University College of Northern Denmark

Technical report

Homework platform

Dmai0914 Group1

Cosmin Gabinat

Iliyan Stoev

Janis Indrans

Nikita Cimaskevic

Content

Introduction

In the 21st century the computer technologies have become available to a higher percentage of the world’s population. In more developed countries, especially in western countries, the personal computers and smartphones are available to mostly everyone, these technologies being part of their daily life. With such a high availability, computer technologies became an important tool of the community, especially in storage and administration of different types of data. Being widely used, different purposes and requirements appear, creating the need of different computer software to run on these machines and manipulate the data according to the requirements.

Schools are part of the institutions that deal with large amount of data. Thousands of students and employees create, submit and update new data to the school every day. This process can be problematic and time consuming. This is why lately, it became more and more common to find computer technologies implemented in schools. These technologies help in dealing with sorting, reading and storing the data. However, the use of computer technologies and computer software in schools has a very wide variety of purposes. One specific purpose of computer technologies in schools is represented by the use of online platforms required for submitting documents. This helps in saving both time and physical resources like paper. These are the reason that this type of online platform represents a valuable asset for any modern school nowadays.

Problem statement

The problem of delivering the homework in elementary schools can be sometimes a time consuming activity. The children may forget to bring it to school, and they parents cannot keep track of their children’s activity.

Considering the above mentioned information, this project is aiming on formulating and answering a problem statement that will improve the issue of computer technologies use in relation to the process of submitting homework in schools.

The problem statement is formulated as follows: “**How can schools avoid the daily time and resource consuming process of delivering homework, by using computer technologies? Could this be designed so it can be supported by different platforms and devices?”**

The idea is to design and develop an online platform that would allow pupils to submit their homework online. This platform could be used by elementary schools, for pupils between 10 and 16 years. Another target group is represented by teachers, as the platform is also aiming to allow them to create and upload homework for their pupils, as well as visualizing the homework that are already uploaded by them. The last target group is represented by the parents of the pupils that are studying in a school that uses this platform, allowing them to find and book tutors for their children.

This report is aiming to provide the technical documentation of a distributed system. This system is more exactly an online homework platform aiming to help schools ease the process of submitting homework, which is currently time and resource consuming for both students and teachers. By providing schools with such a system, we also help them implement a more sustainable policy, by reducing the amount of paper used in their process.

First of all, in what concerns the implementation of this platform in schools, the most important requirement is that all students and teachers have access to internet. This is why, for now, this homework platform is intended for schools in Denmark, where all students are guaranteed to have access to internet, that will allow the use of the platform.

Secondly, the main purpose of this online platform is to give students the ability to upload their homework as text files, on the school server. In the same time, teachers will be able to create and submit assignments on the server, from a win form client, in order to eliminate the waste of paper and time carrying this task during class time. Besides submitting assignments, teachers will also be able to download the homework submitted by the students. All the assignments and homework submitted to the database will be available for browsing in the future.

Furthermore, this platform can also be used for booking a tutor for the students. The tutors can fill in the dates there are available for tutoring, on their win form client, while the students will be able to book a tutor from the web client.

Domain model

Domain model for current project is not the most complex one as it consists only from 6 classes and one of them is a super class. After some drafts where made and discussed the final domain model where made. The following classes where designed: Person, Child, Teacher, Assignement, TutoringTime and Homework. The names for the fields were given in logical sense and the ones that might be confusing are explained.

Figure x:"Domain Model"

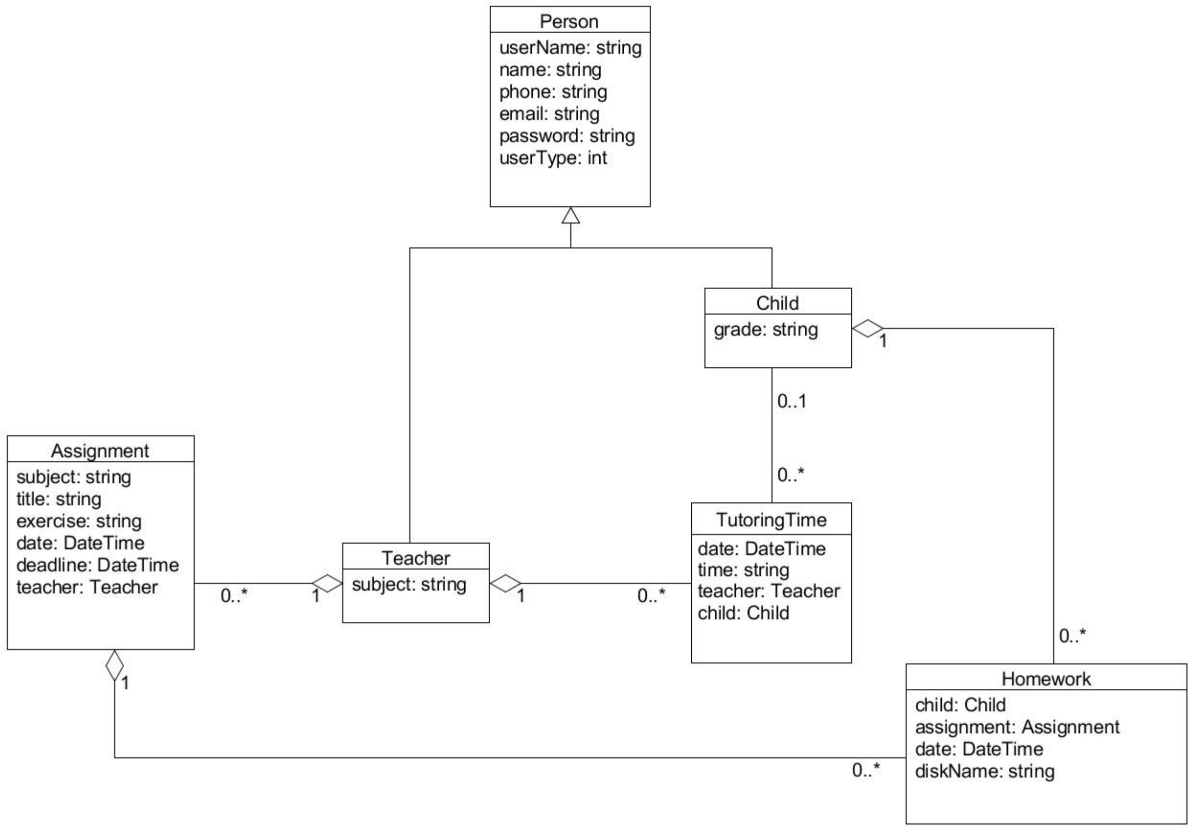


Figure 1 Domain model

Person: allows to create an object of type Person that has 6 fields to store its state. The names of the fields are: userName, name, phone, email, password and userType.

- userName: stores the login name of the current person so it can be used to log in to the system

- password: stores the password of the current person so it could be used to log in to the system
- userType: stores the number that represents type of the user so it could be used to separate functionality of the system between child and teacher

This class is a superclass for two subclasses – Child and Teacher.

Child and Teacher: allows to create an object of type Child / Teacher and has one field to store its state. As both of these classes inherits from class Person they share the same fields and functionality as Person. The name of the Child's field is grade and Teacher's field is subject.

- grade: stores the grade in which the child is studying in. For example - A1
- subject: stores the subject of the teacher which he/she is teaching to the kids. For example – Math

Assignment: allows to create an object of type Assignment and has 6 fields to store its state. The names of the fields are: subject, title, exercise, date, deadline and teacher. This object cannot exist without an object of type Teacher which is an aggregation. Every Assignment must have 1 Teacher.

- subject: stores the study subject of the teacher who makes the assignment
- title: stores the title of the assignment so it could be used to find the right assignment
- exercise: stores a detailed exercise text which is written by teacher so the child could read it and do his homework
- date: stores the date when the assignment where made
- deadline: stores the date of until when the homework must be handed in by uploading a file
- teacher: store an object of a type Teacher which made an assignment

Homework: allows to create an object of type Homework and has 4 fields to store its state. The names of the fields are: child, assignment, date and diskName. This object cannot exist without an object of type Assignment and Child. In order to create it the Child and the Assignment must exist at the same time. This is also an aggregation. Homework class was created in order to record the info of all the home works that is being uploaded by the children. Every Homework must have 1 Child and 1 Assignment.

- child: stores an object of a type Child who is uploading his homework
- assignment: stores an object of a type Assignment that the homework uploaded for
- date: stores the date of when the homework was uploaded
- diskName: stores the path of where the homework file is located, which was uploaded by the child

TutoringTime: allows to create an object of type TutoringTime and has 4 fields to store its state. The names of the fields are: date, time, teacher and child. TutoringTime object cannot exist without an object of type Teacher, which is another aggregation. The purpose of this class is to create available tutoring times of teacher which could be booked by a child. Every TutoringTime may have 0 or 1 Child and it must have 1 Teacher.

- date: stores the date when the tutoring time is available
- time: stores the time of the tutoring time when it is available
- teacher: stores an object of type Teacher who created the tutoring time
- child: stores an object of type Child who booked the tutoring time, but may be also null if no one booked it

All of the objects serves for CRUD (Create, Read, Update and Delete) operations. For this project not all of the operations are implemented at the current time, but might be done in future development.

Associations

Person class is generalized in two subclasses – Child and Teacher, but doesn't have any direct associations with other classes.

Child class is associated with classes TutoringTime and Homework. One Child can have 0 to many Homeworks and 0 to many TutoringTimes.

Teacher class is associated with classes TutoringTime and Assignment. One Teacher can have 0 to many TutoringTimes and 0 to many Assignments.

Assignment class is associated with classes Homework and Teacher. One Assignment can have 0 to many Homeworks but it has to have one and only one Teacher. Assignment stores all the information about Teacher.

Homework class is associated with classes Assignment and Child. One Homework has to have one and only one Child and one and only one Assignment. Homework stores all the information about Child and Assignment.

TutoringTime class is associated with classes Child and Teacher. One TutoringTime has to have one and only one Teacher but may have 0 or 1 Child. TutoringTime stores all the information about Child and Teacher.

Architecture

First step in creating software is define appropriate requirements, after this step we are ready for creating system architecture which is black box of the system and foundation based on functional and non-functional requirements which represents Quality attributes. Well-made and thought-out architecture structure ensures high performance, system stability, flexibility and maintenance.
The most important requirement that should be taken in consideration is the concurrency which makes the system multi-client system, which means that clients are able to interact with system through different devices. For instance one client communicates through internet browser, another client interacts with the Winform. There should be possible to add different clients like mobile client as well.

There is a lot of architecture patterns to choose from. Based on system requirements and learning plan main architecture pattern was chosen which is Distributed System Architecture. Distributed system gives unique opportunity to connect separate software and hardware components by network by passing messages and look as whole complete system to user.

It is a lot of advantages in taking distributed system as well as some obstacles which should be overwhelmed. The most important advantages are concurrency, scalability and loosely-coupled system.

The primary disadvantage is that all components communicate with each other through the network and it should be taken a lot of attention to create proper security in order to secure information from external threats.

Within Distributed System Architecture it also contains Client-Server architecture.

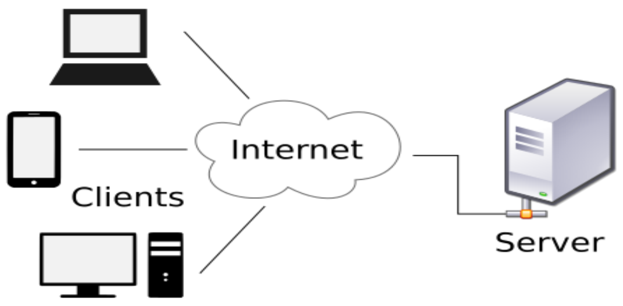


Figure 2 Client-Server

The Client-Server architecture provides the separation of tasks or workloads between the providers called servers and providers called clients by communicating through the network. To design client-server architecture properly it should be defined appropriate amount of logic to reside on client and server sides.

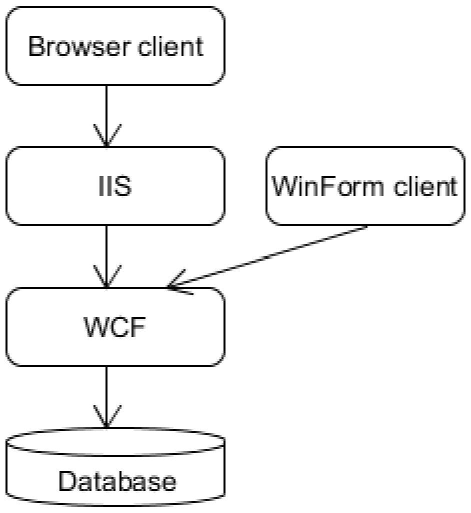


Figure 3 System architecture

System design

To see full convey of the system design, it is very important to describe each part of the designed system according to main architecture which was mentioned above.

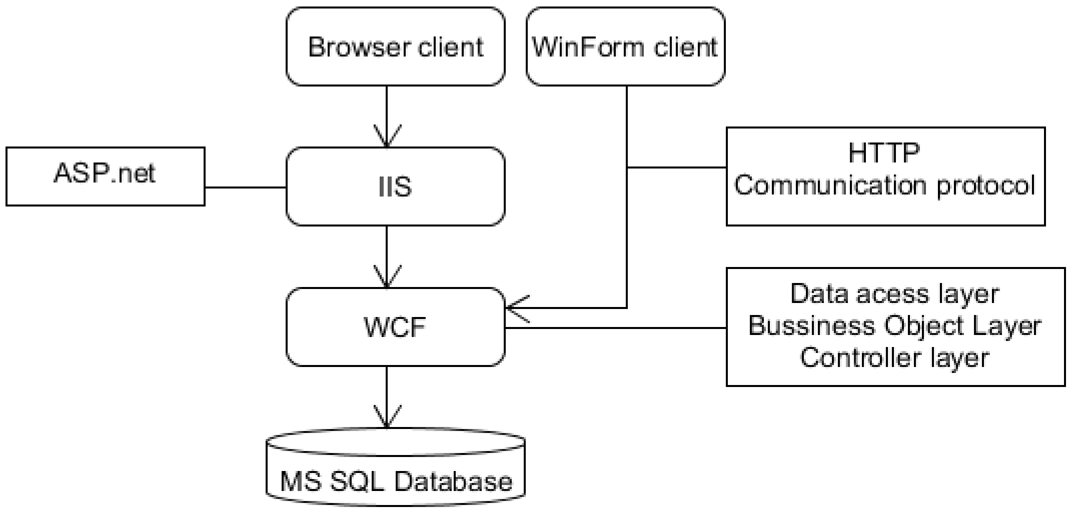


Figure 4 System design

Common description

For better grasp of Distributed System design and system components interaction, it should be mentioned common communication process. There are two different clients in current system WinForm and Web client which are intended to interact with WCF by passing messages with HTTP protocol. Web client sends messages to IIS, which connects to WCF services. The WinForm client connects directly to WCF service. WCF has connection to database and retrieves appropriate information from there and sends response back.

3-tier pattern

3-tier pattern is intended to segregate functionality into different parts, which are being located on physically separate computer or server. In current project all parts are located on LocalHost, but there is still possible to separate them and set up on different servers. This approach provides excellent maintainability because each tier is independent and changes in one do not affect system as a whole. Flexibility is the strong suit of 3-tier pattern as well, because each tier can be managed independently.

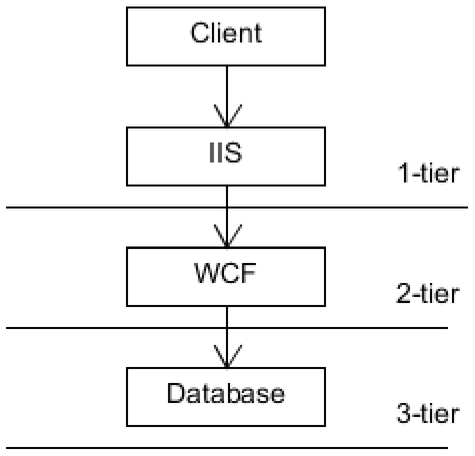


Figure 5 3-Tier Pattern

Internet information server

"The Web Server (IIS) role includes Internet Information Services (IIS) 7, which is a unified Web platform that integrates IIS, ASP.NET, Windows Communication Foundation, and Windows SharePoint Services."[1]

Database

For the purpose of storing and retrieving information MS SQL Database was chosen as a part of Microsoft technology stack.

Windows Communication Foundation (WCF)

"Windows Communication Foundation (WCF) is a framework for building service-oriented applications."[2]WCF is primary thing that makes the system Distributed. The major function is to send data from endpoint to another endpoint where endpoint can be client of a service that requests data from a service endpoint. The main advantage of WCF is loosely-coupled services, where each client can connect as long as contracts are met and does not depend on platform. In order to separate main functionality, so it might be good idea to apply "Layered Architectural

[1] https://technet.microsoft.com/en-au/library/cc753433(v=ws.10).aspx, 2015-12-12
[2] https://msdn.microsoft.com/ru-ru/library/ms731082(v=vs.110).aspx, 2015-12-12

Style” [3].The current WCF library are divided into 3 different layers in order to make high cohesion and separate common functionality between them.

Endpoint

“All communication with a Windows Communication Foundation (WCF) service occurs through the endpoints of the service. Endpoints provide clients access to the functionality offered by a WCF service.”[4]

Endpoint consists three main parts which are :

- the address
- the binding
- the contract

The address: the address shows to potential service consumer where that endpoint is located.

The binding: the binding specifies how to communicate with endpoint. It should include communication protocol, the encoding, security requirements.

The contract: the contract represents functionality provided by current endpoint.

Data Access Layer

Data access layer is separated segment of logic which deals with connection to database. In order to retrieve information Data Access Layer should consist classes with connection for every class from model layer. To create connections ADO.NET transactions were used.

Model Layer

Model layer represents the business entities which will be used as main classes for creation and manipulation objects.

[3] https://msdn.microsoft.com/en-us/library/ee658117.aspx, 2015-12-12
[4] https://msdn.microsoft.com/en-us/library/ms733107(v=vs.110).aspx, 2015-12-12

Controller layer

Control layer represents the business logic of the system. This is kind of a bridge between presentation layer and Data Access Layer where could be inserted some logic for manipulating information between storing or retrieving it from database.

Communication protocol s

The main issue which has been chasing developers and architects during the growth of computer industry was how to make all machines communicate between themselves, sending and receive messages. Moreover, there was a lot of different hardware, software and different systems. To sort out that kind of problems technicians come up with idea to create some standards which will define certain rules for communication between physical entities.

OSI model

OSI model is a conceptual model that characterizes and standardizes the communication functions of a telecommunication or computing system without regard to their underlying internal structure and technology.

OSI model consists 7 abstract communication layers where first level is lowest layer and seventh layer is highest.

- Application
- Presentation
- Session
- Transport
- Network
- Data link
- Physical

As software developers we are interested in first 4 layers and those one will be mentioned more precise.

Application

Application layers works with software and retrieves information from the user.

Presentation

Presentation layer convert data received from application layers to appropriate format which is acceptable for transferring data and vice versa.

Session

Session layer creates sessions between applications and keeps connection long time.

Transport

Transport layer is used to define rules for data transportation and keeping it save. Protocols described are TCP, UDP.

TCP/IP

TCP/IP is stack of protocols which broadly used in networks and internet. The name of this stack is defined from two main protocols TCP and IP. TCP/IP has 4 layers which completely covers all OSI model functionality even though does not use all 7 layers which are described in that model.

There are 4 layers in TCP/IP protocol stack:

- Application layer
- Transport layer
- Internet layer
- Link layer

Application layer

Application layer includes the protocols used by most applications which provide users data exchanging services through the network. For this project mainly was used HTTP protocol which provided connection in WWW.

Transport layer

Transport layer in TCP/IP can solve problem of non-guaranteed data transfer and assure right sequence of coming data. The transport protocols in TCP/IP stack define exactly the application information dedicated for.

There are two main protocols TCP and UDP. The TCP protocol provides reliable connection for application and always checks if transferred data was transported to receiver with the right sequence, request data in case of loss and exclude data duplication. The UPD is connectionless protocol which is unreliable and does not check if receiver received the data, mainly used for streaming audio and video.

In our developed system was used TCP for transferring data.

Internet layer

Internet layer is in charge for defining the host and sending packets through the internet. Address is specified by IP which has the hierarchy.

Link layer

Link layer describes how data is transferring on physical level. The main protocol is Ethernet, which is intended for define for what machines in the network belongs appropriate packet.

Concurrency

The biggest question when it comes to building a distributed system is: How do we handle multiple operations against the same server, and what if these operations was from multiple clients?

The best answer to the above questions should be – Transactions ("Call from a client to server, the server calculates or updates a resource"). Simple examples for a transaction could be: Bank Transfer, User creation, Booking.

In order for a transaction to ensure accuracy, completeness and data integrity there are some properties commonly known as "ACID properties" that should be in place.

Atomicity – The property states that a transaction should be "all or nothing", meaning that all the operations are executed or none.

Consistency – Assumes that the operation will bring the database from one consistent state to another. All data written must be valid according to all defined rules(constrains) (Brian's Slides Session 9).

Isolation - In a database system where more than one transaction is being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system. No transaction will affect the existence of any other transaction.

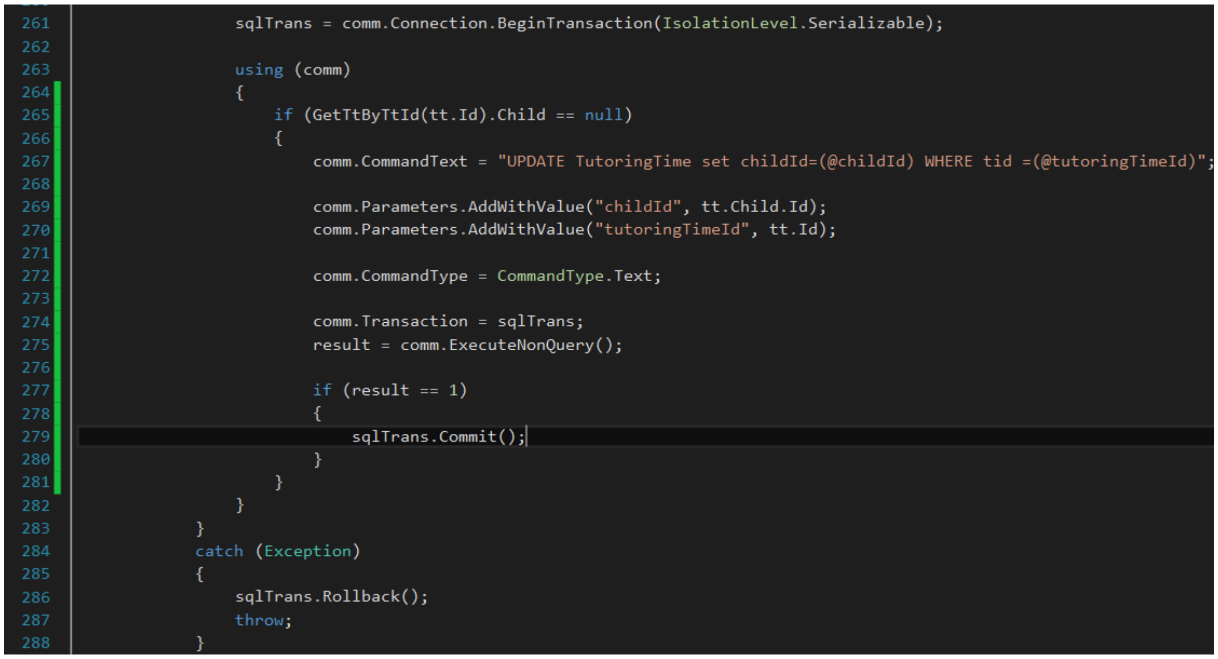
Durability - Means that once a transaction has been committed, it will remain so, even in the event of power loss, crashes, or error.

Since the subject of talking in this chapter is Concurrency, the attention should be on the Isolation property of Transactions. There are two approaches for handling Concurrency in a distributed system: Optimistic (Optimistic concurrency control works on the assumption that resource conflicts between multiple users are unlikely (but not impossible), and allows transactions to execute without locking any resources. Only when attempting to change data are resources checked to determine if any conflicts have occurred. If a conflict occurs, the application must read the data and attempt the change again) and Pessimistic (Pessimistic concurrency control locks resources as they are required, for the duration of a transaction. Unless deadlocks occur, a transaction is assured of successful completion).

In the following project the problem where multiple users want to access and/or change the same resource at the same time, is with the Booking Tutor functionality. For this particular part of the program, Pessimistic Concurrency Control has been chosen, since there is a risk that multiple transactions can interfere with each other.

Implementation:

The following block of code shows the Data Access Layer implementation of RegisterBooking and how the transaction has been implemented and how the Isolation Level has been set:

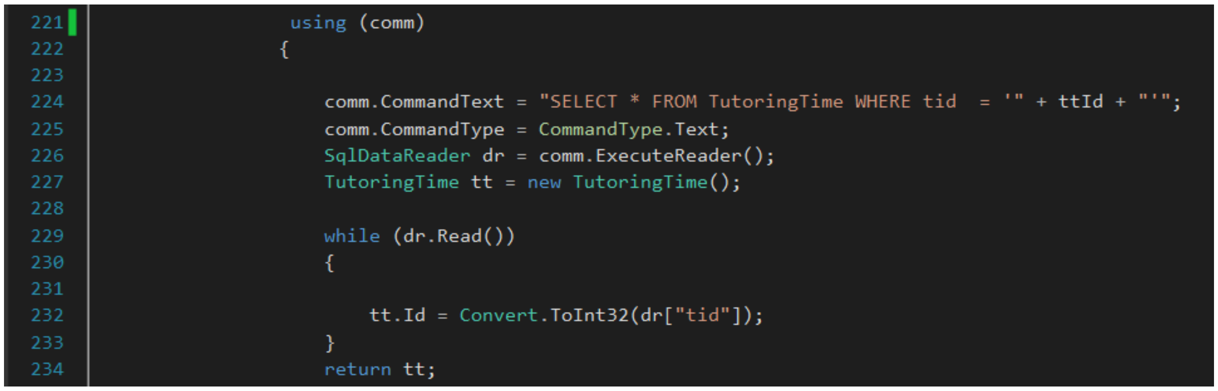


```
261         sqlTrans = comm.Connection.BeginTransaction(IsolationLevel.Serializable);
262 
263         using (comm)
264         {
265             if (GetTtByTtId(tt.Id).Child == null)
266             {
267                 comm.CommandText = "UPDATE TutoringTime set childId=(@childId) WHERE tid =(@tutoringTimeId)";
268 
269                 comm.Parameters.AddWithValue("childId", tt.Child.Id);
270                 comm.Parameters.AddWithValue("tutoringTimeId", tt.Id);
271 
272                 comm.CommandType = CommandType.Text;
273 
274                 comm.Transaction = sqlTrans;
275                 result = comm.ExecuteNonQuery();
276 
277                 if (result == 1)
278                 {
279                     sqlTrans.Commit();
280                 }
281             }
282         }
283     }
284     catch (Exception)
285     {
286         sqlTrans.Rollback();
287         throw;
288     }
```

Figure 6 Isolation level in DataAccessLayer

In line 261 the BeginTransaction method has been called which takes a parameter of IsolationLevel set to Serializable, in that way the database has been locked when the method call has been performed.

Next a SELECT statement has been performed via GetTrByTtId method



```
221 using (comm)
222 {
223 
224     comm.CommandText = "SELECT * FROM TutoringTime WHERE tid  = '" + ttId + "'";
225     comm.CommandType = CommandType.Text;
226     SqlDataReader dr = comm.ExecuteReader();
227     TutoringTime tt = new TutoringTime();
228 
229     while (dr.Read())
230     {
231 
232         tt.Id = Convert.ToInt32(dr["tid"]);
233     }
234     return tt;
```

Figure 7 Isolation level in DataAccessLayer

It is needed because of the need to check if the desired Tutoring Time has already been booked. This is done by checking if the Child object is null, if it is null then the desired Tutoring Time is still available and the code continues after it, else the Transaction has been rolled back Line: 286.

If the above check has passed successfully, the ExecuteNonQuery method is called (Line: 275) and stored into the result field which will return the number of rows affected after the execution of the statement. Finally, another check is performed, if there is at least 1 row affected, the Transaction commit method is called, else again the Transaction has been rolled back.

The implementation of this functionality can assure that whenever two or more users want to Book the same Tutor in the same Time and Date, the fastest will win.

Implementation

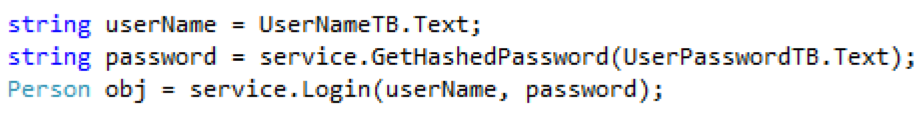
This chapter will only cover the implementation of a few of the functionalities of "Homework OnP". These functionalities have been chosen according to the business value of their respective user story.

Login functionality

First of all it has to be mentioned that the login functionality has not been implemented directly from the template, but instead created manually.

Keeping in mind that this platform uses a WCF service, there is a set of methods and calls made from the client to the service and through all the layers in the service. Also it is important to mention that the login functionality is implemented on two different clients, a web client and a win form client. Each of these clients works with a different type of objects. As such, only Child objects are able to log in and use the web client interface, while the win form allows only Teacher objects to log in. However, a single Login method is used for both cases.

The username and password are passed from the client to the service calling the Login(string username, string password) method. At this point, the password is already hashed by calling a method from the service and returning the hashed value of the UserPasswordTB input.



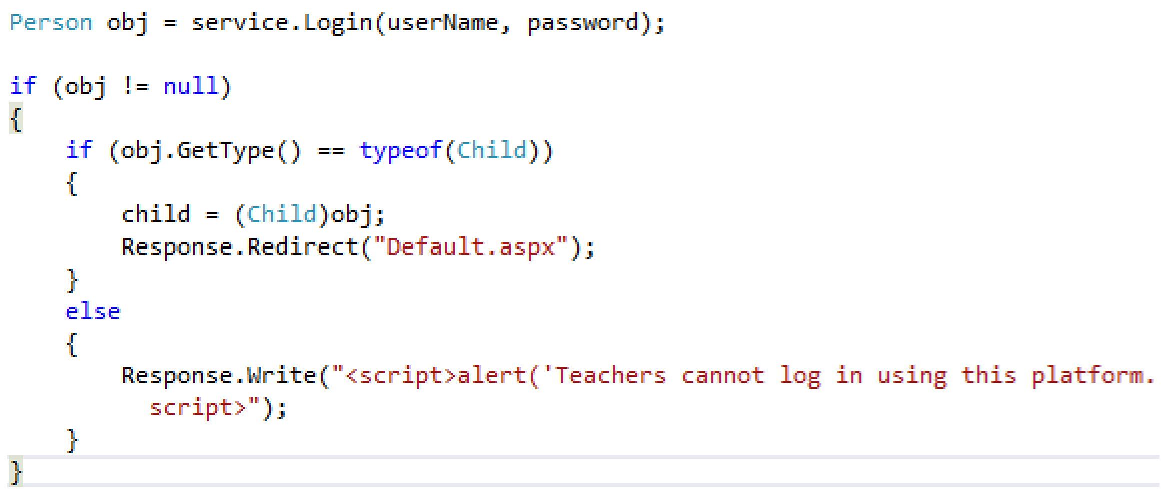
```
string userName = UserNameTB.Text;
string password = service.GetHashedPassword(UserPasswordTB.Text);
Person obj = service.Login(userName, password);
```

Figure 8 WebClient hashing password

The actual hashing of the password happens in the PassHash class from the BLL. In this class there is a method that takes a string parameter and returns the hashed value of this string. The hashing is made using the SHA256CryptoServiceProvider class. A previous version of the platform was using the GetHashCode() method, available for a string object. This has been replaced due to its lack of security.

Following, the service calls the UserCtrl in the Business Logic Layer (BLL) where a Person object is built with the two parameters. This object is further passed to the Data Access Layer (DAL) which queries the database for a user with the specified username and password. If there is any match, a Child or Teacher object is build, according to the user type value, and it is returned through layers all the way to the client, as an Person object.

After returning the object from the database, each client has the duty of checking wheatear the returned object is corresponding to the type allowed in the client, respectively Teacher for the win form client, and Child for the web client. This takes place using the typeOf() method. In this way the platform ensures that no Child will be able to log in on a win form client, and vice versa.



```
Person obj = service.Login(userName, password);

if (obj != null)
{
    if (obj.GetType() == typeof(Child))
    {
        child = (Child)obj;
        Response.Redirect("Default.aspx");
    }
    else
    {
        Response.Write("<script>alert('Teachers cannot log in using this platform.
          script>");
    }
}
```

Figure 9 - WebClient - object type check

A previous version of the login functionality included two login methods in the service, one for each client. An Object type was returned from the lower layers, and it was casted in each login method, to the corresponding type, before returning it to the client. In this way the service was carrying all the work in connection to the login functionality.

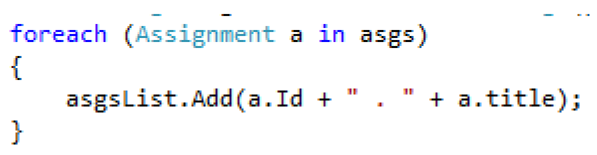
Submit homework functionality

It represents the main functionality of this online platform, even though the login functionality has been covered first. This is because users were required to be logged in before being able to submit their homework.

Similarly to the previous functionality, the submit homework process goes through the same layers. Even though the upload of the document happens on the client side, a Homework object is created in the BLL and is added to the database, containing the title, exercise, date, the disk name of the uploaded document, and the assignment for which it was submitted.

In this case, the SubmitHomework method in DAL returns an integer that specifies if any row has been changed in the database. This integer is returned all the way to the web client, and according to it a message is displayed to the user confirming of denying the upload of the document.

An important thing to explain here is how the assignmentId is used. In order to create an assignment, the platform offers a combo box where the user can select the assignment he/she wishes to upload homework for. Even though a list of Assignment objects is returned from the service, the combo box is populated with an array of strings formed by concatenating the assignmentId with the name of the assignment.



```
foreach (Assignment a in asgs)
{
    asgsList.Add(a.Id + " . " + a.title);
}
```

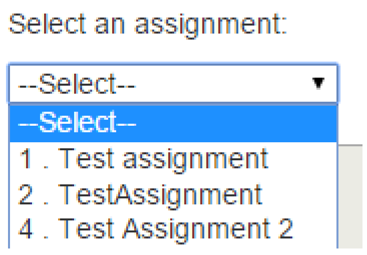
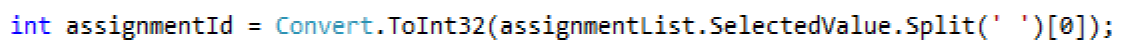


Figure 10 - Populating list of strings with assignment id and name

Having the combo box populated with strings, the selected index is used to extract the Assignment object from the list of assignments retrieved earlier from the service. This object is then used on writing the title and exercise of the assignment in a text box, so that the user can see the exercise he selected.

However, the assignment id sent to the service for creating homework object is extracted differently. The selected value from the combo box is broken into an array of string using the Split(‘ ‘) method. In this way considering Figure 3, the first string will always be the id of the assignment.



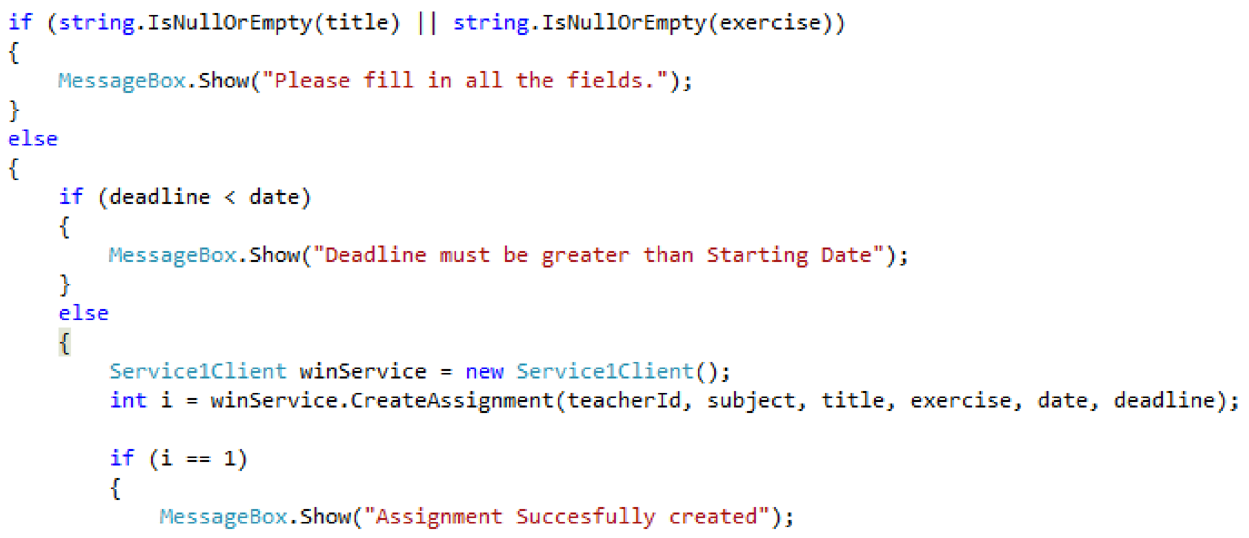
```
int assignmentId = Convert.ToInt32(assignmentList.SelectedValue.Split(' ')[0]);
```

Figure 11 - Retrieving assignmentId from combo box

Create assignment functionality

This method can only be accessed from the win form client, the functionality being available only for the teachers. The assignment is created in the same way as the homework, passing all the parameters from the win form client (title, date, deadline, exercise and the teacher id), all the way to the AssignmentCtrl in BLL, where an Assignment object is built, and in DAL its information is stored in the database.

A set of checks have been created in the win form client, in order to make sure that the user does not create an invalid assignment. In the figure below it can be seen how first the title and exercise fields are checked to make sure they are not null. If these checks pass, the dates are also checked to make sure the deadline is not set before the starting date of the assignment, fact which would make the assignment invalid. Only if all these checks pass, the values will be forwarded to the service, calling the "CreateAssignment" method. The returned integer will be checked and according to it, a message will be displayed to the user.



```
if (string.IsNullOrEmpty(title) || string.IsNullOrEmpty(exercise))
{
    MessageBox.Show("Please fill in all the fields.");
}
else
{
    if (deadline < date)
    {
        MessageBox.Show("Deadline must be greater than Starting Date");
    }
    else
    {
        Service1Client winService = new Service1Client();
        int i = winService.CreateAssignment(teacherId, subject, title, exercise, date, deadline);

        if (i == 1)
        {
            MessageBox.Show("Assignment Succesfully created");
```

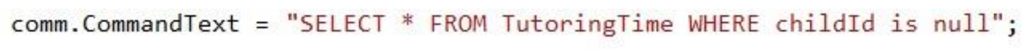
Figure 12 – Create assignment checks in win form client

Book tutor functionality

This functionality is one of the important functionalities for current system as it shows the knowledge of concurrency. This functionality enables children parents to book the tutor which could help to improve their child’s knowledge in a specific subject. It uses 3 main methods for the transaction to be successful. One of these methods registers Booking, anothther checks if TutoringTime is still available and the other retrieves all the available tutoring times from the database.

User interface in this situation is a web browser which accesses the web server (IIS), which is called WebClient in the current system, and retrieves the information to display in browser. The server then communicates and calls the methods from WCF server which then goes through the three layer architecture from controller to database. Both methods has a returns.

One of the method is called “GetAllAvailableTutoringTimes()” and maintains the same name in every layer. This method is used to get all the available TutoringTime objects. This is done by defining query which will return all the TuturingTime objects where the childId is NULL.



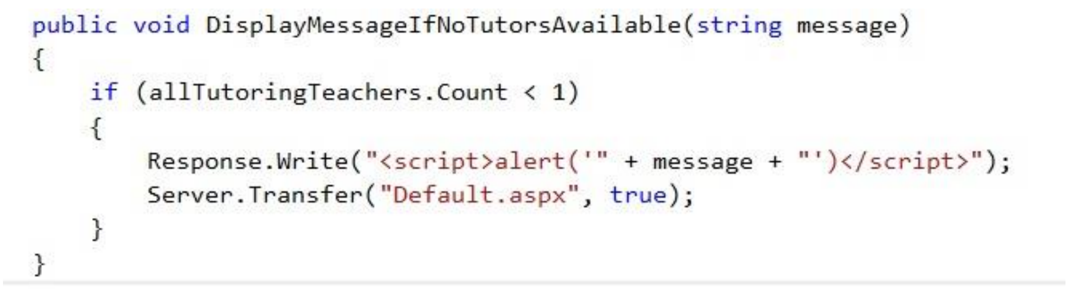
```
comm.CommandText = "SELECT * FROM TutoringTime WHERE childId is null";
```

Figure 13 query in TutoringTimeDb

This query allows to get all the TutoringTime objects which are available for booking. If the childId is null it means that it’s available. If there is a childId it means it is taken. When list of objects is returned to the WebClient it is of type Array so it’s been converted in to the list again by adding the following expression: “.OfType<TutoringTime>().ToList()”.This will Ensure that the Array will be changed to list. This list than is saved in global variable to be used in sorting operations. There are 2 more global variables that stores the information sorted out from the available tutoring time list. One list stores the available teachers and the other stores the available teacher subjects. This is done to make sure that the teacher names and the subjects in drop down lists doesn’t repeat as the teacher may have multiple tutoring times (appointments). Every time the page is loaded or the post back is triggered it updates the available tutoring time list.

Drop down lists are populated based on the available tutoring time list so that if the teacher has no tutoring times available it won’t be showed in the dropdown list. Same with the subject. The sorting method is called: SetAllTutorsAndSubjects(List<TutoringTime> list). It takes as a parameter the list of tutoring times, makes some checks on it, including the date check so it does not display the past days and adds the sorted objects to the lists. When this is done the lists are used to populate the drop down lists.

These lists are also used to render the calendar and set design as it was planned by enabling only the dates that has available TutotingTime. Also it paints the calendar cells in green to visually provide the information about the available TutoringTimes. Information in the calendar is changed every time when drop down list index changes. In situations when the changes are made by another user the pop-up messages are displayed to notify the user if someone already booked the teacher and there system can't find the required data by using if statements. This prevents system from crashing and misunderstandings of user. The special method for displaying and transferring user to default page was made.

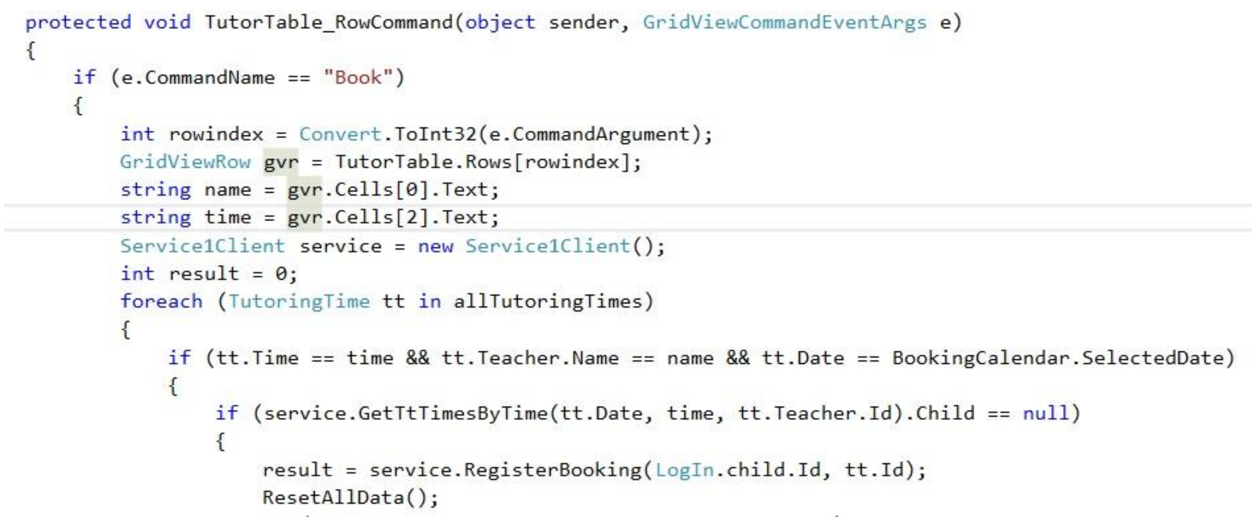


```
public void DisplayMessageIfNoTutorsAvailable(string message)
{
    if (allTutoringTeachers.Count < 1)
    {
        Response.Write("<script>alert('" + message + "')</script>");
        Server.Transfer("Default.aspx", true);
    }
}
```

Figure 14 method for error handling

It takes a parameter of string to allow to customize the message for any situation. This message is usually used when there are no tutoring times available at all. Also there are some simple messages to notify user of any change. When the day is chosen the grid view, which is used as a table, is populated with available tutoring times based on the selected values of drop down list.

In grid view there is a button column that when it is pressed gets the grid view row and that has the button and returns the information of the row.



```
protected void TutorTable_RowCommand(object sender, GridViewCommandEventArgs e)
{
    if (e.CommandName == "Book")
    {
        int rowindex = Convert.ToInt32(e.CommandArgument);
        GridViewRow gvr = TutorTable.Rows[rowindex];
        string name = gvr.Cells[0].Text;
        string time = gvr.Cells[2].Text;
        Service1Client service = new Service1Client();
        int result = 0;
        foreach (TutoringTime tt in allTutoringTimes)
        {
            if (tt.Time == time && tt.Teacher.Name == name && tt.Date == BookingCalendar.SelectedDate)
            {
                if (service.GetTtTimesByTime(tt.Date, time, tt.Teacher.Id).Child == null)
                {
                    result = service.RegisterBooking(LogIn.child.Id, tt.Id);
                    ResetAllData();
```

Figure 15 booking tutor

Then it finds the tutoring time chosen by using foreach loop and if statement, calls the method from service to check if the tutoring time is still available and then calls another method, RegisterBooking(LogIn.child.Id, tt.id), from service to register booking by passing child id and tutoring time id. This method returns value from data base about if the transaction was successful to check and display an error message if it is required. Also there are checks before the booking registration that returns error messages. Register booking method is where the concurrency appears. In TutoringTimeDb this method uses IsolationLevel.Serializable, which makes sure that only one user at the time can access the method. The others are placed in queue. This is the highest isolation level.

Conclusion

The main task during the long time period of working, programming and designing was to answer the question of proposed problem statement: "**How can schools avoid the daily time and resource consuming process of delivering homework, by using computer technologies?"** The answer was found by creating and applying new Homework platform, which was made in time with the right choice of system design, architecture and system development approach. Now students can check from wherever they want their assignments uploaded by teachers and give fast response to the teachers by uploading homework.

The right choice of design contributed to solve and answer another question of problem statement: “**Could this be designed so it can be supported by different platforms and devices?**” To solve this problem was used WCF framework, which made possible to connect different devices and platform as long as they meet the appropriate standards. In future versions it is possible to add more platforms easily by adding additional functionality to WCF.

Current project was intended to give some insights about designing and setting up distributed system, the main goal of the outcome was to gain knowledge about how to overcome obstacles in such complicated systems. According to the final results it should be said that expected outcome is achieved. Developer team has fully implemented Distributed System, created different clients according to the requirements, made and explained concurrency considerations and finally gained knowledge about protocols and how to apply them in real projects.

References

1. https://technet.microsoft.com/en-au/library/cc753433(v=ws.10).aspx, 2015-12-12
2. https://msdn.microsoft.com/ru-ru/library/ms731082(v=vs.110).aspx, 2015-12-12
3. https://msdn.microsoft.com/en-us/library/ee658117.aspx, 2015-12-12
4. https://msdn.microsoft.com/en-us/library/ms733107(v=vs.110).aspx, 2015-12-12
5. Repository link : https://github.com/IliyanStoev/dmai0914_Sem3_Gr1.git