Curriculum information

9.4 Appendix D – Analysis Models



9.5 Appendix E – Dynamic Model

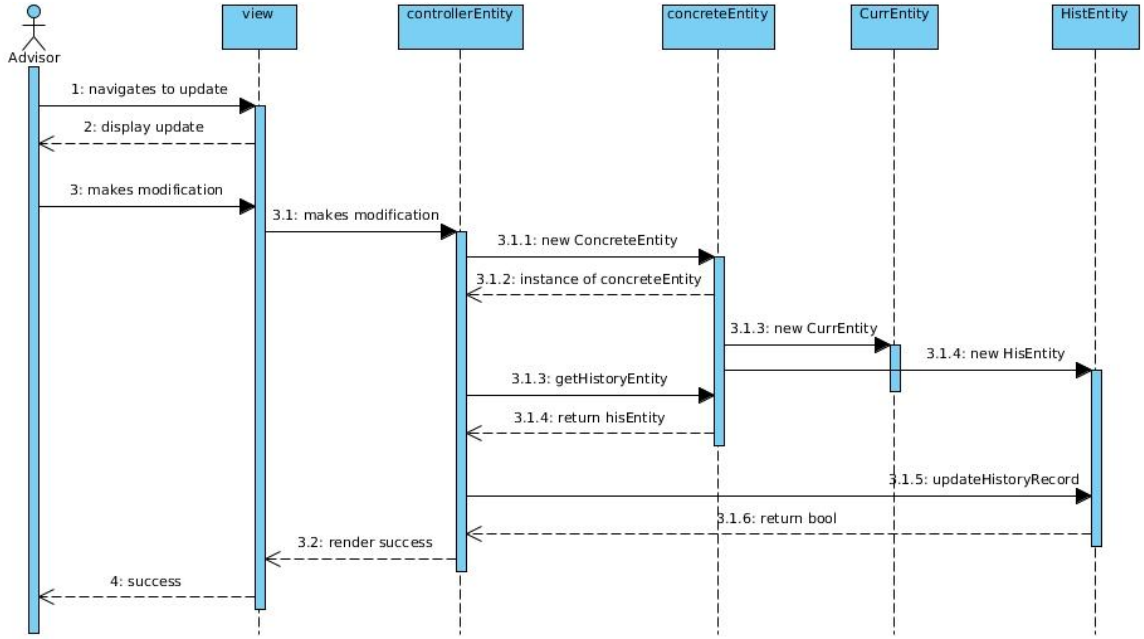


Figure 9-7 General Sequence Diagram for Update of CRUD for an entitie.

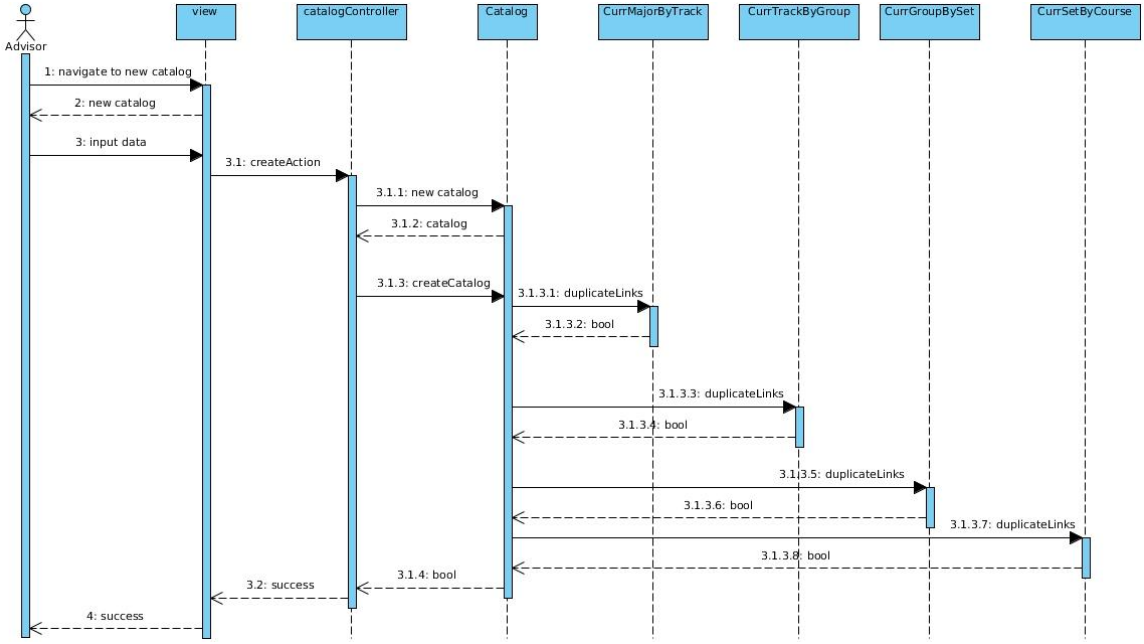


Figure 9-8 Sequence diagram for Use Case ID: 3 -- Generate catalog for advisor

9.6 Appendix F – E-R Diagrams

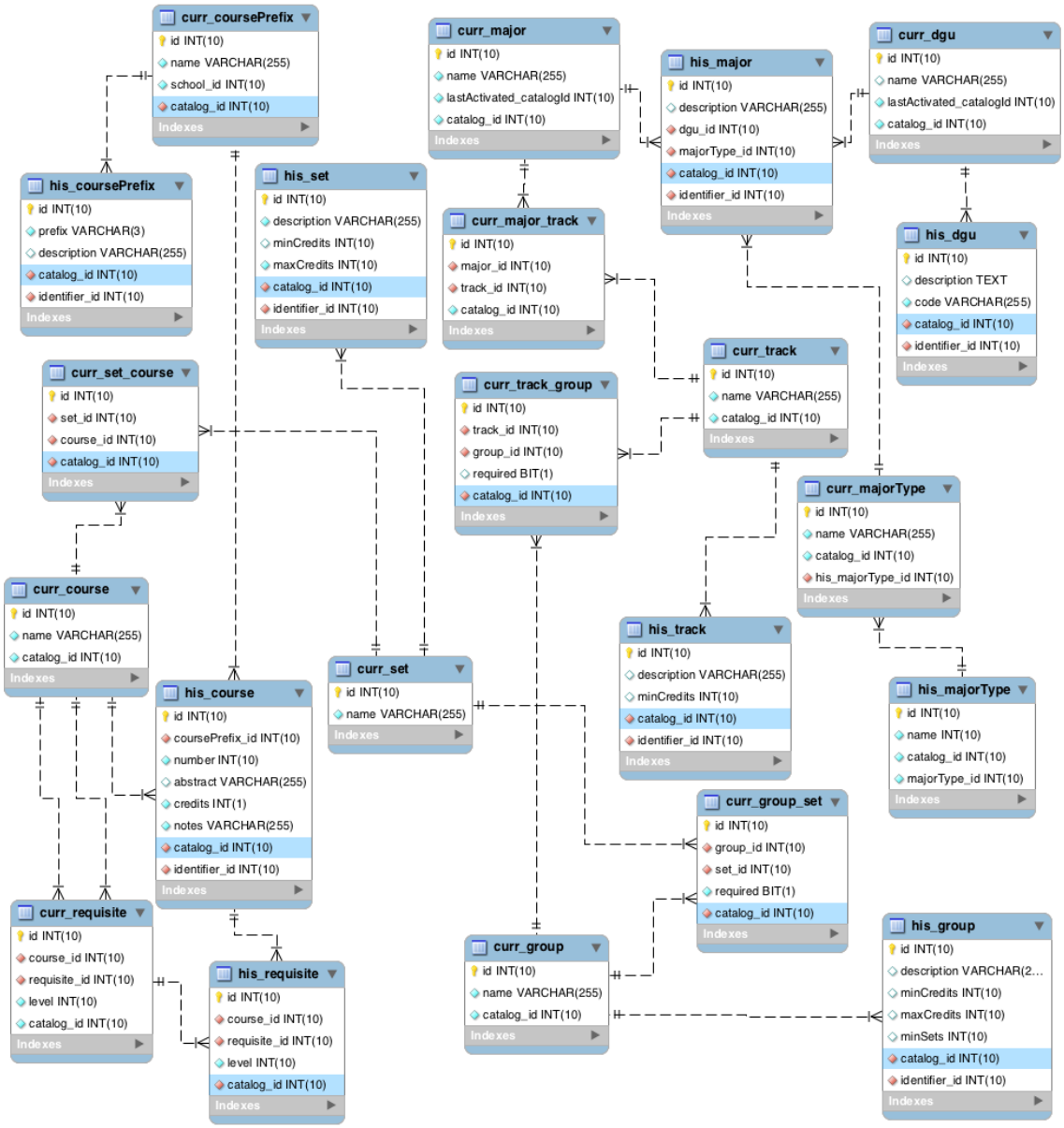


Figure 9-9 Detailed overview of Class Diagram

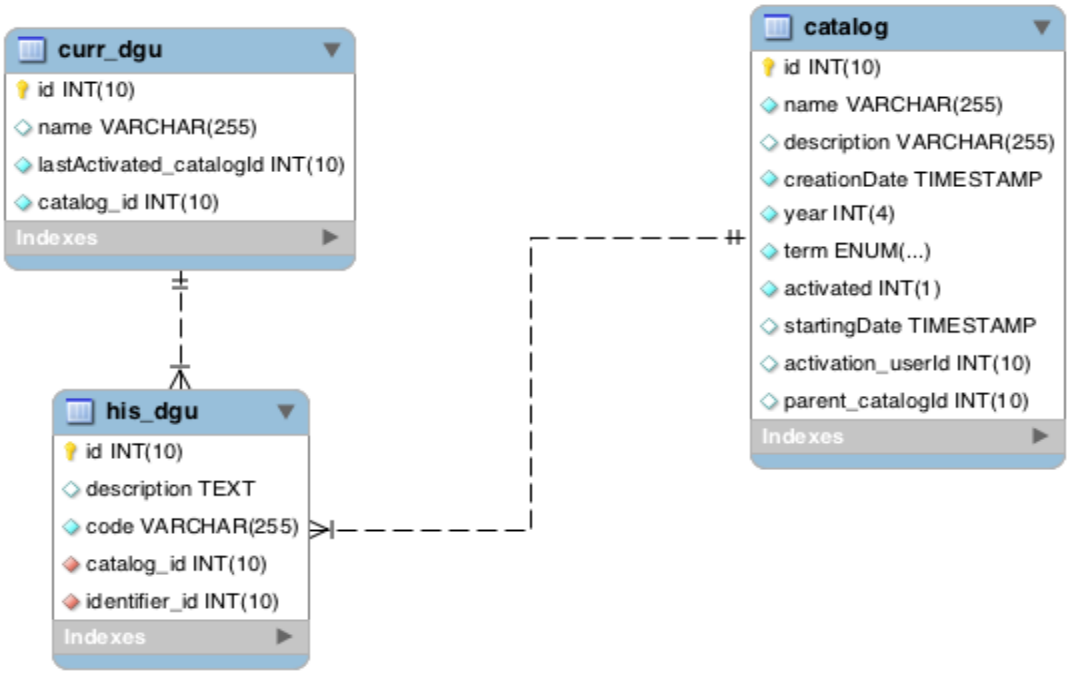


Figure 9-10 Bird's eye view - DGU class diagram

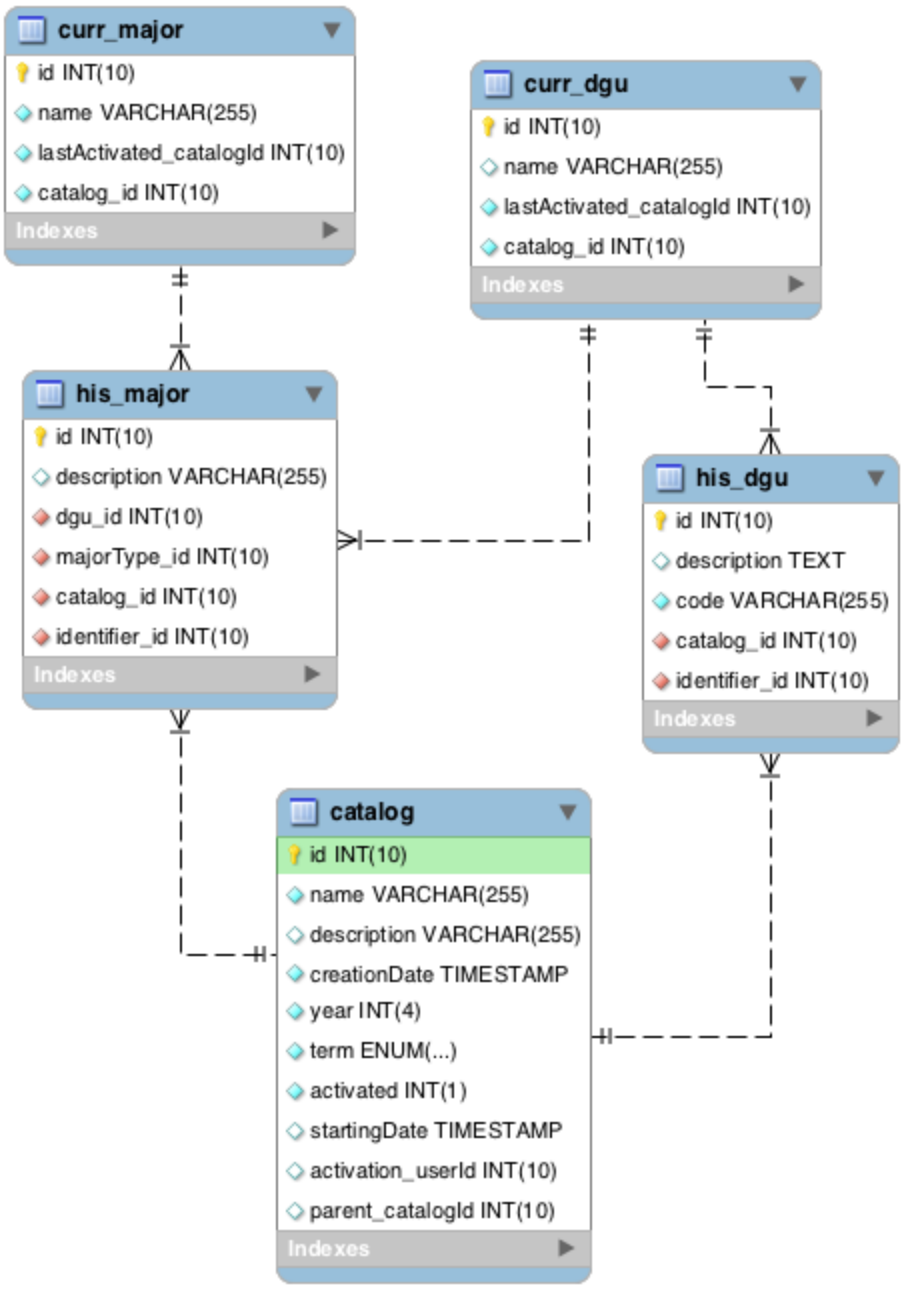


Figure 9-11 Birds-eye view DGU_MAJOR relation

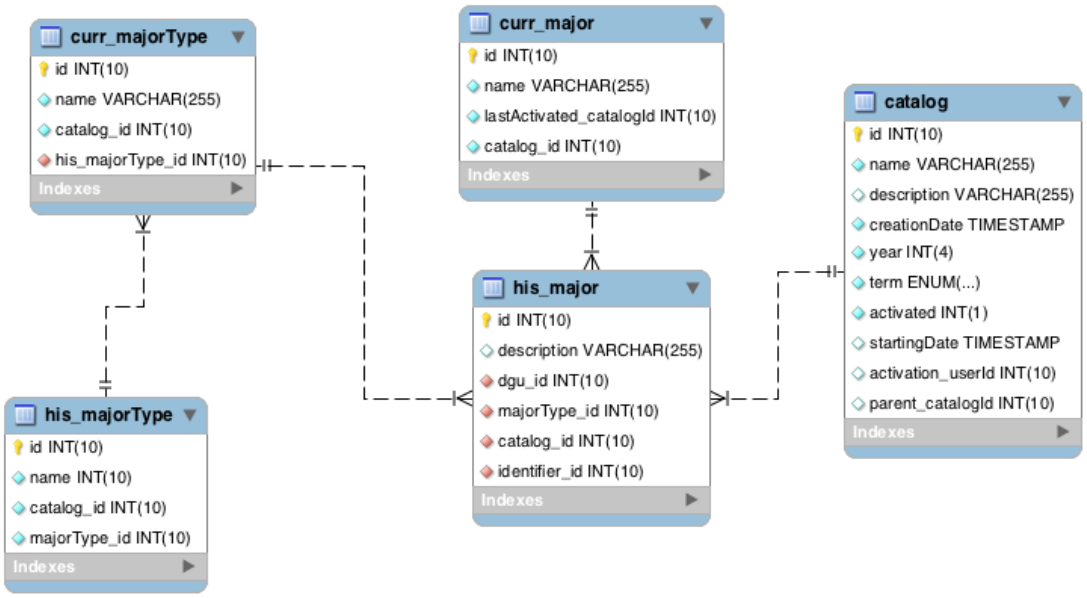


Figure 9-12 Birds eye view major

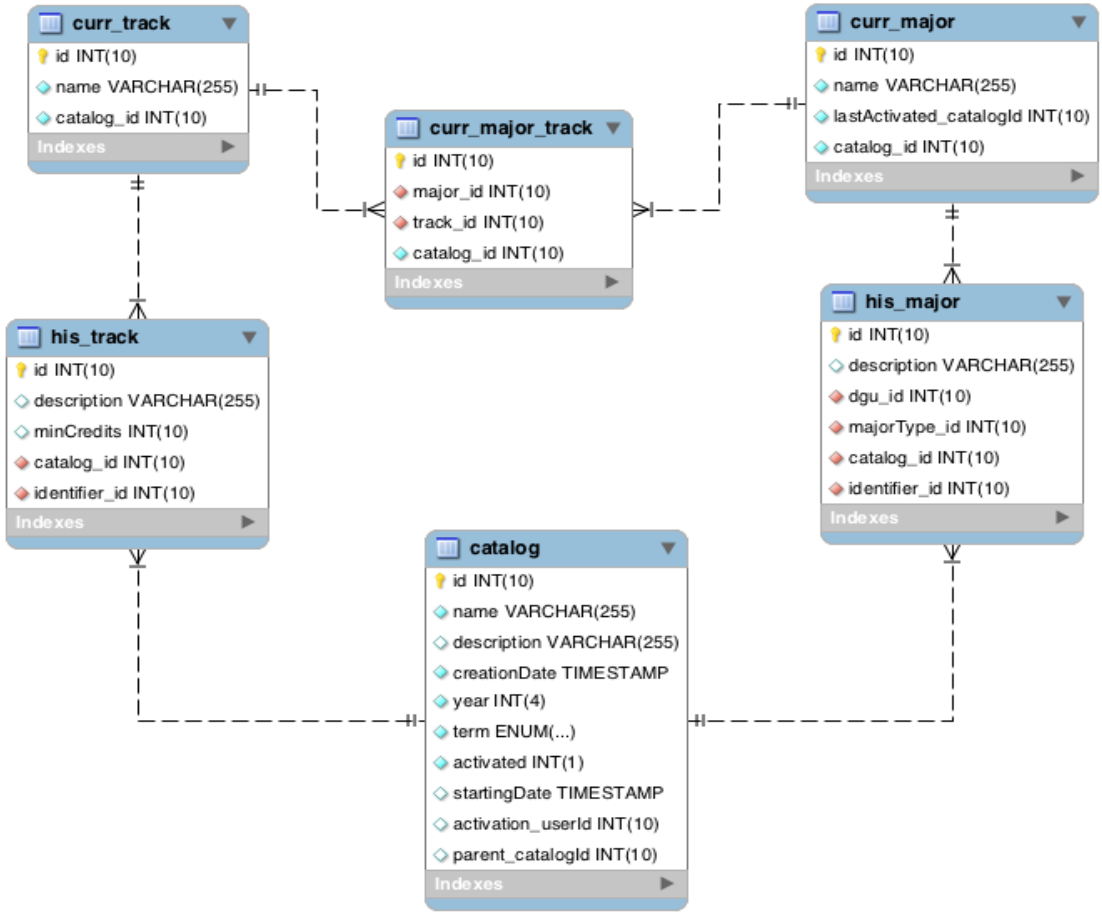


Figure 9-13 Birds eye view major_track relation

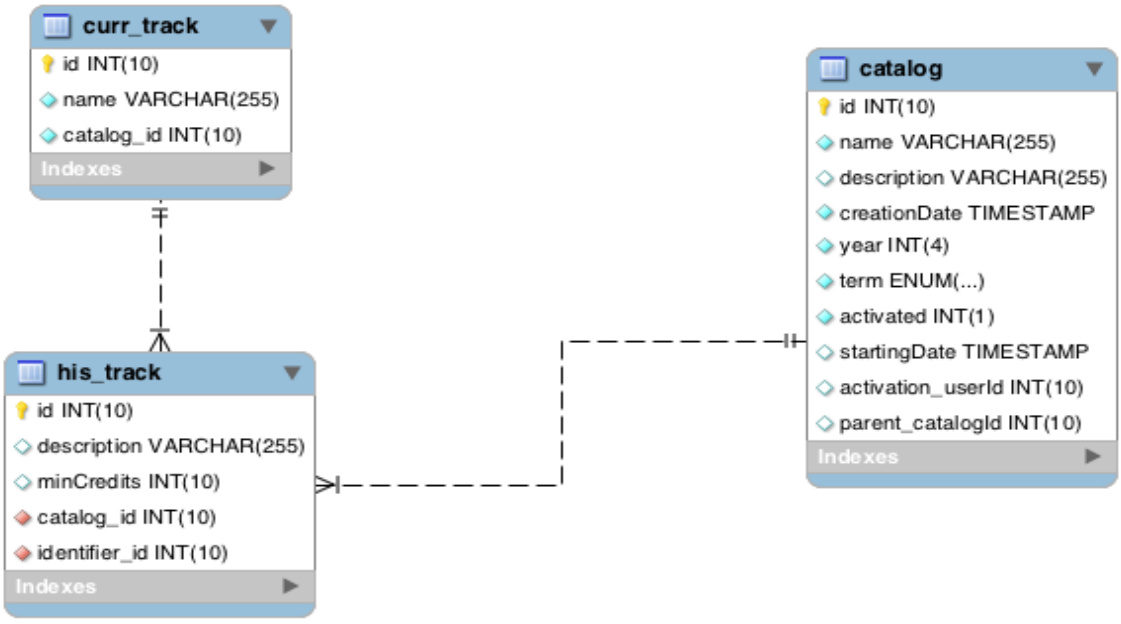


Figure 9-14 Birds eye view track

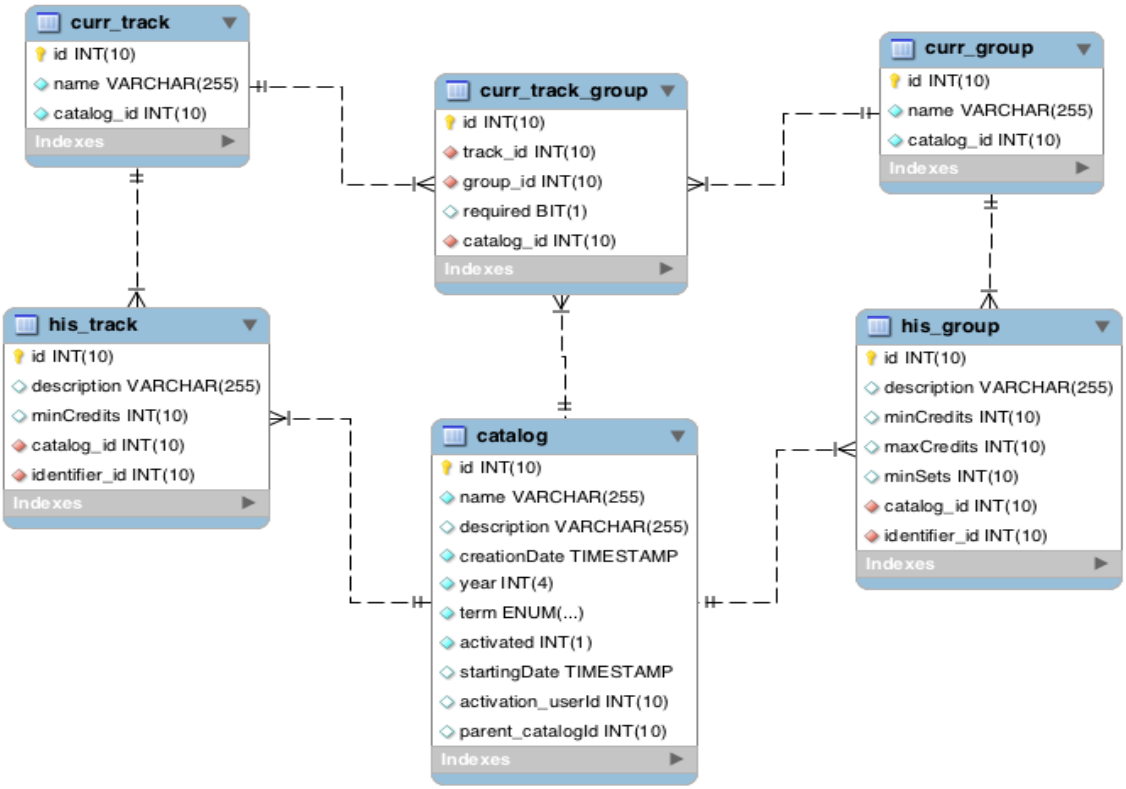


Figure 9-15 Bird's eye view track_group relation

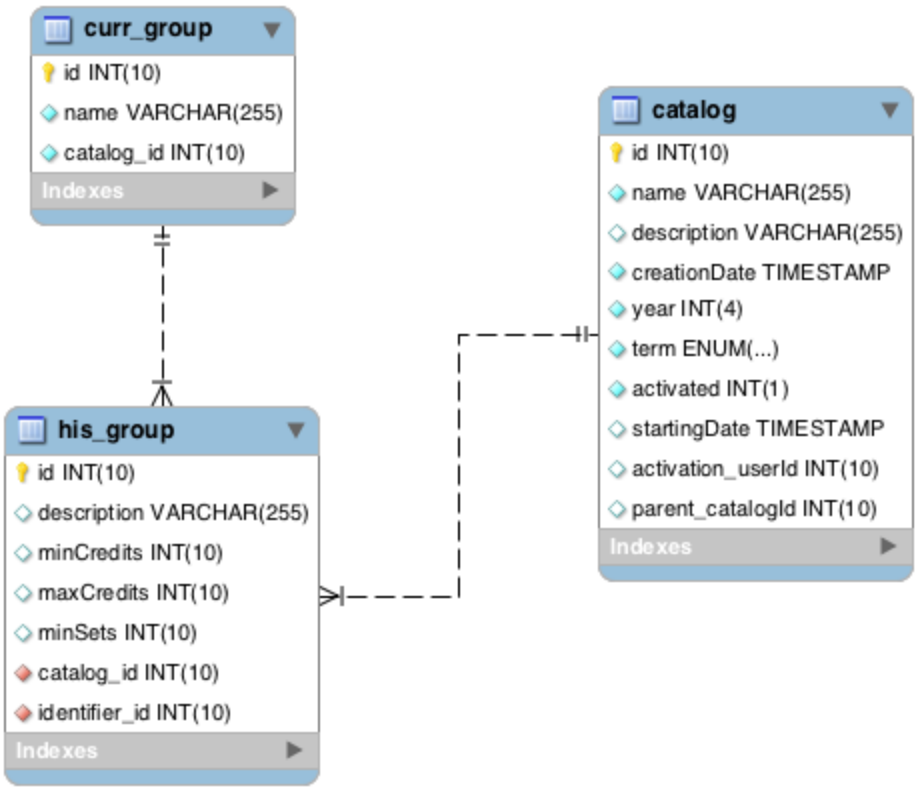


Figure 9-16 Birds eye view group

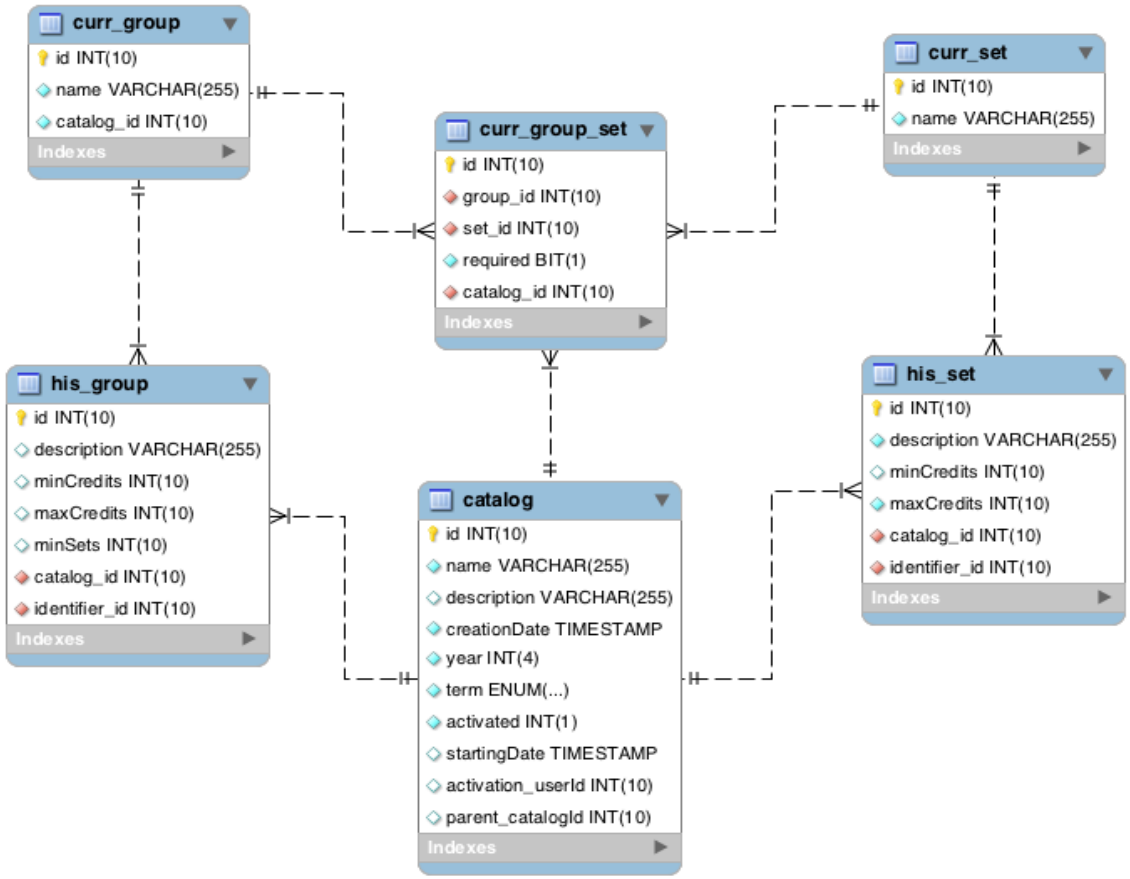


Figure 9-17 Birds eye view group_set relation

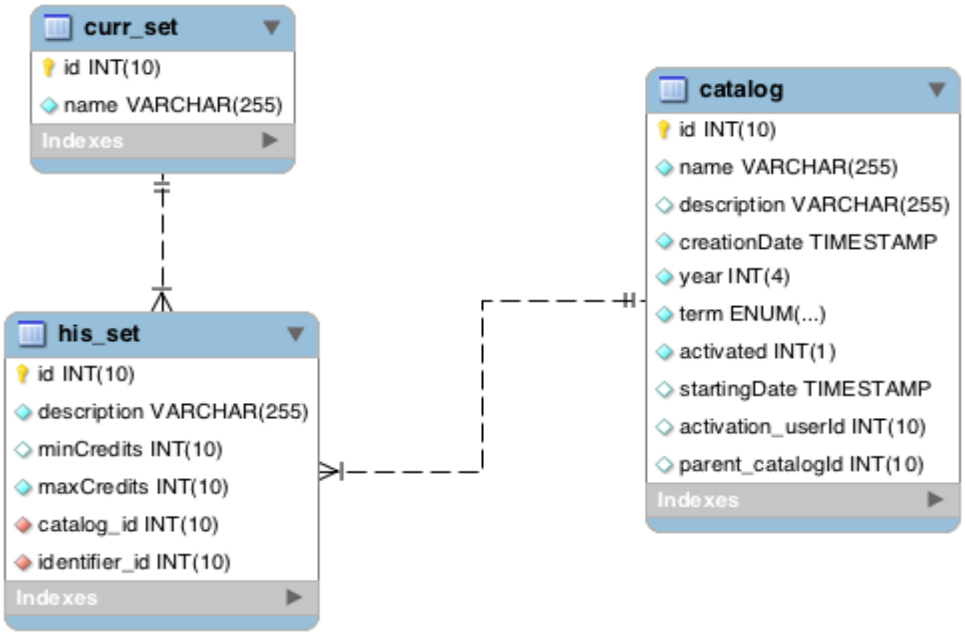


Figure 9-18 Birds eye view set

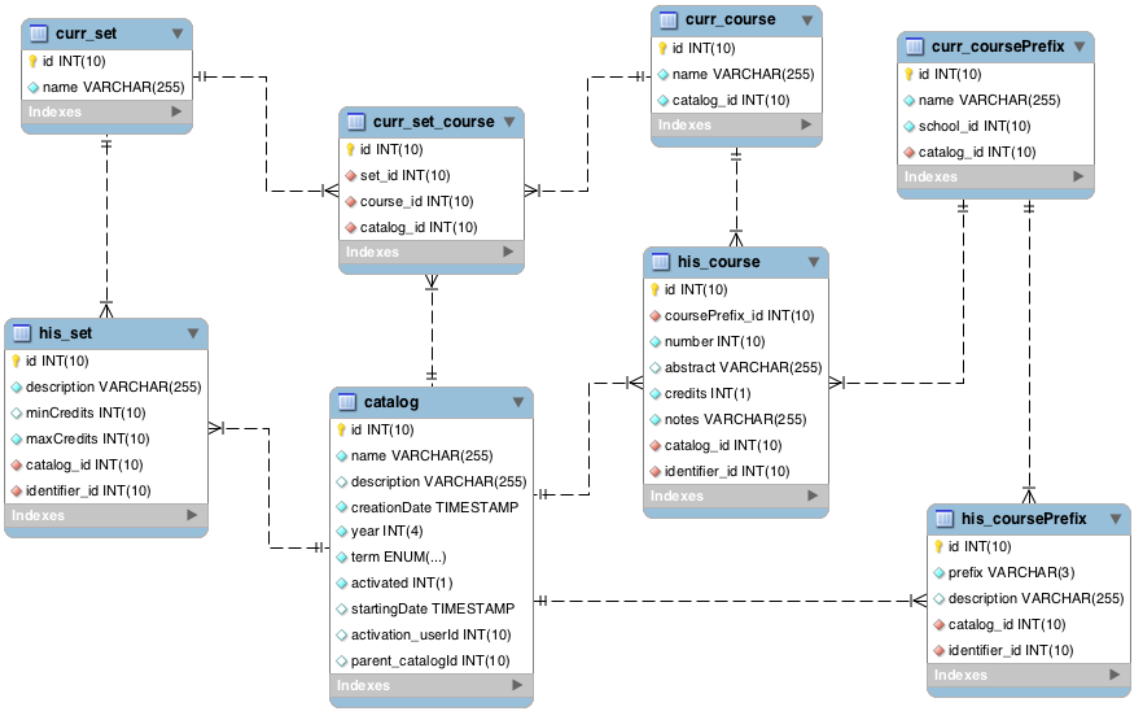


Figure 9-19 Birds eye view set_course relation

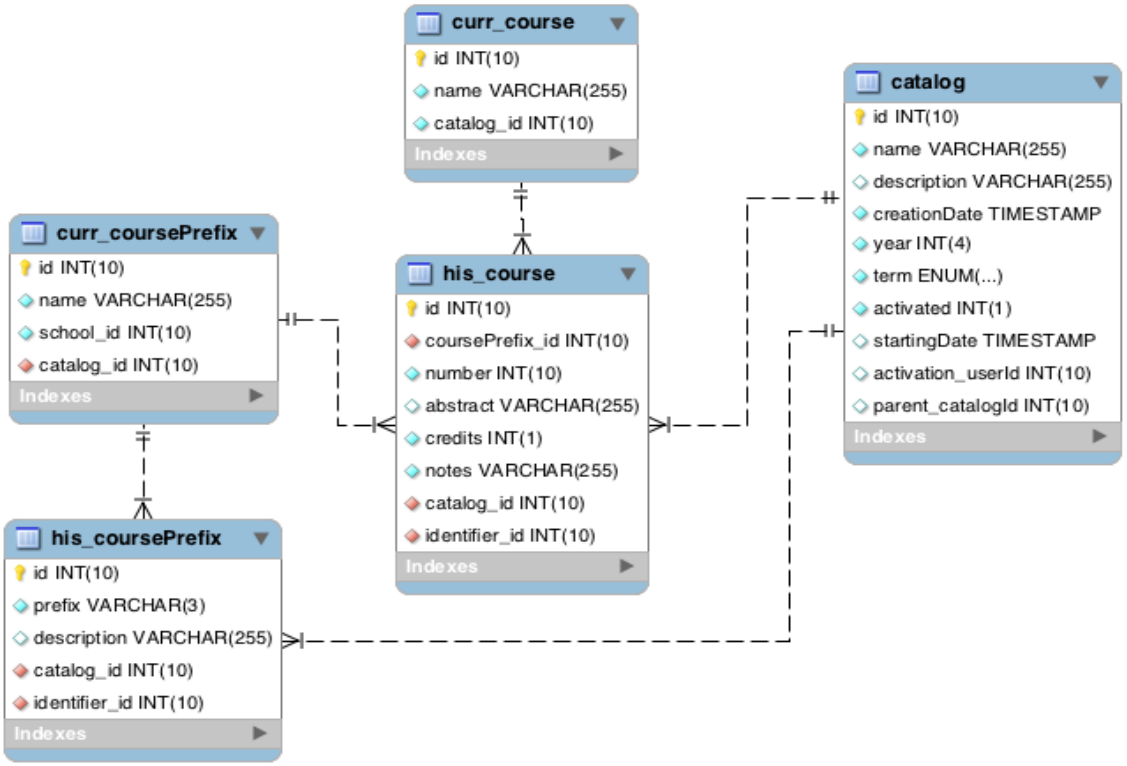


Figure 9-20 Birds eye view course

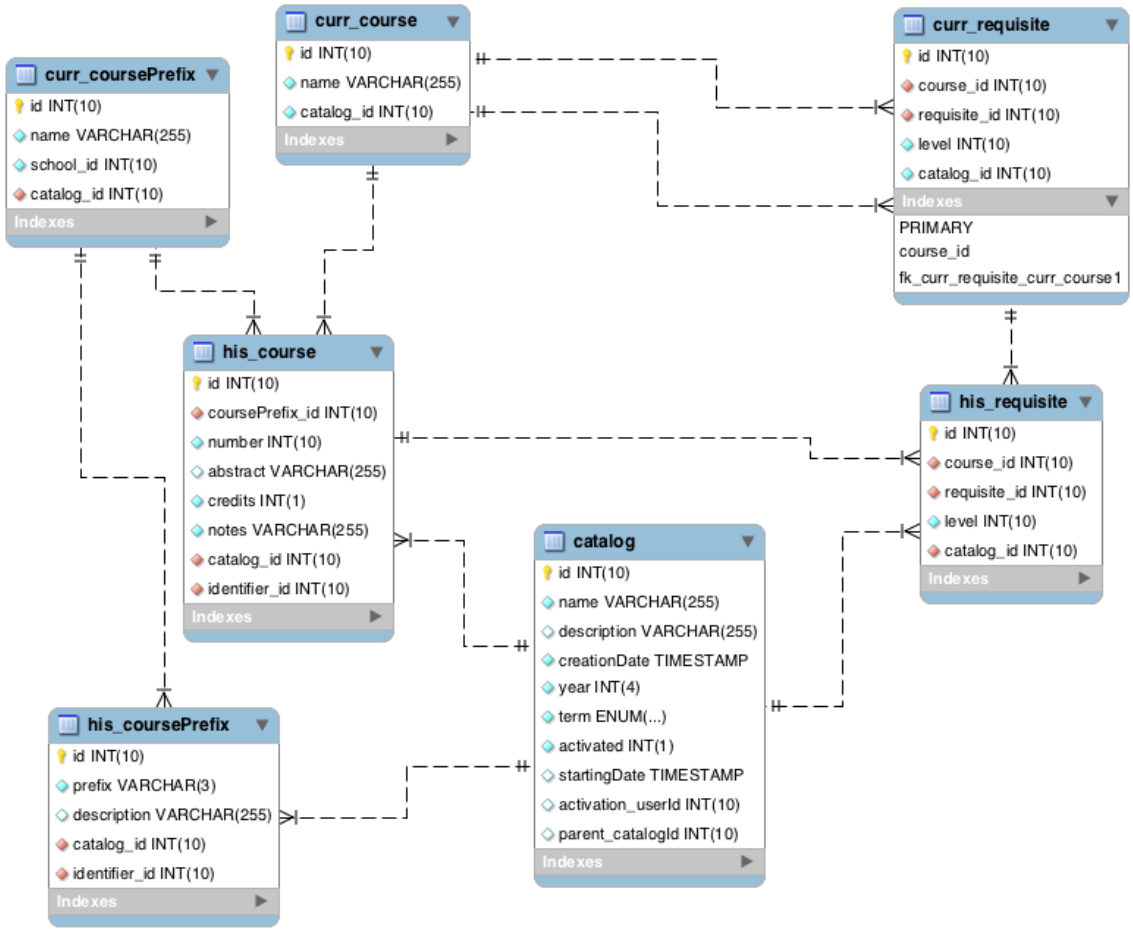


Figure 9-21 birds eye view course_requisite relation

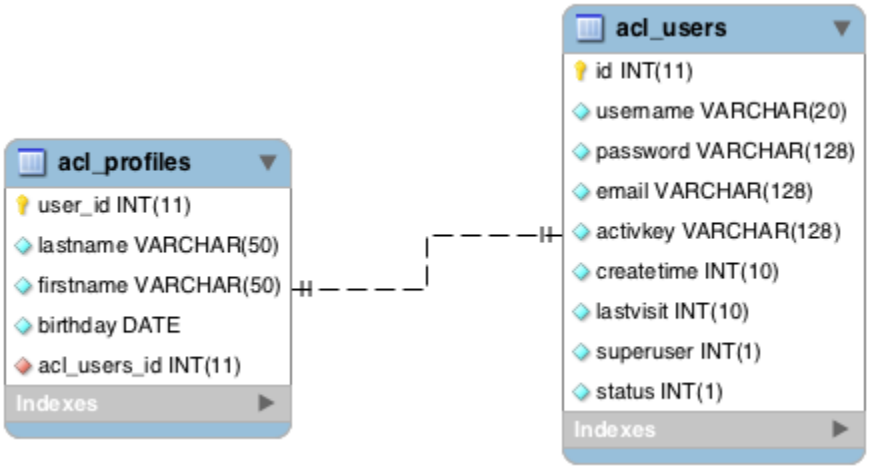


Figure 9-22 ACL-user relation

g

10. References