Concordia University
Department of Computer Science
and Software Engineering

Software Process
SOEN 341/4 --- Winter 2009 --- Section S

SOFTWARE DESIGN

Team members information	
Name	**SID**
Marc-André Moreau	9347100
Mathieu Dumais-Savard	6095275
Julia Lemire	9402969
Phuong-Anh-Vu Lai	5644399
Sébastien Parent-Charette	9178821
Andreas Eminidis	9377603
Corey Clayton	9349200
Eric Chan	9365079

TABLE OF CONTENTS

TABLE OF FIGURES

1. INTRODUCTION

This document examines the architecture of the web based application scheduling software designed and implemented by team7. It includes a description of the different modules of the user interface, a mock-up of the logic to be used and a detailed break-down of the database.

2. Architecture

2.1 – Overview

The team7 scheduling system, hereby referred to as PlanZilla, uses a 3-tier software architecture. This means the presentation, the logic and the data are separated. This has the advantage of making the code more reusable and maintainable since the tiers can be modified without necessarily affecting each other. Figure 2.1.1 shows the different components of a 3-tier architecture:

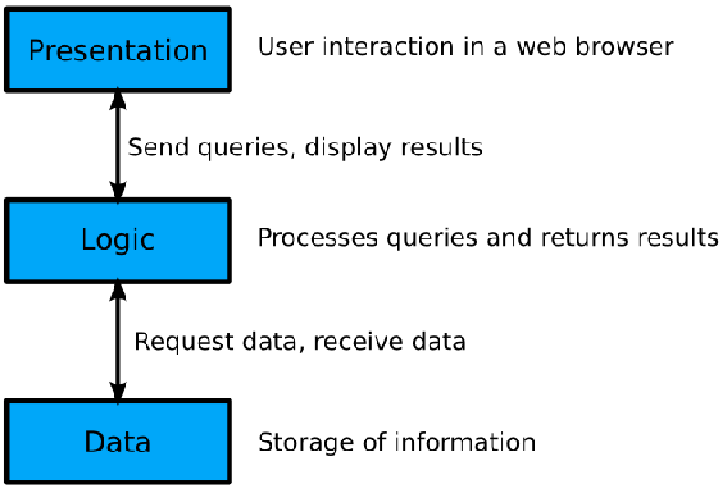

Figure 2.1.1: 3-tier Architecture

In the case of PlanZilla, the presentation is what is presented to the user in the form of a website. This part is done using jQuery, a JavaScript library that makes it easier to support a broad range of browsers while providing a rich set of easy to integrate controls and plug-ins.

The logic tier is composed of multiple PHP modules for the various tasks the application needs to handle. This includes user authentication, scheduling, registration and courses.

The third tier is the database. The database used in this project is MySQL. The multiple PHP modules interact with the database in order to query data requested by a user in the presentation tier, and return the results in a format ready to be used by the presentation tier.

Figure 2.1.2 shows a diagram of the 3-tier architecture used for PlanZilla.

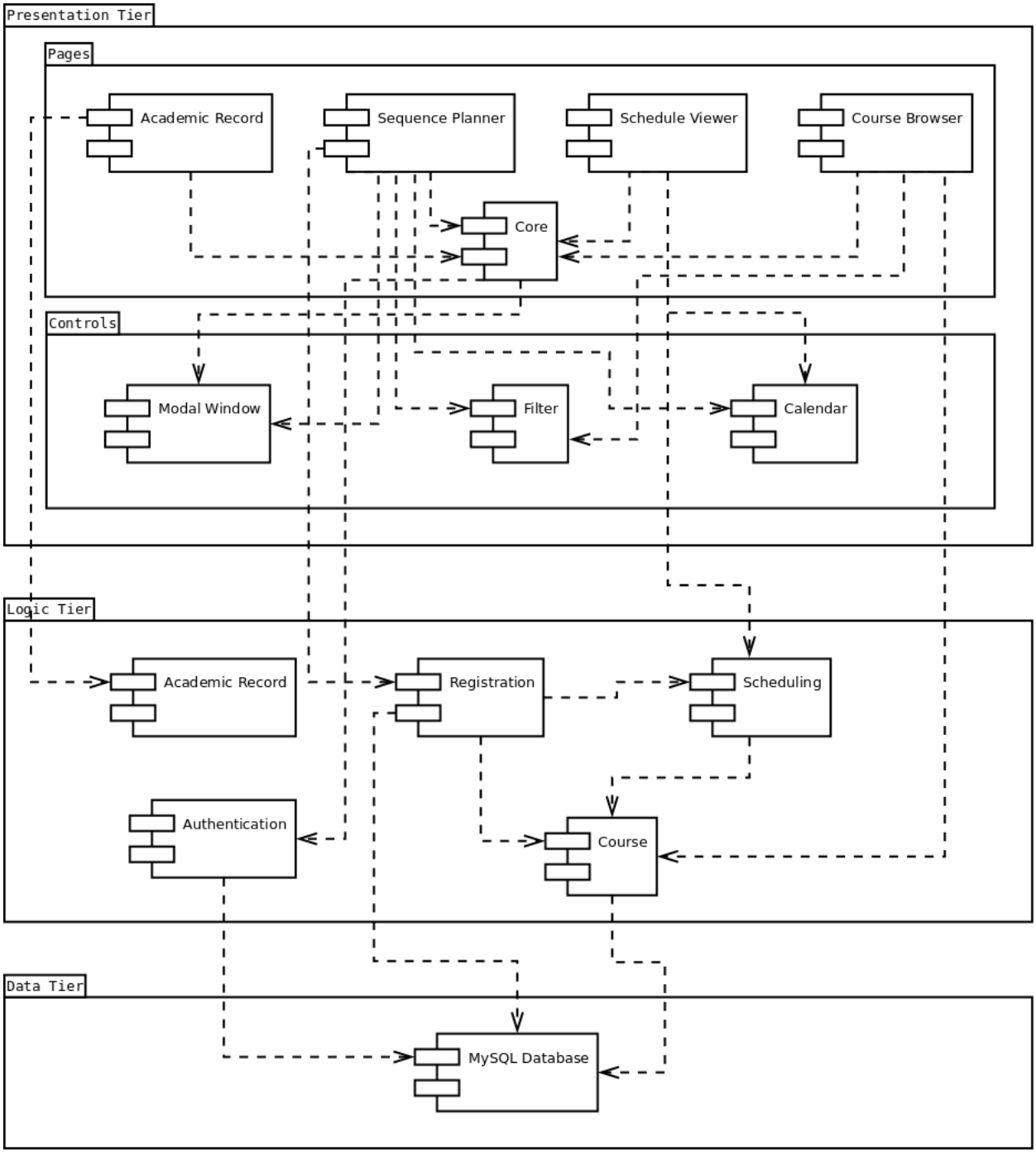

Figure 2.1.2: UML Component Diagram

3. Presentation Tier

3.1 Tabs

3.1.1 Core

The core is the main page of the website. It contains the tab controller. It provides access to the authenticated and the unauthenticated sections with the use of an authentication module. Other formatting features like headers, footers and the main menu are managed in this module.

3.1.2 Academic Record

The application has a tab called Academic Records, in which students can view their transcripts and other useful academia related information. The Academic Record module in the presentation tier only handles the presentation of the data that is provided by the analogous Academic Record from the second tier, while this information stored in the MySQL database.

3.1.3 Sequence Planner

The Sequence Planner is a handy tool that can generate potential schedules from a list of given courses that a student wants to register for. It has the ability to propose courses from the student's course sequence. The user also has the option of setting special constraints, such as not having classes late on Friday night or early Monday morning.

3.1.4 Schedule Viewer

The schedule viewer shows a student's schedule in a calendar view. A schedule is only available if the student has registered for courses.

3.1.5 Course List Browser

The course list browser can be accessed by both an authenticated and an unauthenticated user. It provides a full list of the courses offered with some general information. This can be useful for both registered students, who want to learn more about the courses they have to take, and for people currently not enrolled in any program but would like to know more about a give course.

3.2 Controls

3.2.1 Modal Window

At different places in the application a modal window is needed, such as for the authentication of a user. Instead of rewriting the same code multiple times, a modal window module is used.

3.2.2 Filter

The filter is a control used for the auto completion of a text entry in various modules of the presentation tier. For instance, if a user starts typing “Software” the filter could suggest the auto completion “Software Engineering”.

3.2.3 Calendar

The calendar module is used for the presentation of schedules in a sleek calendar view. Both the schedule viewer and the sequence planner access this module.

4. Logic Tier

4.1 Authentication

The Authentication module is responsible for properly validating a user for the duration of their session on the website. It receives a user's credentials and validates them according to the information stored in the database. Passwords are stored with the following hashing function: sha256(sha256(password) + random_salt).

4.2 Academic Record

The Academic Record module in the logic tier has the task of gathering information, such as the grades of a student, and returning this data in the form of a full student transcript. This is to be readily displayed by the Academic Record module from the presentation tier.

4.3 Registration

The Registration module is used when a student has chosen a schedule and now wants to register the selected courses. Registration first uses the Course module to verify that the schedule is valid and that the student can take all the courses selected. The student is then registered in these courses. In the case of an invalid schedule, such as when a course becomes full right before the student wants to register, then the Registration module will deny the request and a brief reason for the error will be given.

4.4 Scheduling

The Scheduling module takes a list of courses and a list of constraints, and generates all of the possible schedules resulting from them. The list of possible schedules is then sent back to the calling module, which in this case is the Sequence Planner module. Scheduling also depends on the Course module in order to verify the validity of the generated schedules. This is important to ensure that a student satisfies all prerequisites of a given course.

4.5 Course

The Course module is used to access course information. It also to handle the logic of validating a course schedule with regards to course dependencies and the courses a student has already taken.

5. Data Tier

5.1 Overview

Figure 5.1.1 shows an overview of the database architecture for PlanZilla:

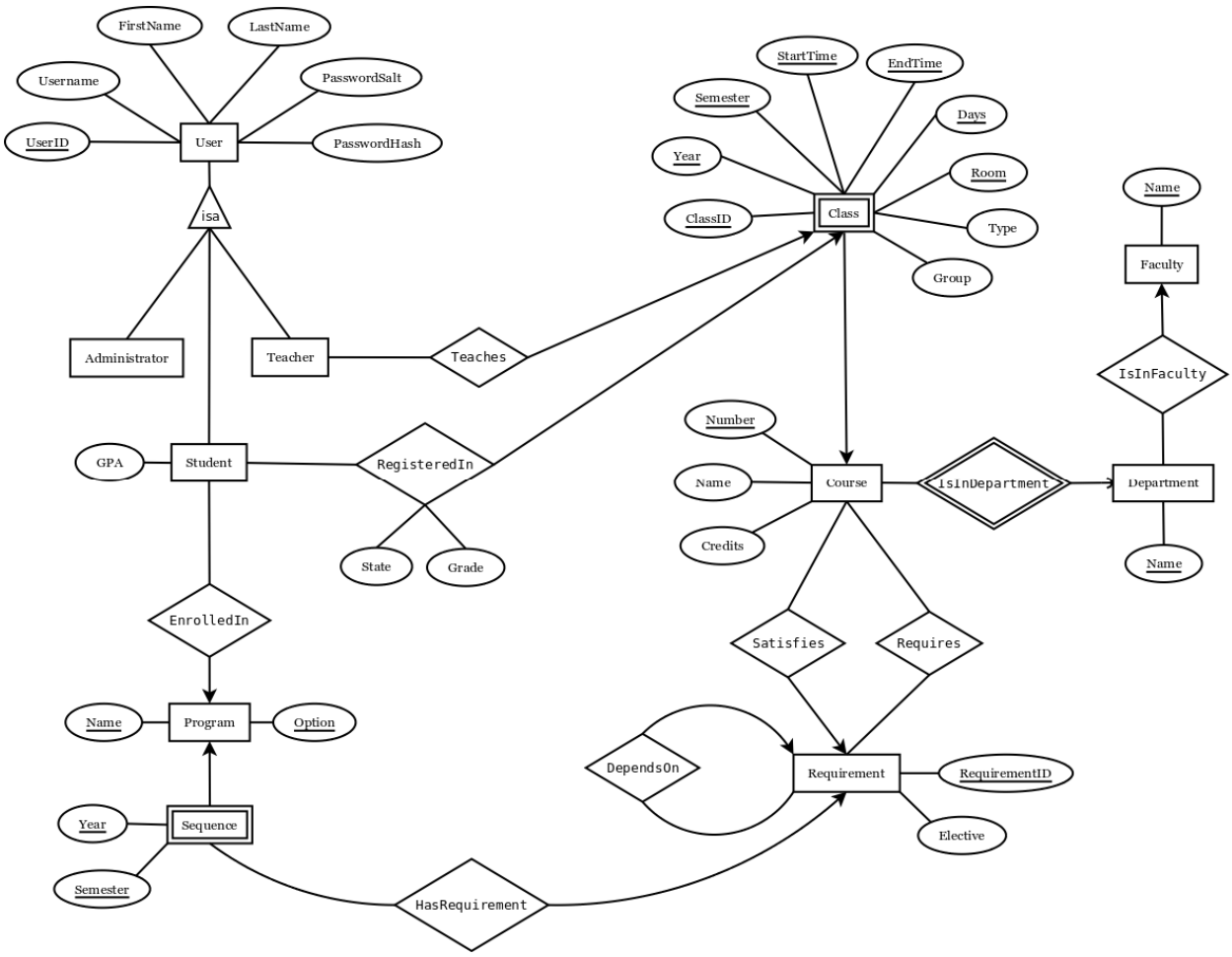

Figure 5.1.1: Entity-Relationship Diagram

Each part of the above entity-relationship diagram is explained more thoroughly in the following sections.

5.2 Users

Figure 5.2.1 shows the different types of users that can access the system.

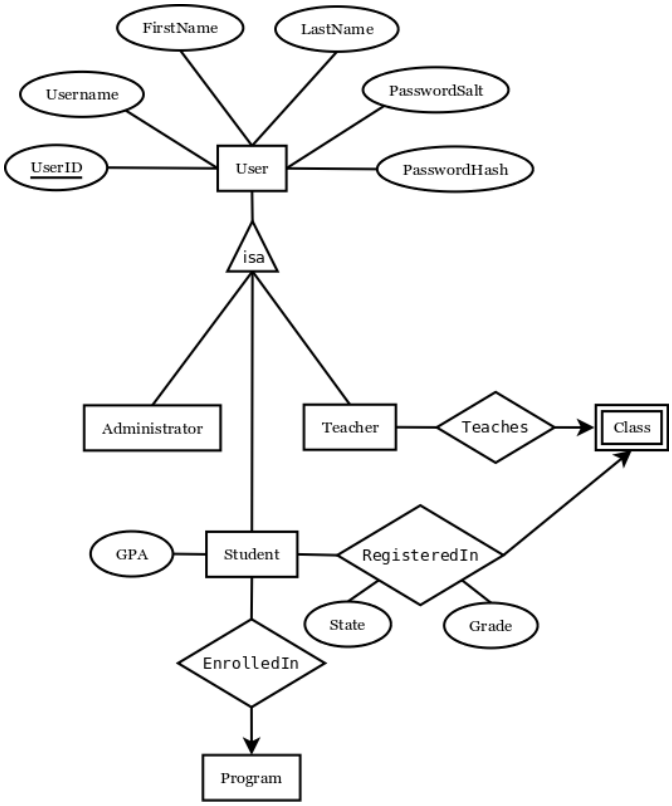

Figure 1.2.1: The different types of users that can access PlanZilla.

5.2.1 Administrator

Users in the administrators table have administrative powers in the database. This would usually be reserved for the staff of the school that have access to operations that other types of users do not. This may include adding new courses and manually registering students for courses that are already full.

5.2.2 Student

Students are a special type of users that can be enrolled in a program and registered in courses. It inherits all of the attributes from User, and has an extra attribute for the student's GPA. The RegisteredIn relationship is used for both present and past courses: if a student has successfully passed a course, then the State attribute should say "Pass" and the Grade attribute should contain the associated grade. If the course is currently being taken, then State should say "Current" and no grade should be available yet.

5.2.3 Teacher

Teachers, like students, are a special type of users. They can teach different classes. The Teaches relationship is made with class and not course. This is because it is possible to have many teachers for one course. For instance, there may be one teacher giving a lecture and another teacher in charge of tutorials and labs.

5.3 Courses

Figure 5.3.1 focuses on how a course's information is listed in the database.

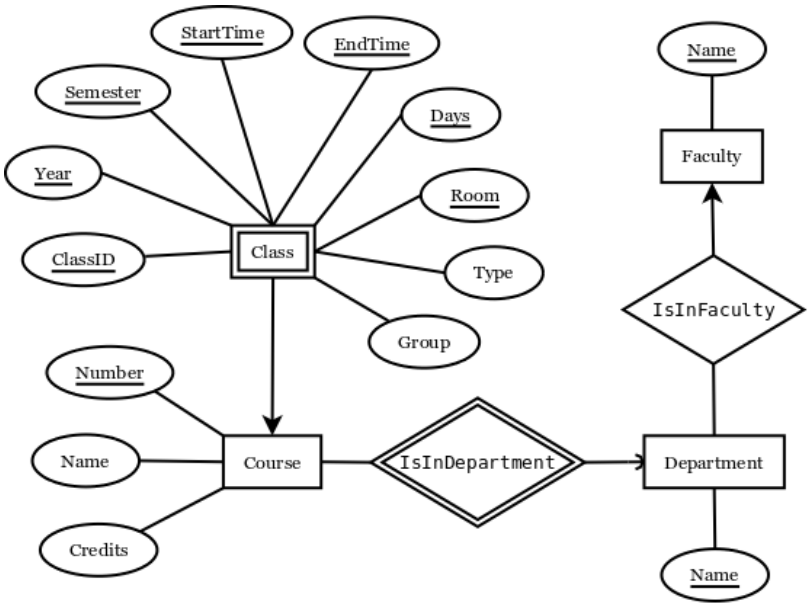

Figure 2.3.1: Courses in the database

5.3.1 Course

A course is uniquely identified by its department and course number. For this reason, each course must be related to Department with the IsInDepartment relationship. The IsInDepartment relationship uses short four letter department names such as COMP or SOEN. The Name attribute of Course is simply the full name describing the course. The number of credits is also stored within the course.

5.3.2 Class

For each course there may be a different number of classes. A class is a weak entity, as it depends on a course to be uniquely distinguished from other classes. This is where you find important information such as semester, year, time, day(s), group and room. The type attribute is used to identify the class as being a lecture, a tutorial or a lab. Because of the large number of attributes required to uniquely identify a class, an extra attribute has been added: ClassID. The ClassID is simply a convenience in order to relate teachers to classes.

5.3.3 Department

A department is one of the departments by which a course is given. Examples of this are COMP (Computer Science) and SOEN (Software Engineering). The four letter abbreviation is used to relate courses to departments, but the Department entity also stores the full name of the department. As departments are part of a faculty, each department is related to the faculty it belongs to.

5.3.4 Faculty

Each faculty has an abbreviation of a few letters that is used to relate faculties to departments, such as ENCS with COMP. Like departments, the Faculty entity stores the full name of the faculty. For instance, ENCS stands for Engineering and Computer Science.

5.4 Requirements

Figure 5.4.1 depicts how the prerequisites for a course are stored in the database.

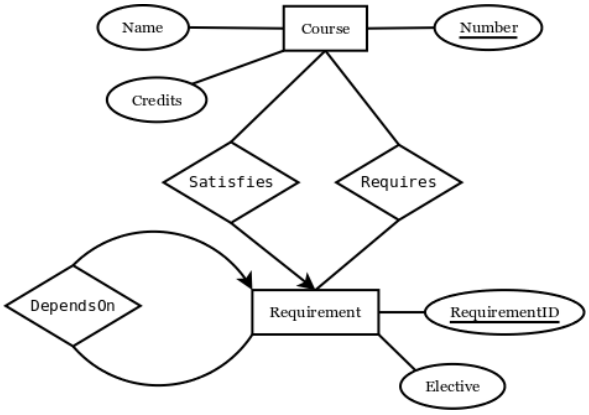

Figure 5.4.1: Representation of requirements.

5.4.1 Simple Requirements

The simplest case of a requirement is a course which is dependent on one or more other courses. For example, let course A be dependent on courses B and C. In the database, course A would be related to a requirement with the Satisfies relationship. That requirement would in turn be related to courses B and C using the Requires relationship.

5.4.2 Elective Requirements

In many programs students have to take at least one elective course. The elective course is usually chosen from a predefined set of possible courses. For example, let course A be dependent on a technical elective. Course A would be related to a Requirement, that would in turn be related to all the possible courses through the Requires relationship. The difference between a requirement where you have to take all of the courses required and one where you only have to pick one of them is made with the Elective attribute. Whenever Elective is not null, it is set as the name of the elective, such as “Technical Elective”.

5.4.3 Multiple Requirements

There are even more complex cases of requirements that require an extra relationship DependsOn that relates a requirement to another requirement. For instance, let course A be dependent on B+C or D+E. This can be represented in the database by an elective requirement depending on two other requirements (B+C and D+E).

5.5 Programs

Figure 5.5.1 shows how a program is stored in the database.

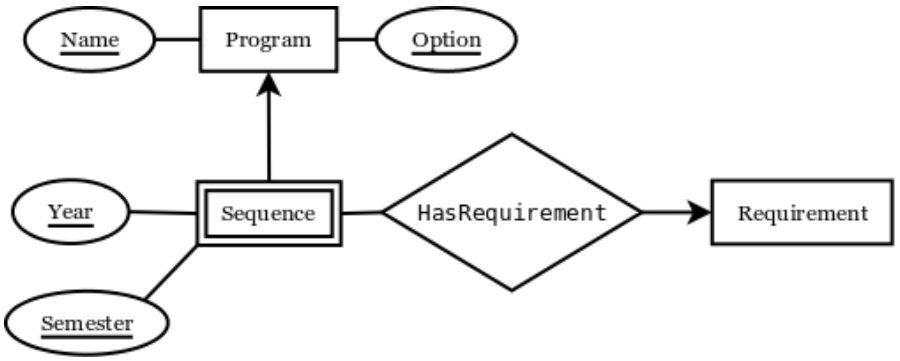

Figure 5.5.1: Representation of a program.

5.5.1 Program Options

A program can have many possible options. However, each option of a program corresponds to a different course sequence. For this reason, a Program is uniquely defined by the program name and its option. An example of a program would be engineering, with an option being software engineering.

5.5.2 Course Sequence

A course sequence is usually presented to the student as a full list of courses to be taken according to each semester of the program. This is why a semester is identified by the Program it is associated with, a year and a semester. The courses that have to be taken to meet the requirements of the sequence are represented using Requirement entities, just like with courses.