

**SFL ONLINE**

*Bachelor’s Thesis*

**Ekin Naz EREN – 213CS2301**

**Tolga ŞIVGIN – 214CS2013**

Supervised by

Assoc. Prof. M. Taner Eskil

January 2019

# **Abstract**

**SFL Online**

While the faculties of our university has a course information system, namely course online, the Prep school of foreign language in our university lacks one. Due to prep program having a different working mechanism then the faculty, it needs to have its own system. SFL Online is an online course management and student information system designed for the use of students and instructors of the SFL program. Its main purpose is to provide a grade and attendance information system between students and instructors of SFL program. We have tried to cover as many functionalities as possible to fit the needs of the SFL students and instructors.

# DEFINITIONS, ACRONYMS AND ABBREVIATIONS

SFL: School of Foreign Languages

MVC: Model View Controller (A software architecture pattern, which we use in this project.)

CRUD: Create Read Update Delete

UI: User Interface

CHAPTER 1

# INTRODUCTION

This chapter will inform you about the general problem

## Purpose of the System

SFL Online is a web-based system, whose purpose is to provide an efficient and online way for instructors of the foreign languages school to enter attendance and grade information of students, and share them with other instructors and students easily. Also, purpose of the system includes providing functionalities to students, so that they can view their grades and attendances entered, calculate their passing grade conditions and view their weekly course schedule.

## Scope of the System

The system provides functionalities to students and instructors. Also, the system includes a superuser (Admin) whom has extended functionalities and permissions, such as having direct access to managing records hold in the database.

As stated above, the system includes of three actors, which consists of students, instructors and an admin. Since each of these system users have different functionalities, each one of them has different interfaces according to their roles in the system.

The interfaces specialized for these users will provide useful and required functionalities, according to the actor’s role in the School of Foreign Languages. Actor’s role will also determine the authorizations they have in the system, having only access to functions they are authorized according to their roles. For example, an instructor can edit and enter a student’s grades, while a student can only view them.

Since all our actors share certain attributes in the database regarding their personal information (E-mail, Name & Surname, Password etc.) they are all inherited from an abstract table named “Person”. All actors are an instance of “Person”, and they share these attributes regarding personal information. They are specialized as “Student”, “Instructor” or “Admin” with a discriminator role column in the database. Login authentication is a common functionality for all actors. While logging in to the system, “Person” table is searched if the login information is correct, and the system redirects the user to the corresponding interface matching the user’s role.

There is no such function as registration to the system, like in the faculty. Admin adds students as they’re enrolled in the prep program, and instructors if they are newly hired, sending them an e-mail through ışık mail with the information of their initial login information.

Every user logged in to the system can view their personal information. They can’t change their personal information. They are allowed to change their password at any time. If a user forgets their password and cannot authenticate, the login screen will have an option to send them an e-mail.

Instructors can enter the attendances of the students, whom belong in the classrooms and the courses they are teaching. Similarly, they can enter the grade information of the students in a class, of the course they are lecturing in that class. Instructors take the attendances weekly, but can change the attendance of any week at any time for flexibility. They can enter grades of any assignment, homework and midterm given. They can view the class lists of the classes they are teaching, and view a student’s academic status, grade status and personal contact information. They can write announcements for the classes and courses they are teaching, for all the students in that class to see. Instructors can also see their weekly class schedule, and see which day and time they are teaching a course at a class.

Students in a class can view their attendance and grade information entered, for the courses they are taking. They can see their academic status, such as which class and track they belong to. If they finished a module (semester), they can view the past grade averages of the modules they have finished previously. They can view their weekly course schedule, and see at which day and time they have lectures. They can view announcements by their instructors, for the courses they take. They can calculate the average grade they need and what grade they have to take from the exit exam in order to pass and become a freshman at the University. We call this calculator “Prep Passing Grade Calculator”, a module we have designed by request, for Işık University SFL program.

Admin of the system is in charge of the database operations. The admin can see all the users in the system, and add, edit or delete them. When new students register to the prep school, the admin will add all these students to the system. Similarly, when students finish prep school, the admin will delete them from the system. The admin can also add, edit or delete instructors. Also, the admin can add new classrooms, tracks, and grade types. Admin can change the passing conditions, such as editing the average percent effectiveness of a grade, or the percentage taken of the exit exam. Admin can set a module as a currently active module, which is the same thing as setting the currently active semester in Campus Online. Finally, it is admin’s job to enroll students in classrooms, and also assign instructors to classrooms & courses they will teach.

## Design Goals

SFL Online is an online course management and student information system designed for the use of students and instructors of the SFL program. Thus, the main target is to implement all offered functionalities, the functional requirements, mentioned in Requirement Analysis Document, in shortly RAD.

SFL Online system suggests a usable, reliable, performance, supportable, secure platform. It is targeted to have user-friendly interfaces and fast response times which will allow the users to make their operations quick and easy.

## Objectives and Success Criteria of the Project

There is no such web-based existing system dedicated to Foreign Languages School in our University, such as Campus or Course Online used in the institute. The instructors of SFL share the attendance and grade information of the classes with each other through Google Drive. Students don’t always have the opportunity to view their attendance information or grades unless they ask their instructor, or the instructor shares the attendance & grade sheet with them via E-mail.

Analyzing the issues listed above, the instructors have to share multiple excel files throughout the semester in Google Drive, and it can be confusing and time-taking to update, reach and browse throughout all the files. Students have limited access to these sheets, and may not view them at all times unless shared by instructors.

Another issue the SFL instructors told us is that the students of the SFL are unsure of the passing conditions of the program. Instructors are asked by the students throughout the year about the passing conditions of the exit exam, and their academic average. To clarify these questions for the students, we will add a calculator for students, which they can use to calculate their passing conditions, given their grade averages and exit exam results.

From demand and to make these processes summed up in one common place, and to provide an online, faster, reachable, usable and easier way, we have decided to make our thesis project about providing an online system for SFL.

The main success criteria of the project is to provide and develop the all functions mentioned in this section and scope of the system section, specifically the functions regarding attendance and grade information. Assuming that all the functions are working properly in the system and the main criteria of success is fulfilled, the second success criteria is to make sure these functions are useful and respond fast to the user actions.

Therefore, objectives of the SFL Online system are to have high performance by providing fast responses and services so that the user of the system can complete his/her task faster. Another objective of the project would be usability, by providing usability means the users can complete their tasks with less moves, or mouse clicks in other words. Finally, since the system will hold the information about users and the SFL, the system and the database should be reliable, secure and loss-prone.

CHAPTER 2

# LITERATURE REVIEW

## Current Software Architecture

While there is no existing system dedicated to SFL program, we took references from similar systems used in our school. Two systems are used in our school’s bachelor program, Course Online [2] and Campus Online [3]. Course Online is taken as the main reference point as functionality for our system. Course Online provides course management functions for students and instructors of faculty, which are the functions that we’re striving to develop for the SFL program. However, these systems have been implemented a long time ago, in early 2000’s. Architecture used to build the current systems are unknown, but they were written in Visual Basic and developed in Asp.Net. Therefore, in terms of user-interfaces, security, response-time, speed and other functionalities these systems are outdated. Course Online is not used by most of the instructors to enter attendance and grade information even if it supports these functionalities, due to its outdated and unfriendly user interface. They load attendance and grade information of courses as pdf, excel or word files instead. We want to make SFL Online more appealing and usable for both instructors and students, so these functionalities are actively used. Newer technologies should be used for a faster, more secure and user-friendly system. In terms of database, especially in Campus Online, the database infrastructure is complicated. More relations among different tables cause slower operations, sometimes it can cause to deadlock. While SFL Online will not have a database as big as Campus Online, database relations, operations and complexity shall be aimed to be straightforward and fast as possible. The increasing population of students is also causing more network traffic, which is no longer sufficient enough to support a large number of users for the current software.

## Proposed Software Architecture

Our proposed design pattern of SFL Online is MVC. Model subsystems maintain domain knowledge, and does not depend on any view or controller subsystem (Entity Objects). View subsystems display it to the user (Boundary Objects), and Controller subsystems manage the sequence of interactions with the user (Control Objects). We used MVC model because, in our system, entity objects and data will be in Model, and Controller can be called bridge. Controller provides communication and interaction between Model and View. View can be called interface between users and the system, thus it has boundary objects. These project outputs are logically separated, which reside under on the same machine. MVC pattern is provided by the ASP.NET MVC Framework.

In our component diagram, we’ve used Layered Architectural Style to represent MVC, for readability and reusability. We support a system that has been implemented with the layered architecture. According to this layered architecture, there is a hierarchy of layer, each layer using services are offered by the lower layers. We implement our project with the layered architecture, all of the functions are called from services.

Essential requirements are to develop the features of SFL Online efficiently. SFL Online needs to have its own unique implementation, with functionalities alike Course & Campus Online and more functionalities added. Optional requirements, in other words desired features, are going to be implemented. We are going to develop the system with ASP.NET MVC Framework that is a framework that uses C# infrastructure. Our system is an object-oriented system. This is a technique of coding and we will use database in the part of back-end. The front-end part will include interfaces for users. The interface pages are using web services.

CHAPTER 3

# Proposed System Analysis

This chapter is provides information about the system functionality.

## Overview

SFL Online is an online course management and student information system designed for the use of students and instructors of the SFL program. The system also has an administrator, whom manages the system and has full authentication to the system. The admin also sets which module is currently active. The next section (Functional Requirements) contains a list of all the functions about what any actor will be able to do in the system. Nonfunctional Requirements describes the systems behavior, which will affect the systems success criteria and performance, and includes system, interface and implementation details. Basically, a functional requirement describes *what* our software system should do, while non-functional requirements place constraints on *how* our system will do so.

## Functional Requirements

This section contains what each of our actors in the system must be able to do in the system.

Functional Requirements describe the high-level functionality of the system. They are the requirements the system must have, and corresponding actors are able to perform. Each actor has different functional requirements, and there are some common requirements for all actors. These common functionalities are associated with account-related functionalities.

**Functional Requirements shared by all actors:**

* All actors in the system must authenticate first to use the system. (Login)
* All actors must have a password recovery option in case they forget their password and are unable to authenticate, with their password reset and sent to them via e-mail.
* Once authenticated, all actors must be able to change their passwords at any time.
* All actors can view their personal information. (Name, Email, etc.)

**Functional Requirements for the Students:**

* Students shall be able to see their academic information, such as which track and class they belong to.
* Students shall be able to view their weekly course schedule.
* Students must be able to see all the courses given in the class they belong.
* Students shall be able to calculate their passing conditions, from being a prep student to a freshman in the faculty, given their grade average and exit exam score they should achieve as inputs.
* Students shall be able to see their grades entered for assignments, midterms and quizzes for each course.
* Students shall be able to see how much percentage does a grade has effect in a course. (For example, midterms having 30% while assignments have 15%)
* Students shall be able to see their weekly attendances for each course.
* Students who have finished at least one module shall be able to view their previous grade averages of completed module. They will also be able to see the total academic average score of their total completed modules.
* Students shall be able to view announcements.

**Functional Requirements for the Instructors:**

* Instructors shall be able to see the courses in the classes they are teaching.
* Instructors shall be able to view the list of students in the classes they teach.
* Instructors shall be able to see a student’s contact and academic information, such as their e-mail to contact and their grade averages from the class list.
* Instructors must be able to enter attendances of the students they are teaching in a course.
* Instructors must be able to enter the grades of midterms, quizzes and assignments of students in the courses they are teaching.
* Instructors shall be able to see their weekly course schedule, seeing which classes and courses they teach at a certain day and time in week.
* Instructors shall be able to write announcements.

**Functional Requirements for the Admin:**

* Admin shall be able to display personal information and account information of students and instructors.
* Admin shall be able to add, edit or delete students or instructors from the system.
* Admin shall be able to change what percentage a grade will have effect in a course. (Change grade percentages)
* Similarly, Admin must be able to change the exit exam passing conditions. (e.g. change passing condition from 70 out of 100 to 65 out of 100 points)
* Admin shall be able to set which module is currently active in SFL Online.
* Admin shall be able to add, edit or delete classes from the system database.
* Admin shall be able to declare which track a class belongs in.
* Admin shall be able to see the quota of a class, and won’t be able to add new students to a class if the class quota is full.
* Admin shall be able to add, edit or delete courses from the system database.
* Admin shall be able to enroll students to classes according to their first exit grade, placing them in tracks and classes automatically and in bulk, corresponding to their first exit grade.
* Admin can also enroll a student in a class by hand.
* Admin shall be able to enroll instructors into courses in a class they shall teach.

## Nonfunctional Requirements

This section describes user-level requirements those are not directly related to functionality.

### Usability

SFL Online will include three different interfaces for each actor. These interfaces will include all the functions an actor must perform. User interface of each actor shall look similar in design, but perform differing functions according to the actor. For usability, these interfaces should be easy to use and understand, with the actor being able to perform their tasks without explanation. They should also be appealing to the eye while being simple in design. Users should be able to reach their goals with minimal number of clicks.

### Reliability

SFL Online shall be secure, and will not allow any unauthorized user to enter the system. Only users permitted and defined by the admin in the database shall be able to authenticate to the system, using their account information. There is no “registration” to the system by users who is not an admin. EF ORM is used in the project to keep the database, using a code-first approach. OWIN Cookie authentication and ASP.NET Identity for OWIN is used to provide reliability while logging in. In addition, for it to be reliable, the system should not be down or crash in case of errors, and should be running 90% of the time.

### Performance

SFL Online is designed to be used by multiple numbers of users. Therefore, the system must allow at least 1500 parallel users. Response time should be fast, with a maximum of 4 seconds waiting time in regular site traffic. AJAX HTTP GET or POST requests may be used to receive information and feedback, and task completions of users without reloading the page.

### Supportability

The system should be reachable over any browser (maybe with the exclusion of older versions of internet explorer) in a standard computer. Management and maintenance of SFL Online belongs to the system administrator. Maintenance of SFL Online should be able to be done easily, and in certain time intervals. Interfaces for database altering, tables and instances should be shown in administrator panel.

### Implementation

The system will be implemented on Visual Studio platform. C# will be used as the programming language and ASP.NET will be used as a framework in the process. EF will be used as an ORM tool. We will be using a “code-first” approach, which means we will construct the Models first and the virtual relations between them, and the database will be created based on these models [7]. Model-view-controller design methodology is going to be implemented in this project, also provided by .NET. User Interface should be web-based (accessible via WWW Browser).

In front end, HTML5, Razor, CSS and JavaScript will be used. JQuery and AJAX methods will be used as JavaScript libraries. Bootstrap library will be used for prettier designs and responsiveness.

### Interface

The interface should be easy to use for all actors, as usability is the main concern. There should be no guide to instruct the actor about what and how to do their tasks. The interface shouldn’t be too colorful and eye-tiring, instead a simple color scheme should be used. The colors of school might be used. Menu navigation should change according to actor. There will be a side menu navigation in our interfaces. Similar tasks and functions should be put together.

### Packaging

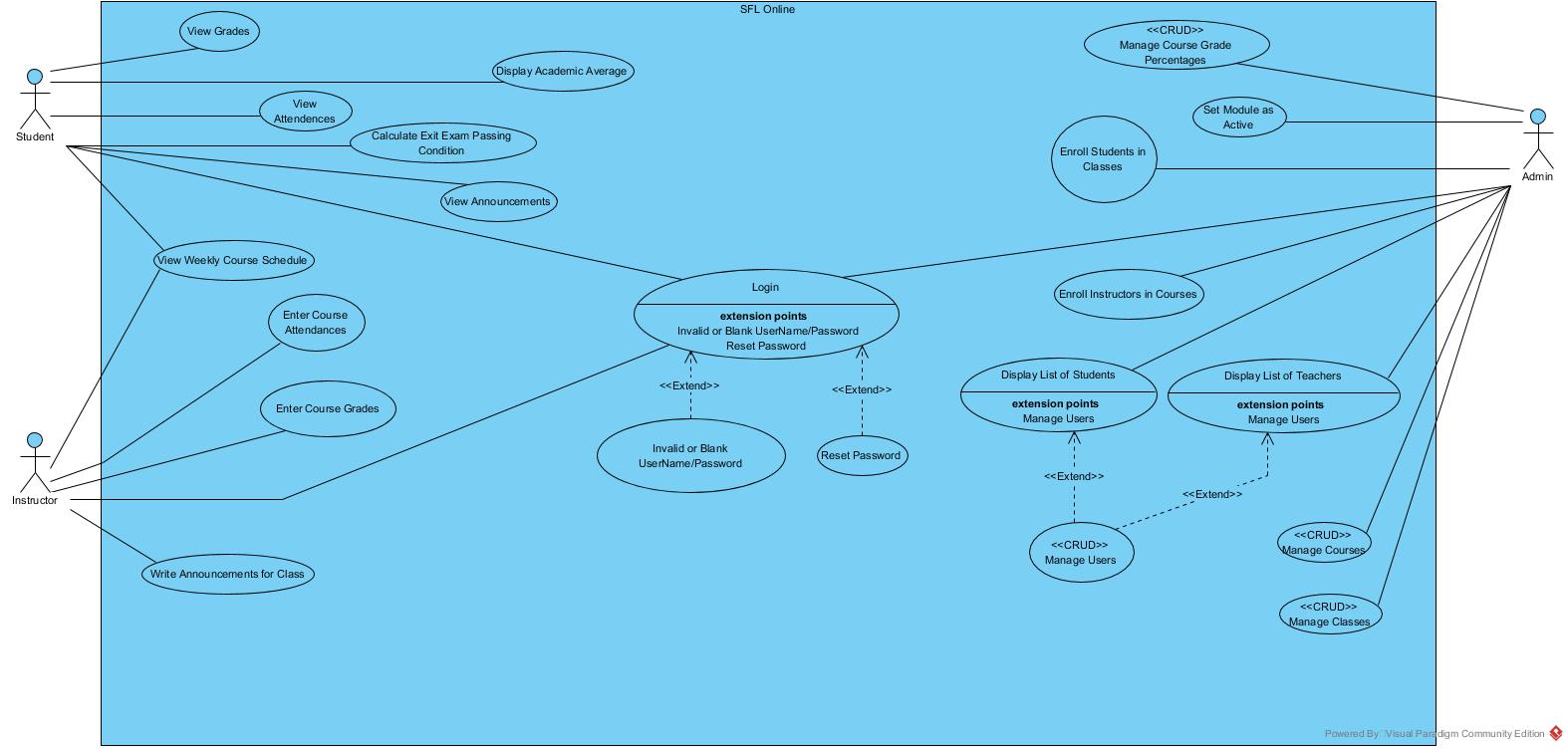
We are planning for the project to be used by the SFL program, with the project fulfilling the functional requirements of entering attendance and grades, and calculating exit exam passing conditions, but other constraints on the actual delivery of the system are not determined. Thus, packaging requirements will be decided in the future.

### Legal

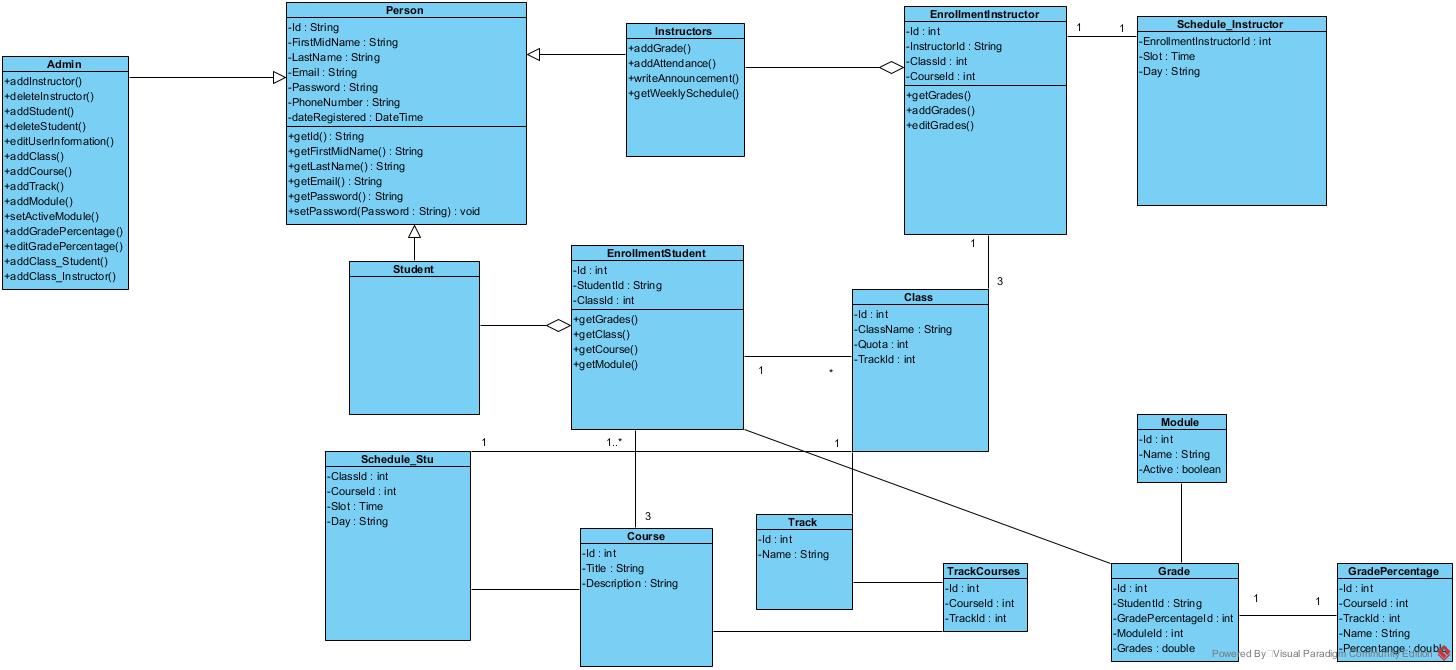
The software is provided "as is", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and no infringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the software or the use or other dealings in the software. (MIT License)

## System Models

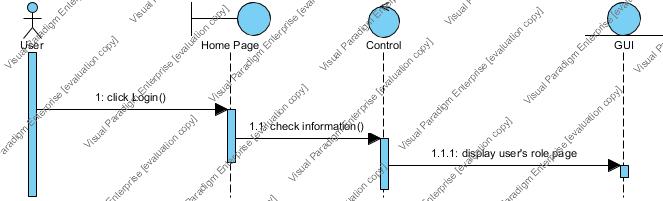
### Use case model



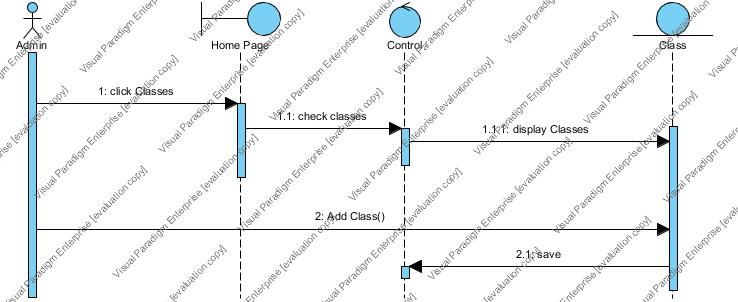
### Object model

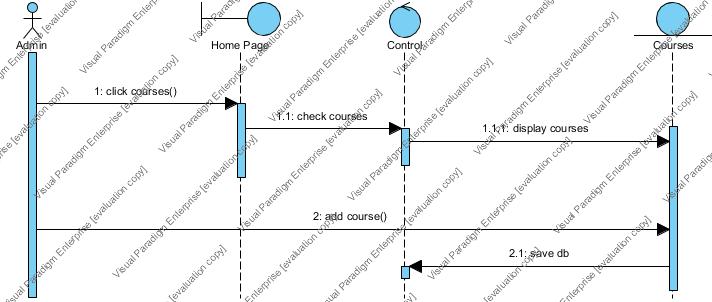


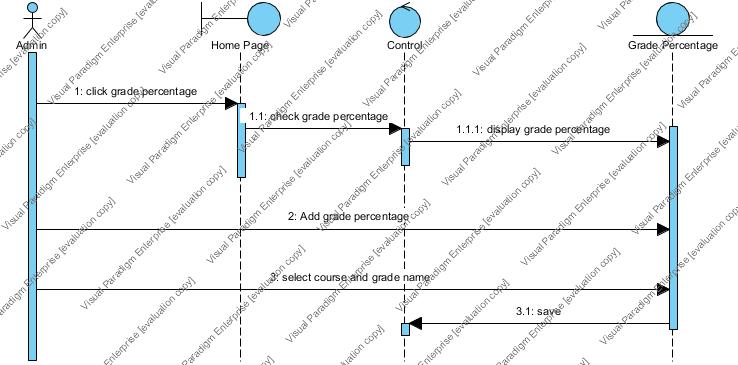
### Dynamic models

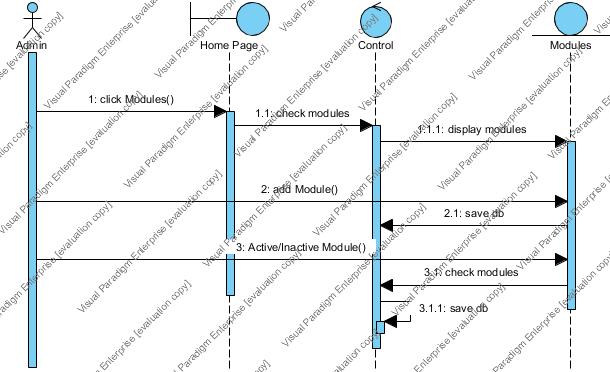
Login

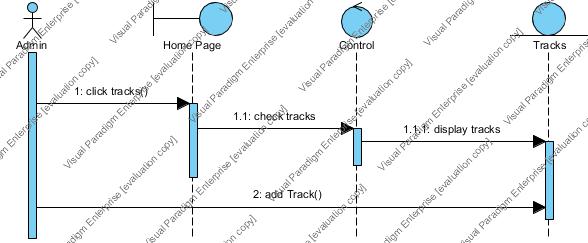
Admin Add Class

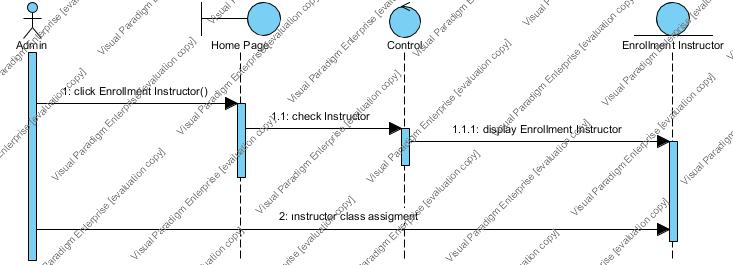


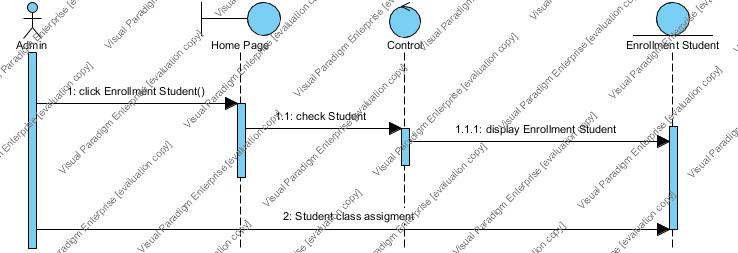
Admin Add Course

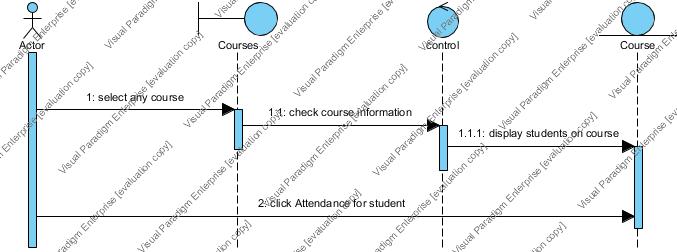
Admin Add Grade

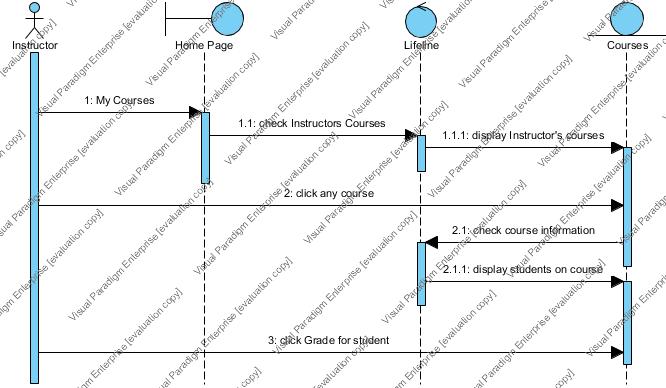
Admin Add Module

Admin Add Track

Admin Enrollment Instructor

Admin Enrollment Student

Instructor Attendance

Instructor Add Grades

## System Decomposition

SFL Online is made of three major actors that are named students, instructors and system administrators. Each actor in the system has different authorities. Operations are divided into actor authorities. We have three actors to implement. We also have common account functionalities for all the actors, such as logging in and changing password. So, we have four different sub-systems for interfaces, corresponding to each actor and one of their common functionalities.

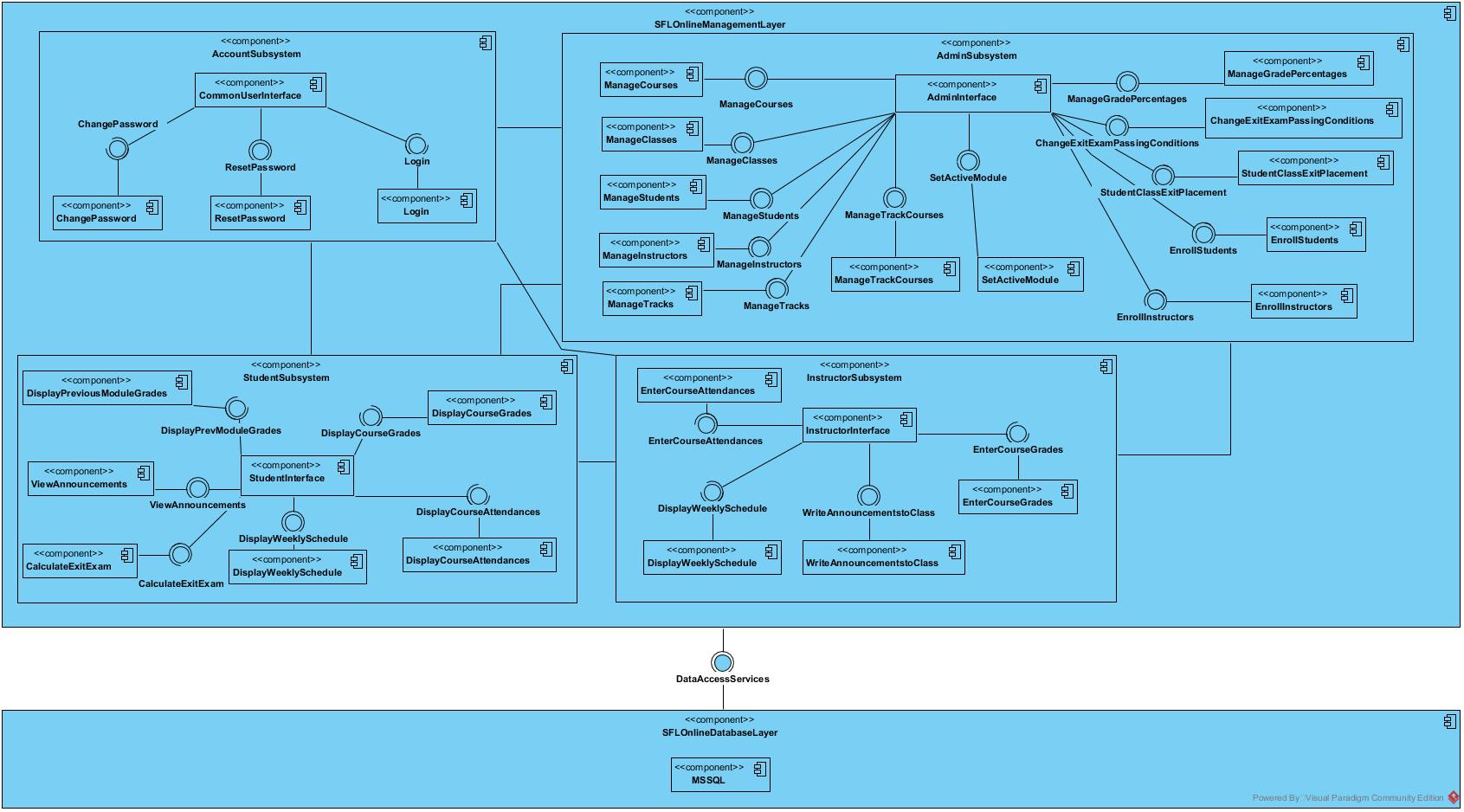


Figure 1: Coupling view of Subsystem Decomposition

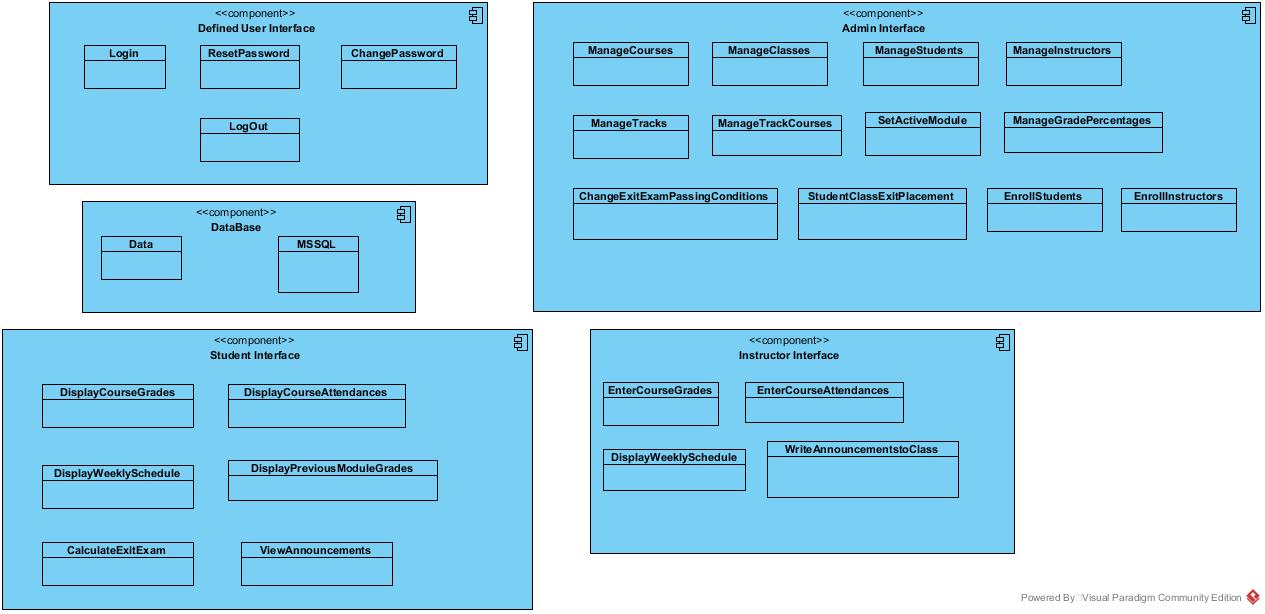


Figure 2: Cohesion view of Subsystem Decomposition

CHAPTER 4

# IMPLEMENTATION DETAILS, TESTS, AND EXPERIMENTS

## Object Design Trade-offs

* Instead of using a Database-First approach, we've decided to use code-first approach in the entity framework due to the fact it provides auto-generated code, reducing development time.
* Instead of using ASP.NET Identity infrastructure, we've used Identity partially with OWIN. This is due to Identity, when 'Individual User Accounts' is selected as authentication when first creating the project, it creating it's own models and database with many tables and columns for each table, such as “twofactorloginenabled”, “concurrenystamp”, etc. It had many columns unnecessary for our project, and changing the default Identity tables required a lot of extra code and time. On the other hand, these default tables and Identity package come with many useful functions, and provide extra security, such as hashing passwords. We've instead provided more than sufficient security by using OWIN middleware and Identity for OWIN. Also, all form posts validate anti forgery tokens to provide security. Use of many unnecessary attributes would also increase the complexity of the database and code. We've also wanted to create our own databases from code, to have more flexibility over it. Trade-offs we've chosen for this case are development time, simplicity and flexibility over security.
* Instead of defining Email, Password, User ID, etc. attributes for each actor model (Instructor, Student and Admin), we have inherited all the actor models from Person model, defining these attributes only once in Person model, and passing these attributes to the specialized actor models. Thus, complexity is decreased.
* Instead of using a service or defining another table, we have given admin the functionality to set the modules as 'active' or 'inactive' from the system. Active module is a bool attribute.
* Instead of using AJAX methods, we're refreshing the page on most form posts. AJAX post method is used only in a few forms in the system. We've traded-off development time over response-time.
* While we have invested development time for UI to look appealing, our main focus was for the system functional requirements to work properly. As a result, our UI’s look simple but easy to use. All of our functional requirements are satisfied in the project.

CHAPTER 5

# Conclusions and Future Work

We have managed to implement all the functional requirements in our code. Due to the fact there is no previous existing system for SFL program, and the differences in class, semester and grading mechanisms compared to the faculty, creating a database and relations was challenging. Yet, we think we have managed to build a prototype system that will be the reference system for a SFL system later to be developed, or improved by using our system.