Club Manager

Management Application

GROUP MEMBERS

IVO VASILEV NEKOV ivox0043@edu.eal.dk

BOGDAN IULIAN bogdan.Iulian.v@gmail.com

CUSTOMER: JIMMI BUSK Facebook @Jimmi Sit Busk @White Tiger Muay Thai Odense

Contents

Project expectations and organization

Our team consists of two members - Bogdan and Ivo.

We will use different set of tools through the whole work process.

We will approach the project by using the Agile methodology that help us split the work into small tasks that can be tackled easier. We will be using Trello as Scrum board. We have chosen the platform because it has all the functions that we need and is familiar to us from the first semester project.

In accordance to the Agile methodology, the work will be divided into sprints. At the end of each sprint we will deliver pieces of product for which we will ask for feedback. The role of Scrum Master will be passed between both of us.

Through the whole process, we will be working together but from time to time there will be single person tasks. To be able to share the documentation and keep track of progress our main storage platform will be Google Drive. As a feature of Trello, Google drive can be linked to it.

For artifact creation, we are using UML tools – UMLET and draw.io (which is implemented in Google Drive).

Our code is being uploaded on GitHub, so both member can commit changes to it.

Learning Goals

To evaluate our project quality and our personal progress through the process, we have agreed upon several learning goals. They will be our main pointers if we have fulfilled them.

Knowledge (Programming)

The student has knowledge about:

1Pv1. Specifications of abstract data types How data is represented in a proper way, e.g. repositories.

1Pv2. Criteria for program quality E.g. GRASP, SOLID, code standards, naming conventions etc.

1Pv3. Abstraction mechanisms in modern programming languages E.g. Interfaces, classes, inheritance, delegates/events…

Knowledge (Technology)

The student has knowledge about:

1Tv2. The facilities and functions of modern database systems

Skills (Programming)

The student can:

1Pf1. Specify and construct algorithms

1Pf2. Use programming language to realize algorithms, design patterns, abstract data types, data structures, design models and user interfaces

1Pf3. Use a modern integrated development tool, including version control systems E.g. C#, SQL, Visual Studio, GitHub

1Pf4. Realize models in a database system and construct programs which use a database interface

1Pf6. Design applications based on a layered software architecture E.g. 3-layered architecture

1Pf7. Use software components/libraries E.g. API to connect to Excel-spreadsheet, Database, testing-tools

1Pf8. Prepare documentation in relation to valid de-facto standards in the field E.g. Sequence Diagrams, Design Class Diagrams, Operations Contracts, test-harness, code comments etc.

1Pf9. Use up-to-date techniques and tools for testing and quality assurance E.g. Test-Driven Development, Module testing, integration testing, reviews, software design patterns, GRASP / SOLID etc.

1Pf10. Evaluate qualitative and quantitative properties of algorithms and data structures E.g. considerations concerning properties regarding time, space and scalability

Skills (Systems Development)

The student can:

1Sf1. Model and design It systems

1Sf2. Use an appropriate software architecture

1Sf5. Use appropriate design patterns

Competences (Programming)

The student can:

1Pk1. Participate as a professional programmer in development and maintenance projects

1 Pk2. Keep up to date with current programming languages, development tools, programming technology and program design

Group Contract

To have better organization and work ethics we have agreed upon setting some rules that will prevent conflicts and misunderstanding. That is our group contract.

§ Compulsory attendance. Call/text cancellation if you are unable to attend, or call/text if you are delayed.
§ Tasks would be spread evenly through both members.
§ We must keep appointments and times and be honest about if we discover that one of us has problems complying with an agreement, so we can take it into account.
§ We write decision log every time we are together.
§ Group logbook is used to nuggets and decision record.
§ We help each other professionally and show understanding for each other.
§ We provide and use constructive and specific criticism - be solution-oriented.
§ Work focused with good / high work ethic. We do our best!
§ Good cooperation: flexibility, mutual respect, structured and constructive.
§ We will use our classmates and teacher when in doubt and ask for help in the process.

We have started out by gathering the requirements from our product owner. We have analyzed the problem and found a solution. The main thing here is to find out the purpose of the application, the actions that it should perform and identify the stakeholders. This step was to create the Business model/Business case.

The Club

The Muay Thai Club “White Tiger” is a private business. The club is located at “Carlsgade 4”, Odense C. The gym is run by the owner Jimmy Busk and his partner Eva Schulz. Jimmi just became the World Fightsport & Martial Arts Council representative of Denmark and his responsibilities largely grew.
His business is providing martial arts activity for a wide range of students, from kids and women to amateur and professional fighters.

Existing system

The current process involves manual work mainly, which means that there is no established software that can be used to fulfil our PO's everyday duties. All the details about the club's members are remembered by heart, the tournaments and events are scheduled manually using pen and paper.This can become a problem on the long run because the documents can be lost or the information can change anytime. The fee payment represents a big problem for the PO because he is forced to remember all the dates when the members paid their fees.

Benefits and limitations:

There are different benefits that our project will bring to the business, the solution is radically changing the way of working.

Because the number of members that attend the classes is growing and the product owner has more responsibility coming with his new position, the need of improving the quality of his work is crucial. He knows that it will become harder and harder to remember everyone, so he needs a smart and fast solution to do this job for him. The current members are developing their skills from day to day and he expects more and more students to desire to attend competitions, this will bring extra work for him to remember all the information when he's signing them up for tournaments. On top of that, his new position requires him to set up competitions and promote events, so having a way of storing information about these events will be necessary.

The limitation that occurs for the moment will be the use of specific technology to make it work, such as a windows-running computer and access to a database.

Option selection:

From the beginning of the project we had a clear idea about how are we going to solve the problem. Because we were familiar with his needs before starting the project, we have gathered information about what he needs in advance, the effect of this situation being the fact that we found a solution very fast. We knew that the best solution for us is to use a database to store the data and we built upon this idea. After a long discussion in which we polished our solution, we presented it to our product owner and he agreed upon everything we suggested. In the end, we had the green light for our solution, a desktop application that will allow him to store all the information he needs about his fighters and tournaments, creating new events and sending email notification to his students for different purposes. All the data will be stored in a database because of its consistency and durability.

Risk assessment:

Because the solution is not making use of any third-party resources except the database, the risks are very reduced. With all that said, the risk that is present now is that the product owner will have to pay to use a database. We informed him about the situation and he agreed to go all the way with our solution and to pay for their services. Even so, the risk of changing his mind still exists and because of that we will deal with this problem by setting up a local database if this problem will occur.

Project approach:

To tackle the project in an effective way, we will use the Agile approach to deliver constant results that can be evaluated for feedback by the product owner and other external actors. For the product itself, we will use the combination of C# programming language that we are familiar with along with Visual Studio and for the database management we will use the SQL language along with Microsoft SQL Server Management System.

For each artifact in our report we have used criteria.
They show us if we have fulfilled the requirements and give us more knowledge of the structure and representation of each one of the artifacts.

Business model

Key Partners

What are our Key Partners?	The key partner will be the owner of the club/PO and members of the club.
Which key resources are we acquiring from partners?	The Key resource that we acquire from is our PO. He has broad knowledge of his current working process.
Which key activities do our partners perform?	The key activities that our partners perform will be solely the communication with us.

Key Activities

The Key Activities that our Value proposition require are:
Production

- Creating a working software using C#, SQL, UML and Agile methodology.

Problem solving

- Communication with PO
- Planning Sprints
- Making documentation
- Manual (see appendix)

Key Resources

The key resources we need to finalize the projects are:
Intellectual

- Knowledge of programming languages
- Knowledge of business process
- Management strategies

Human

- Teamwork
- Communication with PO

Value proposition

Our value proposition is a software which can store club member information (personal details, sports career details, payment fee). The software will be able to store event data (tournaments, seminars, other) and provide a notification system that will be used for fee deadline or upcoming events. The app will try to ease the manual labor and make the process organized.

We will be delivering a product which focuses on:

- Usability - simple wireframe/UI
- Performance - quick and reliable
- Free of charge - no financial benefits will be received from PO

Which one of our customer's problems are we helping him to solve?

- organization of membership/fighter information
- organization of event information
- notification of club member on more than one platform (the existing platform was Facebook)
- reduce the amount of time

Customer relationship

What Type of relationship does each of our Customer Segments expect us to establish and maintain with them?
Which ones have we established?
How are the integrated with the rest of our business model?
How costly are they?

We won't deal with the members of the club at all (but we said that the members are key partners).
We have established relationship with the PO.

Channels

How are our Channels integrated?
Which ones work best?
Which ones are most cost-efficient?
How are we integrating them with customer routines?
Channel phases

1.Awareness

- How do we raise awareness about our company's products and services?

The product owner will be able to notify the members of the club about their financial obligations.

2.Evaluation

- How do we help customers evaluate our organization's Value Proposition?

We have pinpointed the advantages of having a semi-automatic solution to his organizational habits.

3.Purchase

- How do we allow customers to purchase specific products and services?

We are not going to make profit out of it.

4.Delivery

- How do we deliver a Value Proposition to customers?

We will deliver the value proposition in different stages for 7 weeks period.
He will be able to see the progress and make modifications/additional features etc.

5.After sales

- How do we provide post-purchase customer support?

We will have a manual for the application. It will be a sort of customer support. However, he will be able to contact us via mail/mobile whenever has some difficulties.

Quality Assessment

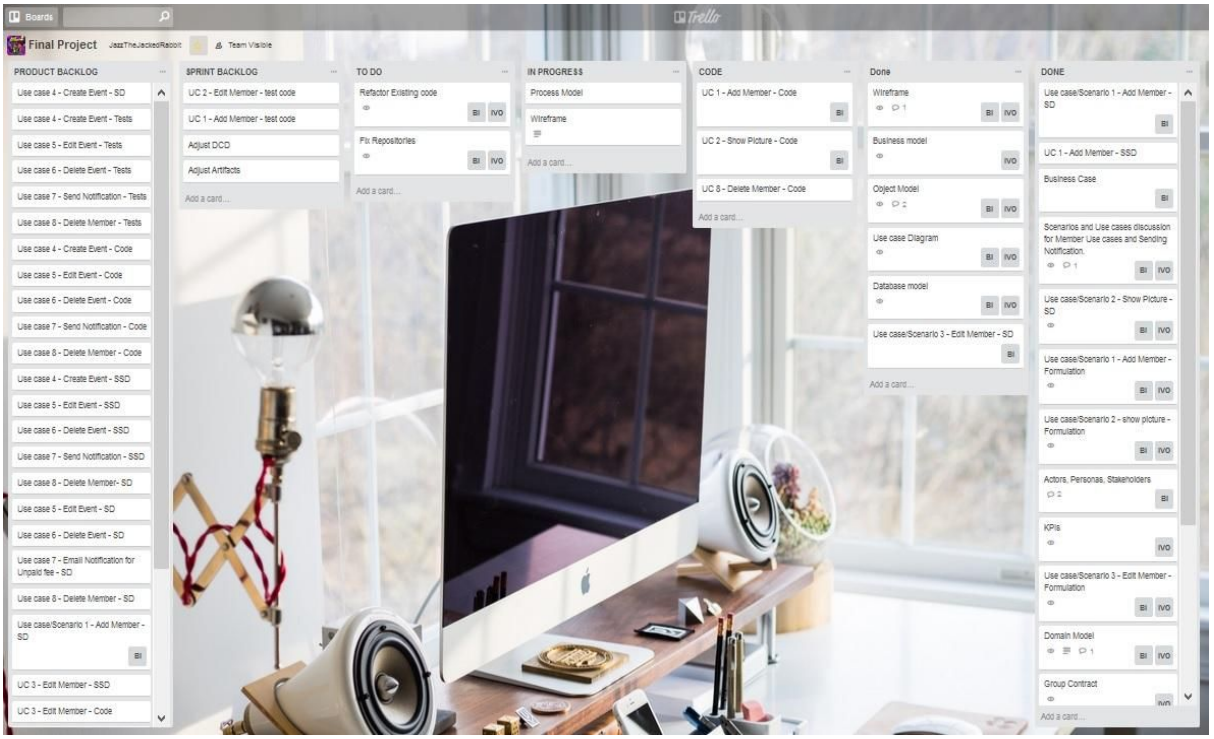
How safe should the data in the system be?	Our system would store some sensible information such as private details, so the system should be safe.
How many users and data should the system be able to handle?	Only the product owner will be able to use the application, so the system should handle only one user.
How fast should an average user learn to use the system?	Due to the simplicity of the GUI and the and guideline that we have, the system wouldn't be hard to be studied.
Who are average users?	We expect the user to become fluent with the program in several hours most. Our average user will be non-technical.

Agile methodology

The methodology used in this project is Agile, which gives us the possibility to go back and forth on the work already done. This means that requirements can change throughout the project, also artifact can be updated and modified. Working closely with the Product Owner can guarantee that the system will fulfil his needs.

Scrum board

Organization was very important for our group.
To be able to organize, keep track of the work process and spread the work evenly, we have decided to use Scrum board. The specific platform was Trello.
This kind of tool was familiar to us from previous project, but during the current project we have managed to use it to its full potential.



Personas:

Morten is the one of the club's trainers and he is employee since the beginning of the club. He is 40 years old, from which 15 years he spent in Thailand mastering the art of Muay Thai. Before starting to teach other people, he was a professional fighter having many titles and decorations along his career.

After an unfortunate motorcycle accident, he was forced to stop his fighting career and he decided to pass the Thai legacy to other people willing to learn this old mastery. He is taking classes every day, teaching in the same measure kids, amateurs, women and professional fighters.

Maya is a 32 years old professional Muay Thai fighter and she is running the club along her husband. She is multiple world champion at her division and she is practicing this art since she was in high school. In her daily life, she is combining the work with the pleasure by teaching a special class only for women.

Stakeholders:

Morten
Maya
Club members

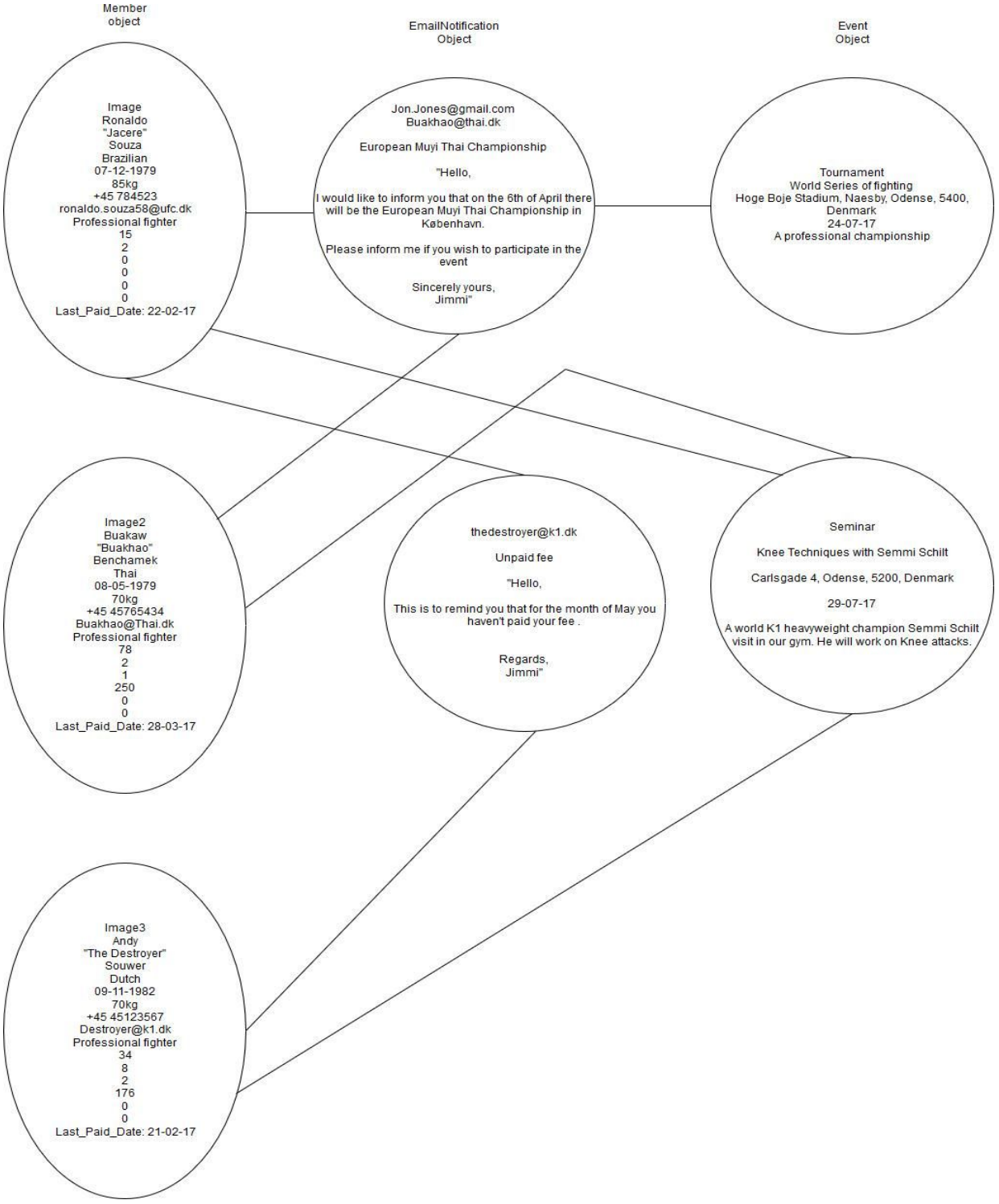
Object model

Object model defines the system in terms of objects and classes and is like system model prior to development of the program. Describes the relationship between instances of objects from the conceptual classes.

Our criteria for this artifact are:

- Ability to see the multiplicity between the conceptual classes;
- It is created before Domain model;
- The objects have example data in them that could be an actual object in the domain;

- Object relationship is showed using lines;



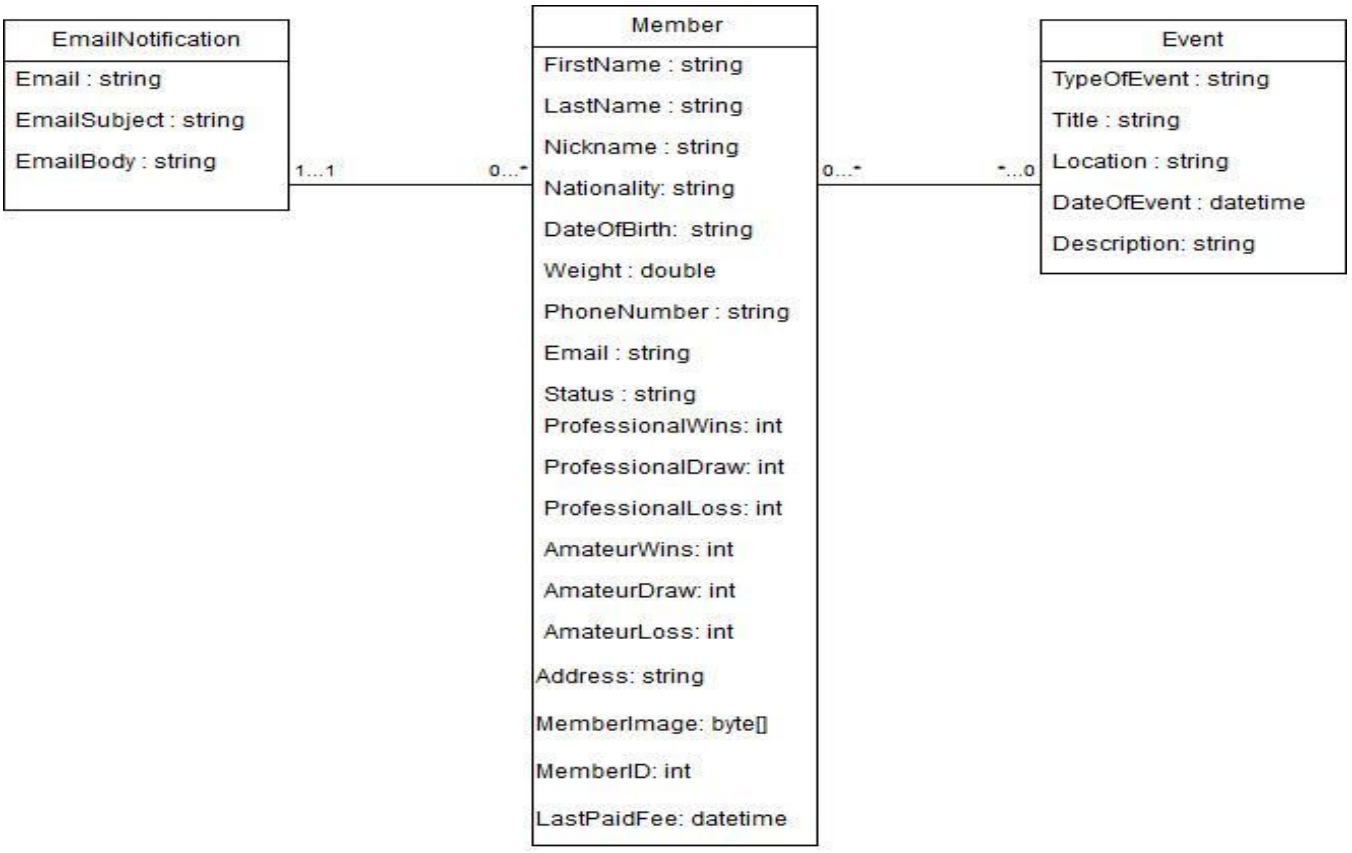
Domain Model

After making our Object model, we had the resources to make the conceptual classes that our system would use. The diagram includes all the attributes that are needed. We followed this criteria for creating the artifact:

- Contains only conceptual classes (classes that can be found in real world domain example

Inventory not repository etc.)

- Uses language that customers should understand, technical terms from the domain i.e.
- There should be names on the classes, and they should be very small and if it's more than one word it should be written like this "Member".
- Contains variables only not the methods and the variables does not need variable types and does not need to differentiate variable from property etc.
- Conceptual classes have lines between them that defines the relationship, many-to-many or one-to-many etc.



Class diagram

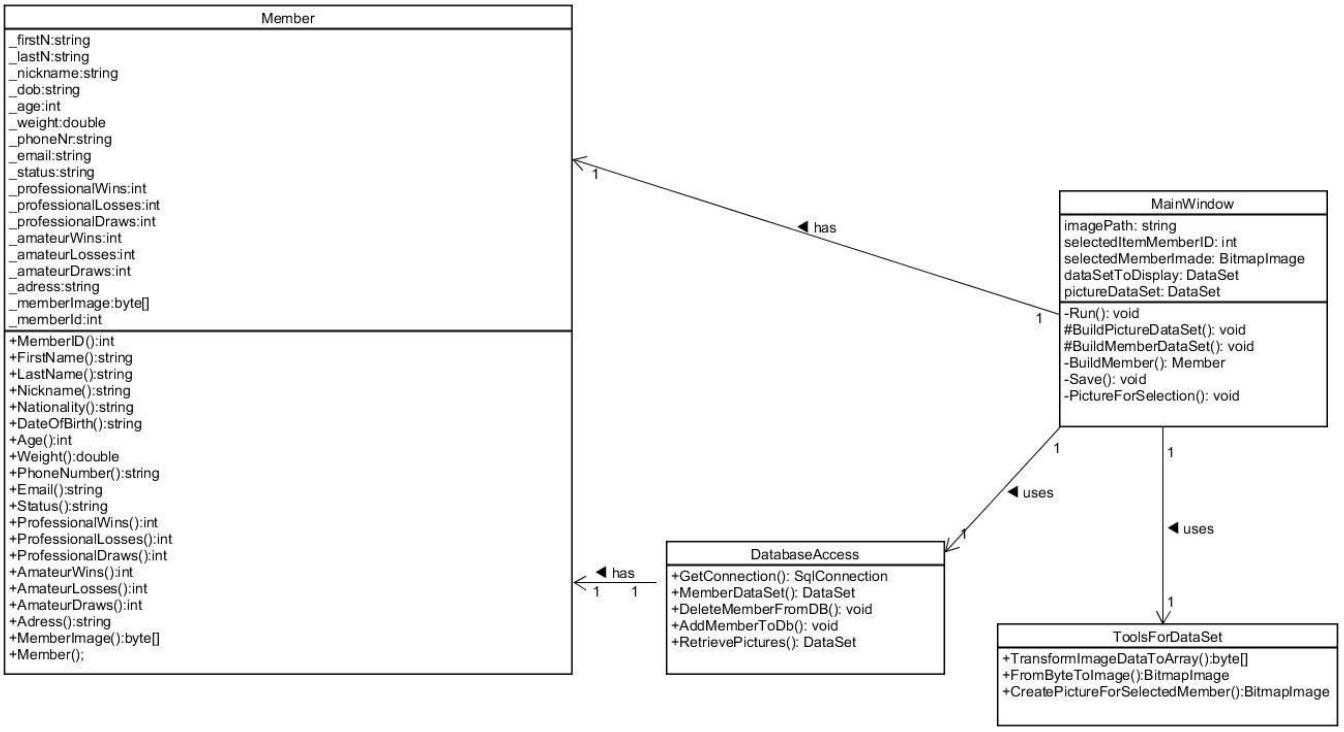
The **Class diagram** is the next thing that we did. It represents the relationship between the classes, object, attributes and operations in the system. It is a detailed plan of the whole software system. It is used as a tool for other software developers as well. It gives exact overview of the state of the application.

In our case before starting the code and after that, it changed a lot. We are going to share two DCDs. The first one will represent our current system.

The second one will represent the system after making the refactoring (See appendix). It will be the one that we are aiming for. We have recognized what kind of errors we have made in the architecture. These are the criteria that we have used for creating the artifact:

- Should represent software classes
- Connection between two classes need a word describing the relationship

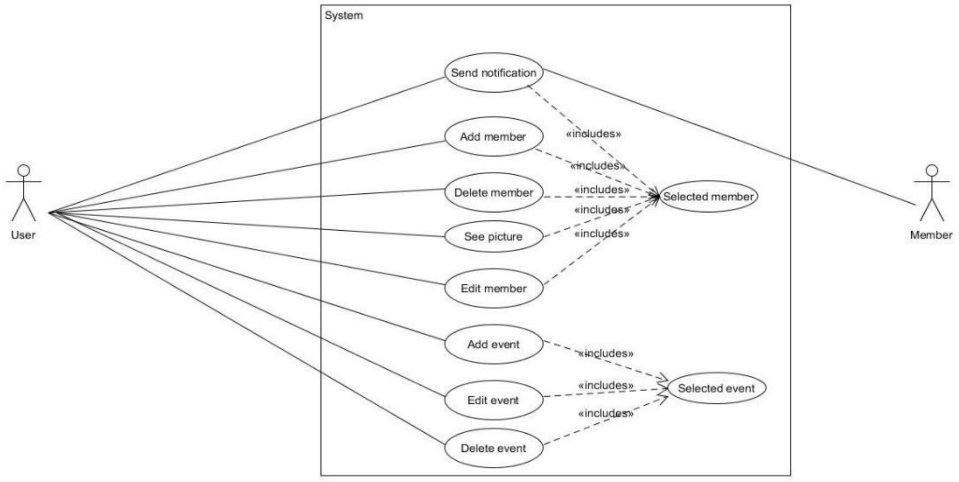
- Public methods/variables are introduced with “+”
- Private methods are introduced with “-”



Use case diagram

In advance of the use cases actual creation we have set up the use case diagram of the system to capture the dynamic aspect of the system.

In the diagram, we summarized the actions that the users of the system will be able to perform and few of the relationships between them.



Use case and scenario criteria

After identifying the personas and actor we started with the Scenarios. With the scenarios, we have a better view of the whole working process and helps with defining the Use cases. It should be written in narrative form and without any terminology. The use cases capture the goal of an action, the trigger event that starts a process, and then describe each step of the process including inputs, outputs, errors, and exceptions. Use cases are often written in the form of an actor or user performing an action followed by the expected system response and alternative outcomes. We used it to guide the design and development by using appropriate names for every process. After creating the use cases, we did the Use case diagram which shows visually the that the users interact with the system.

SSD and SD

From the Use cases, we have made the SSDs and the SDs for each of them. The SSD visualizes how a user interacts with the system for a specific use case and the SD shows the interaction between two objects. We have SDs for the most complicated tasks in our system. Criteria:

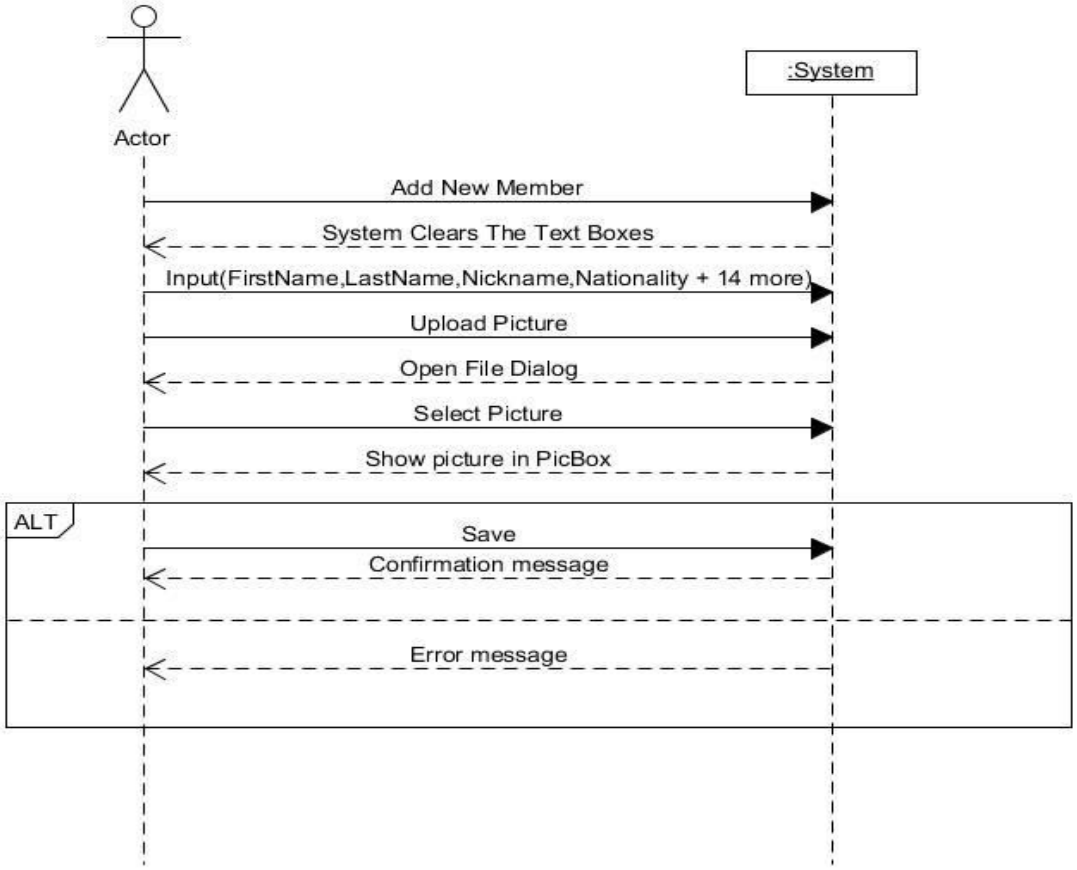
- Needs to be easy to follow.
- Must be made by specific standard - UML.
- Must explain every act between the actor and system, input and output.
- Represents the objects and classes involved.
- Represents the Use Cases, made by them.

Use case 1 – Add member

Morten goes to the “Member” tab.
The tab displays the member detail form and a list with all the members.
Morten presses “Add new” button.
The program clears the form fields.
Morten fills the form with the correct information.
Mortens wants to upload a picture of the member.
He presses “Upload Picture” button.
He selects the picture from the computer.
He confirms the selection.
He presses “Save”.
The program displays a confirmation message and clears the form.

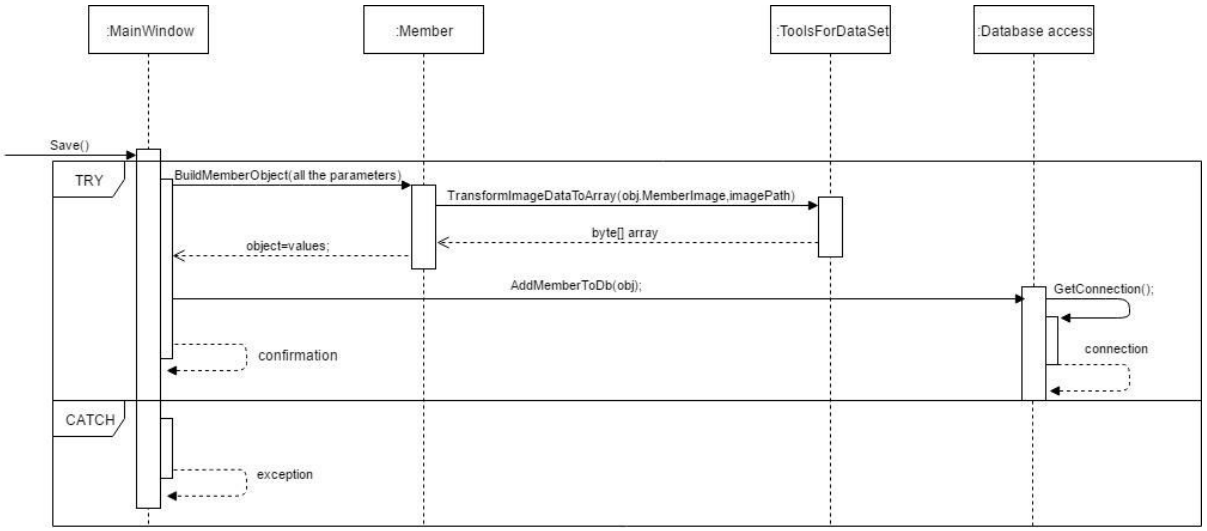
Basic flow: The user wants to add a new member. He goes to the “Member” tab and selects the “Add new” button. He fills the available form with the correct information and clicks “Save”.
Exception flow: The information must be in the correct format.
All the fields need to be filled.
Precondition: The member is not in the system.
Postcondition: The member exists in the system.

System Sequence Diagram



- Sequence diagrams illustrate interactions and sequence in a kind of fence format, in which each new object is added to the right.
- Sequence diagram clearly shows sequence or time ordering of messages.
- It has large set of detailed notation options.
- Lifelines represent either roles or object instances that participate in the sequence being modeled. Lifelines are drawn as a box with a dashed line descending from the center of the bottom edge. The lifeline's name is placed inside the box.
- To show an object (i.e., lifeline) sending a message to another object, you draw a line to the receiving object with a solid arrowhead (if a synchronous call operation) or with a stick arrowhead (if an asynchronous signal). The message/method name is placed above the arrowed line.
- Guards are used when a condition must be met for a message to be sent to the object. To draw a guard on a sequence diagram in UML you place it above the message line and in front of the message name.
- Alternatives, loops, optional, parallel elements is drawn using a frame. The word "alt", "loop", "opt", "par" is placed inside the frame's name box.

Sequence Diagram



Code Snippets

```
98          private void Save()
99          {
100             try
101             {
102
103                 Member obj = BuildMember();
104                 DatabaseAccess.AddMemberToDb(obj);
105                 MessageBox.Show("Success");
106             }
107             catch
108             {
109                 MessageBox.Show("Error occured! Please try again!");
110             }
111         }
```

- The “Save” method is used to merge two methods that are working in different classes: “BuildMember” from the Main Window class and “AddMemberToDb” from the Database Access class. The method was built to provide the functionality of two methods in a single call, fact that is following the “Single responsibility” principle stated in SOLID. It will display a confirmation message if the process will succeed or it return an exception if something failed.

```
74          private Member BuildMember()
75          {
76              Member obj = new Member();
77              obj.FirstName = firstNameTB.Text;
78              obj.LastName = lastNameTB.Text;
79              obj.Nickname = nicknameTB.Text;
80              obj.Nationality = nationalityTB.Text;
81              obj.Email = emailTB.Text;
82              obj.Age = Convert.ToInt16(ageTB.Text);
83              obj.Weight = double.Parse(weightTB.Text);
84              obj.PhoneNumber = phoneNumberTB.Text;
85              obj.Adress = adressTB.Text;
86              obj.Status = statusTB.Text;
87              obj.AmateurWins = Convert.ToInt16(amWTB.Text);
88              obj.AmateurLosses = Convert.ToInt16(amLTB.Text);
89              obj.AmateurDraws = Convert.ToInt16(amDTB.Text);
90              obj.ProfessionalWins = Convert.ToInt16(proWTB.Text);
91              obj.ProfessionalLosses = Convert.ToInt16(proLTB.Text);
92              obj.ProfessionalDraws = Convert.ToInt16(proDTB.Text);
93              obj.DateOfBirth = dObTB.Text;
94              obj.MemberImage = ToolsForDataSet.TransformImageDataToArray(obj.MemberImage, imagePath);
95              return obj;
```

- “Build Member” method it’s used to read the information from the UI and send it to the correct propriety of the object it creates. For the “MemberImage” propriety, it calls the “TransformImageDataToArray” method from the “ToolsForDataSet” class that will convert the image found by the “imagePath” to an array of bytes and send it back to the propriety.

```
46          public static void AddMemberToDb(Member obj)
47          {
48              using(SqlConnection connection = GetConnection())
49              {
50                  SqlCommand cmd = new SqlCommand("AddMemberToDB", connection);
51                  cmd.CommandType = CommandType.StoredProcedure;
52                  cmd.Parameters.Add(new SqlParameter("@FirstName", obj.FirstName));
53                  cmd.Parameters.Add(new SqlParameter("@LastName", obj.LastName));
54                  cmd.Parameters.Add(new SqlParameter("@Nickname", obj.Nickname));
55                  cmd.Parameters.Add(new SqlParameter("@Weight", obj.Weight));
56                  cmd.Parameters.Add(new SqlParameter("@Age", obj.Age));
57                  cmd.Parameters.Add(new SqlParameter("@PhoneNumber", obj.PhoneNumber));
58                  cmd.Parameters.Add(new SqlParameter("@Email", obj.Email));
59                  cmd.Parameters.Add(new SqlParameter("@Nationality", obj.Nationality));
60                  cmd.Parameters.Add(new SqlParameter("@DateOfBirth", obj.DateOfBirth));
61                  cmd.Parameters.Add(new SqlParameter("@Status", obj.Status));
62                  cmd.Parameters.Add(new SqlParameter("@Adress", obj.Adress));
63                  cmd.Parameters.Add(new SqlParameter("@AmateurWins",obj.AmateurWins));
64                  cmd.Parameters.Add(new SqlParameter("@AmateurLosses",obj.AmateurLosses));
65                  cmd.Parameters.Add(new SqlParameter("@AmateurDraws",obj.AmateurDraws));
66                  cmd.Parameters.Add(new SqlParameter("@ProfessionalWins",obj.ProfessionalWins));
67                  cmd.Parameters.Add(new SqlParameter("@ProfessionalLosses",obj.ProfessionalLosses));
68                  cmd.Parameters.Add(new SqlParameter("@ProfessionalDraws",obj.ProfessionalDraws));
69                  cmd.Parameters.Add(new SqlParameter("@MemberPicture", obj.MemberImage));
70                  cmd.Parameters.Add(new SqlParameter("@PictureName", obj.FirstName + " " + obj.LastName));
71                  cmd.ExecuteNonQuery();
72              }
```

- The “AddMemberToDb” method is used to save the object into the database. It’s making use of the “using” statement that will use the connection returned by the “GetConnection” method to communicate with the database. It is using a stored procedure called “AddMembetToDB” to add all the parameters of the object into the correct place in the table.

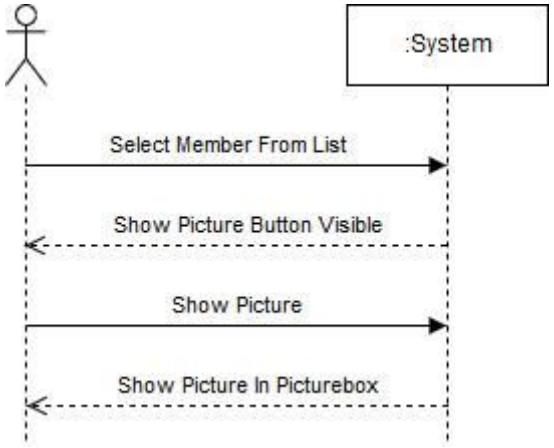
Use case 2 - Show picture of Member

Morten goes to the “Member” tab.
The tab displays the form and the list with the member in our system.

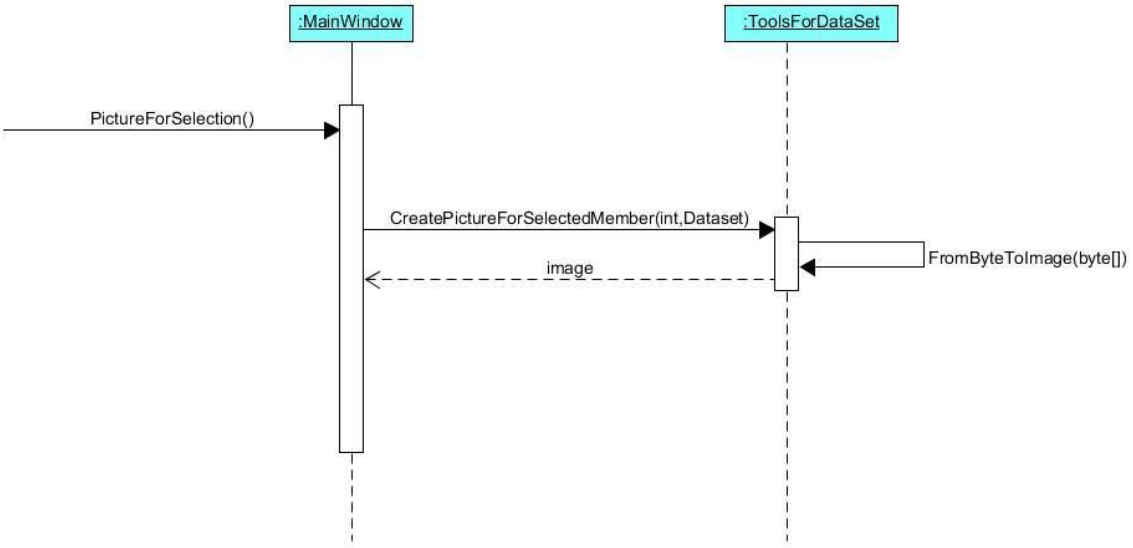
He selects the desired member.
He presses “Show Picture” button.
The picture is displayed in the form.

Basic Flow:
The user wants to see the picture of a member. He goes to the member tab. He selects a member from the list of members. He presses the “Show picture” button. The picture is displayed in the form.
Pre-conditions:
The picture box is empty.
Post-condition:
The picture box contains the member image.

System Sequence Diagram



Sequence Diagram




```
141     private void MemberGrid_SelectionChanged(object sender, SelectionChangedEventArgs e)...
160     private void PictureForSelection()
161     {
162         selectedMemberImage = ToolsForDataSet.CreatePictureForSelectedMember(selectedItemMemberID, picturesDataSet);
163     }
```

- The “PictureForSelection” method is used to bind the picture box in the GUI with the picture

that the “CreatePictureForSelectedMember” method from the ToolsForDataSet will return.

```
63          public static BitmapImage CreatePictureForSelectedMember(int id, DataSet dsForPicture)
64          {
65              BitmapImage imageFromDB = new BitmapImage();
66              byte[] imageArrayFromDS;
67
68              foreach (DataTable table in dsForPicture.Tables)
69              {
70                  foreach (DataRow row in table.Rows)
71                  {
72                      int memberIdFromRow = Convert.ToInt32(row["MemberID"]);
73                      if (id == memberIdFromRow)
74                      {
75                          imageArrayFromDS = (byte[])row["MemberPicture"];
76                          imageFromDB = FromByteToImage(imageArrayFromDS);
77                      }
78                  }
79              }
80              return imageFromDB;
81          }
```

- The “CreatePictureForSelectedMember” method is taking as parameters an integer and a dataset. It declares a BitmapImage called “imageFromDB” that will become the returned object. The method will loop through all the tables and the rows of the dataset passed as parameter and convert the content of the “MemberID” row into an integer. Then it will check every result against the integer value that we passed as parameter and if it will find a match, it will transform the content of the “MemberPicture” row into a byte array. Then it will bind the “imageFromDB” to the result of the “FromByteToImage” method and return the object.

```
38          public static BitmapImage FromByteToImage(byte[] array)
39          {
40              if (array == null || array.Length == 0)
41              {
42                  return null;
43              }
44              else
45              {
46                  var image = new BitmapImage();
47                  using (var mem = new MemoryStream(array))
48                  {
49                      mem.Position = 0;
50                      image.BeginInit();
51                      image.CreateOptions = BitmapCreateOptions.PreservePixelFormat;
52                      image.CacheOption = BitmapCacheOption.OnLoad;
53                      image.UriSource = null;
54                      image.StreamSource = mem;
55                      image.EndInit();
56                  }
57                  image.Freeze();
58                  return image;
59
60              }
61          }
```

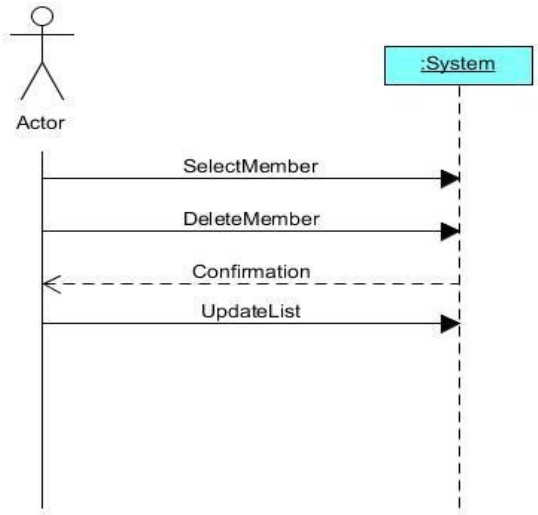
- The method “FromByteToImage” is used to convert an array of bytes that is passed as parameter into a BitmapImage. In our case, it will take the array that we converted from the dataset and transform it to an image using a memorystream and specific techniques.

Use case 3 – Delete a member:

Morten goes to “Member” tab.

The tab displays a list with all the members.
He chooses the member.
Then he clicks on “Delete Existing Member”.
Confirmation window shows.
He presses “I confirm”.
The system shows confirmation message, clear the form and refresh the list.
Basic flow: The user wants to delete an existing member. The user selects a member from the list. He confirms the delete. The list is refreshed and the member is removed.
Exception flow:
There will be no exception as the button won’t be active until a member is selected from the list.
Precondition: The member is existing in the system.
Postcondition: The member is no longer existing.

System Sequence Diagram



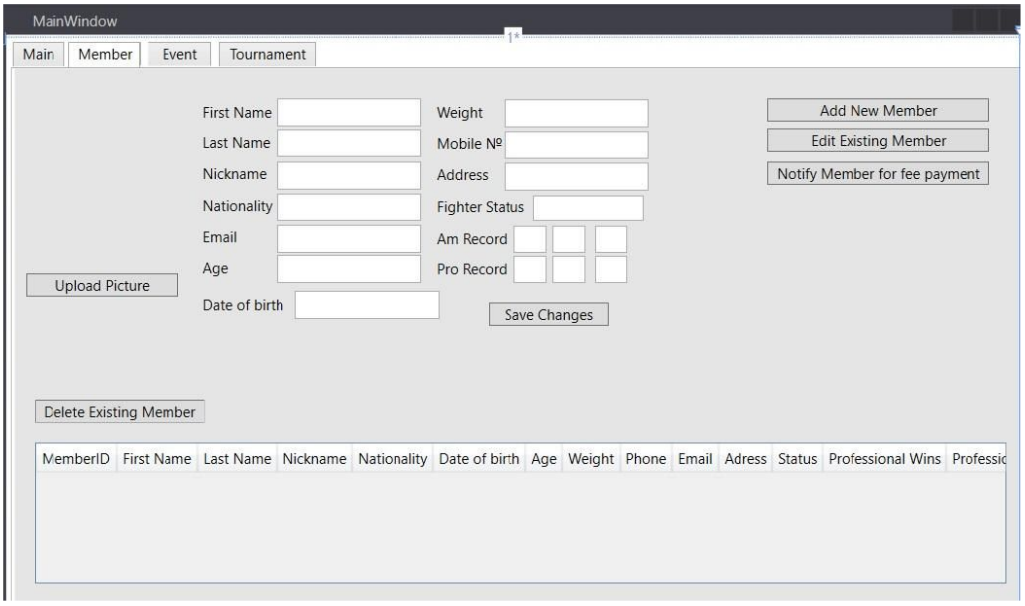
Solution design

The initial design of our application was planned to follow the 3-layer architecture: Presentation layer(UI), Business layer (repositories, objects, etc.) and Data layer (database tables and stored procedures).

Presentation layer

This layer of our application consists of a graphical user interface (GUI) that will fulfil the user’s needs. The framework we chose to work with in the beginning of the project was Windows Forms but after long discussion and comparisons we decided to go further with the Windows Presentation Foundation (WPF) framework due to several reasons:

- It is a newer platform thereby it offers more functionality than the Windows Forms
- The presence of XAML that allowed us to gain knowledge on the design field
- Data binding that allows a cleaner separation between data and layout
- Performance friendly due to hardware acceleration



Business layer

The business layer is working like a bridge between the presentation layer and the data layer and it consists of 3 sub-layers:

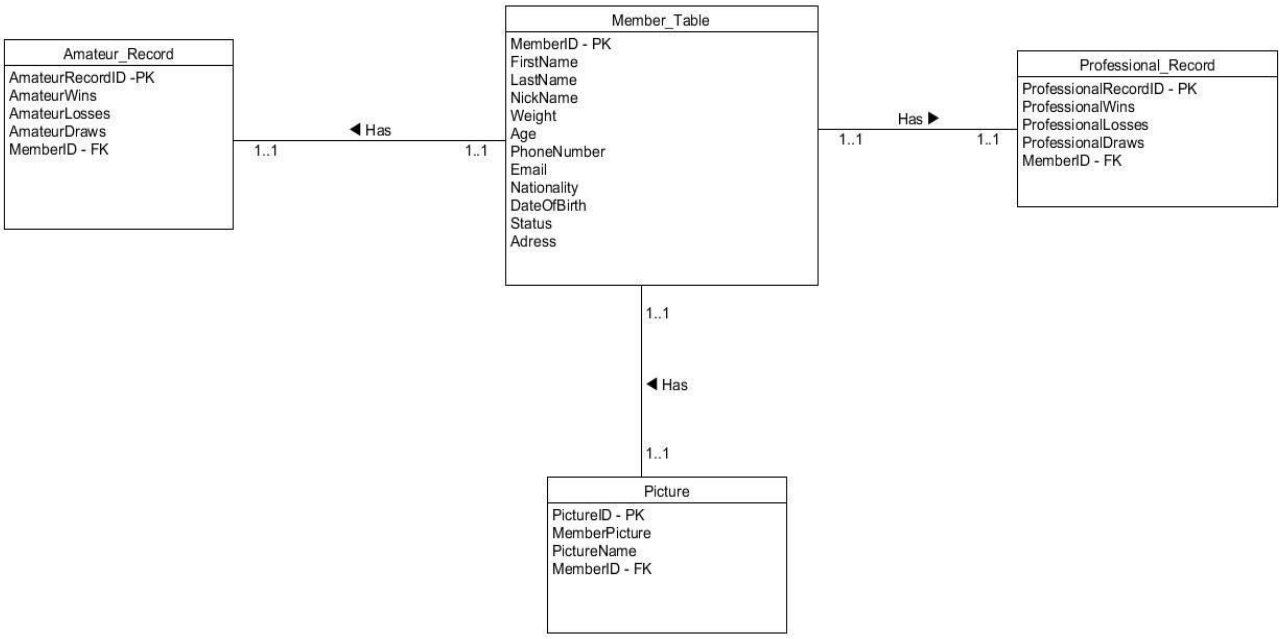
- The business logic layer:
- The data access layer: Database access class
- The value object layer: Member class

Repository pattern: The initial plan was to use the repository pattern in our application to provide a middleman between our storage unit and user interface. The plan was to create a repository for each of our business objects and implement straightforward methods that would deal with the CRUD data from the database. Due to lack of experience and resources (time, workforce) we have suspended the plan and converted it to a more easy-to-deal-with plan that allowed us to move forward with the project and be able to deliver shippable pieces of work at the end of each sprint.

Data layer

Our data layer consists of an SQL Database that will store all the data in cause.

Database Diagram



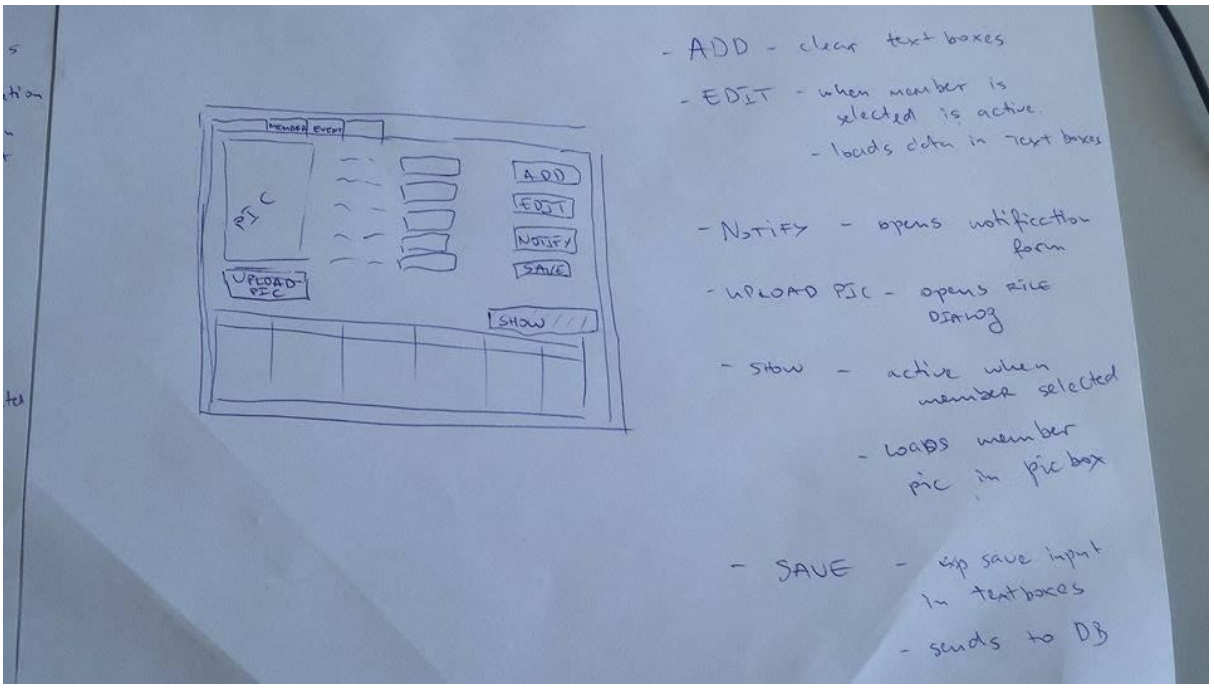
Description of each sprint

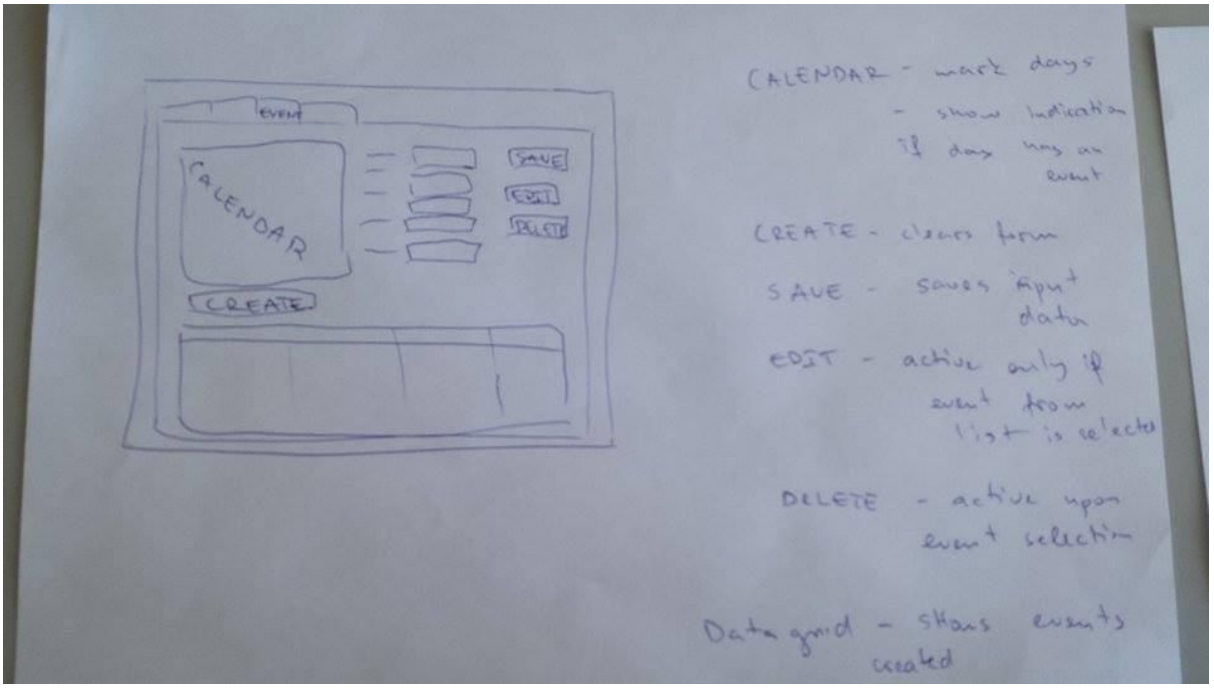
Sprint 1

In our first sprint, we started by brainstorming and figuring out what our goals were for the current project. We had already an idea for the solution and its purpose. We set up all the platforms that we are going to use for sharing, gathered a list and planned what kind of artifacts we are going to need to present to the product owner and for our final report. After arranging a meeting with our PO, we've created a wireframe which was the way to make him visualize and understand what our proposition for solution is. On the meeting, we gathered the most sufficient information regarding is business routines and details that were needed to create the initial artifacts - Business case, business model, personas, stakeholders. All our task was divided between both of us using Trello.

To measure our workflow and to be able to control eventual gaps, we have been set up a burndown chart directly connected to our scrum board that show us the workflow of each sprint that will give us an idea about how good we managed the tasks.
In the end of the week we did an evaluation of what was done during the sprint and made a plan for next sprint.

Wireframe for initial system which gives a bit more details about the functionality and visualization presented to the PO.





Sprint 2

We brainstormed (or similar action) all the use cases we need for the project from which we continued to go into detail with three of them that we considered crucial for the further development. Along with them, the scenarios and the expected outcomes.

During the sprint, we also created the initial models for our business, the Object model and the Domain model which undergo changes later after the meeting with the PO because he required new functionality and small changes of the existing plan.

By the end of the sprint, the wireframe got its last shape and was ready for the implementation.

Sprint 3

In the beginning of the sprint we had a meeting with one of our teachers to receive feedback on the existing work and clarify if we were on the right track.

We started with the first use case for which we have created the diagrams and visualized the process that will fulfill. We continued with the wireframe implementation in accordance with the use case for which we have fulfilled the functionality that the use case will hold.

We started with the initial setup for our solution by creating 2 of the classes and populate them with the correct content that would allow us to fulfill the use case.

The meeting with the PO was short. Our goal was to show him the user interface and tell him some details about the process that he will face for fulfilling the goal of the use case.

Sprint 4

In the beginning of the sprint we planned the content of the week and we set up a goal we have to reach by the end of the sprint.

We have started by planning the database model, the tables we need and the relations between them. We tried to create them according to the process of normalization to prevent future changes in the database and risking altering the content.

After we finished with the tables, we checked the artifacts to make sure that we followed the plan and moved to the next step, connecting the program with the database.

In this phase, we encountered problems that we haven't been able to solve in time due to lack of experience, fact that forced us to change the plans to reach the sprint goal - the first use case functional.

The solution we found was not how we expected but it helped us fulfill the sprint goal and in the end of the week, we presented the work to our PO that was able to add a member with the information he needs to the database.

Sprint 5

In the beginning of the week we rehearsed the problems we encountered in the last sprint and we thought about how they will affect the next use case. We started to work on the use case 2 and we could finish the artifacts quickly saving time for the coding process.

The lack of experience with WPF was a step-back for us because we spent a lot of time searching about data bindings to understand the process and use it accordingly.

The next challenge for us represented the process of adding picture because it brought changes into the database, fact that we tried to avoid.

In the end of the sprint, we updated all the artifacts that encountered changes during the sprint.

Sprint 6

In this sprint, we focused mainly on trying to look back and check where can we improve.

We have started a refactoring process but we brought unwanted changes that harmed the solution

so we had to spend more time on it.

On the way, we could create the code for another use case - Delete member.

After refactoring, changes appeared in the artifacts as well so we noted down what we need to touch in the next sprint and started to work on the final report.

To use the last sprint wisely, we planned the report content in advance and wrote down a list of artifacts that we need to include.

Sprint 7

The sprint was focused on building the report.

We split the tasks equally between us to save time and to allow both of us to have a personal touch on it. We compared the list of content we have with the other groups and asked for feedback on our artifacts.

Conclusion

In conclusion, during the past 6 weeks we tried to cover all the learning goals in the curriculum. Our plan was to use as many techniques from the ones we have studied during the year and display a high level of professionalism.
Unfortunately, lack of experience played an important role in the end because we encountered difficulties that we couldn't overcome in time forcing us to reconsider the process and to use techniques that we are aware there are not the best ones.
For ourselves, this project represents a huge step forward. We tried to go over all the material from the curriculum. We learned about the process from planning phase to the working product. While managing both customer and group relationship we managed to establish a strong teamwork which delivered results.

APPENDIX

Use cases:

Use case 3 – Edit member:

Morten goes to "Member" tab.
The tab displays the member detail form and a list with all the members.
Morten presses "Edit" button.
He chooses the member.
Then he clicks on "Edit".
The program loads the information in the form.
Morten modifies the desired details.
Morten presses "Save Changes".
The system shows confirmation message, clear the form and refresh the list.

Basic flow: The user wants to edit an existing member. The user selects a member from the list. The user details are loaded. The user edits the desired information. He saves the information. The list is refreshed and the member info is updated.
Exception flow:
The “Edit” button is active only after selection of a member.
Precondition: The member is existing in the system.
Postcondition: The member information has been updated.

Use case 4 – Create an event

Morten goes to the “Event” tab.
The tab displays a calendar for the current month.
Morten wants to create a new event.
He selects a day from the calendar.
The list box will display all of the events for the selected date.
He presses “Create New Event”.
A window will pop up and display a form.
He fills the form.
He presses “Save”.
The program displays a confirmation message, closes the form and updates the list of events for current date.

Basic flow: The user wants to add a new event. He goes to the “Event” tab and selects a day. He presses “Create New Event” button. He fills the form available with the correct information and clicks “Save”.
Exception flow: The information must be in the correct format.
All the fields need to be filled.
A day from the calendar must be selected.
Precondition: The event doesn’t exist.
Postcondition: The event exists in the system.

Use case 5 – Edit an existing event

Morten goes to the “Event” tab.
The tab displays a calendar and a list of events.
Morten selects an event from the list.
Morten presses “Edit an event”.
The event form pops up with the information loaded.
He makes the changes.
He presses “Save”.
The program displays a confirmation message, close the window, clears the form and updates the list.

Basic flow: The user wants to edit an existing event. He goes to the “Event” tab and selects the desired event. He clicks on “Edit”. He makes the changes and presses “Save”. The information is updated.
Exception flow: The information has to be in the correct format.
All the fields need to be filled.
Precondition: The event exists in the system.

Postcondition: The event exists in the system with the updates.

Use case 6 – Delete an existing event

Morten goes to the “Event” tab.
The tab displays a calendar and a list of events.
Morten selects an event from the list.
Morten presses “Delete”.
Confirmation message shows up.
He presses “I confirm”.
The event is deleted from the system and the list is updated.

Basic flow: The user wants to delete an existing event. He goes to the “Event” tab and selects the desired event. He clicks on “Delete”. He confirms the action. The event is deleted and list updated.
Exception flow: There is no exception flow.
Precondition: The event exists in the system.
Postcondition: The event is no longer existing.

Use case 8 – Delete a member:

Morten goes to “Member” tab.
The tab displays a list with all the members.
He chooses the member.
Then he clicks on “Delete Existing Member”.
Confirmation window shows.
He presses “I confirm”.
The system shows confirmation message, clear the form and refresh the list.

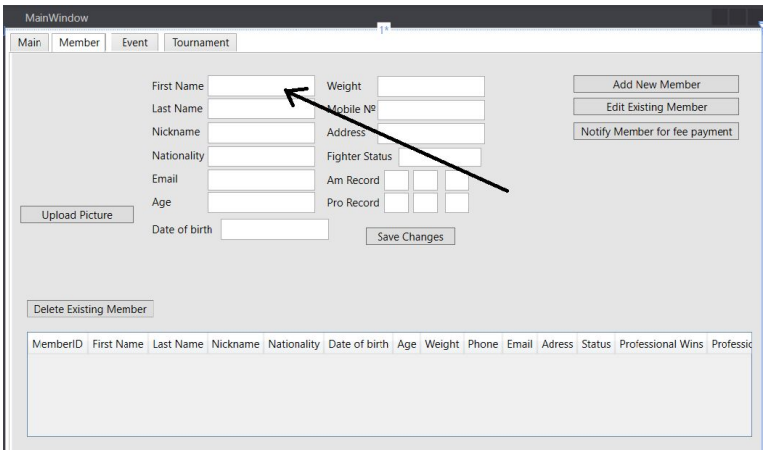
Basic flow: The user wants to delete an existing member. The user selects a member from the list. He confirms the delete. The list is refreshed and the member is removed.
Exception flow:
There will be no exception as the button won’t be active until a member is selected from the list.
Precondition: The member is existing in the system.
Postcondition: The member is no longer existing.

USERS GUIDE: Software’s manual

WARNING!
Because the program is still in the development phase, after adding or deleting a member, the list will be loaded again with the new information from the database but the old information will still be present in the list!
We have already formulated the solution for this issue and it will be fixed soon.

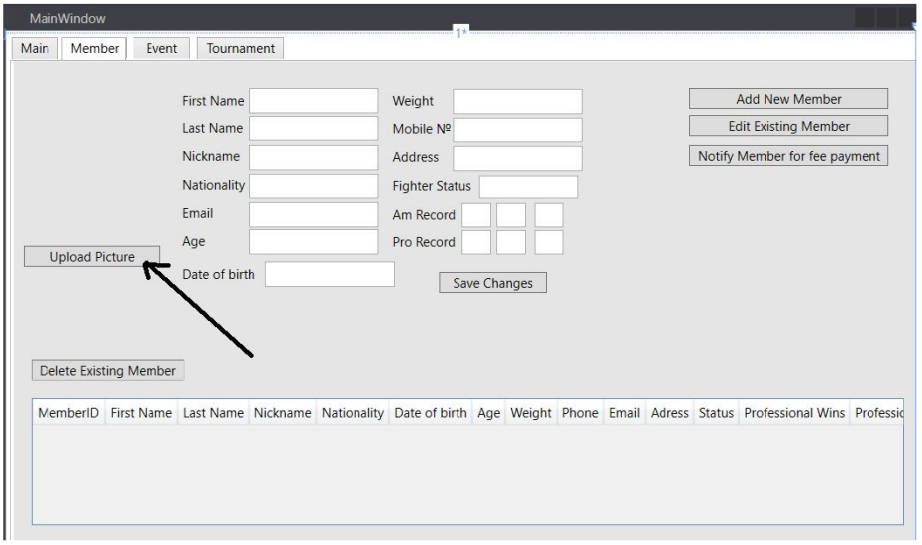
ADD MEMBER:

In order to add a new member to the system, you should follow the following rules:



- Introduce the correct information in the text boxes
- The program has no input validation for the moment, therefore you should respect the following conventions:- First name, Last name, Nickname, Nationality, Email, Status, Date of birth and Address fields accept both letters and numbers as input.
    - Age, Weight, Mobile Number,Am Record and Pro record accept ONLY numbers as input.

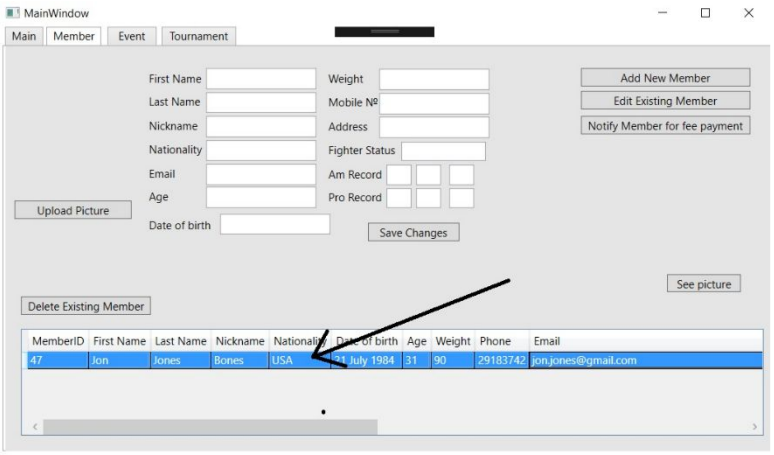
- Upload a picture from the computer before saving by pressing the “Upload Picture” button, select the desired document and confirm:



- Press “Save Changes”.

SEE PICTURE:

- Select the member from the list



- Press “See Picture” button.

Delete member

- Select a member from the list like in the above example
- Press “Delete Existing Member”

Design Class Diagram

We decided to upload the DCD for our future plans after refactoring.