

UNIVERSITATEA "ALEXANDRU IOAN CUZA"
FACULTATEA DE INFORMATICĂ



TIMESHEET MANAGEMENT

Redinciuc Daniel-Ilie

Sesiunea: Iulie, 2019

Coordonator științific

Lect. Dr. Ionuț Pistol

**Avizat,
Îndrumător Lucrare**

Titlul, Numele și prenumele _____

Data _____ Semnătura _____

DECLARAȚIE
privind originalitatea conținutului lucrării de licență/disertație

Subsemnatul(a)Redinciuc Daniel-Ilie.....

domiciliul încomuna Cordăreni, jud. Botoșani.....

născut(ă) la data de20.07.1994....., identificat prin CNP1940720071378....., absolvent(a)
al(a) Universității „Alexandru Ioan Cuza” din Iași, Facultatea de ...Informatică..... specializarea
.....Informatică în limba engleză....., promoția2019.....,

declar pe propria răspundere, cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal
și dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență
cu titlul:

_____ Timesheet Management _____
_____ ela

borată sub îndrumarea _____ Lect. Dr. Ionuț Pistol _____, pe
care urmează să o susțin în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență/disertație să fie verificată prin orice
modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului său într-o
bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea
facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diploma sau de
disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul
cercetării pe care am întreprins-o.

Data azi,

Semnătură student

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „Timesheet Management”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași, 26.06.2019

Absolvent Daniel-Ilie Redinciuc

Contents

1. Introduction	5
1.1. Motivation.....	5
1.2. Existing solutions.....	6
1.3. Next steps	6
2. Technologies used.....	7
2.1 Microsoft SQL Server – T-SQL	7
2.2 Entity Framework.....	8
2.3 HTML Agility Pack.....	9
2.4 XPATH.....	9
2.5 Regex.....	10
2.6 ASP.NET	11
2.7 MVC.....	12
2.8 HTML5	13
2.8 CSS.....	14
2.9 JavaScript	15
2.10 JQuery	15
3. Timesheet Management	16
3.1 Application functionalities	16
3.2 Application architecture	17
3.3 Database structure.....	18
3.4 Features implementation	20
3.4.1 Student role	26
3.4.2 Admin role.....	31
4. Final conclusions	34
4.1 Conclusions	34
4.2 Future directions.....	34
5. Documentation	35

1.Introduction

1.2. Motivation

In order to write this paper, I have studied time management solutions and technologies used in industry for creating applications that respond to this problem.

The objective of this thesis is to create a web application that can help the students of the Faculty of Computer Science of Iasi organize their schedule based on the timesheet created by the faculty staff, adding the possibility to adjust it to their needs, but also to add custom events, independently by the faculty schedule.

In the business world, the problem of time management represents quite a huge point of interest as numerous studies ^[1] have concluded that a good time management culture inside of a company, but also outside of it has a series of clear advantages. Developing skills in this direction benefits both the employer and the employee, as the individuals become more punctual and disciplined, their efficiency and performance are increased, the planning can be respected and the targets are met, but this is also reflected in the amount of stress that is being reduced, the confidence of the individual increases as he can achieve his goals and have a clear image of what is going to happen next and last but not least, a good time management leaves time for fun and relaxation.

The necessity of developing a timesheet management solution for the students of our faculty is motivated also by a recent study ^[2] of ecal.com, which made a survey on 1000 people from Australia, aged 18-24. The respondents confirmed the need for such solution, as 70% of them replied that they are using digital calendars to organize their time, such as Mobile Calendar, Google Calendar and Outlook Calendar. Quite a big percent is still taken by the classic method, of organizing the schedule on paper: 28.3%. Only an insignificant percent of 1.7% replied that they are using unreliable methods, like trying to remember what they have to do, relying that someone close to them will remind them or simply not schedule their time at all.

1.2. Existing solutions

The most spread method of managing time is using an agenda. At first, this was a notebook with a page for each day of the day, but as the technology progresses, this was replaced by software calendars, most important ones being Google Calendar, Outlook Calendar and mobile calendars.

There are no exact information about how many users are actively using Google Calendar, but according to techjury.net ^[3], Google has 1.2 billion monthly active users, and since the calendar is part of the account we can assume that the number of people who are using this service is quite high. Google Calendar main feature is the possibility of adding events, by specifying the date, starting hour, ending hour and optionally, a title and a short description. An event can also include attendees, who are people from your community that will be notified about the event invitation via an email.

In the event, there is also the possibility to include a location where the event is going to take place. Google Calendar is also integrated with Google Maps, therefore the location field might be a very useful detail in case the meeting is going to be a physical one. The invites attendees have the possibility to accept, reject or propose a different time, by also including a comment on why the proposed time is better for the meeting. Every event will generate a notification to remind the participants that the event from their agenda is going to start soon.

The number of potential users for an application developed for the students of the Faculty of Computer Science Iasi is about 1600. This is an estimation made on the numbers provided by the university, which advised that yearly there are 1586 of students plus the number of potential interested teachers and admins.

1.3. Next steps

In the next chapters, I am going to present the technologies that have been used, and the way I have implemented this targeted solution using the presented technologies, together with a user guide of my application. My goal is to develop a scalable solution, capable of improving the student experience at this faculty and also help the students get ready for the industry practices, but also to offer a nice looking alternative to the timesheet of the faculty. What is different in this application from the others presented above is the direct integration with the website of the faculty, so the classes' events won't be required to be added manually.

2. Technologies used

2.1 Microsoft SQL Server – T-SQL

Microsoft SQL Server is a management system for relational databases, released by Microsoft on April 24, 1989 ^[4]. Its primary functionality is to store and retrieve data requested by another software applications which are running on the same computer or that are connecting to the database which is running on a server.

A database represents a collection of tables, views, indexes, triggers, functions and stored procedures. In order to create and interact with the database, Microsoft also create a procedural language extension for SQL Server, named T-SQL.

The command to create a table is “CREATE TABLE [schema].[tablename]”, but is it also mandatory to specify a list of fields that this table will contains. A table can store numbers (INT, DOUBLE, FLOAT), text strings (varchar and nvarchar – which also supports the Unicode characters) or dates (DATETIME and SMALLDATETIME).

When creating a table, T-SQL allows us to add data validation constraints. For example, if we want a field to be filled in mandatory, we should add the constraint “NOT NULL” in the field definition at table creation or we can add it later via the command “ALTER TABLE”. This will allow us to change the field validation, but a database clean-up might be required as all the existing data must match the new constraint.

The most important constraints in a SQL Database are the constraints for primary key and foreign key. A primary key represents a unique identifier at table level, so two or more lines in a table cannot have the same identifier. Because we would have to check before inserting in a table if the primary key hasn't been used yet, via T-SQL we can remove this check by specifying to the primary key that it has to auto-increment. This is being done via the command IDENTITY(seed, increment), so if we give to a primary key the IDENTITY(1,1) property, each row will be automatically receive primary key of the last entry, incremented by 1. A foreign key represents a field in one table which is primary key in another. This helps to create a good linking between tables.

In T-SQL, another popular feature is the usage of Stored Procedures. A stored procedure represents a series of SQL commands, encapsulates in an object, which can perform certain actions based on the input parameters or on a fixed logic. Ideally, they should perform updates on the database on a large number of rows with very little processing. The usage of cursors is not recommended, as a Stored Procedure's performance is decreasing as the number of computing operations are increasing.

2.2 Entity Framework

Entity Framework is an open-source Object Relational Mapper (ORM) framework for .NET applications supported by Microsoft. It was released in 2008 and its purpose is to simplify the mapping between software objects and the tables from a relational database. Entity Framework takes care of creating the database connections, executing commands, reading queries output and converting it to software objects. This is Microsoft recommendation for new .NET applications, over technologies like LINQ or ADO.NET.

Entity Framework is one of the most flexible ORM's, as it allows us to map single entities to multiple database table or multiple entities to a single table:

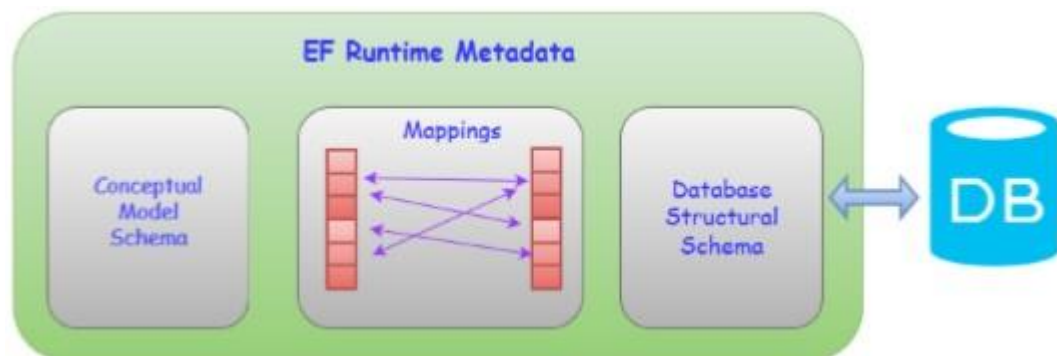


Fig.1 Entity Framework model [5]

In Visual Studio 2017, Entity Framework is not included as default, but can be downloaded via the Nugget Package Manager. There are different approaches on how to link the software application to the database via the Entity Framework. One of them

is the “Database First” approach. For this we would need to already have an existing database or to create it via a T-SQL script as described in the previous subchapter. When this approach is used, we have to specify the address of the server, the username and the password and the database that we want to import. Entity Framework does a lot more than just creating a connection between the application and the database, but also generates a graphical schema of the database, but illustrating the tables’ definitions and the relation between them. At the same time, a database context is created, together with database models, which are the SQL tables mapped as classes in the software application.

2.3 HTML Agility Pack

Html Agility Pack is a HTML parser that can be used to read/write DOM and has support for XPATH. This is the most used web-scrapper tool in C#, being downloaded more than 15 million times ^[6].

An instance of the `HTMLOdoment` class from Html Agility is built from an html file loaded from the local environment or extracted from a website via a `Http Client`. The instance of the class identifies the nodes of the HTML and allows us to extract a certain node or a series of nodes that match the same requirements. In order to extract the information from a node, properties `InnerHTML` or `OuterHTML` can be used.

2.4 XPATH

XPath is a query language that is used to extract nodes from XML documents. The first version of XPATH has been released in 1999 and although multiple versions have been released after that, the first version remained the most used one

Used in combination with HTML Agility Pack, XPath is used to extract the desired node. Most used commands are ^[6]:

- “nodename” – it will select the node with the specified name
- “/” – it will select the root node
- “//” – it will select the nodes from the current node

- “@” – it will select the desired attributes
- “.” – it will select the current node

2.5 Regex

Regular expressions are widely integrated in almost every text editor and all the programming languages I've used so far had the possibility to use regular expressions either natively or via an external module. A regular expression is a special text string that defines a search pattern. The concept of regular expression appeared in the 1950's, thanks to Stephen Cole Kleene, an American mathematician which described what a regular language is.

This technology can be used in combination with XPath and HTML Agility Pack for cleaning up an extracted HTML document. Let's say we want to extract all the references to external web sites. For this we would need to create a Regex variable, with the following form: `"href=\"\\.\\"/>`

In C#, the Regex library is returning an array, which has on the first position the regex match and on the second one the string that have been found via the wildcard searching. These are some of the basic wildcards that are commonly used in regular expressions:

Wildcard and character matching ¹⁰⁰		
.	Matches any single character.	a.c matches aac, abc, azc, a2c, a\$c etc.
\d	Matches a decimal digit character (i.e. 0-9).	a\d matches a1, a2, a3 etc. but not aa, ab etc
\D	Matches a non-digit character.	a\D matches ab but not a1, a2, a3 etc
\s	Matches any whitespace character (space, tab, newline).	ab\s matches ab d but not abcd, abxd etc
\S	Matches any non-whitespace character.	ab\S matches abcd, abxd etc. but not ab d
[...]	Matches the characters specified in the brackets. You can specify a range of characters by specifying the first and last characters in the range, separated by a hyphen. You cannot use other special characters within the [] — they will be taken literally.	9[aeiou] matches 9a, 9e, 9i etc. and 9A, 9E, 9I etc, but not 9b, 9c, 9B, 9C etc 9[a-z] matches 9a, 9b, 9z etc. and 9A, 9B, 9Z etc. but not strings such as 91, 99 or 9([0-9#*] matches any single E.164 character (digits 0-9, hash key or asterisk)
[^...]	Matches anything except the set of specified characters.	[^a-z] matches any non-alphabetical character [^0-9#*] matches anything other than an E.164 character

Fig 2. Regular expressions ^[7]

2.6 ASP.NET

ASP.NET is an open source web application framework designed for web development. It was released on 2002 by Microsoft and can be used to develop web sites, web applications and web services. Although is not very popular among freelancers or startups, the .NET platform remains popular in huge corporations.

The .NET framework can theoretically support any programming language. However, in order to be compatible with .NET, a programming language must respect CLI (Common Language Infrastructure) specifications. CLI standard has been created by Microsoft and defines what features a programming language must provide in order to be .NET-compliant. The most used programming language for .NET is C#, which have been developed by Microsoft at the same time while working the .NET framework.

C# is a Full Object Oriented Programing Language, everything being an object. It's not possible to have global functions, variable or constants ^[8]. It has native support for all the basic variable types like integers, Boolean, strings, etc.

C# provide type safety, which means that all objects are initialized by zero dynamically, error messages will be produced when trying to access an uninitialized variable, there are automatic checks on arrays (check the out bounds) and others.

Another essential feature of this programming language is the ability to handle exceptions. This is being done via the “try-catch” block. There are several specialized exceptions that can be caught (example: `DivideByZeroException`), but there is also the possibility to catch all the exceptions via the generic “Exception” one.

2.7 MVC

MVC (Model-View-Controller) is an architectural pattern which dives the application in 3 parts (Fig.3). Initially this pattern was designed and used for GUI (Graphical User Interface) applications, however this became the most used pattern for web applications.

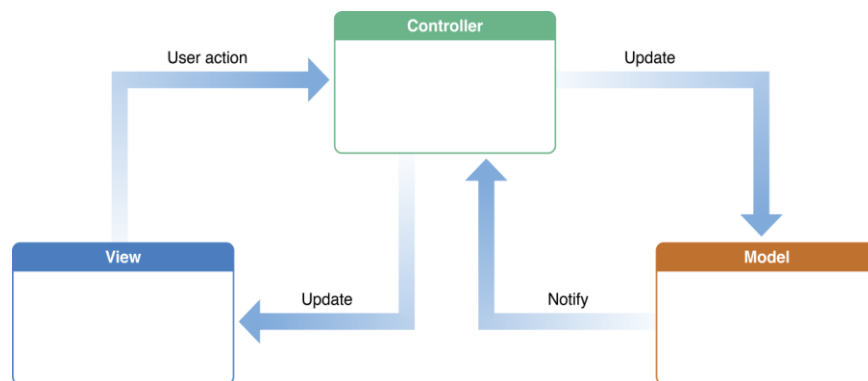


Fig. 3 MVC workflow [9]

View:

It is the most important part for the end user, as the user cannot see the model or the controller, but only what is displayed on the screen. The view is responsible to represent information as tables, charts, lists, diagrams and so on. The view is responsible with showing the model in a user-friendly way.

Controller:

The controller is the core component of the MVC and represents the intermediary between one or more application views and one or more model objects. The end user sees the view and wants to trigger an action. But the action is not handled by the view, but all the commands are sent to the controller, which is validating the input, and updating the model.

Model:

Some prefer that the model to define how the data should be structured but also process the data entirely. However, when using Entity Framework and .NET's MVC, the models are generated automatically from database tables, but if that is not enough, we can create additional models. A model is responsible with encapsulating the data. In object oriented programming languages it doesn't represent anything more than a class. A model can be a person, an animal, a class, a table if that is relevant to the application. The model processing is done in the controller.

Using MVC has plenty of advantages, such as faster development times, having different views for the same model and very SEO-friendly. Since its modular structure, it can allow parallel development and different team working on each part. The disadvantages of this pattern is the increased complexity, as multiple technologies must be used (ideally, the developer should understand both the front-end and the back-end parts of the project)

2.8 HTML5

HTML5 is the fifth version of HTML (HyperText Markup Language), which was released in 2008. The first version of HTML was released in 1993. HTML is the standard language for documents designed to be displayed in a web browser.

The web browsers can receive the HTML Document either from a web server or locally and they are able to render its content into multimedia web pages. However, the pure HTML pages are not what a user would expect to see in 2019. The W3C (World Wide Web Consortium) recommended the usage of CSS over HTML since 1997.

The most used HTML tags are ^[10]:

- `<html>` is the main element, the root of the document, which specifies that the document is html
- `<head>` is used to specify all the head elements like title of the page (`<title>`), define CSS via the `<style>` tag etc. It is also recommended to add here the links to JavaScript or Bootstrap files.
- `<body>` is used to define the main part of the html. Here is where we add images, list, tables, etc.
 - `` is used to define the beginning of a unordered list
 - `` is used to define a list element
 - ``, ``, `<u>` are used for text formatting
 - `` is used to create a link to another page
 - `<form>` - is used to mark the beginning of a form
 - `<input>` is used to define which type of input is expected in a form
 - `<table>` is used to mark the beginning of a table
 - `<th>`, `<tr>`, `<td>` are tags that are used inside of the `<table>` tag to define the header, row and column.
 - `<div>` is used to create a division in the HTML page

2.8 CSS

CSS (Cascading Style Sheets) is a style sheet language used to describe a document written in a markup language, which was released in 1996 ^[11]. CSS allow to separate the content from presentation and it greatly improves the User Interface, by adding beautiful colors, fonts and style over HTML.

The CSS is usually provided via Bootstrap templates, which are CSS files that contain nice stylings for every HTML elements and that can be used with just a little understanding of CSS styling.

Some of the main CSS elements are ^[12]:

- `display` : there are 4 main values used for this:
 - `block`: this allows the elements to take as much space as they can and usually cannot be place in the same horizontal line with other display modes
 - `inline`: wraps the element tightly around them

- inline-block: combines the block and inline properties
- none: the element is completely hidden
- width and height: the names of these properties are self-explanatory. The units are px, em, rem, % or auto
- margin and padding: these are used to define the space between elements
- color: you can specify the color of a elements as a hex value or a RGB value
- background: it can be used to set a certain color or images in the background of elements
- font: used to stylize text

2.9 JavaScript

JavaScript is another programming language used in web development and it can change both the HTML and the CSS, but can also perform calculation and validate data. It can work with numbers, strings, objects, arrays and functions ^[13].

One of the most used feature of JavaScript is the possibility to change HTML Content. This is mainly perform via the “getElementById” method of the document, which receives as parameter the id of a HTML Element. After that, we can use the DOM property “innerHTML” to change its content. JavaScript is a case sensitive language, so the methods and properties must be used as declared.

With JavaScript we also have access to AJAX (**A**synchronous **J**avaScript **A**nd **X**ML), which is a combination of a XMLHttpRequest object, JavaScript and HTML DOM. With the help of AJAX, we can update a webpage, without the need to refresh it, we can send or receive data from the server after the page has loaded and se can send data to the server in the background ^[14]

2.10 JQuery

JQuery is a JavaScript library released in 2006. It is used to simplify the animation creation process. This is the most used JS library due to the large amount of features and due to the fact that it JavaScript is more difficult to learn and using already implemented JS function from a JQuery library is more efficient.

3. Timesheet Management

3.1 Application functionalities

Our application has the following functionalities:

- The interface is easy to use (Annex 1)
- The solution is well structured, every layer of the application being separated, but interacting with the other ones.
- The access to pages is restricted based on login and user roles
- The web application has 2 areas, one for admins and one for students
- Student accounts can be created via the web application
- Users can login into their accounts
- Users have access to an account page where they can fill in information
- Students can select to which group they have been assigned by the faculty
- Based on the group of the student, a list of classes are assigned to the student
- The student can see the schedule as a list
- The student can remove classes from his/her schedule
- The student can replace a class with another class of the same subject
- The student can provide a feedback on the reason behind the change
- The student can add classes which were not promoted and that are not usually linked to his/her group
- The student can see all of the subjects taught in the faculty, access their discipline files and the linked classes
- The student has access to a visual calendar of all the classes.
 - The student can add custom events, edit and remove existing ones
- The student can use to “Pomodoro technique” to track the time spent on learning
- The admin has access to see a list of existing students and can view, edit and delete their profiles
- The admin has access to see a list of existing subjects and classes and can view, edit and delete them
- The admin has access to a refresh page, where he can retrigger the application synchronization with the website of the faculty
- The admin has access to graphical reports about the application usage

3.2 Application architecture

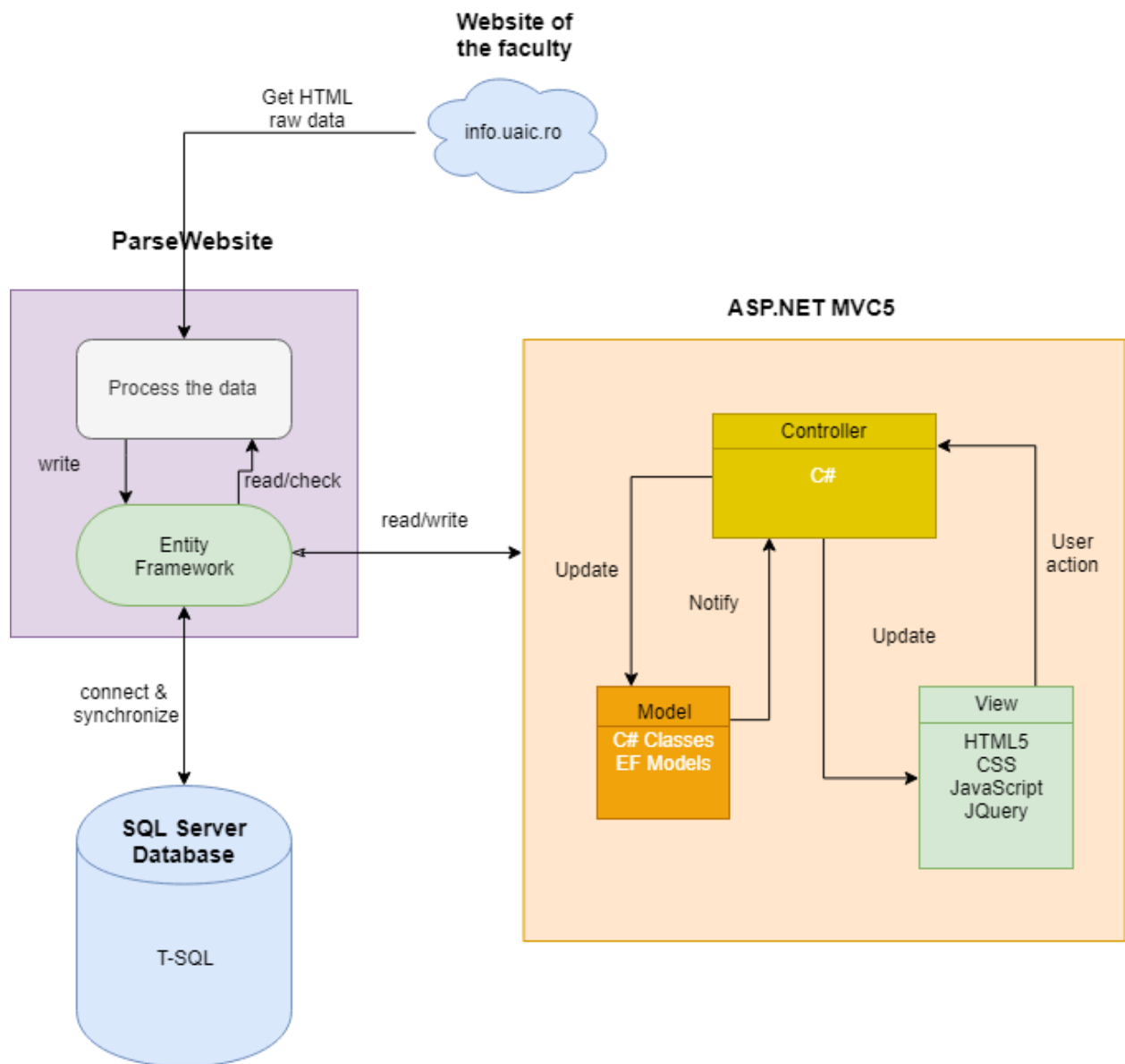


Fig 4. Application architecture

3.3 Database structure

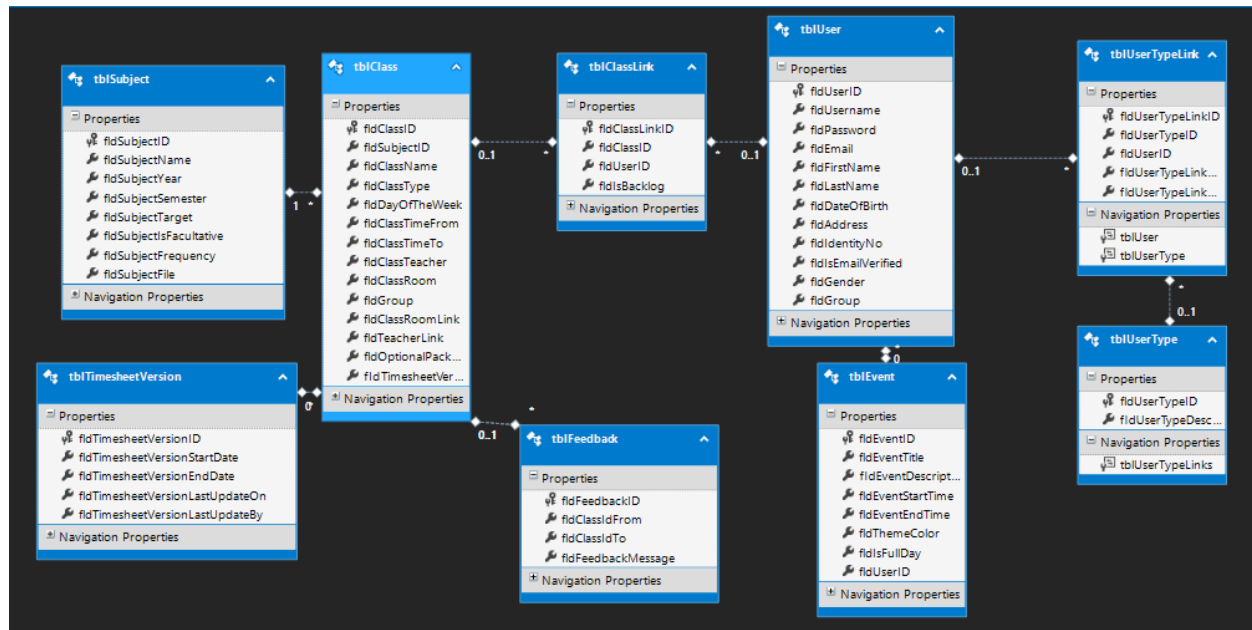


Fig 5. Database structure

The above scheme has been automatically generated by Entity Framework.

We have the following tables:

- **tblUser**: this table is storing information about a user of the application. It contains info about the name (*fldFirstName* + *fldLastName*), username (*fldUsername*), password (*fldPassword*), email (*fldEmail*), date of birth (*fldDateOrBirth*), address (*fldAddress*), identity number (*fldIdentityNo*), gender (*fldGender*) and information of the group where the user is allocated (*fldGroup*).
- **tblUserType**: this table is used to define different roles. Besides the unique identifier, it contains the description of the role (*fldUserTypeDescription*). Currently, we have 2 roles: student and admin, but new roles can be added in case of need.
- **tblUserTypeLink**: this table represents the junction table between **tblUser** and **tblUserType**. It contains 2 foreign keys: *fldUserTypeID* (**tblUserType**) and *fldUserID* (**tblUser**). It also allows to define a start and ending date for a certain role (*fldUserTypeLinkStartDate*, *fldUserTypeLinkEndDate*). The reason why I chose

to create the user type in different table is to keep the database normalized. Having a junction table, it supports theoretically a user having multiple roles at once. Also if I would've chose to create a field in tblUser to check the roles, that would mean that we need to alter the tblUser table every time in order to add a new column, which is not a good approach for maintaining a database.

- **tblSubject:** this table is used to store information about the disciplines available at the faculty. It stored the subject name (*fldSubjectName*), the program to which it addresses (*fldSubjectTarget*), the year and semester when it's being taught (*fldSubjectYear* + *fldSubjectSemester*) and the link to the pdf containing the subject file. Also, since some subjects are facultative, we are also storing this information in *fldSubjectIsFacultative*.
- **tblClass:** this is one of the most important table. Here is where we have the actual information about a class. This has the foreign key to tblSubject (*fldSubjectID*) and to tblTimesheetVersion (*fldTimesheetVersionID*) as we need to have the information well linked in case of refresh. The table also stores information about the class name (*fldClassName*), the type of the class (*fldClassType* – course, exam, laboratory etc), the day of the week (*fldDayOfTheWeek*), start time of the class (*fldClassTimeFrom*), the end time of the class (*fldClassTimeTo*), the name of the teacher (*fldClassTeacher*), the room where the event takes place (*fldClassRoom*) and info about the optional package in case the class is an optional (*fldOptionalPackage*)
- **tblTimesheetVersion:** as changes can happen in the official timesheet, we need to keep our application up-to-date, but we want to also keep a history of the previous classes. Therefore we are linking the classes with the current timesheet version. In this table we are storing the start date of the version (*fldTimesheetVersionStartDate*), the end date of the version (*fldTimesheetVersionEndDate*) and tracing field (*fldTimesheetVersionLastUpdateOn* and *fldTimesheetVersionLastUpdateBy*). When a new version is created, the end date is set to the maximum date value and the previous version ending date should be set to the starting date of the new version
- **tblClassLink** is the junction table between tblUser and tblClass. It defines which classes the user should attend. It contains the foreign keys from these 2 tables and also a field that we can use to mark that a certain class is actually a backlog class from a previous year that the user didn't promote (*fldIsBacklog*)
- **tblFeedback** is the table where we keep track of all the class changes. In order to improve the timesheet, when a user changes a class, we are collective info about

what is the reason of making this change. It contains foreign keys to tblClass (*fldClassIDFrom*, *fldClassIDTo*) and also the feedback message (*fldFeedbackMessage*)

- **tblEvent** is the table where we store information about custom event in the calendar of a student. We store here the title of the event (*fldEventTitle*), the description (*fldEventDescription*), the covered period of time (*fldEventStartTime* + *fldEventEndTime*), the color of the box (*fldThemeColor*), the check if the event is going to take all day or not (*fldIsFullDay*) and a foreign key to tblUser to see which is the owner of the event (*fldUserID*)

In the database we also have a stored procedure name “dbo.spUpdateClassLinks”. This is responsible with recreating the class links when the user changes the group or when the timesheet is being refreshed.

3.4 Features implementation

First of all, I will present the module of the application responsible with parsing the website in order to have a clear idea on how do we process the raw html data and how do we store it in the database.

First, we need to extract the subjects as these are foreign keys in class table. The subjects for the bachelor program are found at <https://www.info.uaic.ro/programs/informatica-ro-en/>, and the ones for the master program at <https://www.info.uaic.ro/studii-de-master/>. In order to get the HTML document information, we use HTML Agility Pack nugget package. In order to get the raw html data we need an Http Client, so the method dealing with this must be asynchronous. The GetStringAsync method is going to return a string, but we cannot yet process it, therefore we are passing the raw data to the HtmlAgilityPack class HtmlDocument, which will identify the nodes and will allow the parsing. As we want to make sure that the parsing of the subject happens before we start parsing class, we are making the method as a Task and we waiting for it to finish.

```
var httpClient = new HttpClient();  
var html = await httpClient.GetStringAsync(mastersUrl);  
var htmlDocument = new HtmlDocument();  
htmlDocument.LoadHtml(html);
```

Both the bachelor subjects and the master subjects have the same structure: they are stored in a multiline JavaScript variable. Since we want the parsing to be as generic as possible, we are using another Task method named “ParseSubjectLink”, which receives two parameters: the link that needs to be parsed and the program related to this link (“Bachelor”/”Master”). We are extracting the entire HTML page and stored it in an **HTMLDocument** instance. Here is where we can use the XPATH to extract the needed data. We can observe that the JavaScript variable is located in the main div, in the script section. In order to extract the text only we are passing the full XPATH to the **SelectNodes()** method of the **HtmlDocument** class:

```
var subjectHtml = htmlDocument.DocumentNode.SelectNodes("//div[@id='main']/div/div[1]/div[1]/div[2]/script/text()");
```

Fig. 6: Extracting JavaScript variable content using XPATH

All the bachelor subjects can be found in a single JS variable, however, because for the master degree we have multiple specializations, we have a root page with links to all of them. As we said, we want to keep the ParseSubjectLink as generic as possible, so we are extracting from the root page all the links, and calling the parsing method for each link. In order to extract the sub-links, we use XPATH again (Fig.17). The links are placed in an unordered list (ul). There are 6 specializations, but only 5 of them are using fixed subjects, the last one focusing on researching. Here is where we can put to good use the ability of XPATH to also do computations, so we can specify that only the first five links should be extracted, by using the position() property of XPATH.

```
var subjectURLS = htmlDocument.DocumentNode.SelectNodes("//div[@id='main']/div/div[1]/div[1]/div[2]/ul/li[position()<6]");
foreach (var subjectURL in subjectURLS)
{
    string subjectURLFull = Regex.Match(subjectURL.InnerHtml, regexLink).Groups[1].Value;
    subjectURLFull = "https://www.info.uaic.ro" + subjectURLFull;
    ParseSubjectLink(subjectURLFull, "Master").Wait();
}
```

Fig. 7 Extracting all the master subjects links

The extracted links are actually relative links, so we cannot navigate to them as they are. Therefore, we have to attach them to the root address, which is <https://www.info.uaic.ro>. We also have to clear the links by using a regular expression. The “regexLink” pattern from Fig.7 is defined as:

```
“string regexLink = “href=\\“(.*?)\\””;
```

This pattern is looking for portions of text which starts with “href=” keyword. Then we use the dot wildcard that will extract any single character and we also apply the star (*) which will indicate to take as many as possible. The ending sign is the quote which will mark the ending of the link. The Match method from the Regex class is returning an array, which will have on the first position the match, and on the second position will have the portion of text which have been extracted due to wildcards. This is the part that we are interested in, because we want the actual relative link, without the “href” string, so that we can concatenate and call the parsing method with a valid link.

Now that we solved the problem of extracting the links, we’ll present how we parse the JS variables (Fig. 8) that are holding all the subjects information. The raw version of the text returned by the XPATH contains a container and a variable named *rawData*. The information stored here is very well structured and can be parsed line by line. In order to extract its content, we have to use regular expressions again. The pattern is “`(. *\\s. *)*`”. This will extract any character including newline character, which is limited between the “” character.

```
var container = document.getElementById("root");
var rawData = '1|1|01|Advanced Software Engineering
Techniques|7|http://www.info.uaic.ro/~webdata/planuri/master/en/MISS1101.pdf|https://profs.info.uaic.ro/~adiftene/studenti.html
1|1|02|Formal Methods In Software Engineering|8|http://www.info.uaic.ro/~webdata/planuri/master/en/MISS1102.pdf|https://sites.google.com/site/filcoursefms/
1|1|0301|Java Technologies|7|http://www.info.uaic.ro/~webdata/planuri/master/en/MISS110301.pdf|
1|1|04|Distributed Operating Systems|8|http://www.info.uaic.ro/~webdata/planuri/master/en/MISS110401.pdf|
1|1|04|Reverse Engineering|8|http://www.info.uaic.ro/~webdata/planuri/master/en/MISS110402.pdf|
1|1|04|Advanced Topics in Neural Networks|8|http://www.info.uaic.ro/~webdata/planuri/master/en/MISS110403.pdf|https://sites.google.com/view/rbenchea/advanced-chapters-of-
neural-networks
1|2|05|Advanced Topics of Machine Learning|8|http://www.info.uaic.ro/~webdata/planuri/master/en/MISS120501.pdf|
1|2|05|Concurrent and distributed programming|8|http://www.info.uaic.ro/~webdata/planuri/master/en/MISS120502.pdf|
1|2|05|Event-based Systems|8|http://www.info.uaic.ro/~webdata/planuri/master/en/MISS120503.pdf|
1|2|06|Human-Computer Interaction|7|https://profs.info.uaic.ro/~busaco/teach/courses/hci/
1|2|06|Software Quality|7|http://www.info.uaic.ro/~webdata/planuri/master/en/MISS120602.pdf|
1|2|06|Pragmatic Aspects in Engineering of Cloud Systems|7||
2|1|01|General Fundamental Discipline III|8||
2|1|02|Fundamental Discipline for Software Engineering V|8||
2|1|03|Fundamental Discipline for Software Engineering VI|8||
2|1|04|Professional/Research Practice III|6||
2|2|05|Fundamental Discipline for Software Engineering VII|8||
2|2|06|Fundamental Discipline for Software Engineering VIII|8||
2|2|07|Complementary Discipline I|8||
2|2|08|Dissertation Preparing|6||';
```

Fig 8. Multi-line JS variable holding the discipline information

The structure of the variable is the following: on each new line we have a separate subject and its elements are separated by “|” :

- on the first position we have the subject year
- on second we have the semester
- on third we have the info if a subject is facultative or not
- on fourth position we have the subject name
- on fifth we have the link to the discipline’s file PDF.

Since the subjects rarely change we don’t recreate them at each parsing, but we check if it already exists. We are creating a **context object** and creating subject instances of the **tblSubject** model class. Once the **SaveChanges** method is being called, the

instances are inserted in the database. This is the most time consuming action, so we are calling inserting new lines in the database after the creation of each subject, but we are storing everything and save everything is the end. This made a huge improvement in terms of performance.

```
foreach (var line in rawData.Split('\n').Select(s => s.Trim()).ToArray())
{
    string[] items = line.Split('|').Select(s => s.Trim()).ToArray();
    var newSubject = new tblSubject();
    newSubject.fldSubjectYear = Int32.Parse(items[0]);
    newSubject.fldSubjectSemester = Int32.Parse(items[1]);
    newSubject.fldSubjectIsFacultative = items[2].ToUpper().Contains("F") ? 1 : 0;
    newSubject.fldSubjectName = RemoveDiacritics(items[3]);
    newSubject.fldSubjectFile = items[5];
    newSubject.fldSubjectTarget = target;

    var checkIfExists =
        context.tblSubjects.FirstOrDefault(s => s.fldSubjectName == newSubject.fldSubjectName);
    if (checkIfExists == null)
    {
        context.tblSubjects.Add(newSubject);
    }
}
context.SaveChanges();
```

Fig 9: Subject parsing and saving via Entity Framework

In the JS variable we don't have however information about optional packages. Those are in found at the same links, in the html headers on type H4.

Before starting to parse the classes, we have to make sure that we keep track on the versions of the timesheet. Therefore we are creating a method named "CreateNewTimesheetVersion". Here we are using Entity Framework to extract the current version.

```
var currentVersion = (from c in context.tblTimesheetVersions
                      where c.fldTimesheetVersionStartDate < DateTime.Now
                      where c.fldTimesheetVersionEndDate > DateTime.Now
                      select c).FirstOrDefault();
```

If there is none, we create the first one. If exists, we are ending the current one by setting the end date to current date and time, and create a new version with unlimited ending. Until a version will be created, this will be our indicator that this is the valid version that should be used in the MVC application, but also in the class parsing.

The class parsing is the based on the timesheets available at the following link: https://profs.info.uaic.ro/~orar/globale/orar_complet.html. On this page we have a line for each time frame, but behind that we have a link which shows all the classes for a discipline.

We are using the same regular expression for relative links extraction. Because we don't want to parse the same links multiple times, we are storing them in a list after parsing a link and checking if we haven't parsed it yet.

Although for the disciplines we have 2 versions of the page, one in Romanian and one in English, in the timesheet page the things are not that well-structured and the classes' names can be either in English or in Romanian. In order to insure a correct linking between subjects and classes, we are using a dictionary, and every classes will be name in the Romanian language. Since most of the students are using this language, this is the most convenient approach.

```
private static Dictionary<string, string> Translations = new Dictionary<string, string>();  
  
1 reference | 0 exceptions  
private static void DefineTranslations()  
{  
    Translations.Add("Advanced Programming", "Programare avansata");  
    Translations.Add("Algebraic Foundations of Computer Science", "Fundamente algebrice ale informaticii");  
    Translations.Add("Algorithm Design", "Proiectarea algoritmulor");  
    Translations.Add("Computer graphics", "Grafica pe calculator");  
    Translations.Add("DBMS Practice", "Practica SGBD");  
    Translations.Add("Numerical calculus", "Calcul numeric");  
    Translations.Add("Object-Oriented Programming", "Programare orientata-obiect");  
}
```

Fig 10. C# dictionary for subject names

All the classes links are parsed by the same method, called "ParseClassLink", which will have the same behavior for all the links. All the html classes links have the same structure: they contain 2 tables. In the first table we have the recurrent classes, which take place weekly at the mentioned hours. The second table contains one time events, such as exams, projects or a recovery for a lost class. We want to also store the date in the fldDayOfTheWeek field from tblClass and we want them to appear only in the mentioned date in the visual calendar from MVC project that we will present below.

We are defining an array which contains all the names in Romanian for every day of the week. Then when we are parsing the tables line by line, if the line contains one of the weekday names, we know that is a header and we are skipping. Until another weekday name is encountered, we know that all the subsequent subjects are in that day. Each table line contains the hour when the class starts and ends (columns 1 and 2), the class name (column 3), class type (column 4), the groups that can attend (column 5), teachers (column 6), room (column 7) and optional package (column 8).

For the group, classroom, teacher there can be multiple values. If we encounter a character that suggests there might be multiple values, we are splitting the string,

removing the html syntax via regex and save the elements in the database, hyphen separated, in a single string (Fig. 11)

```
#region Teachers
case 5:
    //in case there are multiple teachers, make a split and apply the title regex on each part
    if (cell.InnerHtml.Contains("<br>"))
    {
        string teachers = String.Empty;
        var teachersRaw = Regex.Replace(cell.InnerHtml, "\\s+", string.Empty).Split(new string[] { "<br>" }, StringSplitOptions.None);
        foreach (var teacher in teachersRaw)
        {
            teachers += Regex.Match(teacher, regexTitle).Groups[1].Value + " - ";
        }
        newClass.fldClassTeacher = teachers.Remove(teachers.Length - 2);
    }
    else //else apply the regex on the main part
    {
        newClass.fldClassTeacher =
            Regex.Match(Regex.Replace(cell.InnerHtml, "\\s+", string.Empty), regexTitle)
                .Groups[1].Value;
    };
    break;
```

Fig 11. Parsing the teachers cell and checking if multiple values are filled

Next we are going to present the most important part of the project, the web application where we are putting to good use the information that we obtained from the faculty website and we stored in the database.

It has the following structure:

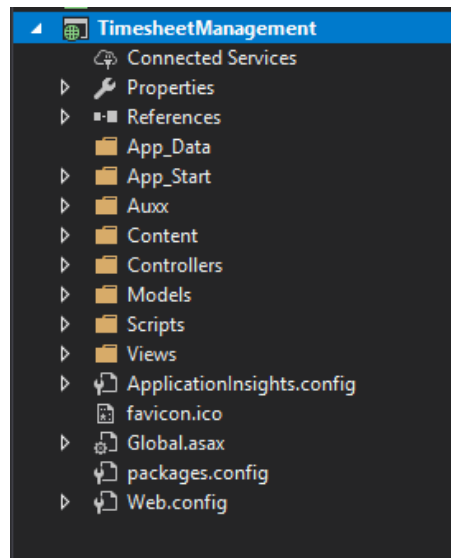


Fig 12. The MVC project parts

In the **App_Start** folder we have the BundleConfig, where we are defining which bootstrap files we are using and also linking the JavaScript and JQuery scripts.

In the **Auxx** folder, we have the **Utils** class which contains useful methods that can be used in all the classes, like the method which is encrypting string using the MD5 encryption algorithm which we use to encrypt the password.

In the **Content** folder, we keep everything useful for creating a modern design to our website, like fonts, images and Bootstrap files

In the **Controllers**, **Models** and **Views** folder are keeping the core elements of a MVC application.

In the **Scripts** folder we are keeping JavaScript files and JQuery libraries.

Next, we will present the 2 existing roles and what functionalities are available on each page. The access to each page is restricted, so without an account, only the home page can be accessed

3.4.1 Student role

From the Register page the only account that can be created is the one with the “student” role. The components used for these pages are: **AccountController**, **AccountModel** and the Login and Register views. The view files are Razor views. This engine allows us to switch from html code to C# code only by using the “@” sign. On Login and Register page we have 2 forms that are calling the post methods **LoginVerify** and **Register**. In the C# controllers, we can have different methods with the same name with we use Data Annotations. In the **AccountController**, we have 2 methods called **Register**. One is marked as being a get method and the other one a post method. When we are accessing the page, the get method is being called to display the form, and once the submit button is being pressed, the POST method is called, which receives an Account model which is filled with the form information. We are using Entity Framework to check if an account doesn’t already exists, and if it doesn’t, we are hashing the password and create the user in the database:

```
var check1 = context.tblUsers.FirstOrDefault(u => u.fldUsername == account.fldUsername);
var check2 = context.tblUsers.FirstOrDefault(u => u.fldEmail == account.fldEmail);
account.fldPassword = CreateMD5(CreateMD5(account.fldPassword));
context.tblUsers.Add(account);
```

After the login is confirmed, we are using sessions to store the user id, this way we can deny the access or redirect to student or admin pages

Once the student credentials are confirmed, he is being redirected to the student home page. In order to keep a navigation bar on all the pages, we are using a layout. This contains the navigation bar html and CSS code, then in the body part we are calling

a C# method name **RenderBody()**. This makes sure that all the web pages that are using a certain layout will be rendered in the space remained unoccupied by the layout elements.

```

<span class="icon-bar"></span>
<span class="icon-bar"></span>
<span class="icon-bar"></span>
</button>
@Html.ActionLink("TSM", "Index", "StudentHome", new { area = "" }, new { @class = "navbar-brand" })
</div>
<div class="navbar-collapse collapse">
<ul class="nav navbar-nav">
<li>@Html.ActionLink("Home", "Index", "StudentHome")</li>
<li>@Html.ActionLink("My Schedule", "Index", "MySchedule")</li>
<li>@Html.ActionLink("My Account", "Index", "StudentInfo")</li>
<li>@Html.ActionLink("Unallocated subjects", "Index", "ClassBacklog")</li>
</ul>
<ul class="nav navbar-nav navbar-right">
<li>@Html.ActionLink("Log in", "Login", "Account")</li>
<li>@Html.ActionLink("Log off", "LogOff", "Account")</li>
</ul>
@* @Html.Partial("_LoginPartial") *
</div>
</div>
<div class="container body-content">
@RenderBody()
<hr />
<footer>

```

Fig. 13: The layout of the navigation bar for student role

Once on the home page, the student has to first fill in account information. This is being handled by the StudentInfoController which is passing to the view a model of type UserGroup. This class contains an object of tblUser and a list of strings. The user part is populated with the available information we have about the logged user so that we can display it in the form via GET. The list of strings is populated with all the known groups, which are extracted from tblClass. Once we user is filling the account information, the controller is triggered again by the user action and an instance of UserGroup is passed as parameter, now having all the required information to personalize the user's schedule. Once the group was set or changed, we are calling a stored procedure which is create class links between the user and the classes available for the selected groups.

Next, the user can see his classes and personalize his schedule. For this, we have another controller named MyScheduleController. The index action results method is working with the ScheduleModel. This model contains 4 lists of ClassModel type. I've split the classes in 4 categories: mandatory, optional, exams and others (which contains the facultative classes).

```

public class ScheduleModel
{
    public List<ClassModel> MandatoryList { get; set; }
    public List<ClassModel> ExamsList { get; set; }
    public List<ClassModel> OptionalList { get; set; }
    public List<ClassModel> OthersList { get; set; }
}

```

The model is passed to the view, which is creating 4 tables containing these classes. For each row we are displaying a button that allows to change the hour, teacher or group where you want to attended, if the class is eligible. For courses, a student is not able to change the assigned group. Once a student clicks on the switch button, the id of the class link is being passed to the ClassController. This extracts the class and the subject and builds a list of classes for the same subject can be used as an alternative. If a class is already added in your profile, it is not displayed. Once you select another class, the link with the old one is deleted and the old class will be visible in the alternatives list.

In the same view we also have a button that is redirecting to the VisualSchedule view, which is using the Full Calendar JQuery library in order to build a nice looking schedule. In order to create the calendar we are using the AJAX technology. We create a JavaScript function called “RefreshCalendar”, which will call the server side and expects a JSON document containing a list of events that are to be displayed. The method called is GetCalendarData from MyScheduleController. There we use Entity Framework to load all the classes and all the events of the user and pass them back in a JSON.

```

List<EventModel> data = this.LoadData();
result = this.Json(data, JsonRequestBehavior.AllowGet);

```

All the received data will be stored in a JS array named **events**. Once we process the received data, we are calling the GenerateCalendar from this array. We are doing this by calling the **fullCalendar** function from the FullCalendar JQuery library ^[15] and will create boxes for the given events.

All the output will be shown in the div which the id “calendar”

```

<div id="calendar"></div>

```

```

function RefreshCalendar() {
    events = [];
    $.ajax({
        type: "GET",
        url: "/MySchedule/GetCalendarData",
        success: function(data) {
            $.each(data,
                function(i, v) {
                    events.push(
                        {
                            eventID: v.IsClass||v.EventId,
                            title: v.EventTitle,
                            description: v.EventDescription,
                            start: moment(v.EventStartTime).format('YYYY-MM-DD HH:mm'),
                            end: moment(v.EventEndTime).format('YYYY-MM-DD HH:mm'),
                            backgroundColor: v.EventThemeColor,
                            allDay: v.IsFullDay,
                            isClass: v.IsClass!=null ? v.isClass : 0
                        });
                });
            GenerateCalendar(events);
        },
    },

```

Fig 14. Ajax function to refresh the calendar

```

function GenerateCalendar(events) {
    $('#calendar').fullCalendar('destroy');
    $('#calendar').fullCalendar({
        contentHeight: 400,
        defaultDate: new Date(),
        timeFormat: 'h(:mm)a',
        header: {
            left: 'prev, next, today',
            center: 'title',
            right: 'month, basicWeek, basicDay, agenda'
        },
        eventLimit: true,
        eventColor: '#378006',
        events: events,
        eventClick: function(calEvent, jsEvent, view) {
            selectedEvent = calEvent;
            $('#myModal #eventTitle').text(calEvent.EventTitle);
            var $description = $('<div/>');
            $description.append($('<p/>')
                .html('<b>Start: </b>' + calEvent.start.format("DD-MMM-YYYY HH:mm a"))));
            if (calEvent.end !== null) {
                $description.append($('<p/>')
                    .html('<b>End: </b>' + calEvent.end.format("DD-MMM-YYYY HH:mm a"))));
            }
            $description.append($('<p/>').html('<b>Description: </b>' + calEvent.description));
            $('#myModal #pDetails').empty().html($description);

            $('#myModal').modal();
        }
    });
}

```

Fig 15. Generate the calendar

We also implemented a modal to be displayed when an event is clicked to display information about room, teacher and class type. We also added edit and delete button on it, in order to perform these actions upon the selected event

```
<div id="myModal" class="modal fade" role="dialog">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal">&times;</button>
        <h4 class="modal-title"><span id="eventTitle"></span></h4>
      </div>
      <div class="modal-body">
        <button id="btnDelete" class="btn btn-default btn-sm pull-right">
          <span class="glyphicon glyphicon-remove"></span> Remove
        </button>
        <button id="btnEdit" class="btn btn-default btn-sm pull-right"
style="margin-right: 5px;">
          <span class="glyphicon glyphicon-pencil"></span> Edit
        </button>
        <p id="pDetails"></p>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default" data-
dismiss="modal">Close</button>
      </div>
    </div>
  </div>
</div>
```

When the student clicks on the edit button, another modal pops asking for the new data. If the event is a class, the user is being redirected to be page where he can see the alternatives. Once a different class is being selected, he is being redirected to the Feedback page, where there is a dropdown with possible answers. After the feedback is given, we are inserting the line in tblFeedback, storing the message, class id from and class id to.

```
var newFeedback = new tblFeedback
{
    fldClassIdFrom = feedback.FromClass,
    fldClassIdTo = feedback.ToClass,
    fldFeedbackMessage = feedback.ReasonOfChange
};

context.tblFeedbacks.Add(newFeedback);
context.SaveChanges();
```

Another page to which the student has access is the unallocated subjects. The controller for this action is ClassBacklogController. The controller is calculating the subjects that are already available in the profile of the user and it's building a list of SubjectModel, which is being displayed by the view. We have information about the

name of the subject, the target year and semester, the discipline file and a button to add backlog classes in your profile. If there is no available class in this semester and the user needs to wait until the next semester, the user receives a notification about that.

```
var allocatedSubjects = (from s in context.tblSubjects
    join c in context.tblClasses on s.fldSubjectID equals c.fldSubjectID
    join cl in context.tblClassLinks on c.fldClassID equals cl.fldClassID
    where cl.fldUserID == loggedUserId
    where cl.fldIsBacklog != 1
    select s.fldSubjectID).ToList();

var allSubjects = (from s in context.tblSubjects
    select s.fldSubjectID).ToList();

var unallocatedSubjects = new List<SubjectModel>();

var unallocatedSubjectsIDs = allSubjects.Except(allocatedSubjects).ToList();

foreach (var sub in unallocatedSubjectsIDs)
{
    var currentSubject = context.tblSubjects.First(s => s.fldSubjectID == sub);
    var auxSubject = new SubjectModel();
    auxSubject.fldSubjectID = currentSubject.fldSubjectID;
    auxSubject.fldSubjectSemester = currentSubject.fldSubjectSemester;
    auxSubject.fldSubjectName = currentSubject.fldSubjectName;
    auxSubject.fldSubjectYear = currentSubject.fldSubjectYear;
    auxSubject.fldSubjectFile = currentSubject.fldSubjectFile;
    auxSubject.fldSubjectIsFacultative = currentSubject.fldSubjectIsFacultative;
    auxSubject.fldSubjectFrequency = currentSubject.fldSubjectFrequency!=null?"1":"0";

    unallocatedSubjects.Add(auxSubject);
}

return View(unallocatedSubjects);
```

Fig. 16. ClassBacklog Controller

3.4.2 Admin role

This role can currently be granted only by a sys admin. An admin has access to the StudentListController, SubjectListController, TimesheetRefreshController and to the ReportingController.

In the StudentListController, the admin can see a list of all available users, can edit them, can delete them and can see their schedules. Also, an admin can add new users. If the admin initiates a user deletion, a confirm box is being displayed. Once confirmed, the user and all his information is being deleted. We are using the StudentDelete method for this action and we are checking if the user initiating this action is logged and is an admin:

```

0 references | 0 requests | 0 exceptions
public ActionResult StudentDelete(int id)
{
    int loggedUserID = 0;
    try
    {
        loggedUserID = (int)Session["userID"];
        using (var context = new TSMEntities())
        {
            var isAdmin = context.tblUserTypeLinks.FirstOrDefault(utl => utl.fldUserID == loggedUserID);
            if (isAdmin.fldUserTypeID != 2) //not admin
                return View("_AccessDenied");
            else
            {
                var studentToDelete = context.tblUsers.FirstOrDefault(u => u.fldUserID == id);
                var studentLinkToDelete = context.tblUserTypeLinks.FirstOrDefault(u => u.fldUserID == id);
                context.tblUserTypeLinks.Remove(studentLinkToDelete);
                context.tblUsers.Remove(studentToDelete);
                context.SaveChanges();
            }
            return RedirectToAction("Index", "StudentList");
        }
    }
    catch (Exception)
    {
        return View("_AccessDenied");
    }
}

```

Fig. 17 Delete a student account

The SubjectList controller is managing both subjects and classes. The admin can select which area to be accessed. The ManageClasses view will display a list of all the available classes, and the list of subjects are displayed by ManageSubjects view. The admin can modify details about any subject or class and even delete them. The deletion of a class will cause all the class links to be deleted as well, since a class link cannot exist without a class id due to the foreign key constraint. With Entity Framework, we don't have to worry about deleting each dependency. As long as our tables are well linked, it is enough to remove the desired entry and all the dependent data will be cleared as well. Also, when deleting a subject, all linked classes and class links will be removed as well, so this functionality should be used wisely

The admin is the one that can retrigger the timesheet version, via the TimesheetRefresh controller. The view represents just a big red button, which the admin can click. The command will be handled by the Refresh action handler from the controller, which will trigger the Parse Website project functionalities that we presented above. A new timesheet version will be created and the website of the faculty will be scrapped again for class information. The class links are not remade at this steps and the

parsing of the website is quite slow, so it's better to recreate the links on each profile when accessing the schedule, if new version is detected

```
0 references | 0 requests | 0 exceptions
public ActionResult Refresh()
{
    //await ParseWebsite.Program.RefreshData();
    try
    {
        using (Process myProcess = new Process())
        {
            myProcess.StartInfo.UseShellExecute = false;
            myProcess.StartInfo.FileName = "D:\\ParseWebsite\\ParseWebsite\\bin\\Debug\\ParseWebsite.exe";
            myProcess.StartInfo.CreateNoWindow = true;
            myProcess.Start();
            return RedirectToAction("Index", "AdminHome");
        }
    }
    catch (Exception)
    {
        return View("AccessDenied");
    }
}
```

Fig 18. Retrigger the timesheet refresh as admin

The last feature of the admin role is maybe the most interesting one. We are using HighChart JQuery library to create pie charts which could improve the student experience. We can extract for example the gender distribution of the students. The view for each report is being built by a JavaScript function which is using the HighChart library to display the data. In the function, we are mentioning the method which should provide input data. We provide the url `"/Reporting/GetGenderData"`, which means that we expect the data to be loaded from the Reporting Controller, from the `GetGenderData` method. This returns a JSON containing the value and name. Here we are using a model with 3 attributes: Male, Female and Others named `RatioGender`. In any report, we would need to have attributes for every option so that the percentages to reflect the reality.

```
public ActionResult GetGenderData()
{
    int male = context.tblUsers.Where(u => u.fldGender == "Male").Count();
    int female = context.tblUsers.Where(u => u.fldGender == "Female").Count();
    int others = context.tblUsers.Where(u => u.fldGender == "N/A").Count();

    RatioGender obj = new RatioGender();
    obj.Male = male;
    obj.Female = female;
    obj.Others = others;

    return Json(obj, JsonRequestBehavior.AllowGet);
}
```

4. Final conclusions

4.1 Conclusions

The Timesheet Management application has been successfully implemented, creating a very interesting experience both for admin and student roles. I believe that this product was developed to answer a need to have a platform where the students can provide feedback on their needs and the admins can extract reports and see where there is room for improvement. Also the fast pace of the modern society becomes pretty overwhelming, so besides learning new technologies and programming techniques, a future software engineer should get used starting from the faculty studies that a good time management can make a difference in productivity and level of achievements. With the implemented features, a student can learn the benefits of being organized, using learning techniques and practice task prioritization

4.2 Future directions

This application can be further improved. These are some directions of how it can get even better:

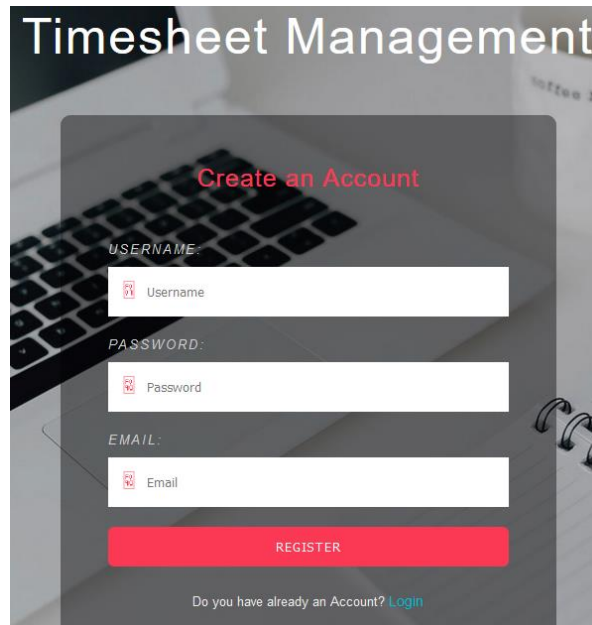
- The platform could be linked with the database of the faculty so that the new students could be added automatically;
- A mobile application could be developed in order to increase the usage of the schedule;
- A machine learning algorithm could be added in order to detect user's custom events and make suggestions of recurrent events;
- Another interesting feature would be to integrate voice assistants in order to create, edit, or delete events from the calendar with your voice;
- Learning techniques could be added on the page to help students optimize their learning routines;
- To have everything in one place, a communication feature could be implemented to replace the existing external groups.

5. Documentation

- [1] <https://www.managementstudyguide.com/time-management-benefits.htm>
- [2] <https://ecal.com/70-percent-of-adults-rely-on-digital-calendar/>
- [3] <https://techjury.net/stats-about/gmail-statistics/>
- [4] https://en.wikipedia.org/wiki/Microsoft_SQL_Server#T-SQL
- [5] https://www.tutorialspoint.com/entity_framework/entity_framework_overview.htm
- [6] https://www.w3schools.com/xml/xpath_syntax.asp
- [7] https://docs.pexip.com/admin/regex_reference.htm
- [8] <http://ecomputernotes.com/csharp/cs/features-of-c>
- [9] <https://developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html>
- [10] <https://www.geeksforgeeks.org/most-commonly-used-tags-in-html/>
- [11] https://en.wikipedia.org/wiki/Cascading_Style_Sheets#Syntax
- [12] <https://zellwk.com/blog/9-important-css-properties-you-must-know/>
- [13] https://www.w3schools.com/whatis/whatis_js.asp
- [14] https://www.w3schools.com/xml/ajax_intro.asp
- [15] <https://fullcalendar.io/>

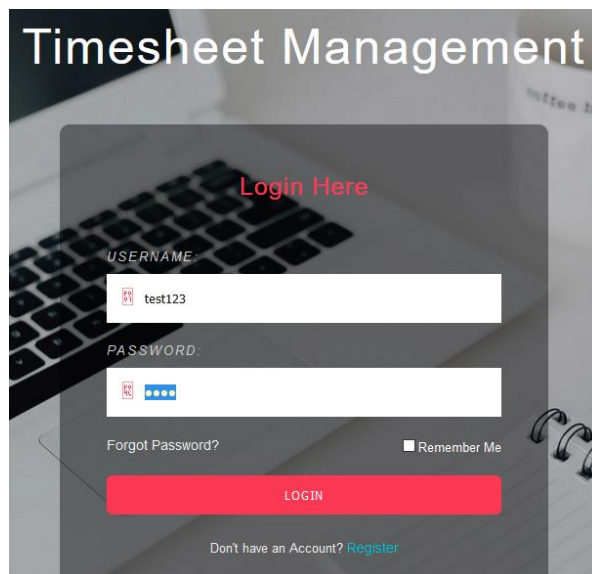
User Guide

1. In order to create an account, access the Register page and enter the username, password and email:



The screenshot shows the 'Timesheet Management' application interface. At the top, the title 'Timesheet Management' is displayed in a large, white, sans-serif font. Below the title, the heading 'Create an Account' is written in a smaller, red, sans-serif font. The registration form consists of three white input fields with red icons on the left: 'USERNAME:' with a person icon, 'PASSWORD:' with a key icon, and 'EMAIL:' with an envelope icon. Each field contains a placeholder text: 'Username', 'Password', and 'Email' respectively. Below these fields is a prominent red button with the text 'REGISTER' in white, uppercase letters. At the bottom of the form, there is a link that says 'Do you have already an Account? [Login](#)' in a small, blue, sans-serif font.

2. After the successful registration, you will be redirected to the login page:



The screenshot shows the 'Timesheet Management' application interface for the login page. At the top, the title 'Timesheet Management' is displayed in a large, white, sans-serif font. Below the title, the heading 'Login Here' is written in a smaller, red, sans-serif font. The login form consists of two white input fields with red icons on the left: 'USERNAME:' with a person icon and 'PASSWORD:' with a key icon. The username field contains the text 'test123' and the password field contains four blue dots. Below these fields, there is a link that says 'Forgot Password?' in a small, blue, sans-serif font, followed by a checkbox labeled 'Remember Me' and a red button with the text 'LOGIN' in white, uppercase letters. At the bottom of the form, there is a link that says 'Don't have an Account? [Register](#)' in a small, blue, sans-serif font.

3. After the login, since you have no info filled, you will be redirected to account page:

My Account

First name:

Last name:

Username:

Email:

Gender:

Date of birth:

Group:

Address:

Save

4. After the details are filled you can access your schedule. The classes will be organized in 4 categories: mandatory, optional, exams and others.
-

Calendar

Mandatory classes:

Name	Room	Day	From	To	Teacher	Type	Group	Change
Grafica pe calculator	C405	Marti	18:00	20:00	Lect.dr.GhirvuLucian	Laborator	I3B2	<div style="background-color: #4CAF50; color: white; padding: 2px 10px; display: inline-block;">Switch</div>
Calcul numeric	C112	Miercuri	12:00	14:00	Lect.dr.IgnatAnca	Curs	I3B	
Calcul numeric	C412	Miercuri	14:00	16:00	Lect.dr.IgnatAnca	Laborator	I3B2 - I3B7	<div style="background-color: #4CAF50; color: white; padding: 2px 10px; display: inline-block;">Switch</div>
Calcul numeric	C412	Miercuri	14:00	16:00	Lect.dr.IgnatAnca	Laborator	I3B2 - I3B7	<div style="background-color: #4CAF50; color: white; padding: 2px 10px; display: inline-block;">Switch</div>
Grafica pe calculator	C112	Joi	14:00	16:00	Lect.dr.GhirvuLucian	Curs	I3B	

5. If you want to change a class press on the switch button and all the alternatives will be displayed:

Name	Room	Day	From	To	Teacher	Type	Group	Switch
Sisteme de operare	C412	Luni	08:00	10:00	Colab.AioaneiDragos	Laborator	I1A7	Switch
Sisteme de operare	C411	Luni	12:00	14:00	Lect.dr.VidrascuCristian	Laborator	I1X1	Switch
Sisteme de operare	C403	Luni	18:00	20:00	Colab.AioaneiDragos	Laborator	I1B5	Switch
Sisteme de operare	C412	Luni	08:00	10:00	Colab.AioaneiDragos	Laborator	I1A7	Switch

6. Once you change the class, you will be asked for a feedback:

TSM
Home
My Schedule
My Account
Unallocated subjects
Log in
Log off

We would love to hear your feedback on this change 😊

Please choose an option
Please choose an option
I have a job and the hour is not convenient for me
I cannot wake up to early
I cannot focus this late
The new teacher is helping me learn more
The current teacher methods are not working for me
I heard it's easier to promote the new class
I heard it's very difficult to promote the old class
Others

© 2019 - Timesheet Management

7. From the My Schedule page, you have the option to see the visual calendar, by pressing the red button:

Calendar

Mandatory classes:

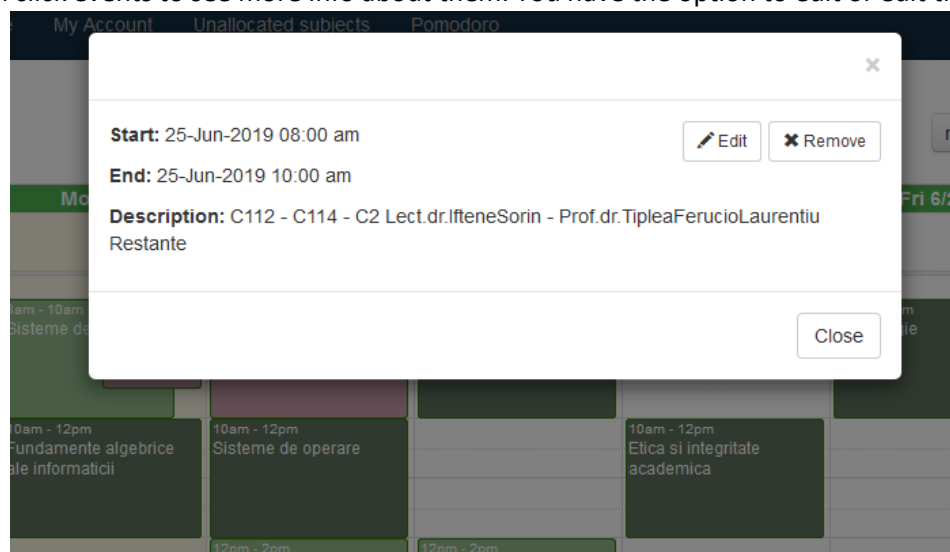
<
>
today

Jun 23 – 29, 2019

month
week
day
agenda

	Sun 6/23	Mon 6/24	Tue 6/25	Wed 6/26	Thu 6/27	Fri 6/28	Sat 6/29
all-day							
8am		8am - 10am Sisteme de proiectare algoritmi	8am - 10am Fundamente algebrice ale informatiei	8am - 10am Programare orientata-obiect		8am - 10am Pedagogie	
9am							
10am		10am - 12pm Fundamente algebrice ale informatiei	10am - 12pm Sisteme de operare		10am - 12pm Etica si integritate academica		
11am							
12pm			12pm - 2pm Fundamente algebrice ale informatiei	12pm - 2pm Sisteme de operare			
1pm							
2pm		2pm - 4pm Proiectarea algoritmi	2pm - 4pm Probabilitati si statistica	2pm - 4pm Limba engleza II			

8. You can click events to see more info about them. You have the option to edit or edit the events



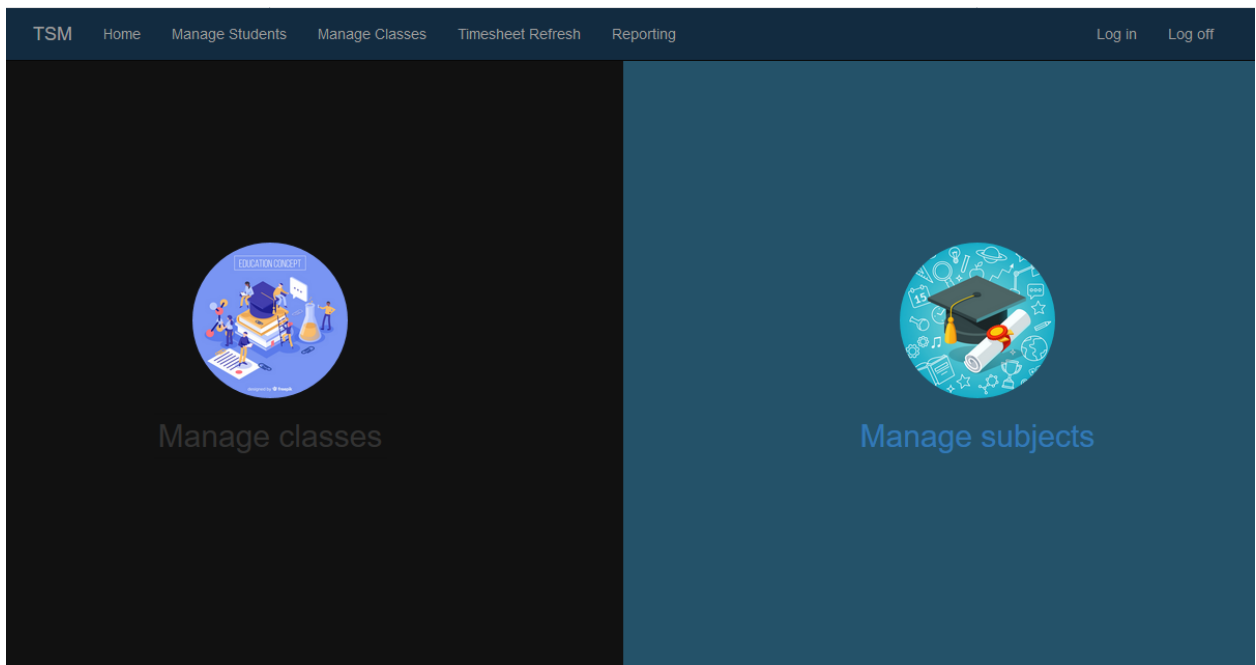
9. In case you cannot find a class, you can search for it in the “Unallocated subjects” tab.

Subject name	Year	Semester	Subject file	More
Structuri de date	1	1	Link	Pick
Logica pentru informatica	1	1	Link	Pick
Matematica	1	1	Link	Pick
Practica - Introducere in programare	1	1	Link	Pick
Limba engleza I	1	1	Link	Pick
Programare competitiva I	1	1		Pick
Educatie fizica	1	1	Link	Pick
Psihologia educatiei	1	1		Pick
Educatie fizica	1	2		Pick
Pedagogie I (Fundamentele pedagogiei + Teoria si metodologia curriculumului)	1	2		Pick

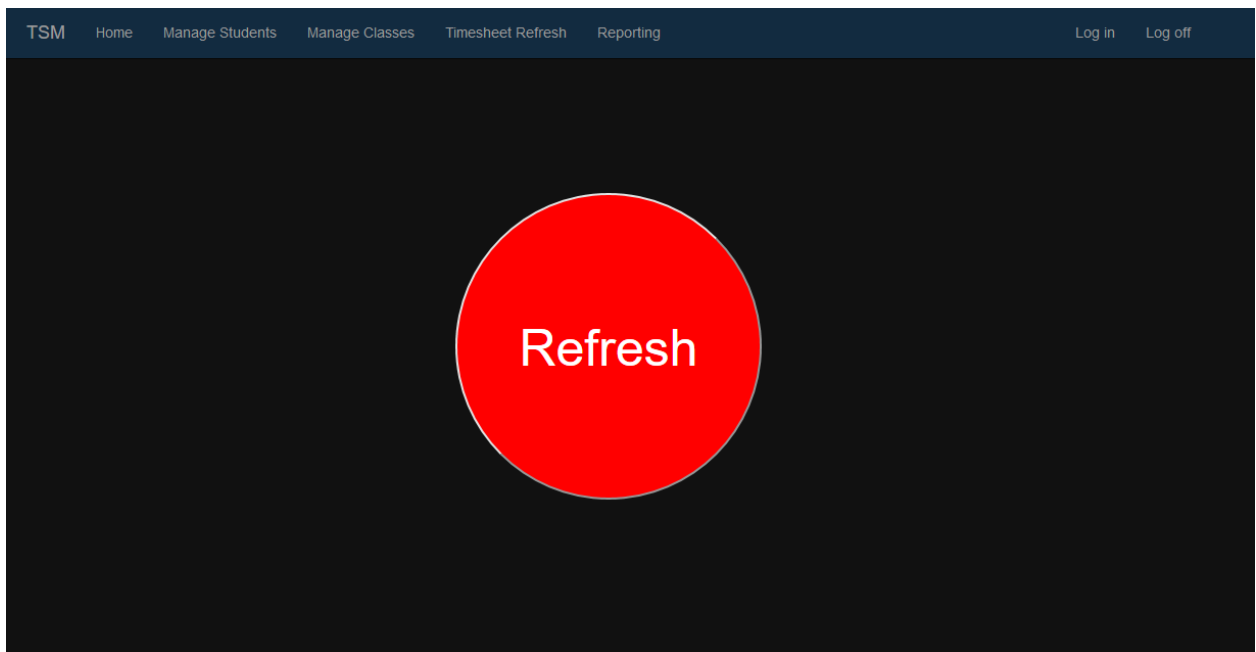
10. As an admin you have access to edit users from the “Manage students” tab:

+ Add new user				
Username	First name	Last name	Email	Actions
daniel	Daniela	Redinciuc	dsasdasd	Edit Details Delete
ana.maria	ana2	maria	ana.maria@gmail.com	Edit Details Delete
test	Test	test	test@yahoo.com	Edit Details Delete
test2	test2	test2	test2	Edit Details Delete
test123	Test	test	asdkhasd	Edit Details Delete

11. You also have access to manage subjects and classes:



12. In the “Timesheet Refresh” tab, the admin can trigger the synchronization with the website



13. In the “Reporting” tab, you have access to reports:

