

**SFL ONLINE**

System Design

6.1.2019

Version 1.0

**Ekin Naz EREN – 213CS2301**

**Tolga ŞIVGIN – 214CS2013**

Prepared for

CSE490 Project



Table of Contents

[**1.** **Introduction** 2](#_Toc535394208)

[**1.1** **Purpose of the System** 2](#_Toc535394209)

[**1.2** **Design Goals** 2](#_Toc535394210)

[**1.3** **Definitions, Acronyms and Abbreviations** 2](#_Toc535394211)

[**1.4** **References** 3](#_Toc535394212)

[**2.** **Current Software Architecture** 3](#_Toc535394213)

[**3.** **Proposed Software Architecture** 3](#_Toc535394214)

[**3.1** **Overview** 4](#_Toc535394215)

[**3.2** **System Decomposition** 5](#_Toc535394216)

[**3.3** **Hardware-Software Mapping** 6](#_Toc535394217)

[**3.4** **Persistent Data Management** 7](#_Toc535394218)

[**3.5** **Access Control and Security** 10](#_Toc535394219)

[**3.6** **Global Software Control** 10](#_Toc535394220)

[**3.7** **Boundary Conditions** 10](#_Toc535394221)

[**4.** **Subsystem Services** 12](#_Toc535394222)

[**5.** **References** 14](#_Toc535394223)

SYSTEM DESIGN DOCUMENT [1]

# **Introduction**

## **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.

## **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.

## **Definitions, Acronyms and Abbreviations**

SDD: System Design Document

RAD: Requirements Analysis Document

SFL: School of Foreign Languages

ORM: Object Relational Mapping

EF: Entity Framework (ORM for .NET)

SQL: Structured Query Language

MSSQL: Microsoft Structured Query Language

SSMS: SQL Server Management Studio

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

CRUD: Create Read Update Delete

UI: User Interface

OWIN: Open Web Interface for .NET

IDE: Integrated Development Environment

## **References**

* RAD of SFL Online
* Course Online
* Campus Online

# **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.

## **Overview**

We are going to develop the system with ASP.NET MVC Framework, which is a framework that uses C# programming language and has model-view-controller design pattern infrastructure. The complete system is divided into smaller sub-systems with own activities.

SFL Online consists of three actors, namely students, instructors and admin. Each actor in the system has different authorities. Operations are divided into actor authorities. All the actors share some common account related operations, such as logging in and out, and changing password. For each of these three actors, plus the account operations, we have four subsystems.

To provide a bird's eye view of the software architecture used in the system;

***Admin Subsystem:*** Provides services to admin to manage the system. It allows admin to perform CRUD operations in the database for classes, courses, student and instructor lists, tracks, courses a track includes(Track Courses), and Exit exam passing conditions. It also allows admin to enroll students by bulk according to their first exit exam grade or one by one, enroll instructors in classes, and set a module as currently active module.

***Instructor Subsystem:*** Provides services that allow instructors to enter grades and attendances for a course they are teaching in a class, view their weekly class schedule and write announcements for a class they are teaching.

***Student Subsystem:*** Provides services to students that allows students to display their grades and attendances for a course they are currently taking, display weekly schedule, display their previous module grades taken if they have previously completed a module, calculate which grade they should take from exit exam in order to become freshman and view announcements.

***Defined User Subsystem:*** Allows the system administrator, instructors or students to login to the system, request a password reset in case of forgotten password and change their password once authenticated.

## **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.

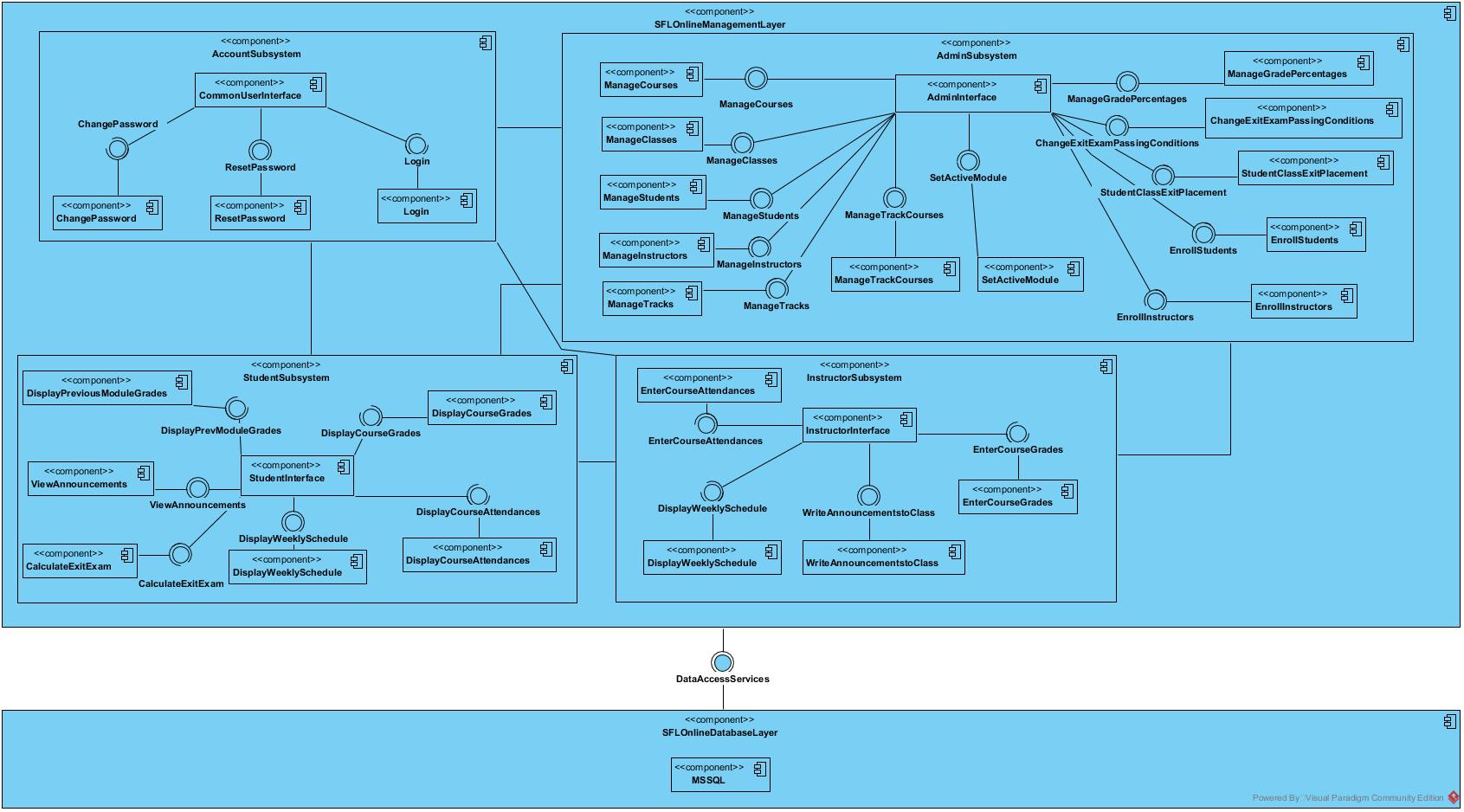


Figure 1: Coupling view of Subsystem Decomposition

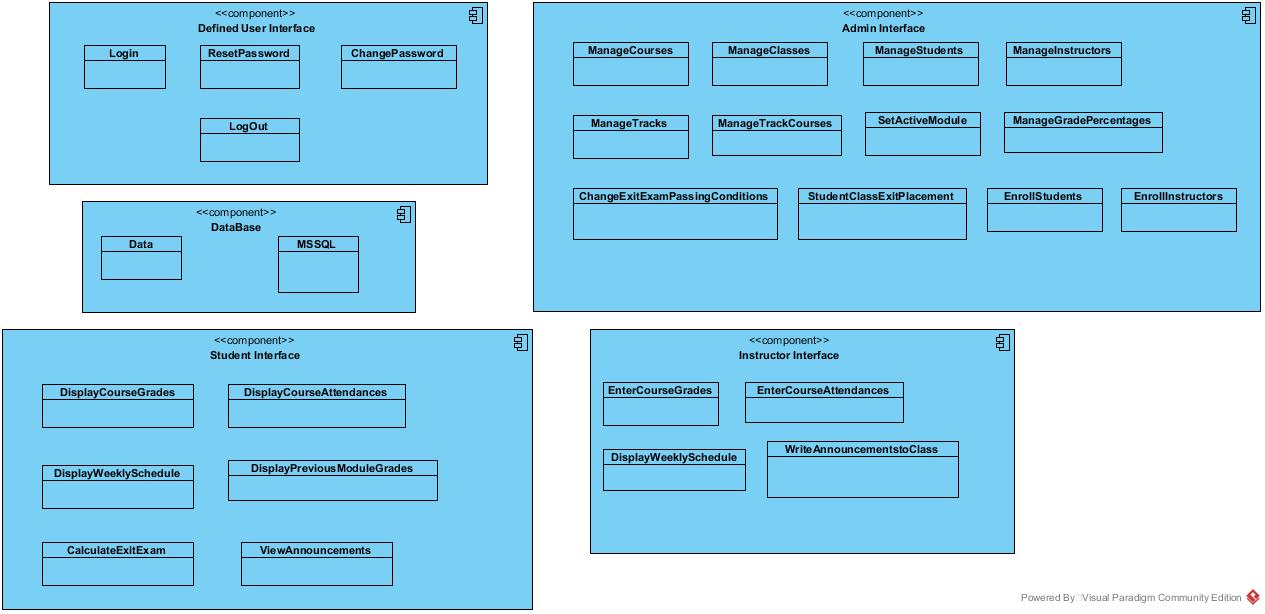


Figure 2: Cohesion view of Subsystem Decomposition

## **Hardware-Software Mapping**

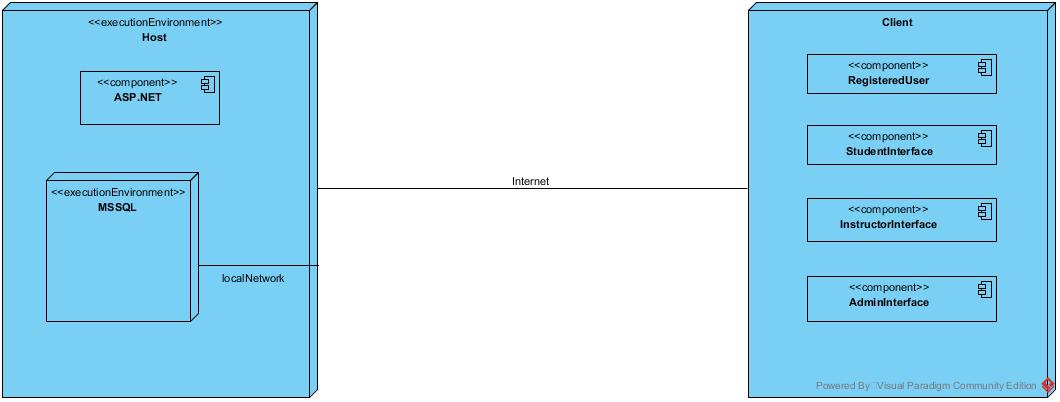


Figure 3: Hardware-Software Mapping Deployment Diagram

We are able to examine hardware-software mapping by two sides that are actors and system.

* In terms of actors, there is no any constraint on hardware-software mapping. SFL Online is a web-based application that developed by C# ASP.NET infrastructure. All different type actor is able to reach to the system with ordinary computer, or mobile devices with internet-connection.
* In terms of system, we are going to demonstrate the system with IIS Express (IIS, Internet Information Services), a web-server provided by Visual Studio. Full version of SFL Online might able to demonstrate with Digital Ocean, Google, Outlook or Amazon Cloud services, and so on.

## **Persistent Data Management**

We are going to work with MSSQL database. Being a Microsoft technology, MSSQL is easy to use and well-integrated with Visual Studio, and visual studio provides SSMS Express to use in itself, without opening another program. Microsoft SQL Server is a relational database management system, or RDBMS, that supports a wide variety of transaction processing, business intelligence and analytics applications in corporate IT environments. It's one of the three market-leading database technologies, along with Oracle Database and IBM's DB2. [4]

We are going to work with object relational model, in shortly ORM. Object relational model provides more power, greater flexibility, better performance and greater data integrity then those that came before it. Some of the benefits that are offered by the Object-Relational Model include:

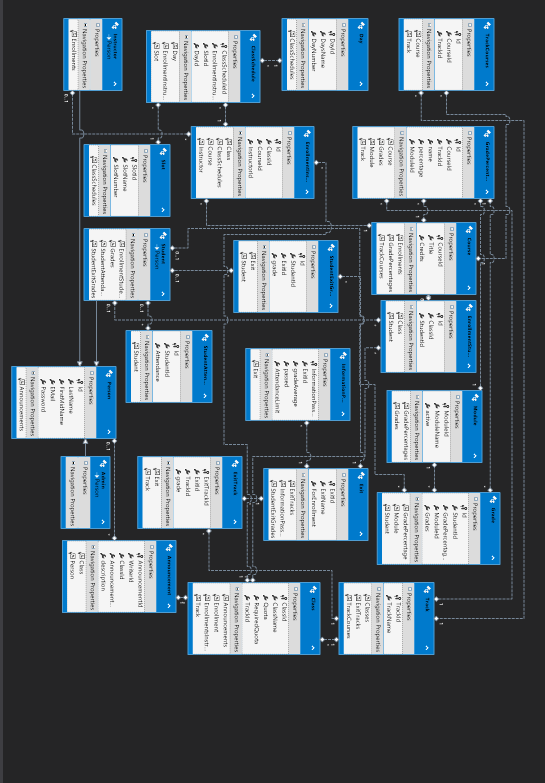
* Extensibility – We are able to extend the capability of the database server; this can be done by defining new data types, as well as user-defined patterns. This allows the us to store and manage data.
* Complex types – It allows us to define new data types that combine one or more of the currently existing data types. Complex types aid in better flexibility in organizing the data on a structure made up of columns and tables.
* Inheritance – We are able to define objects or types and tables that procure the properties of other objects, as well as add new properties that specific to the object that been defined. It also complements object-oriented programming, which we are using.

Entity Framework (EF) will be used as ORM tool, and we will be using a Code-First approach. Code-First is mainly useful in Domain Driven Design. In the Code-First approach, we focus on the domain of your application and start creating classes (model of MVC) for our domain entity rather than design our database first and then create the classes which match your database design. We use this approach for speeding up development, cleaner code generation and database relations.

Entity Framework provides:

* Provides dedicated functionality for CRUD operation (Create, Read, Update, Delete). Easy to implement CRUD operations.
* If we want to replace the data store, it is very easy to replace without modifying the data access logic since all data access logic are present in higher level.
* Easy to manage one to one, one to many and many to many relationships between tables.
* Conceptual model can be represented in a better way.
* It enables developers to visually design models and mapping of database.
* Developer can reduce the code in classes and sub-classes for data access.
* It provides auto generated code, reduces development time and development cost.

**Description of Encapsulation of the Database**



'Person' is an abstract user, which indicates a defined user in the system. All the other actors are inherited by this user in the model. With the help of EF ORM, only 'Person' table is seen as a user in the database, and the user role is defined by a 'Discriminator' column in the Person table, defining a user in a specific role.

* **Person Model:** Holds the common attributes and information shared by all users in the system, namely First-Middle Name, Last Name, School ID, E-mail, password. In the models; Admin, Instructor and Student inherit from this model. Instructor and Student has different connections to other tables. After migrating the models to the database via EF, Person table creates a 'Discriminator' column, which defines a Person's role in the system. [5]
* **Admin Model:** Holds the same attributes as person table and is inherited from it.
* **Student Model:** Inherited from Person Model.
* **Instructor Model:** Inherited from Person Model.
* **Course Model**: Course model stores all the information about courses. It needs Title as varchar and Credits as int.
* **Class Model**: Class model stores all the information about classrooms. It has it’s unique int id, class name as varchar, quota which is the maximum limit a class can hold in int, requiredquota in int, which indicates the current population of a class, and TrackId as foreign key.
* **Exit Model:** Indicates an exit exam. It has a unique int ID, Exit Name as varchar and the score limit forenrollment.
* **EnrollmentInstructor:** Instructors that teaches courses in classes. It needs class,course and instructor as foreign key, and has it’s unique int id.
* **EnrollmentStudent:** Students that belong in a classroom. It needs class and student as foreign key. It has it’s unique int id.
* **StudentExitGrade:** An exit grade score of a student. It takes student and exit models as foreign key. It needs grade field as float.
* **Module:** A semester is named module in SFL. It needs a unique int id, module name in varchar and active in bool to show if that module is active or inactive.
* **GradePercentage:** Shows how much a certain type of grade has effect in a course (Homework has 15% effect for Reading course for example). It needs a unique int ID, name in varchar, percentage in int, and takes course, track and module as foreign key.
* **TrackCourses:** Shows courses that belong in a track. It needs a unique int ID, and takes course and track as foreign key.
* **Grade:** Shows a grade of a student in a course. It needs student, grade percentage and module as foreign key, grade as float and a unique int identifier.
* **StudentAttendance:** Shows a student’s total attendance. It takes student as foreign key, has unique int id and attendance as int.

Day and slot indicates days in a week and slots from 9 am to 5 pm respectively. They are used for weekly schedule display of instructors and students.

EF makes sure if a row in a table related to another table by Foreign Key is deleted, the rows including that deleted data in the related table are deleted too, preventing crashes and errors in the database and in our system.

All uniqueness check is controlled in models.

## **Access Control and Security**

In case of security, user’s access to the system with a uniquely defined e-mail and password at login page. OWIN Cookies and ASP.NET Identity are used for system security. Each user has a role in the system, and a model inherited from the Person table according to their role, then defined in the database via ORM. User accesses services by the actor role, and redirected to their respective pages according to their role.

Each user, which are instructors, students and admin, has their permissions and interfaces defined. After login, each actor is redirected to their respective home pages, providing them an interface to perform their operations. If an actor tries to reach another actors function or page, a function or page that they’re not allowed or authorized to do, they are thrown back to the login page and cannot reach that page. System administrator has full management authentication, and can control and manage the system and the database. Thus, because of front-end structure, an authenticated user in a specific role cannot reach another actor’s permissions.

## **Global Software Control**

SFL Online is a web-based system that uses HTTP GET/POST requests. Thus, having an internet connection is required to access the system. The system uses cookies in authentication, and keeps the user’s first name, e-mail and role on a thread. After successfully authenticating, the user is redirected to their respective page according to their role. Users role is detected on login, so that the user is redirected correctly to their respective home page, and cannot access to a page that he/she does not have permission to display.

Accessible pages are synchronized over internet-connection. The synchronization is satisfied among queries on server. To succeed this, database connection is also required. After database connection has been done, synchronization operations are satisfied.

## **Boundary Conditions**

**Start-up:**

* The system must be run in Visual Studio platform, via IIS Express, through any browser.
* SFL Online needs to have a stable internet connection to work properly.

**Shutdown:**

Shutdown might occur with two-ways;

* Physical hardware might be interrupted.
* IIS might be stopped.

**Error behavior:**

* Internet-connection might be unconnected.
* Physical hardware might be interrupted.
* IIS Express might have problems.
* Database might be lost.
* Database connection might be unconnected.
* User might have attempted to login with invalid credentials. (These exceptions are getting caught by the system and handled properly.)

While we’re not implementing start and stop server functionalities, when the project is live, admin will need server functionalities in order to do maintenance in the system.

SFL Online is initialized by the system admin invoking the ‘Initialize System’ use case. Once the initialization of the system is completed, the departments and the curriculum information are registered onto system by the system admin. In addition, the system admin initialize the server by invoking the ‘Start Server’ use case. After that, when the system is brought from non-initialized state to steady-state, the system is opened and free for the system users (Students, Instructors, and Admin) to login and perform their tasks. When the system is terminated by system admin invoking the ‘Shutdown Server’ use case. All users that currently logged in to the system will be disconnected. This situation occurs for instance, the system admin closes the system for a maintenance before the registration day(s). No user can login to the system until the system is initialized by the admin again.

The use cases mentioned above, namely ‘Initialize System’, ‘Start Server’, ‘Shutdown Server’ can be seen below:

|  |
| --- |
| *Use case name:* InitializeSystem |
| *Participant actors:* Initiated by Admin |
| *Flow of events:* 1. Upon successful login, the Admin executes the commands to register the class, course, track, module, grade percentage, exit exam information according to the SFL’s rules. |
| *Entry Condition:* The Admin logs into the database server that the SFL Online uses. |
| *Exit Condition:* Initialization of the system is completed and the prep school information of the university are registered onto the database server. |

|  |
| --- |
| *Use case name:* StartServer |
| *Participant actors:* Initiated by Admin |
| *Flow of events:* 1. Upon successful login, the Admin executes the command that is used for the start the server. |
| *Entry Condition:* The Admin logs into the server machine that the SFL Online services built on. |
| *Exit Condition:* SFL Online services are available and waits for connections from user of the SFL Online. |

|  |
| --- |
| *Use case name:* ShutdownServer |
| *Participant actors:* Initiated by Admin |
| *Flow of events:* 1. Upon successful login, the Admin executes the command that is used for the shut down the server. |
| *Entry Condition:* The Admin logs into the server machine that the SFL Online services built on. |
| *Exit Condition:* SFL Online services are unavailable and connections which were already activated are disabled. |

# **Subsystem Services**

The subsystem decomposition of SFL Online system, we divide the system into smaller subsystems with strong coherences.

The subsystem separation shows the entities of following subsystem

* Defined User Subsystem
* Admin Subsystem
* Instructor Subsystem
* Student Subsystem
* Database Subsystem

***Defined User Subsystem***

This subsystem manages account activities that are common to all users.

This subsystem is provided by:

* Login
* Reset Password (In case of forgot password)
* Change Password (After logging in)
* Logout

***Admin Subsystem***

This subsystem provides for admin to manage all the system.

This subsystem is provided by:

* Manage Courses
* Manage Classes
* Manage Students
* Manage Instructors
* Manage Tracks
* Manage Track Courses
* Set Active Module
* Manage Grade Percentages
* Change Exit Exam Passing Conditions
* Student Class Exit Placement
* Enroll Students
* Enroll Instructors

***Instructor Subsystem***

This subsystem provides to enter and edit grade and attendance information in specific courses of a class.

This subsystem provided by:

* Enter Course Grades
* Enter Course Attendances
* Display Weekly Schedule
* Write Announcements to Class

***Student Subsystem***

This subsystem provides to display grade and attendance information in specific courses of a class.

This subsystem is provided by:

* Display Course Grades
* Display Course Attendances
* Display Weekly Schedule
* Display Previous Module Grades
* Calculate Exit Exam
* View Announcements

# **References**

1. Bruegge B. & Dutoit A.H.. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Prentice Hall, 3rd ed.
2. Course Homepages Management System of ISIK University (2003), [Course Online](https://course.isikun.edu.tr/)
3. Campus Automation System (2002) University of Işık,[**Campus Online**](http://campus.isikun.edu.tr/)
4. <https://searchsqlserver.techtarget.com/definition/SQL-Server>
5. [Implementing Inheritance with the Entity Framework 6 in an ASP.NET MVC 5 Application](https://docs.microsoft.com/tr-tr/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/implementing-inheritance-with-the-entity-framework-in-an-asp-net-mvc-application)