

**SFL ONLINE**

Object Design

Version 1.0

**Ekin Naz EREN – 213CS2301**

**Tolga ŞIVGIN – 214CS2013**

Prepared for

CSE490 Project



Table of Contents

[**1.** **Introduction** 2](#_Toc535406844)

[**1.1** **Object Design Trade-offs** 2](#_Toc535406845)

[**1.2** **Interface Documentation Guidelines** 3](#_Toc535406846)

[**1.3** **Definitions, Acronyms and Abbreviations** 3](#_Toc535406847)

[**1.4** **References** 3](#_Toc535406848)

[**2.** **Packages** 3](#_Toc535406849)

[**3.** **Class Interfaces** 4](#_Toc535406850)

[**4.** **References** 8](#_Toc535406851)

**OBJECT DESIGN DOCUMENT [1]**

# **Introduction**

## **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.

## **Interface Documentation Guidelines**

Front-end design:

* Static and dynamic content, such as CSS & Bootstrap files, scripts and images are kept in a file called content.
* ASP.NET Core Razor Pages and Razor syntax are used for the view pages of the project. ASP.NET Core Razor Pages is a page-focused framework for building dynamic, data-driven web sites with clean separation of concerns. [2]
* Each actor in the system has their specified layout page where assets – sidebar navigation, page content area, footer – are held. This layout is the same for all pages of an actor, with content or body area changing accordingly.
* Bootstrap 4 is used for styling UI elements.
* HTML forms styled with bootstrap are used for input operations. They use HTTP GET/POST, and secured by validating anti forgery token in controllers and views.
* ASP.NET Identity role authorization are implemented in pages for security.

## **Definitions, Acronyms and Abbreviations**

SFL: School of Foreign Language

RAD: Requirements Analysis Document

SDD: System Design Document

UI: User Interface

OWIN: Open Web Interface for .NET

## **References**

* RAD of SFL Online
* SDD of SFL Online
* Model Implementation

# **Packages**

SFL: School of Foreign Languages

SFLOnline: Name of our solution/project, sln file.

* SFLOnline
  + src

# **Class Interfaces**

Model name: Person

Model attributes – explanation – dependencies:

*Model attributes – explanation:*

* + Id - Unique user id in string, representing the identity number of a student, instructor or admin in the university.
  + FirstMidName – First name of the user, and their middle name if there is any.
  + LastName – Last name of the user.
  + E-mail – University E-mail address of the user.
  + Password – Password of the user.

*Dependencies:*

* None. Person table is the parent model of Admin, Instructor and Student models.

Model Operations:

* Full Name – toString method that returns FirstMidName and FullName of the user joined as one string.
* Getter and setter methods for model attributes.

Model name: Admin

Model attributes – explanation – dependencies:

*Model attributes – explanation:*

* + All user information of the Admin comes from the Person model.

*Dependencies:*

* Admin model is inherited from the Person model, thus it is dependent with Person model.

Model Operations:

* None

Model name: Student

Model attributes – explanation – dependencies:

*Model attributes – explanation:*

* + All user information of the Student comes from the Person model.

*Dependencies:*

* Student model is inherited from the Person model, thus it is dependent with Person model.
* It is also dependent with EnrollmentStudent model.

Model Operations:

* None

Model name: Instructor

Model attributes – explanation – dependencies:

*Model attributes – explanation:*

* + All user information of the Instructor comes from the Person model.

*Dependencies:*

* Instructor model is inherited from the Person model, thus it is dependent with Person model.
* It is also dependent with EnrollmentInstructor model.

Model Operations:

* None

Model name: EnrollmentStudent

Model attributes – explanation – dependencies:

*Model attributes – explanation:*

* + Id – Unique integer identifier of model. Auto increments by one for each entry.

*Dependencies:*

* It is dependent with student and class models.

Model Operations:

* None

Model name: EnrollmentInstructor

Model attributes – explanation – dependencies:

*Model attributes – explanation:*

* + Id – Unique integer identifier of model. Auto increments by one for each entry.

*Dependencies:*

* It is dependent with instructor, course and class models.

Model Operations:

* None

Model name: Class

Model attributes – explanation – dependencies:

*Model attributes – explanation:*

* + Id – Unique integer identifier of model. Auto increments by one for each entry.
  + ClassName – The name of the classroom.
  + Quota – The maximum limit of students a class can hold.

*Dependencies:*

* It is dependent with Track model.

Model Operations:

* None

Model name: Course

Model attributes – explanation – dependencies:

*Model attributes – explanation:*

* + Id – Unique integer identifier of model. Auto increments by one for each entry.
  + Title – The name/title of the course.

*Dependencies:*

* None

Model Operations:

* None

Model name: Track

Model attributes – explanation – dependencies:

*Model attributes – explanation:*

* + Id – Unique integer identifier of model. Auto increments by one for each entry.
  + Name – The name of the track.

*Dependencies:*

* None

Model Operations:

* None

Model name: TrackCourses

Model attributes – explanation – dependencies:

*Model attributes – explanation:*

* + Id – Unique integer identifier of model. Auto increments by one for each entry.

*Dependencies:*

* It is dependent with course and track models.

Model Operations:

* None

Model name: Module

Model attributes – explanation – dependencies:

*Model attributes – explanation:*

* + Id – Unique integer identifier of model. Auto increments by one for each entry.
  + Name – Name of the module
  + IsActive – A Boolean attribute indicating if this module is active or inactive.

*Dependencies:*

* None.

Model Operations:

* None

Model name: Announcement

Model attributes – explanation – dependencies:

*Model attributes – explanation:*

* + Id – Unique integer identifier of model. Auto increments by one for each entry.
  + AnnouncementName – Title of the announcement
  + Description – Body content of the announcement.

*Dependencies:*

* It is dependent with Person and Class models.

Model Operations:

* None

Model name: ClassSchedule

Model attributes – explanation – dependencies:

*Model attributes – explanation:*

* + Id – Unique integer identifier of model. Auto increments by one for each entry.

*Dependencies:*

* It is dependent with EnrollmentInstructor, Slot and Day models.

Model Operations:

* None

Model name: Day

Model attributes – explanation – dependencies:

*Model attributes – explanation:*

* + Id – Unique integer identifier of model. Auto increments by one for each entry.
  + DaysName – Name of the day in a week.
  + DayNumber – Number corresponding to that day.

*Dependencies:*

* It is dependent with ClassSchedule model.

Model Operations:

* None

Model name: Slot

Model attributes – explanation – dependencies:

*Model attributes – explanation:*

* + Id – Unique integer identifier of model. Auto increments by one for each entry.
  + SlotName – Name of the slot corresponding to a timeline (ex. 9-10 am).
  + SlotNumber – Number corresponding to that slot.

*Dependencies:*

* It is dependent with ClassSchedule model.

Model Operations:

* None

Model name: Exit

Model attributes – explanation – dependencies:

*Model attributes – explanation:*

* + Id – Unique integer identifier of model. Auto increments by one for each entry.
  + ExitName – Name of the exit exam.
  + ForEnrollment – A Boolean indicating if that exit exam is the current exit exam to make enrollments.

*Dependencies:*

* It is dependent with ExitTracks and StudentExitGrades models.

Model Operations:

* None

Model name: ExitTrack

Model attributes – explanation – dependencies:

*Model attributes – explanation:*

* + Id – Unique integer identifier of model. Auto increments by one for each entry.
  + Grade – Maximum grade corresponding to the placement in that track by exit exam.

*Dependencies:*

* It is dependent with Exit and Track models.

Model Operations:

* None

Model name: StudentExitGrade

Model attributes – explanation – dependencies:

*Model attributes – explanation:*

* + Id – Unique integer identifier of model. Auto increments by one for each entry.
  + Grade – Grade of a student from an exit exam.

*Dependencies:*

* It is dependent with Exit and Student models.

Model Operations:

* None

# **References**

1. Bruegge B. & Dutoit A.H.. (2010). Object-Oriented Software Engineering Using UML, Patterns, and Java, Prentice Hall, 3rd ed.
2. https://www.learnrazorpages.com/