

**SFL ONLINE**

Requirements Specification and Analysis

Version 1.0

**Ekin Naz EREN – 213CS2301**

**Tolga ŞIVGIN – 214CS2013**

Prepared for

CSE490 Project



Table of Contents

[1. Introduction 2](#_Toc534517525)

[**1.1** **Purpose of the System** 2](#_Toc534517526)

[**1.2** **Scope of the System** 2](#_Toc534517527)

[**1.3** **Objectives and Success Criteria of the Project** 3](#_Toc534517528)

[**1.4** **Definitions, Acronyms, and Abbreviations** 4](#_Toc534517529)

[**1.5** **Overview** 5](#_Toc534517530)

[2. Current System 5](#_Toc534517531)

[3. Proposed System 5](#_Toc534517532)

[**3.1** **Overview** 6](#_Toc534517533)

[**3.2** **Functional Requirements** 6](#_Toc534517534)

[**3.3** **Nonfunctional Requirements** 8](#_Toc534517535)

[3.3.1 Usability 8](#_Toc534517536)

[3.3.2 Reliability 8](#_Toc534517537)

[3.3.3 Performance 8](#_Toc534517538)

[3.3.4 Supportability 8](#_Toc534517539)

[3.3.5 Implementation 9](#_Toc534517540)

[3.3.6 Interface 9](#_Toc534517541)

[3.3.7 Packaging 9](#_Toc534517542)

[3.3.8 Legal 9](#_Toc534517543)

[**3.4** **System Models** 10](#_Toc534517544)

[3.4.1 Scenarios 10](#_Toc534517545)

[3.4.2 Use case model 38](#_Toc534517546)

[3.4.3 Object model 39](#_Toc534517547)

[3.4.4 Dynamic model 40](#_Toc534517548)

[3.4.5 User interface—Navigational paths and screen mock-ups 44](#_Toc534517549)

[4. Glossary 48](#_Toc534517550)

[5. References 50](#_Toc534517551)

REQUIREMENTS ANALYSIS 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.

## **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 [2].

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.

## **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.

## **Definitions, Acronyms, and Abbreviations**

RAD: Requirements Analysis Document

SFL: School of Foreign Languages

Track: Shows the SFL student’s language level. They are assigned to classes according to the track they belong. There are three tracks, Track 1, 2 and 3.

Module: Can be thought as a semester like in bachelor institute. The difference is, there are 4 modules in a school year. Track 1 and 2 students study for full 4 modules (a year), while Track 3 student’s study for 2 modules (half a year).

Exit Exam: An exam taken at the end of the year or half of the year, and determines if a SFL student will pass the prep school and become a freshman at the institute. SFL academic grade average and exit exam grades are calculated to find if a student will pass. Currently, SFL students must have a grade of minimum 70 points of out 100 points in order to pass.

EF: Entity Framework (ORM for .NET)

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

## **Overview**

The rest of our RAD documentation contains Current System section, Proposed System section, Overview of SFL Online section, Functional Requirements section (*includes high-level functionality of the system)*, Nonfunctional Requirements section *(includes usability, reliability, performance, supportability, implementation, interface, operational, packaging and legal requirements),* System Models section, Object Model section, Dynamic Model section and Glossary.

In System Models section, we described scenarios and models of SFL Online. These models include use case models, object model, dynamic model and user interface view (mockup). Each of these models helps us better understand and analyze the system in different ways. Scenarios tell us the functional requirements in detail.

# Current System

While there is no existing system used by SFL, Işık University has two online systems that the students and instructors in the faculty use, Campus Online [3] and Course Online [4]. We are accustomed to both of these systems, and some of the functions in these systems will also be in SFL Online. We have observed and analyzed both of these systems for reference.

SFL Online is more alike Course Online system used in the institute, as idea since the instructors and students will mainly use it to manage their courses, and they will share many functionalities, so it is the main reference point taken as a system. Campus Online shares only some functionalities with SFL Online, such as the student being able to view their weekly schedule and previous module grade averages. Since both of them were developed a long time ago and uses previous technologies, only their functionalities are taken as reference and not the UI and performance. We know the weak points of these systems, and will strive for a user-friendly, high performance and secure system, using newer technologies.

Since there are differences in how the main faculty and the SFL works in the University, such as semester (module in SFL), students belonging in one class and course mechanisms, SFL Online needs to have its own unique implementation, apart from Course Online.

We have also analyzed some similar automation systems of other universities to take as reference. The university automation systems we have observed include Kültür University Cats System [5] and Marmara University Information Management System [6].

# Proposed System

SFL Online proposes to add useful functionalities and give easier accesses for the students and instructors of SFL program with a new web-based system that was lacking before. This system has to be implemented separately from the faculty system (course online), because of the unique academic mechanism of SFL, which is quite different than how things work in the faculty. Therefore, our system will be specialized to be compatible with how SFL works, and fulfill their needs.

These functions the system provides will be described in detail in Functional Requirements section. We aim for fast response times, reliability, user-friendly interfaces, efficiency and the users accomplishing their tasks in less clicks.

Since the system will be developed with new technologies, the interfaces will also be more appealing and interesting to the users.

## **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**

### Scenarios

A scenario is an instance of a use case. These scenarios will describe the use case instances in detail, and involves a situation where an actor takes a given functionality or path to resolve the scenario.

Obtained actor instances are:

I. Tolga: Student

II. Ekin: Student

III. Asım: Instructor

IV. King of SFL: Admin

**Scenario 1:**

|  |
| --- |
| *Scenario name:* Login |
| *Participant actor instances:* Ekin:Student, Tolga: Student, Asım: Instructor, King of SFL : Admin |
| *Flow of events:*  1. Tolga has newly enrolled into University and was placed in the SFL program, Track 3 after the placement test. The admin, King of SFL, has defined him as student in the SFL Online System. His login credentials are sent to his ışık mail. He wants to login.  2. He opens the SFL Online website from his computer browser. He sees the login form in the initial page.  3. The login form asks for his e-mail and password. He fills in the form with his login credentials to authenticate. Then, he clicks the ‘sign in’ button.  4. Tolga gets an “Invalid e-mail or password” notification as feedback. He realizes he didn’t pay attention to the letter cases while entering his password and re-enters his credentials and clicks ‘sign in’ button again.  5. If there is no problem, he is authenticated and redirected to student home page. |

**Scenario 2:**

|  |
| --- |
| *Scenario name:* View Course Grades and Attendances |
| *Participant actor instances:* Ekin:Student |
| *Flow of events:*  1. Ekin wants to see if the midterm results for her Grammar course has been entered. She is also curious about her total attendance situation. She firstly logs in using her credentials and clicks ‘sign in’ button. She is authenticated & redirected to student home page.  2. Student home page has a sidebar menu for navigation. From this menu, she clicks the “My Courses” dropdown button. This dropdown shows 3 buttons with the names of the courses Ekin is taking, which are “Reading & Writing”, “Core” and “Grammar”. From there, she clicks the “Grammar” button.  3. Ekin is redirected to her course page for Grammar. She clicks “Grades” tab from the page. She sees a list of her current grades, and observes that the midterm grade for the course is entered. She is happy with her grade.  4. Then, she clicks the “attendances” tab. She learns that she a full attendance with only one week missing for this course due to the time that she was ill.  5. Since she has acquired all the information she needed, she clicks ‘Logout’ and exits the system. |

**Scenario 3:**

|  |
| --- |
| *Scenario name:* View Weekly Schedule |
| *Participant actor instances:* Tolga:Student |
| *Flow of events:*  1. Tolga forgot which courses he had on Monday and wants to check it out. He firstly logs in using his credentials and clicks ‘sign in’ button. He is authenticated & redirected to student home page.  2. He clicks the “Weekly Schedule” button from the sidebar menu. His weekly schedule is shown in a table per day and time.  3. He sees that he has Core class at 9 am to 11 am and Grammar class at 1 pm to 3 pm on Monday. |

**Scenario 4:**

|  |
| --- |
| *Scenario name:* Calculate Exit Exam Passing Condition |
| *Participant actor instances:* Ekin:Student |
| *Flow of events:*   1. Ekin knows she has a total grade average of 68 points. She wants to know how many points she has to get from the exit exam to pass and become a freshman. She firstly logs in using her credentials and clicks ‘sign in’ button. She is authenticated & redirected to student home page. 2. She clicks the “Prep Passing Grade Calculator” button from the sidebar menu. 3. There is a calculator in the page she’s redirected with two number input fields asking for her academic average and exit exam grade. She enters her academic average which is 68 points, and assumes she’ll get 70 passing points from the exit exam. She clicks the “submit” button. 4. The calculator shows her the passing grade to be 68, and her passing situation as “Prep”, meaning she has failed. 5. She tries again with 75 points as her exit exam and presses “submit” again. This time, her passing grade is calculated as 71 and her condition is shown as “Fresh”, meaning she will pass if she will get 75 points from the exit exam. |

**Scenario 5:**

|  |
| --- |
| *Scenario name:* View Grade Average |
| *Participant actor instances:* Tolga:Student |
| *Flow of events:*   1. Tolga is a Track 3 student. Being a Track 3 student, he is to study for 2 modules. SFL is currently in the end of module 2 with grades entered and classes ended, with students waiting for the exit exam. He has finished module 1 before. He wants to calculate his exit exam passing condition, but doesn’t know his grade average. 2. He firstly logs in using his credentials and clicks ‘sign in’ button. He is authenticated & redirected to student home page. 3. He clicks the “Academic Status” button from the sidebar menu. 4. From here, he sees his past grade average for module 1, and grades for module 2. He also sees that his total academic average grade is 82, when two modules average grade is taken. 5. Since he has learned his total academic average, he can now calculate what grade he has to take to pass from prep school, using prep passing grade calculator. |

**Scenario 6:**

|  |
| --- |
| *Scenario name:* View Announcements |
| *Participant actor instances:* Ekin:Student, Asım:Instructor |
| *Flow of events:*   1. Ekin’s Reading & Writing teacher, Asım, has said in the last lecture that he will announce the midterm dates soon. She wants to know if he has announced them in SFL Online. 2. She firstly logs in using his credentials and clicks ‘sign in’ button. She is authenticated & redirected to student home page. 3. She clicks the “Announcements” button from the sidebar menu. 4. She observes that there is a new announcement from her instructor, Asım, which says that Reading & Writing midterm will be on Friday next week at class hours. |

**Scenario 7:**

|  |
| --- |
| *Scenario name:* Add Weekly Attendance for course |
| *Participant actor instances:* Asım:Instructor |
| *Flow of events:*   1. Asım is an instructor of SFL, who is also defined as an instructor in SFL Online system. He gives Reading & Writing course in classroom 1A, whom he gave a lecture today. He wants to enter the class attendance. He logs in to the system using his credentials and clicks ‘sign in’ button. He is authenticated & redirect to instructor home page. 2. He clicks the “My classes and Courses” button. 3. He is redirected to the page, where he sees all the classes and courses he is currently teaching. He finds 1A class from the table, and clicks in “Reading & Writing” course as a clickable button. From there he clicks “Enter attendances” button. 4. He is redirected to a page with a list of all the students in that class and course. He enters their attendances for the week in inputs fields of the list. |

**Scenario 8:**

|  |
| --- |
| *Scenario name:* Change Current Password |
| *Participant actor instances:* Asım:Instructor |
| *Flow of events:*   1. Asım is currently logged in the system, and at instructor home page. He wants to change his current password. 2. He clicks “Account” button. From here, he has an option to change password. He clicks “change password”. 3. A form appears where it asks for his current password once and his new password twice. 4. He submits the form and redirected back to his account page. A feedback message informs him that his password has been successfully changed. He has to use his new password the next time he will login to the system. |

**Scenario 9:**

|  |
| --- |
| *Scenario name:* Add Grade |
| *Participant actor instances:* Asım:Instructor |
| *Flow of events:*   1. Asım is the reading & writing teacher of the class 2A. He is currently logged in to the system. He wants to enter the homework grades. 2. He clicks “My classes and Courses” button from sidebar. 3. He sees the list of all the courses he is teaching in the system. He selects Reading & Writing course and sees his students list. 4. He clicks on “Grade” button. Afterwards, page shows that he can add the grades of student or set. He clicks on the save. 5. The system may give an error message if an incorrect note has been entered. 6. He is redirected back to the course page, and sees the grades he has entered. |

**Scenario 10:**

|  |
| --- |
| *Scenario name:* Change current active module |
| *Participant actor instances:* King of SFL: Admin |
| *Flow of events:*   1. Module 1 has ended in SFL, and module 2 will start. Admin should set the active module accordingly in SFL Online. He firstly logs in using his credentials and clicks ‘sign in’ button. He is authenticated & redirected to admin panel. 2. From admin panel sidebar menu, he clicks on “Modules” button, and is redirected to modules page. 3. He sees the list of all modules in a table, with a checkbox on the side indicating that module is active. Currently, module 1 is active. 4. He clicks on “Set Active” button near Module 2 column of the table. He clicks on the button to set it as active. 5. The system gives a warning message as modal, saying setting this module as active will set the other active module as inactive. He clicks “Accept” on modal. 6. He is redirected back to the module page, and sees that Module 2 is active now. |

**Scenario 11:**

|  |
| --- |
| *Scenario name:* Add New Students |
| *Participant actor instances:* King of SFL: Admin |
| *Flow of events:*   1. A new school year has started and placement exam results taken by the new students has been announced. It’s time for our good ol’ admin, King of SFL, to define them in SFL Online system. 2. King of SFL logs in using his credentials and clicks ‘sign in’ button. He is authenticated & redirected to admin home page. 3. He clicks the “Manage Users” dropdown button, and from there, he clicks on “Manage Students” button. 4. He registers the new students to the system, entering their personal and account information. 5. He sees all the students he has registered to the system in a table list of students. |

**Scenario 12:**

|  |
| --- |
| *Scenario name:* Remove Students |
| *Participant actor instances:* King of SFL: Admin |
| *Flow of events:*   1. A school year, and module 4, has ended and exit exam results taken by the students of SFL has been announced. King of SFL should delete the students who has successfully passed the prep school and became a freshman in faculty. 2. King of SFL logs in using his credentials and clicks ‘sign in’ button. He is authenticated & redirected to admin home page. 3. He clicks the “Manage Users” dropdown button, and from there, he clicks on “Manage Students” button. 4. He removes the graduated students from the system, pressing “Delete” button from the table, near the graduated student’s row. 5. When he’s done, he observes the table list of students, and sees the ones he has deleted are no longer registered to the system. |

**Scenario 13:**

|  |
| --- |
| *Scenario name:* Add New Instructor |
| *Participant actor instances:* King of SFL: Admin |
| *Flow of events:*   1. A new instructor has started teaching in SFL. King of SFL should define this instructor in the system. 2. King of SFL logs in using his credentials and clicks ‘sign in’ button. He is authenticated & redirected to admin home page. 3. He clicks the “Manage Users” dropdown button, and from there, he clicks on “Manage Instructors” button. 4. He registers the new instructor to the system, entering his/her personal and account information. 5. He sees the instructor he has registered to the system has been added in the table list of instructors. |

**Scenario 14:**

|  |
| --- |
| *Scenario name:* Edit Instructor |
| *Participant actor instances:* King of SFL: Admin |
| *Flow of events:*   1. An instructor in SFL has been recently married and her surname was changed. King of SFL should reflect this change on the system. 2. King of SFL logs in using his credentials and clicks ‘sign in’ button. He is authenticated & redirected to admin home page. 3. He clicks the “Manage Users” dropdown button, and from there, he clicks on “Manage Instructors” button. 4. He finds this instructor from the Instructors table he was redirected to. He clicks on “Edit” button at this instructor’s row. 5. He sees all the information recorded about this instructor. He changes the instructors surname by writing the new surname on “Last Name” input, and clicks “Save”. 6. He is redirected back to the Instructor table, and observes that the change he has made is saved. |

**Scenario 15:**

|  |
| --- |
| *Scenario name:* Enroll Students in Classes |
| *Participant actor instances:* King of SFL: Admin |
| *Flow of events:*   1. A new school year has started, and admin has defined all the new students registered in SFL Online. Now, he must enroll these students in classes. 2. King of SFL logs in using his credentials and clicks ‘sign in’ button. He is authenticated & redirected to admin home page. 3. He clicks the “Enrollment” dropdown button, and from there, he clicks on “Enroll Students” button. 4. He clicks the “Enroll Students” button. 5. All the students in the student table are placed in classes according to their first exit exam grades. |

**Scenario 16:**

|  |
| --- |
| *Scenario name:* Enroll Instructor in Course |
| *Participant actor instances:* King of SFL: Admin |
| *Flow of events:*   1. This year, Asım is to teach Reading course in class 1A. King of SFL should enroll this instructor as he is teaching this course, in this class. 2. King of SFL logs in using his credentials and clicks ‘sign in’ button. He is authenticated & redirected to admin home page. 3. He clicks the “Enroll” dropdown button from the sidebar, and from there, he clicks on “Enroll Instructors” button. 4. He clicks the “Create New” button. From here, he is redirected to a page with a selection of existing class name, course name, and instructor id, all in dropdown lists. 5. He selects Asım’s instructor id, Class 1A and Reading course. He clicks the “Save” button. 6. If there are no database errors, Asım is now enrolled as the teacher of Class 1A’s Reading course. Admin can observe this enrollment details from the Enrollment Instructor table. If there are database errors such as foreign key constraints, Asım is not enrolled when “Save” button is clicked, instead an error message or page describing the error is shown. |

**Scenario 17:**

|  |
| --- |
| *Scenario name:* Add Course |
| *Participant actor instances:* King of SFL: Admin |
| *Flow of events:*   1. A new course called “Core” is added to the curriculum this year. King of SFL should define this course in the system. 2. King of SFL logs in using his credentials and clicks ‘sign in’ button. He is authenticated & redirected to admin home page. 3. He clicks the “Manage Courses” button from the sidebar and is redirected to the course management page. 4. He sees a table of all the courses registered in the system. He clicks the “Create New” button. 5. He enters the course name as “Core” add saves this new course to the system. |

**Scenario 18:**

|  |
| --- |
| *Scenario name:* Edit Course |
| *Participant actor instances:* King of SFL: Admin |
| *Flow of events:*   1. SFL has decided to rename the course name “Reading” to “Reading Practice”. King of SFL should change this courses name. 2. King of SFL logs in using his credentials and clicks ‘sign in’ button. He is authenticated & redirected to admin home page. 3. He clicks the “Manage Courses” button from the sidebar and is redirected to the class management page. 4. He sees a table of all the classes registered in the system. He sees the “Reading” course from this table and clicks “Edit” on the row of this course. 5. He changes the name of this course as “Reading Practice” and clicks save. |

**Scenario 19:**

|  |
| --- |
| *Scenario name:* Add Class |
| *Participant actor instances:* King of SFL: Admin |
| *Flow of events:*   1. SFL has opened a new classroom for Track 2, because all the other classroom’s quotas are full. King of SFL should define this new class in the system. 2. King of SFL logs in using his credentials and clicks ‘sign in’ button. He is authenticated & redirected to admin home page. 3. He clicks the “Manage Classes” button from the sidebar and is redirected to the class management page. 4. He sees a table of all the classes registered in the system. He clicks the “Create New” button. 5. He enters the class name as “2C” and saves this new class to the system. |

**Scenario 20:**

|  |
| --- |
| *Scenario name:* Change Course Grade Percentages |
| *Participant actor instances:* King of SFL: Admin |
| *Flow of events:*   1. The grade percentage for course “Writing” s midterm has been decided to be increased from 20% to 25%. Admin should reflect this change in the system. 2. King of SFL logs in using his credentials and clicks ‘sign in’ button. He is authenticated & redirected to admin home page. 3. He clicks the “Manage Grades” button from the sidebar and is redirected to the grade management page. 4. He sees a table of all the grade percentages for their respective courses registered in the system. He sees the “Writing” course from this table and clicks “Edit” on the row of this course. 5. He changes the grade percentage of this course from 20% to 25% and clicks save. |

**Scenario 21:**

|  |
| --- |
| *Scenario name:* Forgot Password |
| *Participant actor instances:* Ekin: Student |
| *Flow of events:*   1. Ekin opens the SFL Online website from her computer browser and wants to login to the system, but realizes she forgot her password. 2. She tries to remember her password by entering her credentials and passwords she usually uses, but gets “Invalid Username or Password!” message every time. 3. She sees the “Forgot Password?” option on the login page form, and clicks on it. 4. Another form appears, asking for her ışık mail address, saying her new password will be sent to her mail. 5. She enters her ışık mail and clicks “submit”. A feedback message says her password is now reset and sent to her ışık mail. 6. She logs in to her ışık mail and sees that she has a new mail from SFL Online, with her new password set as “\*\*\*\*\*\*”, and she can login with this password, and can change her password after logging in. 7. She goes back to the SFL Online login page, and successfully logs in with the password sent to her e-mail. |

**Scenario 22:**

|  |
| --- |
| *Scenario name:* Enroll Student in Class |
| *Participant actor instances:* King of SFL: Admin |
| *Flow of events:*   1. Admin has enrolled all the students in bulk in scenario 15. Later, he added a student that was registered in the SFL program due to a special condition. The students exit grade is 30, which means this student will belong in track 1. 2. King of SFL logs in using his credentials and clicks ‘sign in’ button. He is authenticated & redirected to admin home page. 3. He clicks the “Enrollment” dropdown button, and from there, he clicks on “Enroll Students” button. 4. He clicks the “Create New Enrollment” button. Class 1A has a quota of 25 students. He selects student id from dropdown list, class 1A from dropdown list, and tries to enroll this student to class 1A. When he tries to add this student, which is the 26th student to class 1A, he gets and error message saying “Class is already full!” 5. He enrolls this student in Class 1B instead, picking class 1B. 6. He is redirected back to the student enrollment table, and observes that the student is successfully enrolled in class 1B. 7. He clicks ‘Create New Enrollment’ button again to see if there are any students that are not enrolled in classes. When no student is left, there is no element left in the dropdown menu to select student. |

**Scenario 23:**

|  |
| --- |
| *Scenario name:* Change Exit Exam Passing Conditions |
| *Participant actor instances:* King of SFL: Admin |
| *Flow of events:*   1. The first exit exams placement and passing conditions has changed. Now students receiving a grade of 0-30 belong to Track 1, 31-55 to Track 2, 56-70 to Track 3, and students that take a point above 70 are passed to freshman. Admin should reflect these changes in SFL Online. 2. King of SFL logs in using his credentials and clicks ‘sign in’ button. He is authenticated & redirected to admin home page. 3. He clicks the ‘Exit Exam’ dropdown button, and from there, he clicks ‘Passing Conditions’ button. 4. From the Passing Conditions table, he finds all the rows for Exit 1 and clicks ‘Edit’. 5. He edits the passing conditions for Exit 1 for each Track1-2-3 and clicks save. |

**Scenario 24:**

|  |
| --- |
| *Scenario name:* Enter Student Exit Exam Grades |
| *Participant actor instances:* King of SFL: Admin |
| *Flow of events:*   1. The first exit exam results has been announced. Admin should enter the student’s exit grades in order to enroll them in classes accordingly. 2. King of SFL logs in using his credentials and clicks ‘sign in’ button. He is authenticated & redirected to admin home page. 3. He clicks the ‘Exit Exam’ dropdown button, and from there, he clicks ‘Exit Grades’ button. 4. From the Exit Exam table, he clicks ‘Create New’. 5. From here, he is redirected to a form to save a student’s exit exam grade. He selects ‘Exit 1’ as exit exam, enters the student ID and the exit exam grade of a student. Then, he clicks save. 6. He repeats the same process for each student that has taken this exit exam. |

**Use Case Scenarios**

**Use Case 1:**

|  |
| --- |
| *Use case name:* Login |
| *Participant actor(s):* Initiated by Student, Instructor, OR Admin |
| *Flow of events:*     1. The user first enters to SFL Online system via web browser. 2. SFL Online presents the “Login Form” to the user. 3. The user enters his/her username into username text field on the screen, also enters his/her password into password text field on the screen. Lastly, the user sends a request to SFL Online by pressing “Login” button on the screen to be logged in. 4. Browser sends request to reach to server. SFL Online checks the username and password from the “Person” table in the database, so that SFL Online allows the user to login. Then it determines role of the user by retrieving the role data from the “Person” table, and redirects to proper screen. |
| *Entry Condition(s):*   * The user enters to login screen. |
| *Exit Condition(s):*   * The user is logged in to the SFL Online system successfully, OR * The user has received an explanation indicating why he/she could not login. |
| *Exceptional Case(s):*   * The form is sensitive for invalid input. * The server might be down. |

**Use Case 2:**

|  |
| --- |
| *Use case name:* Reset Password |
| *Participant actor(s):* Initiated by Student, Instructor, OR Admin |
| *Flow of events:*     1. The user first enters to SFL Online system via web browser. 2. SFL Online presents the “Login Form” to the user. 3. The user clicks on the “Forgot Password?” link-button of the form. 4. A new form appears, asking for the user’s e-mail registered to the system. The user enters their e-mail and clicks “submit”. 5. Browser sends request to reach to server. SFL Online checks if the e-mail inputted by the user exits in the database, in the “Person” table. If it is found, they get a feedback message saying their password has been reset, and sent to their e-mail address. If it is not found in the database, they get a feedback message saying “No such user found.” 6. The user checks their e-mail, and a message from SFL Online should have arrived with their new password. 7. The user should be able to login as stated in Login use case with this new password. |
| *Entry Condition(s):*   * The user enters to login screen. * The user does not remember or know their password to authenticate. |
| *Exit Condition(s):*   * The user has received an e-mail including their new password, and successfully logged in to the system with this password. * The user has received an explanation saying “No such user found!” if the e-mail inputted is not registered in the system. |
| *Exceptional Case(s):*   * The form is sensitive for invalid input. * The server might be down. |

**Use Case 3:**

|  |
| --- |
| *Use case name:* View Weekly Course Schedule |
| *Participant actor(s):* Student |
| *Pre-condition:* Student must be logged in to the SFL Online system. |
| *Flow of events:*   1. The logged-in student clicks the “Weekly Schedule” texted button from the sidebar navigation bar at student home page. 2. Student is redirected to the schedule page specialized for the class he/she belongs. The schedule is shown in a table, and includes days, time slots and lectures in appropriate positions. The data comes from model class that is StudentSchedule. This model includes ID of class the student belongs (reason for taking class ID instead of student ID is, since in SFL, every student in the same class has the same schedule), lectures through time slots. All objects are received and they are sent in to relevant view. |
| *Entry Condition(s):*   * Student actor must be logged in to the system. |
| *Exit Condition(s):*   * Student actor can display their weekly course schedule successfully. |
| *Exceptional Case(s):*   * The server might be down. |

**Use Case 4:**

|  |
| --- |
| *Use case name:* View Weekly Course Schedule |
| *Participant actor(s):* Instructor |
| *Pre-condition:* Instructor must be logged in to the SFL Online system. |
| *Flow of events:*   1. The logged-in instructor clicks the “Weekly Schedule” texted button from the sidebar navigation bar at instructor home page. 2. Instructor is redirected to the schedule page specialized for the classes he/she is teaching. The schedule is shown in a table, and includes days, time slots and lectures in appropriate positions. The data comes from model class that is InstructorSchedule. This model includes ID of the instructor and lectures through time slots. All objects are received and they are sent in to relevant view. |
| *Entry Condition(s):*   * Instructor actor must be logged in to the system. |
| *Exit Condition(s):*   * Instructor actor can display their weekly course schedule successfully. |
| *Exceptional Case(s):*   * The server might be down. |

**Use Case 5:**

|  |
| --- |
| *Use case name:* View Attendance |
| *Participant actor(s):* Student |
| *Pre-condition:* Student must be logged in to the SFL Online system. |
| *Flow of events:*   1. The logged-in student clicks the “My Courses” texted dropdown button from the sidebar navigation bar at student home page. From this button, a number of buttons are shown, with the names of the courses the student is taking. Student clicks on the corresponding course names button he/she wishes to display attendance for. 2. The student is redirected to a page, where two options are visible in tabs; grades & attendances. 3. Student clicks to “Attendances” tab from this page. 4. Student sees his/her attendance information for the course. |
| *Entry Condition(s):*   * Student actor must be logged in to the system. |
| *Exit Condition(s):*   * Student actor can display their attendance information for a course successfully. |
| *Exceptional Case(s):*   * The server might be down. |

**Use Case 6:**

|  |
| --- |
| *Use case name:* View Grades |
| *Participant actor(s):* Student |
| *Pre-condition:* Student must be logged in to the SFL Online system. |
| *Flow of events:*   1. The logged-in student clicks the “My Courses” texted dropdown button from the sidebar navigation bar at student home page. From this button, a number of buttons are shown, with the names of the courses the student is taking. Student clicks on the corresponding course names button he/she wishes to display grades for. 2. The student is redirected to a page, where two options are visible in tabs; grades & attendances. 3. Student clicks on “Grades” tab from this page. 4. Student sees his/her grade information entered, for each grading (homework, midterm etc.) for the course. |
| *Entry Condition(s):*   * Student actor must be logged in to the system. |
| *Exit Condition(s):*   * Student actor can display their grade information for a course successfully. |
| *Exceptional Case(s):*   * The server might be down. |

**Use Case 7:**

|  |
| --- |
| *Use case name:* Calculate Exit Exam Passing Condition |
| *Participant actor(s):* Student |
| *Pre-condition:* Student must be logged in to the SFL Online system. |
| *Flow of events:*   1. The logged-in student clicks the “Prep Passing Grade Calculator” texted button from the sidebar navigation bar at student home page. From this button, the student is redirected to a form-type calculator with two writable textboxes, one for academic average and other for exit exam grade. Below these, there is another read-only textbox, with the label “Prep Passing Grade”. 2. The student enters their academic average grade, and the expected exit exam grade they may get. Then, they press “submit”. 3. According to the calculated prep passing grade from inputs, the calculator responses with a colored text below the form, “Prep” in orange, or “Fresh” in green. “Fresh” means the student will pass to the faculty with the grades he/she has entered, and “Prep” means he/she will fail to pass. |
| *Entry Condition(s):*   * Student actor must be logged in to the system. |
| *Exit Condition(s):*   * Student actor can calculate their passing grade condition successfully. |
| *Exceptional Case(s):*   * The server might be down. * JavaScript might be disabled on the user’s browser. |

**Use Case 8:**

|  |
| --- |
| *Use case name:* Display Academic Status |
| *Participant actor(s):* Student |
| *Pre-condition:* Student must be logged in to the SFL Online system. |
| *Flow of events:*   1. The logged-in student clicks the “Academic Status” texted button from the sidebar navigation bar at student home page. From this button, the student is redirected to a page which shows the grades of each module completed. 2. The student can see their grades and average grades for each module from this page, and see their “Total Academic Average” grade, indicating the average grade of all the modules. |
| *Entry Condition(s):*   * Student actor must be logged in to the system. |
| *Exit Condition(s):*   * Student actor can display their academic grade information for a module, and their academic grade average successfully. |
| *Exceptional Case(s):*   * Student may not yet have completed at least one module. In this case, student does not have an academic average grade to calculate and display. * The server might be down. |

**Use Case 9:**

|  |
| --- |
| *Use case name:* View Announcements |
| *Participant actor(s):* Student |
| *Pre-condition:* Student must be logged in to the SFL Online system. |
| *Flow of events:*   1. The logged-in student can see if there are any new announcements at their main home page. The posting instructor’s name will also be shown. |
| *Entry Condition(s):*   * Student actor must be logged in to the system. |
| *Exit Condition(s):*   * Student actor can display announcements successfully. |
| *Exceptional Case(s):*   * The server might be down. |

**Use Case 10:**

|  |
| --- |
| *Use case name:* View Announcements |
| *Participant actor(s):* Student |
| *Pre-condition:* Student must be logged in to the SFL Online system. |
| *Flow of events:*   1. The logged-in student can see if there are any new announcements at their main home page. The posting instructor’s name will also be shown. |
| *Entry Condition(s):*   * Student actor must be logged in to the system. |
| *Exit Condition(s):*   * Student actor can display announcements successfully. |
| *Exceptional Case(s):*   * The server might be down. |

**Use Case 11:**

|  |
| --- |
| *Use case name:* Write Announcements |
| *Participant actor(s):* Instructor |
| *Pre-condition:* Instructor must be logged in to the SFL Online system. |
| *Flow of events:*   1. The logged-in instructor clicks on “My Classes & Courses” dropdown button from sidebar. Buttons labeled with the courses he teaches appear. He chooses a course, and clicks the button for it. 2. From the page redirected, instructor clicks on “Write Announcement” button. 3. A textbox appears for the announcement instructor wants to write. 4. The instructor can write the announcement in this textbox and click “send”. 5. After sending, this announcement will be visible for all the students in that class and course. |
| *Entry Condition(s):*   * Instructor actor must be logged in to the system. |
| *Exit Condition(s):*   * Instructor actor can write announcements. |
| *Exceptional Case(s):*   * The server might be down. |

**Use Case 12:**

|  |
| --- |
| *Use case name:* Enter Course Attendances |
| *Participant actor(s):* Instructor |
| *Pre-condition:* Instructor must be logged in to the SFL Online system. |
| *Flow of events:*   1. The logged-in instructor clicks on “My Classes & Courses” dropdown button from sidebar. Buttons labeled with the courses he teaches appear. He chooses a course he will enter attendances on, and clicks the button for it. 2. From the page redirected, instructor clicks on “Attendances” button. 3. A list of every student in the class, taking this course from the instructor appears. 4. Instructor selects the week of the attendance taken, and enters the attendance as “Absent” or “Present” for each student. 5. Instructor clicks the “Save” button. He/she views this information saved from the class list. If wrong attendance information has been entered, the instructor can edit attendances at any time. |
| *Entry Condition(s):*   * Instructor actor must be logged in to the system. |
| *Exit Condition(s):*   * Instructor actor can successfully enter course attendances. |
| *Exceptional Case(s):*   * The server might be down. |

**Use Case 13:**

|  |
| --- |
| *Use case name:* Enter Course Grades |
| *Participant actor(s):* Instructor |
| *Pre-condition:* Instructor must be logged in to the SFL Online system. |
| *Flow of events:*   1. The logged-in instructor clicks on “My Classes & Courses” dropdown button from sidebar. Buttons labeled with the courses he teaches appear. He chooses a course he will enter grades on, and clicks the button for it. 2. From the page redirected, instructor clicks on “Grades” button. 3. A list of every student in the class, taking this course from the instructor appears. 4. Instructor selects the type of grade for the course. This type can be midterm, homework, project, etc. 5. Instructor enters grades for each student in that class, taking this course. 6. After entering grades for all student, Instructor clicks “Save” button. |
| *Entry Condition(s):*   * Instructor actor must be logged in to the system. |
| *Exit Condition(s):*   * Instructor actor has successfully entered grades for a course. |
| *Exceptional Case(s):*   * The server might be down. |

**Use Case 14:**

|  |
| --- |
| *Use case name:* Manage Students (CRUD) |
| *Participant actor(s):* Admin |
| *Pre-condition:* Admin must be logged in to the SFL Online system. |
| *Flow of events:*   1. The logged-in admin clicks on “Manage Users” dropdown button from sidebar. A number of buttons appear at click. Admin clicks the “Manage Students” button. 2. Admin is redirected to the list of all students registered in the system, displayed in a table with their personal information written on rows. 3. CRUD operations for Student table are handled in this page. 4. From this table or control panel, the admin can press “Create New” to add a new student. Bulk additions are considered to be implemented. 5. Admin can edit an existing student’s information. 6. Admin can search for a specific student by filtering (Search bar). 7. Admin can delete students. |
| *Entry Condition(s):*   * Admin actor must be logged in to the system. |
| *Exit Condition(s):*   * Admin actor can successfully perform CRUD operations on Student table. * Admin has received an explanation indicating why he/she could not save the changes made/alterations attempted on database. |
| *Exceptional Case(s):*   * The server might be down. |

**Use Case 15:**

|  |
| --- |
| *Use case name:* Manage Instructors (CRUD) |
| *Participant actor(s):* Admin |
| *Pre-condition:* Admin must be logged in to the SFL Online system. |
| *Flow of events:*   1. The logged-in admin clicks on “Manage Users” dropdown button from sidebar. A number of buttons appear at click. Admin clicks the “Manage Instructors” button. 2. Admin is redirected to the list of all instructors registered in the system, displayed in a table with their personal information written on rows. 3. CRUD operations for Instructor table are handled in this page. 4. From this table or control panel, the admin can press “Create New” to add a new instructor. 5. Admin can edit an existing instructor’s information. 6. Admin can search for a specific instructor by filtering (Search bar). 7. Admin can delete instructors. |
| *Entry Condition(s):*   * Admin actor must be logged in to the system. |
| *Exit Condition(s):*   * Admin actor can successfully perform CRUD operations on Instructor table. * Admin has received an explanation indicating why he/she could not save the changes made/alterations attempted on database. |
| *Exceptional Case(s):*   * The server might be down. |

**Use Case 16:**

|  |
| --- |
| *Use case name:* Enroll Student in Class |
| *Participant actor(s):* Admin |
| *Pre-condition:* Admin must be logged in to the SFL Online system. |
| *Flow of events:*   1. The logged-in admin clicks on “Enrollment” dropdown button from sidebar. A number of buttons appear at click. Admin clicks the “Create New Enrollment” button. 2. Admin is redirected to the list of all students enrolled in classes, displayed in a table. 3. A registered student newly added, and placed in a track level after exit exam should be enrolled in a class. Admin clicks the “Create New” button. 4. From here, a form with two dropdown selectable inputs appear, one for class name and other for student ID. If a student is already enrolled in a class, their ID won’t show up in the dropdown selectable input. 5. A student is selected and enrolled into a chosen class, by clicking “Save”. Student is now enrolled in the selected class, and can be seen in the Student Enrollment table. |
| *Entry Condition(s):*   * Admin actor must be logged in to the system. |
| *Exit Condition(s):*   * Admin actor can successfully enroll a student into a classroom. * In case a class quota is full, admin receives a feedback message. * Otherwise, Admin has received an explanation indicating why he/she could not save the changes made/alterations attempted on database. |
| *Exceptional Case(s):*   * The server might be down. |

**Use Case 17:**

|  |
| --- |
| *Use case name:* Enroll Students in Classes |
| *Participant actor(s):* Admin |
| *Pre-condition:* Admin must be logged in to the SFL Online system. |
| *Flow of events:*   1. The logged-in admin clicks on “Enrollment” dropdown button from sidebar. A number of buttons appear at click. Admin clicks the “Enroll Students” button. 2. A table appears for feedback, showing the exit grades required for a student to be placed in a track. 3. By clicking “Save”, all the students in the students table that have an exit grade and not enrolled in a class before are now enrolled in classes, according to their first exit exam grade, and can be seen in the Student Enrollment table. If a class for a track a student belongs is full in quota, they are placed to another class of the same track level. If all classes in a track are full, the student is not enrolled. |
| *Entry Condition(s):*   * Admin actor must be logged in to the system. * Admin actor should have initialized the exit grades of new students in the system. |
| *Exit Condition(s):*   * Admin actor can successfully enroll a students into classrooms in bulk according to their first exit exam grades. |
| *Exceptional Case(s):*   * The server might be down. |

**Use Case 18:**

|  |
| --- |
| *Use case name:* Enter Exit Grades |
| *Participant actor(s):* Admin |
| *Pre-condition:* Admin must be logged in to the SFL Online system. |
| *Flow of events:*   1. The logged-in admin clicks on “Exit Exam” dropdown button from sidebar. A number of buttons appear at click. Admin clicks the “Exit Grades” button. 2. Admin is redirected to the Exit Exam table. He clicks ‘Create New’ button. 3. From here, he is redirected to a form to save a student’s exit exam grade. For each student, he selects the exit exam name, enters the student ID and the exit exam grade of a student. Then, he clicks save. Students with first exit exam grades initialized can be enrolled in classes by bulk. |
| *Entry Condition(s):*   * Admin actor must be logged in to the system. |
| *Exit Condition(s):*   * Admin actor can successfully save exit grades of students. |
| *Exceptional Case(s):*   * The server might be down. |

**Use Case 19:**

|  |
| --- |
| *Use case name:* Enroll Instructors in Courses |
| *Participant actor(s):* Admin |
| *Pre-condition:* Admin must be logged in to the SFL Online system. |
| *Flow of events:*   1. The logged-in admin clicks on “Enrollment” dropdown button from sidebar. A number of buttons appear at click. Admin clicks the “Enroll Instructors” button. 2. Admin is redirected to the list of all instructors currently teaching a course in a class. 3. Admin clicks the “Create New” button. 4. From here, class name, course title, and instructor ID are selected from dropdown inputs. Admin clicks the “Save” button. 5. Instructor is now enrolled, set to be given a course in a specific class. |
| *Entry Condition(s):*   * Admin actor must be logged in to the system. |
| *Exit Condition(s):*   * Admin actor can successfully enroll an instructor to teach in a classrooms course. * If the class already has the course chosen enrolled, admin receives a feedback message from the system. * In case of other errors, Admin has received an explanation indicating why he/she could not save the changes made/alterations attempted on database. |
| *Exceptional Case(s):*   * The server might be down. |

**Use Case 20:**

|  |
| --- |
| *Use case name:* Set Active Module |
| *Participant actor(s):* Admin |
| *Pre-condition:* Admin must be logged in to the SFL Online system. |
| *Flow of events:*   1. The logged-in admin clicks on “Modules” button from sidebar. 2. Admin is redirected to the list of modules registered in the system, displayed in a table. 3. Modules names are shown in the table. Each has a checkbox at their “active” labeled column, with the currently active module having a tick on its checkbox. Each module also has a “Module Active” and “Module Inactive” button. 4. To change the active module, admin presses “Module Inactive” to disactivate the currently active module. Then, he clicks “Module Active” on the module he/she wants to activate. The active module will have a tick on its checkbox. |
| *Entry Condition(s):*   * Admin actor must be logged in to the system. |
| *Exit Condition(s):*   * Admin actor can successfully change and/or set the active module. * Admin has received an explanation indicating why he/she could not save the changes made/alterations attempted on database. |
| *Exceptional Case(s):*   * The server might be down. * Adding and deleting operations can be performed on module table, but should only be necessary at the first release of the website or rare conditions. |

**Use Case 21:**

|  |
| --- |
| *Use case name:* Manage Grade Percentages (CRUD) |
| *Participant actor(s):* Admin |
| *Pre-condition:* Admin must be logged in to the SFL Online system. |
| *Flow of events:*   1. The logged-in admin clicks on “Manage Grade Percentages” button from sidebar. 2. Admin is redirected to the control panel for Grade Percentages table, whom he/she can perform operations on. A table list of all the grade percentages for a specific course are shown. The course name, Track it belongs, type (homework, midterm, etc.), Percentage it effects and which module it belongs are shown. 3. Clicking “Create New” button functions to add a new percentage. A form opens. Course name, Track and Module are selected from dropdown list. Then, the name of the percentage, midterm for example, and the percentage it’ll effect the course are entered. Sending this by clicking “Save” will save this to the system. 4. Grade percentages can be edited clicking “Edit” button at the row of record. 5. Grade percentages can be deleted clicking “Delete” button at the row of record. |
| *Entry Condition(s):*   * Admin actor must be logged in to the system. |
| *Exit Condition(s):*   * Admin actor can successfully perform CRUD operations on Grade Percentages table. * Admin has received an explanation indicating why he/she could not save the changes made/alterations attempted on database. |
| *Exceptional Case(s):*   * The server might be down. |

**Use Case 22:**

|  |
| --- |
| *Use case name:* Manage Tracks (CRUD) |
| *Participant actor(s):* Admin |
| *Pre-condition:* Admin must be logged in to the SFL Online system.  This use case should only be necessary at the first release of the website or rare conditions. |
| *Flow of events:*   1. The logged-in admin clicks on “Manage Tracks” button from sidebar. 2. Admin is redirected to the control panel for Tracks table in the database. He can see all the tracks registered in the system at the table. 3. Admin can add new tracks, by clicking “Create New”, entering track name as input and clicking “Save”. 4. Admin can edit a tracks name. 5. Admin can delete tracks. |
| *Entry Condition(s):*   * Admin actor must be logged in to the system. |
| *Exit Condition(s):*   * Admin actor can successfully perform CRUD operations on Tracks table. * Admin has received an explanation indicating why he/she could not save the changes made/alterations attempted on database. |
| *Exceptional Case(s):*   * The server might be down. * This use-case itself can be considered as exceptional case. |

**Use Case 23:**

|  |
| --- |
| *Use case name:* Manage Track Courses (CRUD) |
| *Participant actor(s):* Admin |
| *Pre-condition:* Admin must be logged in to the SFL Online system.  This use case should only be necessary at the first release of the website or rare conditions. |
| *Flow of events:*   1. The logged-in admin clicks on “Manage Track Courses” button from sidebar. 2. Admin is redirected to the control panel for Track Courses table in the database. This table shows which courses are included and taught in a track. 3. Admin can add a new course to a track. Admin clicks “Create New”. Admin selects the track and course, then clicks save. 4. Admin can edit a record. 5. Admin can remove a course from a track. (Delete) |
| *Entry Condition(s):*   * Admin actor must be logged in to the system. |
| *Exit Condition(s):*   * Admin actor can successfully perform CRUD operations on Track Courses table. * While adding, if a track already has that course “This Track already have this course” feedback message is received. * In case of other errors, Admin has received an explanation indicating why he/she could not save the changes made/alterations attempted on database. |
| *Exceptional Case(s):*   * The server might be down. * This use-case itself can be considered as exceptional case. |

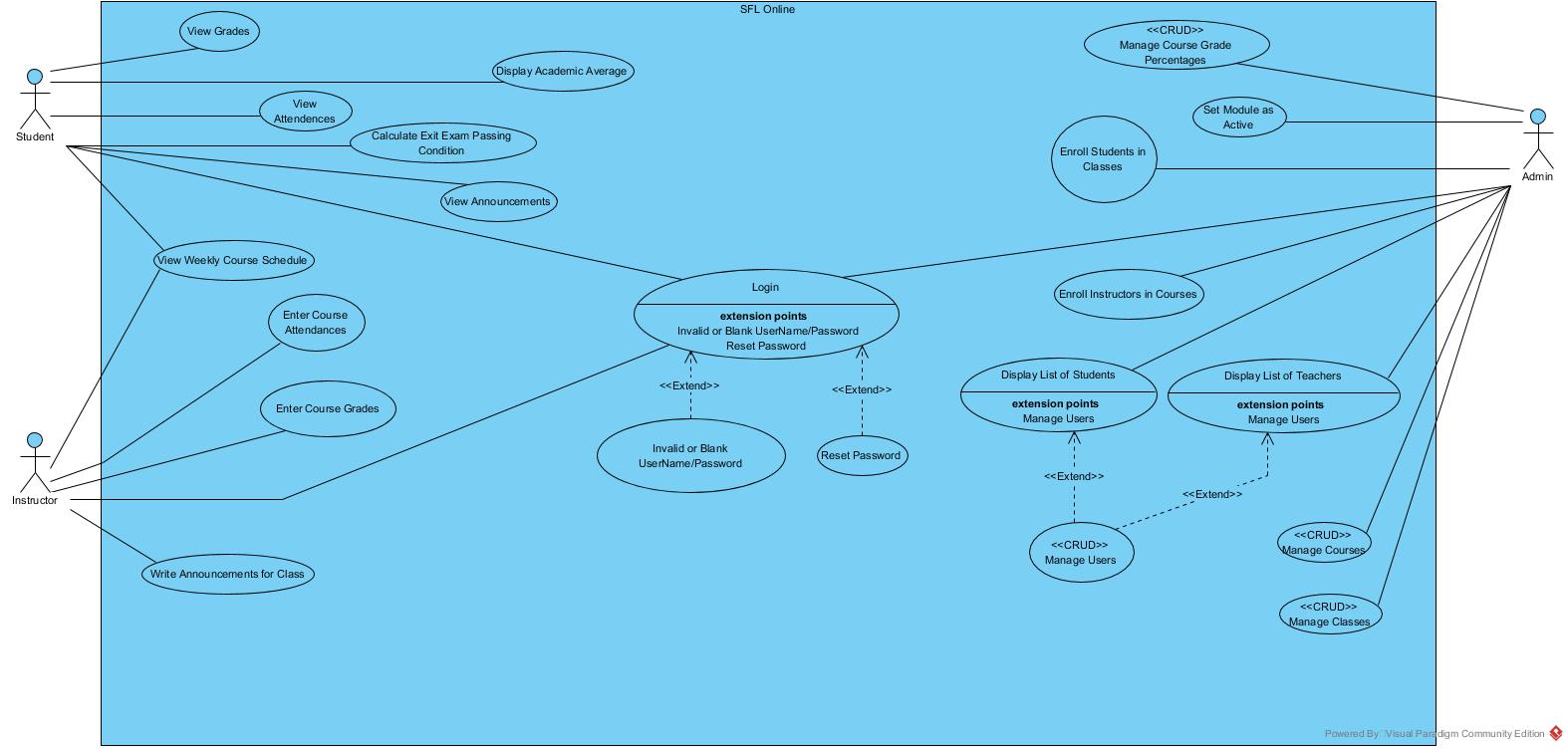
**Use Case 24:**

|  |
| --- |
| *Use case name:* Manage Classes (CRUD) |
| *Participant actor(s):* Admin |
| *Pre-condition:* Admin must be logged in to the SFL Online system.  This use case should only be necessary at the start of the school year or rare conditions |
| *Flow of events:*   1. The logged-in admin clicks on “Manage Classes” button from sidebar. 2. Admin is redirected to the control panel for Classes table in the database. He can see all the classes in SFL, registered in the system at the table. 3. Admin can add new classes. Admin clicks “Create New”, and enters the name of the class, the quota of class, and selects which track it will belong. Admin clicks “Save” to add this new class. 4. Admin can edit class information from the table. 5. Admin can delete class information from the table. |
| *Entry Condition(s):*   * Admin actor must be logged in to the system. |
| *Exit Condition(s):*   * Admin actor can successfully perform CRUD operations on Classes table. * Admin has received an explanation indicating why he/she could not save the changes made/alterations attempted on database. |
| *Exceptional Case(s):*   * The server might be down. |

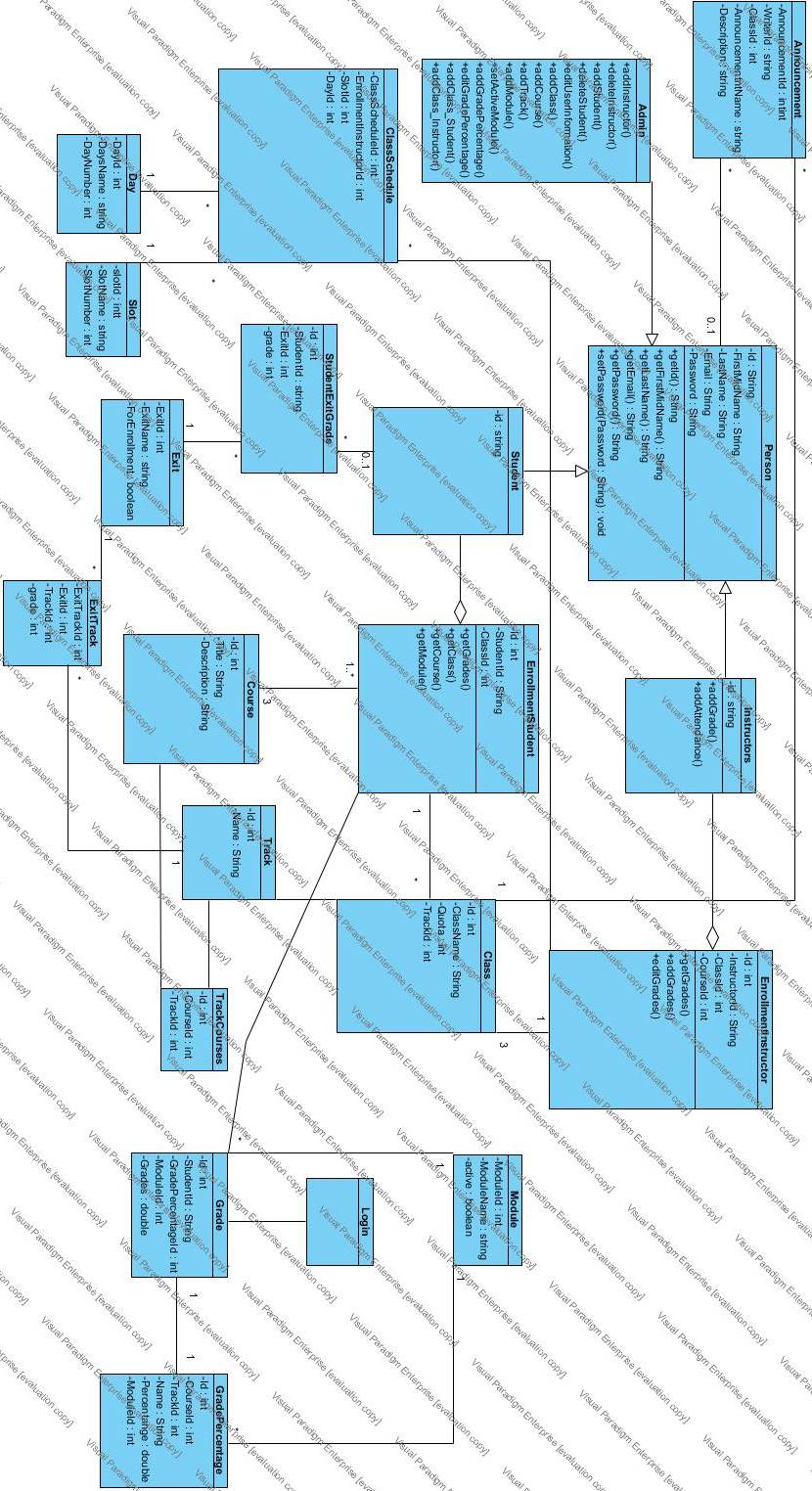
**Use Case 25:**

|  |
| --- |
| *Use case name:* Manage Courses (CRUD) |
| *Participant actor(s):* Admin |
| *Pre-condition:* Admin must be logged in to the SFL Online system. |
| *Flow of events:*   1. The logged-in admin clicks on “Manage Courses” button from sidebar. 2. Admin is redirected to the courses list in a table view. Name and credits of a course are displayed. 3. Admin can add new courses, by clicking “Create New”, entering course name and credit as input and clicking “Save”. 4. Admin can edit a course information. 5. Admin can delete a course. |
| *Entry Condition(s):*   * Admin actor must be logged in to the system. |
| *Exit Condition(s):*   * Admin actor can successfully perform CRUD operations on Courses table. * Admin has received an explanation indicating why he/she could not save the changes made/alterations attempted on database. |
| *Exceptional Case(s):*   * The server might be down. |

### Use case model

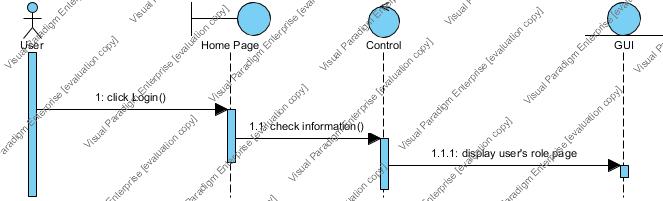


### Object model

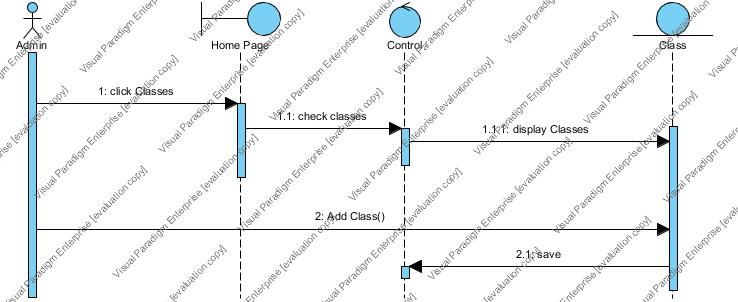


### Dynamic model

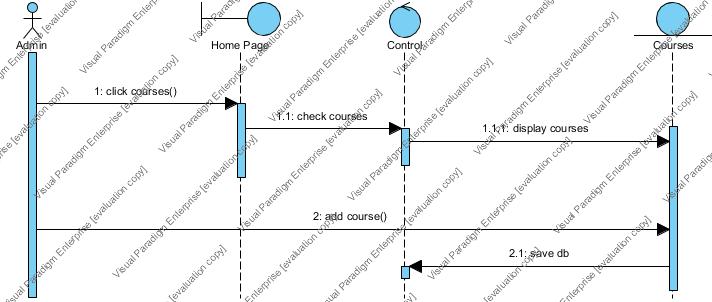
Login



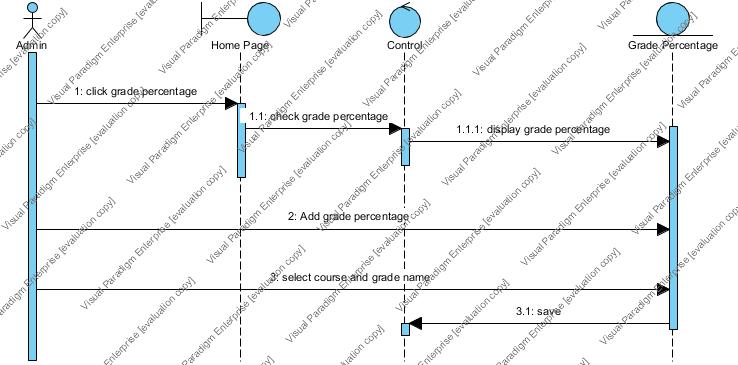
Admin Add Class



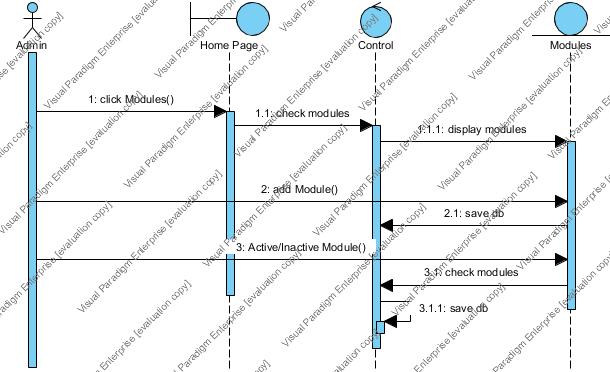
Admin Add Course



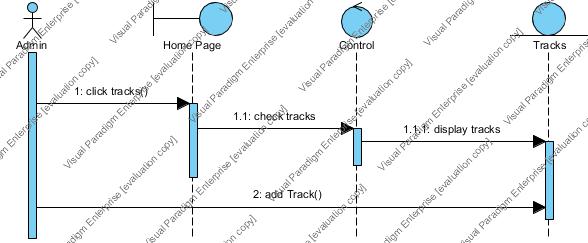
Admin Add Grade



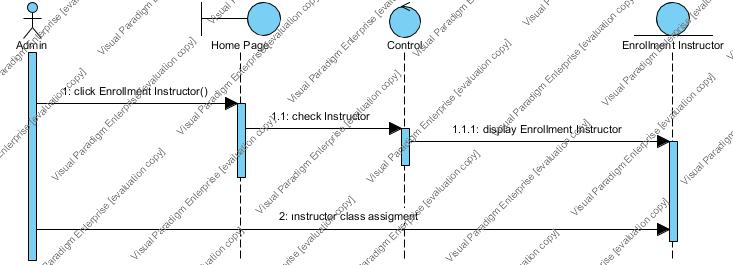
Admin Add Module



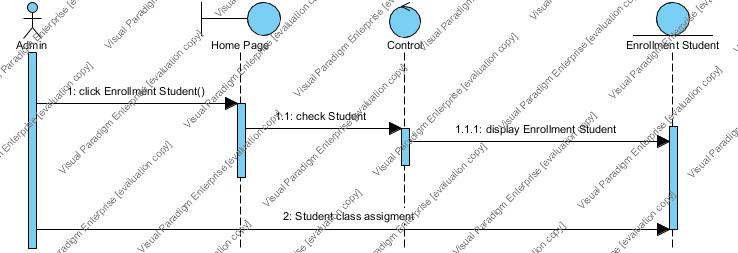
Admin Add Track



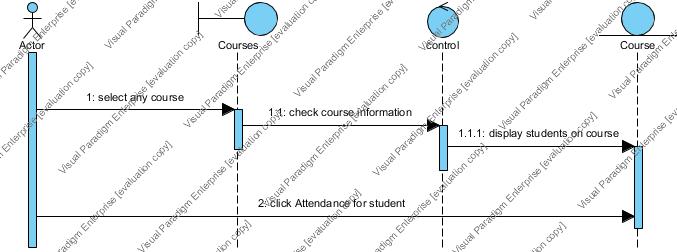
Admin Enrollment Instructor



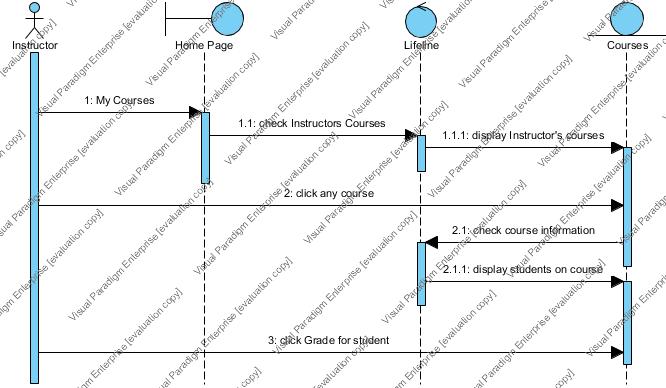
Admin Enrollment Student



Instructor Attendance



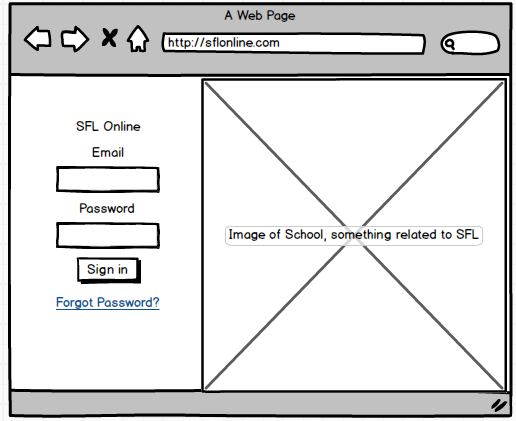
Instructor Add Grades



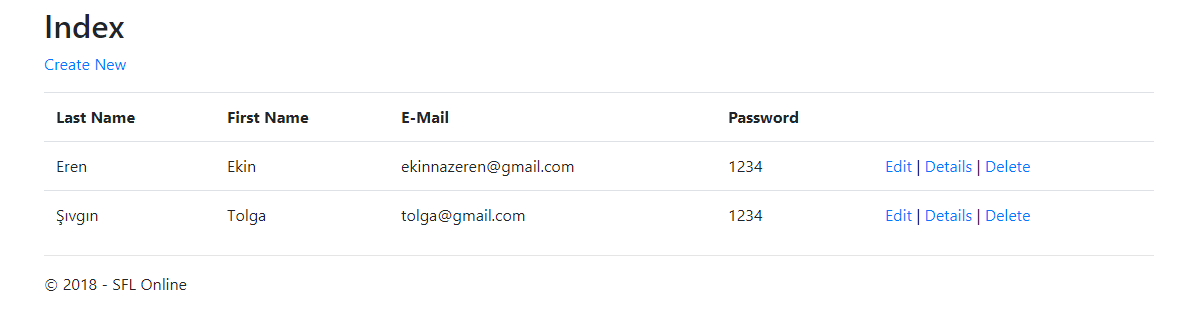
### User interface—Navigational paths and screen mock-ups

Includes mock-ups, prototypes and visual representations and gives an idea of how our system will look like for the users.

1. *Login Page*

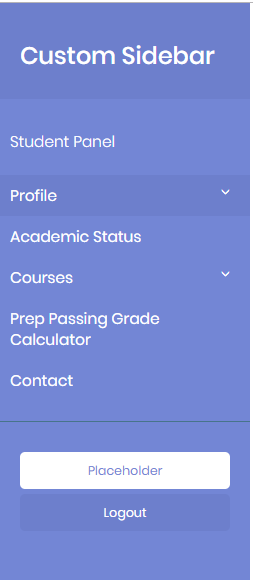
**

1. *Manage Student (CRUD) Table – Admin*

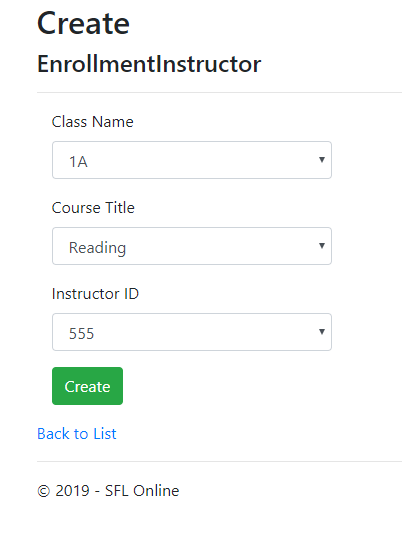
**

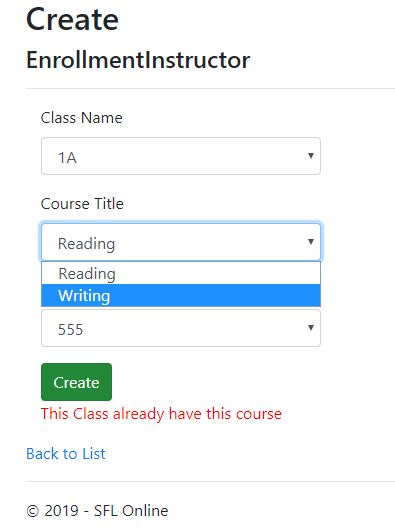
1. *Navigational Sidebar – Student*

Sidebar buttons are demonstrated for student, but the sidebar itself will be used for all actor’s interfaces for navigation.

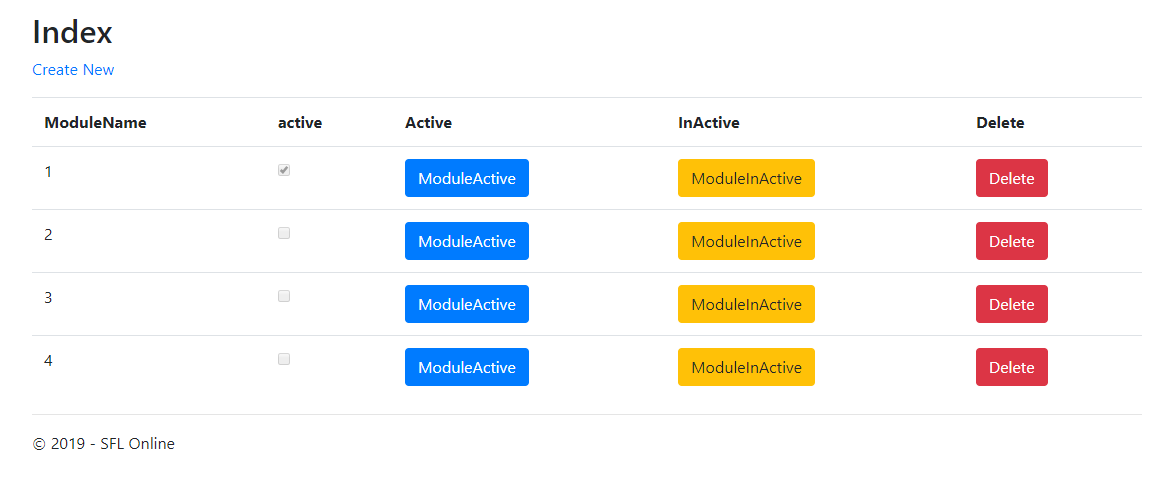


1. *Enrollment Instructor (Enroll Instructor to Class & Course/ Create-Add) – Admin*

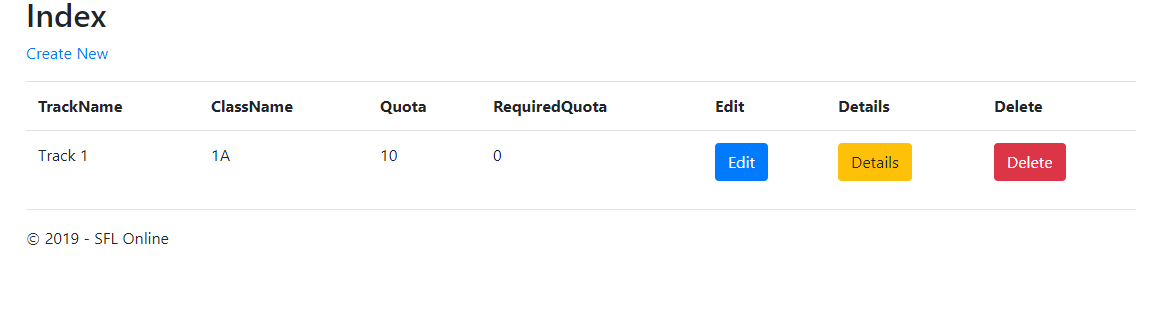




1. *Module Table (CRUD) – Admin*



1. *Classes Table (CRUD) – Admin*



# Glossary

**Actors in the System**

*Actor:* Defined user on the system (i.e. Admin, Student, and Instructor).

*Person*: An abstract table on the database of the system. All the existing actors in the system are inherited from this table. Actors in this table share certain account attributes. All actors are Person, then they are specialized in their respective tables according to their roles.

*Student*: An actor on the system. Represents a prep student in the SFL.

*Instructor*: An actor on the system. Represents an instructor of the SFL program.

*Admin*: The system administrator. The admin has full authorization on the system, and can alter the database, or perform maintenance on the system.

**Terms Concerning SFL**

*Class:* Represents classrooms in SFL that students belong to and instructors teach in. Not to be confused with the programming language “class”, unless stated otherwise.

*Course:* Course topics taught in the classrooms. Some examples are Reading & Writing, Listening, Core, etc.

*Track:* Shows the SFL student’s language level. They are assigned to classes according to the track they belong. There are three tracks, Track 1, 2 and 3.

*Module:* Can be thought as a semester like in bachelor institute. The difference is, there are 4 modules in a school year. Track 1 and 2 students study for full 4 modules (a year), while Track 3 student’s study for 2 modules (half a year).

*Grade Percentage:* The percentage that effects the points required for a student to pass a course. For example, for Reading & Writing course, homework has a 15% effect while midterm has 30% effect on the total grade.

*Academic Average:* The average total grade of SFL Student, according to the modules they have finished. For Track 1 & 2 students, grade averages of all 4 modules are taken as academic average, and module 1 and 2 are taken as academic average for Track 3 students.

*Exit Exam:* An exam taken at the end of the year or half of the year, and determines if a SFL student will pass the prep school and become a freshman at the institute. SFL academic grade average and exit exam grades are calculated to find if a student will pass. Currently, SFL students must have an exit grade of minimum 70 points of out 100 points in order to pass. If a student’s exit grade is less than 70, which is the minimum grade required to pass, they will fail no matter what their academic average grade is.

*Weekly Course Schedule:* Describes a table schedule of which courses a student has at which day and time in the week. It is the same for instructors, but it shows the courses they’re teaching.

**Development**

*Back-end:* Server side of the system. All functionality runs in here.

*Front-end*: Client side of the system. All content & interfaces viewable by the users are displayed in here.

*Template:* Viewable contents.

*CRUD:* Create Read Update Delete. Mostly, this describes admin functions, as the admin can create new entries, edit them and delete them in the database, for all tables.

**Server**

***Request:*** From Client to Server  
***Response:*** From Server to Client  
***Server:*** Receive Request and Send Response  
***Client:*** Send Request and Receive Response

**Frameworks, Technologies & Approaches**

*C#:* Programming language used to develop SFL Online.

*Entity Framework:* Entity Framework is an Object Relational Mapper (ORM) which is a type of tool that simplifies mapping between objects in the software to the tables and columns of a relational database.

*Code-First*: One of the three approaches Entity Framework provides to create an entity model. The developer writes code to specify the model. EF generates the models and mappings at runtime based on entity classes and additional model configuration provided by the developer. We have used this approach while developing the system.

*ASP.NET:* A web development platform provided by Microsoft. It is used for creating web-based applications.

*ASP.NET Identity:* This framework is used for user management/authentication mechanism. OWIN + Identity is used for authentication process in the system.

*OWIN*: Owin defines an abstraction between .NET web servers and web applications. By decoupling the web server from the application, OWIN makes it easier to create middleware for .NET web development.

*Design pattern:* Reusable solution to a commonly occurring problem (i.e. proxy, observer, listener, strategy, model-view-controller, and so on).

*MVC:* A modern design pattern: model-view-controller.

*Html5:* New version of hypertext making language.

*CSS:* An approach to web design for make-up.

*Bootstrap*: A front-end framework for web design for responsiveness.

*JavaScript:* the programming language of HTML and the Web, provides responsiveness

*AJAX:* A technique for accessing web servers from a web page.

*JQuery*: A javascript library, to simplify javascript programming while coding.

*Razor*: Razor is a markup syntax that lets you embed server-based code into web pages using C# and VB.Net. It is not a programming language. It is a server side markup language.

*Function (or method):* Solution to a problem.

*Design methodology:* The development approaches (i.e. Agile, Iterative, Hacking, and so on).

# References

1. Bruegge B. & Dutoit A.H. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Prentice Hall, 3rd ed.
2. [Prep Passing Grade Calculator](http://www.isikun.edu.tr/akademik/sfl)
3. **Campus Automation System (2002) University of Işık,** [**Campus Online**](http://campus.isikun.edu.tr/)
4. **Course Homepages Management System of ISIK University (2003),** [**Course Online**](https://course.isikun.edu.tr/)
5. **Kültür University,** [**CATS**](http://cats.iku.edu.tr/) **(Computer Aided Training & Educational Services)**
6. **Marmara University,** [**Information Management System**](https://bys.marmara.edu.tr/v2/Account/Login)
7. **Official ASP.NET Documents -** [**Get started with Entity Framework 6 Code First using MVC 5**](https://docs.microsoft.com/tr-tr/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application)