



Erhvervsakademi Sjælland

Hovedopgave

Prædefineret information

Startdato:	03-04-2017 09:00	Termin:	jun 2017
Slutdato:	09-06-2017 13:00	Bedømmelsesform:	Dansk 7-trinsskala
Eksamensform:	Mundtlig prøve	ECTS:	15
SIS-kode:	259410 0617 ro14da15-5H 70125 - MDT EKS 7TRIN		
Intern bedømmer:	Michael Claudius		

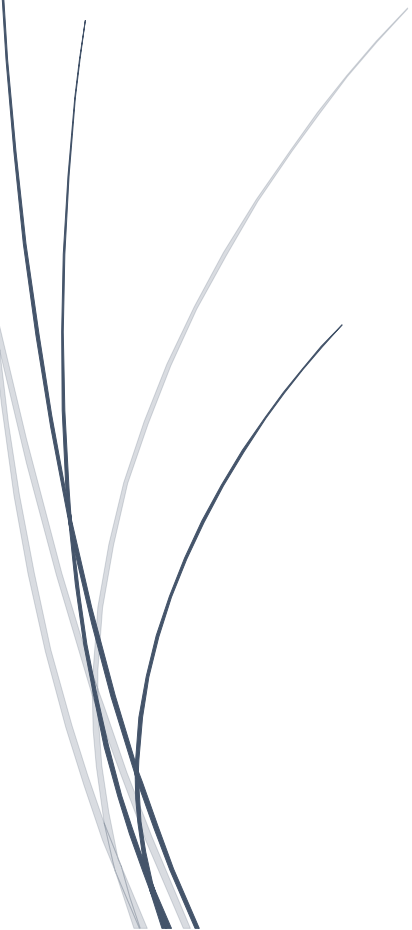
Deltager

Navn:	Stefani Dimitroua Dimitroua
Kandidatnr.:	6765 31739 jun 2017 1951 3970
UNI-C ID:	stef278f
Alt. id:	2510952718
EASJ-id:	stef278f@easj.dk



E-Sport Deals

*Written by Stefani Dimitrova and Jonas Rytter with the
guidance of Michael Claudius*



May 1st – June 9th, 2017
ZIBAT Roskilde
Computer Science
Dissertation report

Table of Contents

Project Establishment Introduction.....	5
Company Description	5
About the company	5
Project description	5
Requirements	5
Problem Statement	6
Problem Formulation.....	6
The Problem	6
Research Questions	6
Solving and Answering Questions	6
Project Scope	7
Resources.....	7
Human Resources.....	7
The Project Group.....	7
Conclusion – Project Group	7
Project Group Risks.....	7
Supervisor.....	8
Product owner	8
Defining workspace and workflow	8
Tools	8
Development Methodologies.....	9
Conclusion	9
Project Plan.....	11
Project Establishment Conclusion	11
Sprint Zero	11
Business Model Canvas	12
User Stories	12
Product Backlog.....	13
Database Design	14
First Attempt.....	14
Entities	14
Identifying Entity Relationships	15
Cardinalities	15

First Attempt – Conclusion	16
Second Attempt.....	16
Entities	17
Identifying Entity Relationships.....	17
Cardinalities	18
Second Attempt - Conclusion	18
Final Attempt – Current Database Solution	19
Architecture.....	20
Design Patterns.....	20
GRASP	20
Proxy Design Pattern	21
Code Reuse Pattern	21
Sprint Zero – Review.....	21
Sprint One – May 1 st – 14 th 2017	21
Sprint Backlog.....	21
User Stories	22
User Story 1 – Secure Information	22
User Story 2 – Become a Mentor	25
Backend	25
Frontend	26
User Story 3 – Create User Account	28
Backend	28
Frontend	29
User Story 4 – Select Game and Role to Browse Mentors	29
Backend	29
Frontend	30
User story 5 – Assign a Supervisor to Booking	31
Backend	31
Frontend	32
User Story 6 – Display All Booking Under a Mentor.....	32
Backend	32
Frontend	32
Sprint Review Meeting	33
Sprint Retrospective	33

Velocity Chart	34
Sprint Conclusion.....	34
Sprint 2 – 15 th – 28 th May 2017	35
Sprint Backlog.....	35
User Stories	36
User Story 7 – Update Mentor Application Status	36
Backend	36
Frontend	37
User Story 8 – See all Supervisors	37
Backend	37
Frontend	38
User Story 9 – User Booking feedback	38
Backend	38
Frontend	39
User Story 9 – Log in.....	40
Backend	40
Frontend	40
User Story 10 – Create Booking.....	41
Backend	41
Frontend	41
Sprint Review Meeting	42
Sprint Retrospective	43
Velocity Chart	43
Sprint Conclusion.....	44
Sprint 3 – 29 th May – 9 th June 2017	44
Product Backlog.....	44
User Stories	45
User Story 11 – Create Supervisor Accounts	45
Backend	45
Frontend	45
User Story 12 – See All Feedback	46
Backend	46
Frontend	46
User Story 12 – Reset Password	47

Backend	47
Frontend	49
User Story 13 – Cancel Booking.....	49
Backend	49
Frontend	49
Sprint Review Meeting	49
Unfinished User Story.....	49
Velocity Chart	50
Sprint Conclusion.....	50
Project Conclusion	51
Process.....	51
Tools	51
Techniques.....	52
Personal Evaluation	52
Stefani Dimitrova.....	52
Jonas Rytter	53
Future plans.....	53

Project Establishment

Introduction

The writing and creation of this report and product is our final evaluation point for our Academy Profession Degree.

This project's purpose was given to us by a company that was in dire need of a software solution for their business plan. We chose this product because we believe it would provide us with a great opportunity to self-explore and develop our skills in software development.

What we hope to achieve during the development of the product is insight and skills in the new technologies that are currently overwhelming the development world and providing better solutions for the market. Having a broader knowledge and experience with different technologies is a key to adapt in this field of work. The report will be our documentation of our research, understanding and practice of new technologies that we will learn throughout the entirety of the project.

The final product that we hope to cross the finishing line with is a working and ready to use solution for the company backed by a detailed documentation of the development process. When we cross that line, we are planning to further develop and enhance our product.

Company Description

About the company

E-sportsdeals is a newly started company that is just stepping into the business world. The focus of the company is the electronic sports market and they are currently helping gamers, teams and communities find a mentor that would take them to the next level of professional gaming.

The product and company owner is Jens Thulstrup Steno Petersen and his partner Victor Folmann. During the development process, Jens will be the one that would take part in the development process of the product because his partner is currently operating from abroad.

The main responsibility that Jens will have towards the process is giving us feedback, suggestions, and additional requirements.

Project description

Despite the company being small, they are currently delivering service to a handful of clientele. However, the company's future business plan is to expand beyond Denmark which means a higher demand of services that needs a software solution.

Currently, the company handles each individual client through private channels. This costs the company time and a lot of paper work.

The vision our client has is to have a simple software solution that would make it easier for his customers to find their desired mentors and communicate with them. He had a few requirements towards the product that we need to satisfy.

Requirements

1. Product should be a web application with a simple and good user interface design
2. Product should have a responsive web design
3. Product should be backed by a database

Problem Statement

Problem Formulation

E-Sport Deals is a company that is newly established on the market and because of that they need an efficient and simple solution for their business. The problem that the company is currently facing is that they have no working solution to manage their workload therefore making it harder to reach their customers. Currently, all the customers are reaching the company through private channels which makes it difficult for the company to handle communications efficiently.

Our client is facing a greatly increasing demand of a system that would solve the problem of communication between his customers and mentors.

The Problem

"How can we develop a new website for E-Sport Deals, using an appropriate framework?"

Research Questions

"What development process will be appropriate for this development project?"

"What are the technological requirements for the project?"

"Which framework will be the best choice?"

"What benefits will be gained from using the chosen framework?"

"How can we test if the user requirements have been met?"

"How do we establish the user requirements for the new website?"

Solving and Answering Questions

The way we are going to solve the problem and answer the question is by firstly being in direct and frequent contact with our client to ensure the requirements are always up-to-date.

From a developer's point of view, we will focus on expanding our knowledge in web development and making a thorough research in what would be the best technology for the best outcome of the product.

Choosing the right development methodology, we would have to analyze and decide based on the different techniques we have been taught during our education, to find the best one.

To find the framework that would fit our project the best, we will evaluate and choose based on the different techniques we have been taught during our education.

Project Scope

The period we have for developing this project is 7 weeks including the inception phase of the project and all sprints.

During this time, we will be creating artifacts to document our process, developing the program itself, testing each part of the system and adapting to feeding from our product owner.

We believe that the most time-consuming part of the process would be documenting and writing the report and delivering a quality product with a finalized documentation. Therefore, we concluded that we will be mainly focusing on developing and thoroughly testing the core functionalities of the system.

Resources

Human Resources

This section will be dedicated to pointing out the people that are involved in the development of the product. Currently, the human resources available are The Project Group, The Project Group's Supervisor, and the Product Owner.

The Project Group

The team developing this product is composed of two ambitious students that will work synergistically to produce quality for the client – Stefani Dimitrova and Jonas Rytter. Having the experience of working in a group together we know our ambitions and boundaries.

Throughout our education, we have been involved in constructing various projects together that resulted in success. Because of that we understand the levels of initiative each of us are willing to contribute to reach a certain goal.

Despite having a successful overall teamwork, we each have our individual strength and weakness that we need to take account for. After evaluating that, we gained an overview that each one of our weakness is compensated by our strengths.

Team member	Strenghts	weaknesses
STEFANI DIMITROVA	Good programmer Good with databases Strong English	GUI design
JONAS RYTTER	Good programmer Good at GUI design Very organized	Databases

Conclusion – Project Group

To conclude, the group we have established is overall balanced. However, to compensate for the weaknesses and because this is a process of expanding our knowledge, we believe that it would benefit us if we get more involved and help each other out understanding the subject our weaknesses are most profound at.

Project Group Risks

1. Data Integrity

Loss of data is always a risk that must be taken account of due to hardware/software inconsistency, human error, etc. Our solution to this problem is to store our data in Github.

2. Unavailability

This risk can happen at any certain point and there isn't any way we can prevent it. Any one of us can fall ill, have personal matters to attend to that could interfere with the development process.

Our solution would be to make sure that whatever responsibility the other one has taken are doable in the period.

Supervisor

Because this project is related to our final evaluation point for our Academy Profession Degree, we have been assigned a supervisor to deliver constructive feedback and guidance in our production process. Having a supervisor involved in this project is very beneficial for both the team and the successful outcome of the process because we are both unexperienced and will make many mistakes along the way.

Product owner

Our Product owner - Jens - has the most crucial role in the creation of the product because he is the one that has the final say in what is acceptable. His main responsibilities are to give us requirements, feedback, and quality testing.

Defining workspace and workflow

Defining our workspace and workflow is an important part of the process to establish a common ground and approach to solving the company's problem.

We discussed and agreed upon working within the **SCRUM** methodologies because we believe that it would best fit our workflow and give us the best way to adapt to possible changing requirements during development process.

Tools

In this paragraph, we will state the tools that we plan to use during the development process.

1. **GitHub**

We will be using GitHub to keep our source code safe and organized to separate it from any new features that we are currently developing.

2. **Visual Studio**

Visual Studio will be the main IDE that we would be developing in.

3. **SQL Management Studio**

We will be using it to manage our SQL database and server.

4. **Discord**

The communication between the team and the product owner will be mainly established through Discord.

Development Methodologies

During the length of our education, we have been introduced to a handful of development methodologies such as Unified Process, Agile approach, Waterfall. To conclude the best fitting methodology for our project, we will analyze how each of them will impact our development.

1. Unified Process¹

Unified process is the first development methodology we got introduced to and spent a great amount of time learning and practicing under it. It is an iterative, incremental, and risk-driven process.

The reason we believe Unified Process is not suitable for our process is because it is under the influence of changes and dynamic workflow. Because Unified Process forced you to work in defined phases (Inception, Elaboration, Construction, and Transition), it is expected that each iteration will have certain activities that would spike and generate documentation early on making it harder to adapt.

2. Agile approach²

Like Unified Process, the Agile approach also promotes iterative, incremental, and risk-driven workflow. However, each iteration is time-boxed and contains the amount of effort the team needs to deliver in the end of it.

Because changes influence our process, Agile with its subset development processes - SCRUM and XP – would strongly benefit us in a way so at the end of the line we will produce a product worthy of the client.

3. Waterfall³

Waterfall is a linear development process which is not adaptable to changes and once all the requirements from the customer have been gathered his presence is not strictly required during the process.

The workflow of this development process would not benefit us in any certain way because we need to be able to adapt to changes and our customer is strongly involved in the development process.

Conclusion

After analyzing all the methodologies, we have knowledge of, we concluded that using the Agile Approach with SCRUM and XP and mix it with some techniques from Unifies Process, would be the most fitting for our development process.

Because the team is relatively small and our client's opinion strongly affects the development course, Agile will allow us to quickly capture and adapt to the dynamics.

¹ <http://www.chiron-solutions.com/chiron-professional-journal/2010/12/20/what-is-the-difference-between-rup-and-scrum-methodologies/>

² <https://www.cprime.com/resources/what-is-agile-what-is-scrum/>

³ <http://www.seguetech.com/waterfall-vs-agile-methodology/>

Agile Techniques

The Agile Framework provides a variety of techniques and tools for a development process. As the framework promotes that artifacts should be used if only they are deemed necessary, we are going to use a few of them with the consideration that some may be dropped or new ones included along the way.

1. **User Stories**
To understand what kind of functionalities our Product Owner wishes the system to execute.
2. **Planning Poker**
To understand the priority and difficulty each user story possesses and estimate our Velocity.
3. **Velocity Tracking**
To keep track of the progress the team makes during a sprint.
4. **Product Backlog**
To have an overview of all the User Stories we plan to do throughout the development process.
5. **Sprint Backlog**
To have an overview of the Estimation and User Stories we plan to do during each sprint.
6. **Sprint Retrospective**
To look back, analyze and reflect how each sprint went and understand how successful the strategy we used to complete the sprint was and estimate if the process will result in delivering a finished product.
7. **Scrum Meetings**
To review the activities performed during the sprint and if the team has managed to complete the estimated velocity. As well as, estimate the effort for the next sprint.
To keep proximity with our Product Owner and ensure that our requirements list is always up to date, as well as the parts of the product we deliver are quality or not.
8. **Sprint Zero**
To establish the necessary environment for the team to begin the development process.
9. **Collective code ownership**
To have the ability for each one of us to modify the other's code if needed and deems that there is a better solution to a problem.
10. **Pair Programming**
To help each other during the times where we would have to focus on developing parts of the system in which our weaknesses are most profound at. That way we will ensure that we create error proof code and establish an understanding and in-depth knowledge of the problem.
11. **Continuous Delivery of Valuable Software**
To ensure that at the end of each sprint we have a part of the system ready to use and work with the mindset that the system must always be ready for release.

Estimating Velocity

When taking this project, both of us had in mind that we were both unexperienced in the Web Development subject and to add to that, we also must learn a new front-end Framework along the way. When looking at the requirements we received from our Product Owner, we discussed that the hardest part of building each user story would be establishing the front-end. When combining the Estimation Points of all the User Stories, we result in having 66 in total and we concluded that we will devote two weeks for each sprint. Because of this, we will need to achieve 22 Estimation points per Sprint.

Unified Process techniques

As well as using techniques from the Agile Framework, we plan to use some from Unified Process.

1. Business Model Canvas

To understand the business domain the system we are building is going to be deployed in.

2. Database Design

To ensure that we understand what entities are involved in the problem domain and how they are related.

3. Sequence Diagram

To visualize a dynamic view of some user stories.

Project Plan

The first week of a project started at the 24th of April and the end date is 9th of June. We decided that each sprint of our process will be 2 weeks.

Because the period is so tight, our daily work schedule will have an average of 8 hours including Weekends.

	Week	1	2	3	4	5	6	7
Sprint / Phase								
Sprint 0		█						
Sprint 1			█	█				
Sprint 2					█	█		
Sprint 3							█	█
Report		█	█	█	█	█	█	█
Deadline								█

Project Establishment Conclusion

The Project Establishment phase of our development process, helped us gain an overview of which would be the most fitting methodology and what techniques and tools would be most beneficial.

In addition, an overview of the total effort needed to realize this product and the willingness the team possesses was established.

To conclude, our analyzation of the techniques and tools we need to realize this product were not consistent because of the problems we encountered during designing of the Database. We could have used more artifacts from Unified Process such as Problem Domain Diagram to ensure a concrete and finalized vision of the Database.










Sprint Zero

The Inception phase of our product will be marked as Sprint 0. The purpose of this sprint is to initiate the project by first understanding the complexity and length of the workload.

In this cycle, we aim to understand the scope of the product by perceiving the Product Owner's Vision, map down the company's business that we are developing the product for, define User Stories and the Product Backlog, design the database, estimate, and prioritize the Product Backlog.

Business Model Canvas

The first meeting we held with our Product Owner was to understand the company's business strategy and structure. We needed this meeting to perceive the area the product will be serving purpose in.

<p>Key Partners </p> <p>Key partners Professional players Professional gaming organizations Gaming communities Schools with e-sport interest</p> <p>Key suppliers Professional players Professional gaming organizations Gaming communities Schools with e-sport interest</p> <p>Key Resources Human resources: Business manager Multimedia manager Human resource manager Professional players</p> <p>Intellectual resources: Knowledge about gaming that professional players provide. Knowledge about the target group.</p> <p>Partners Key Activities Professional players mentor customers Professional Gaming organizations provides professional players Gaming communities provides customers and professional players Schools with interest in e-sport/gaming provides customers</p>	<p>Key Activities </p> <p>What key activities do our Value Propositions require? Mentoring Supplying mentors</p> <p>Customer relationships Dedicated personal assistance</p> <p>Revenue streams Usage fee Lending/Leasing/Renting Brokerage fees Licensing</p> <hr/> <p>Key Resources </p> <p>What Key Resources do our Value Propositions require? Mentoring</p> <p>Distribution Channels Company software</p> <p>Customer Relationships Dedicated personal assistance</p>	<p>Value Propositions </p> <p>What value do we deliver to the customer? The value we deliver to the customers is mentoring in gaming with comfortable hours.</p> <p>Which one of the customers' problems are we helping to solve? We are helping to solve the problem of establishing a credible and easy communication between a customer and a professional gamer. We are also helping gamers to find the best mentorship</p> <p>What bundles of products and services are we offering to each Customer Segment? We offer affordable mentoring hours. An easy way for gamers to find a mentor and communicate with them. Credible communication and resources</p>	<p>Customer Relationships </p> <p>What type of a relationship does each of our Customer Segments expects us to establish and maintain with them? Dedicated personal assistance</p> <p>How costly are they? Initially high Linear reduction</p> <hr/> <p>Channels </p> <p>Through which Channels does our Customer Segments wish to be reached? Through the company software</p> <p>How are we reaching them now? Through the company software which is currently under development. Reaching customers through private channels.</p>	<p>Customer Segments </p> <p>For whom are we creating value? Professional players Gamers who wish to be mentored on a professional level.</p> <p>Who are our most important customers Gamers who wish to be mentored on a professional level.</p>
<p>Cost Structure </p> <p>What are the most important costs inherent in our Business Model Getting mentors</p> <p>Which key resources are most expensive Professional players / mentors</p> <p>Which key activities are most expensive Supplying mentors</p>		<p>Revenue Streams </p> <p>For what value are our customers really willing to pay Resourceful, competent and famous professional players</p> <p>How are the currently paying Private channels</p> <p>How much does each Revenue Streams contribute to the overall Revenues They contribute to all segments in the business</p>		

The business revolves around providing customers with affordable and quality mentoring. The gears that make this business move is mainly the Professional Gamers, Gaming communities, Gaming Organizations, and Schools with interest in e-sports. They provide the business with the Value, Clientele, and Employees. The company helps its customers find the best mentor and provides easy communication between them.

In addition, the relationship the company is trying to maintain with their customers has been marked as Dedicated personal assistance – each mentor has the responsibility to personally handle each customer and provide them with the hours of mentoring they have purchased.

The Product Owner marked the relationship as being linear reductive and by that he means that at the start, the costs for maintaining these relationships will be costly for the overall business, however as the clientele increases, these costs would be compensated.

User Stories

The team and the Product owner held another meeting during which the Product owner addressed core requirements that he had towards the system in development. Based on the requirements he addressed, we mapped down a list of functionalities that the system needs to possess.

However, because the idea the Product Owner has is still new, a lot of the requirements we received were inconclusive and contained very general explanations. As a result, we were unsure of the specifics of the requirements we received and we frequently had to stay in contact with the Product Owner to make sure the user stories we were creating are depicting his needs.

Staying in close contact with the Product Owner, resulted in being beneficial for us because we managed to capture the functionalities that are dire for the system. Despite most of the user stories being crucial, we

discussed with the Product Owner that within this period some of them will be impossible to complete and will be taken into consideration after project time.

After having established an overview of the core user stories, we estimated the effort each of them would require of the team to complete by using the Planning Poker. By having a complete estimation of the core user-stories, we concluded that the total effort needed to build this system within this period is attainable.

Product Backlog

The meeting and discussions with the Product Owner helped us build the system's Product Backlog. Because we have user stories that will be impossible to complete within the project time, we created two backlogs – one with user stories that will be completed within the project time and one that will be considered after project time.

1. Product Backlog with user stories that will be completed with project time

ID	AS A	I WANT	SO THAT	BUSINESS VALUE	ESTIMATION POINTS
1	User	To create an account on the website	I can make a booking	300	5
2	Customer	To buy hours	I can book a lesson with a mentor	300	5
3	User	To see and select the game and role I want	I can see what kind of mentors they offer	300	3
4	Mentor	To be able to see my schedule	I can have an overview of all my booked lessons	200	5
5	User	To be able to cancel a booking	I can notify the mentor if I cant attend the lesson	200	8
6	User	My profile information to be secure	I wouldn't have to worry about being hacked	300	5
7	User	To be able to give feedback after my lessons	Changes or improvements to the lessons could be made	200	3
8	User	I want to be able to log in into the system	I can use the services the system provides	300	5
9	Admin	To be able to see all bookings of all the mentors	I can assign a supervisor for the lesson	200	2
10	Admin	To see all supervisors	I can have an overview of them	200	2
11	User	To be able to reset my password	I can change it if I have forgotten it	300	8
12	Mentor	To be able to send my application	I can become a mentor	300	5
13	User	To be able to make a Booking	I can have a lesson with a mentor	300	3
14	Admin	To be able to change the status of a Mentor	I can Accept or Decline their applications	300	3

2. Product Backlog with user stories that will be completed after project time

The reason we deemed these user stories as impossible to do within the project time is because they are addressing a problem that we are not yet familiar with and we would time to research and understand it.

Product Backlog (Not doable)					
ID	AS A	I WANT	SO THAT	BUSINESS VALUE	ESTIMATION POINTS
6	Mentor/Client	To be able to communicate with my mentor/client	I can learn about his/her experience, expectations, improvement etc.	300	8
11	Supervisor	To be able to spectate lessons	I can make sure that the lessons are going smoothly	200	20
14	Product Owner	To be able to see all feedback and requests that have been made	I can have an overview of them and manage them	100	∞

Database Design

After completing the user stories, we had an easy time to find and address the entities our problem domain will consist of. However, as our database design progressed, we discovered that we had some unnecessary entities and this was one of the main reasons why our database design changed numerous times.

First Attempt

ENTITIES

Access role	Access Role – the purpose of this entity is to store and define each user’s role in the system. Because we have different types of users, we concluded that having an entity that would represent each one of them would be the best solution. Mentor – a Mentor is the entity that people can book to get taught in certain kind of a game or role. A mentor represents a player who has deep knowledge and professional view of strategic and gameplay analyzation. Without a Mentor, there is no Game or Role. In other words, if there is no Mentor in a certain Game that means you cannot be taught in this game and role. User – a User is an entity that can have a different meaning depending on what kind of an access role it has been assigned. We have a Basic User/Customer, an Admin, and a Supervisor.
Mentor	
User	
Ticket	
Booking	
Game	
Role	
Schedule	

The Basic User/Customer is the one that will be hiring a mentor to teach him in specific game and role.

An Admin is someone who have full authority in the system.

Supervisor is someone who will be assigned by the Admin to a booking to make sure a lesson is going in the correct way.

Ticket – represents the quantity and hours our Basic User has purchased.

Booking – represents the hours a Basic User has purchased for a specific Game, Role and Mentor.

Game – represents the games available for mentoring

Role – represents the roles available under a specific game

Schedule – represents a Mentor’s bookings.

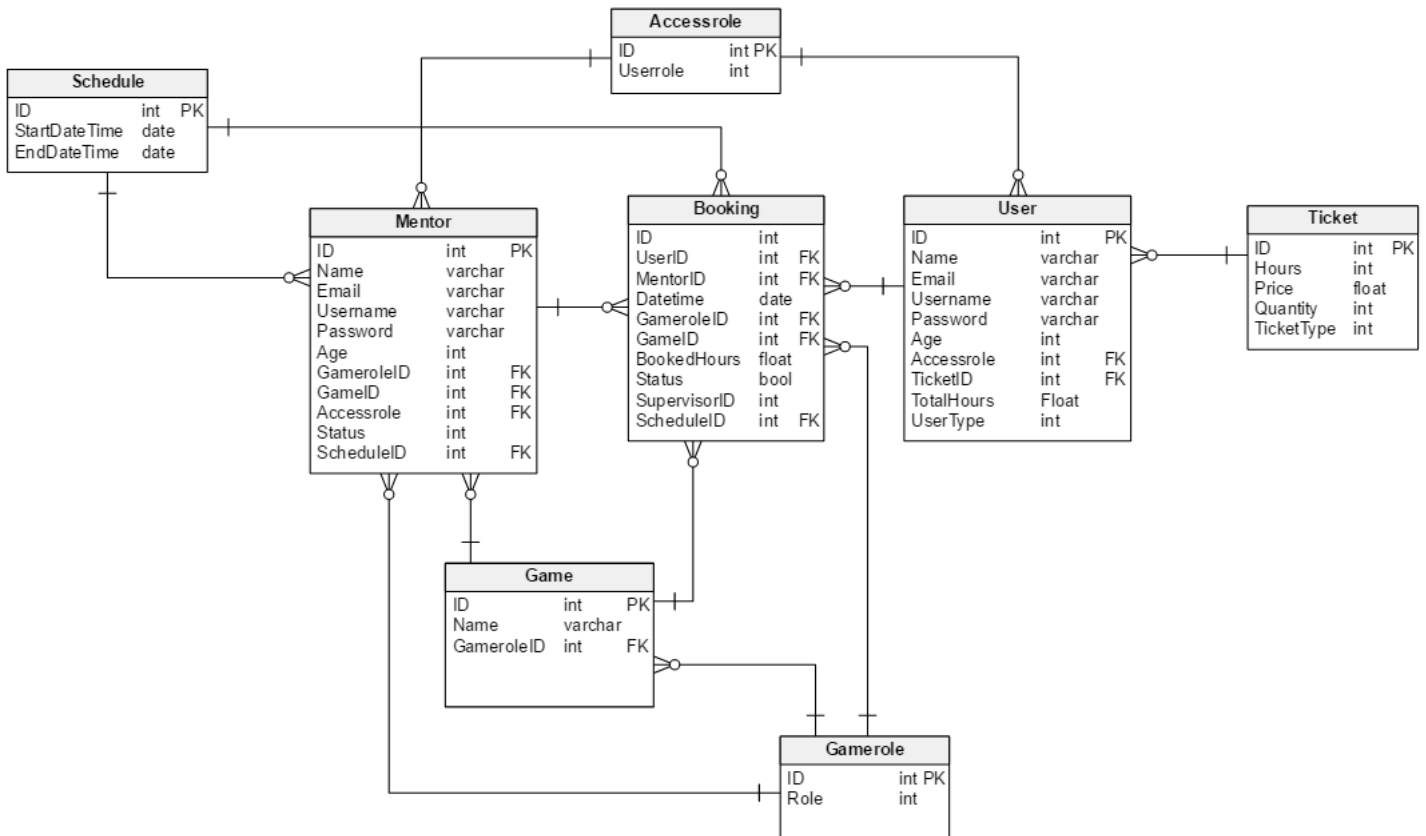
Identifying Entity Relationships

ENTITY	ENTITY	RELATIONSHIP
MENTOR	Booking	1 Mentor can have many Bookings
BOOKING	Mentor	1 Booking can only have 1 Mentor
USER	Booking	1 can have many Bookings
BOOKING	User	1 Booking can only have 1 User
USER	Ticket	1 User can have 1 Ticket
TICKET	User	1 Ticket can have many Users
ACCESSROLE	Mentor	1 AccessRole can have many Mentors
MENTOR	AccessRole	1 Mentor can only have 1 AccessRole
ACCESSROLE	User	1 AccessRole can have many Users
USER	AccessRole	1 User can only have 1 AccessRole
GAME	Role	1 Game can have 1 Role
ROLE	Game	1 Role can have many Games
MENTOR	Game	1 Mentor can have 1 Game
GAME	Mentor	1 Game can have many Mentors
MENTOR	Role	1 Mentor can have many Roles
ROLE	Mentor	1 Role can have 1 Mentor
BOOKING	Game	1 Booking can have 1 Game
GAME	Booking	1 Game can have many Bookings
BOOKING	Role	1 Booking can have 1 Role
ROLE	Booking	1 Role can have many Bookings
SCHEDULE	Mentor	1 Schedule can have many Mentor
MENTOR	Schedule	Many mentors can have 1 schedule
SCHEDULE	Booking	1 Schedule can have many Bookings
BOOKING	Schedule	1 Booking can be under 1 Schedule

Cardinalities

ENTITY	ENTITY	RELATIONSHIP
MENTOR	Booking	1...*
USER	Booking	1...*
USER	Ticket	1...*
ACCESSROLE	User	1...*
ACCESSROLE	Mentor	1...*
USER	Ticket	1...*

GAME	Role	1...*
GAME	Mentor	1...*
ROLE	Mentor	1...*
BOOKING	Game	1...*
BOOKING	Role	1...*
SCHEDULE	Mentor	1...*
SCHEDULE	Booking	1...*



First Attempt – Conclusion

When we completed our first attempt at designing the problem domain of our product, we had a meeting with our Supervisor and he pointed some uncertainties and mistakes with our approach.

To start, the relationships we had between some of the entities were questionable and during the meeting we concluded that the entity Schedule should not exist as an entity because a Mentor’s Schedule is the amount of Bookings he has.

After the meeting, our supervisor advices to dedicate a few days to analyze and find the best solution for the database design.

Second Attempt

Our initial approach suffered a lot of changes. First, we eliminated one of our entities – Schedule.

ENTITIES

ACCESS ROLE
MENTOR
USER
TICKET
BOOKING
GAME
ROLE

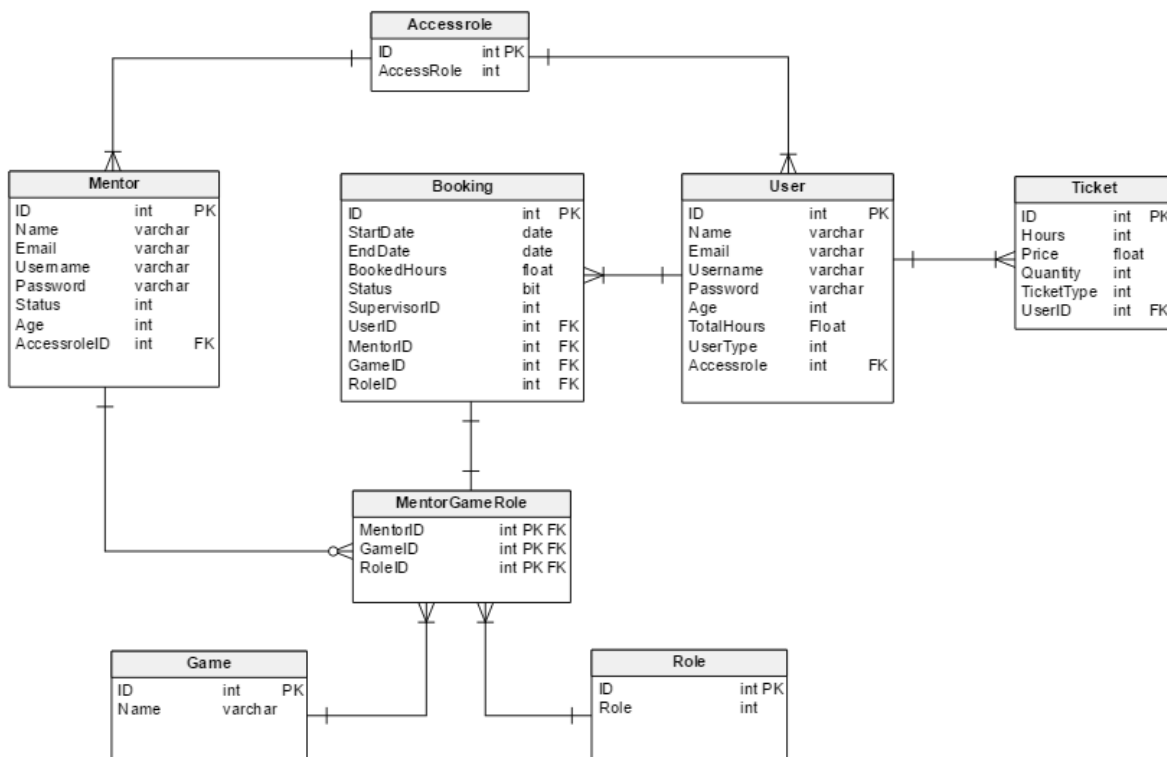
Identifying Entity Relationships

We took a lot of time to discuss the relationships between the Entities Mentor, Game, and Role because this part of the domain resulted in a lot of confusion.

ENTITY	ENTITY	RELATIONSHIP
MENTOR	Booking	1 Mentor can have many Bookings
BOOKING	Mentor	1 Booking can only have 1 Mentor
USER	Booking	1 can have many Bookings
BOOKING	User	1 Booking can only have 1 User
USER	Ticket	1 User can have many ticket
TICKET	User	1 Ticket can only have 1 User
ACCESSROLE	Mentor	1 AccessRole can have many Mentors
MENTOR	AccessRole	1 Mentor can only have 1 AccessRole
ACCESSROLE	User	1 AccessRole can have many Users
USER	AccessRole	1 User can only have 1 AccessRole
GAME	Role	1 Game can have many Roles
ROLE	Game	Many Roles can have 1 Game
MENTOR	Game	1 Mentor can have many Games
GAME	Mentor	1 Game can have many Mentors
MENTOR	Role	1 Mentor can have many Roles
ROLE	Mentor	1 Role can have many Mentors
BOOKING	Game	1 Booking can have 1 Game
GAME	Booking	1 Game can have many Bookings
BOOKING	Role	1 Booking can have 1 Role
ROLE	Booking	1 Role can have many Bookings

Cardinalities

ENTITY	ENTITY	RELATIONSHIP
MENTOR	Booking	1...*
USER	Booking	1...*
USER	Ticket	1...*
ACCESSROLE	User	1...*
ACCESSROLE	Mentor	1...*
USER	Ticket	1...*
GAME	Role	*...*
MENTOR	Game	*...*
MENTOR	Role	*...*
BOOKING	Game	1...*
BOOKING	Role	1...*

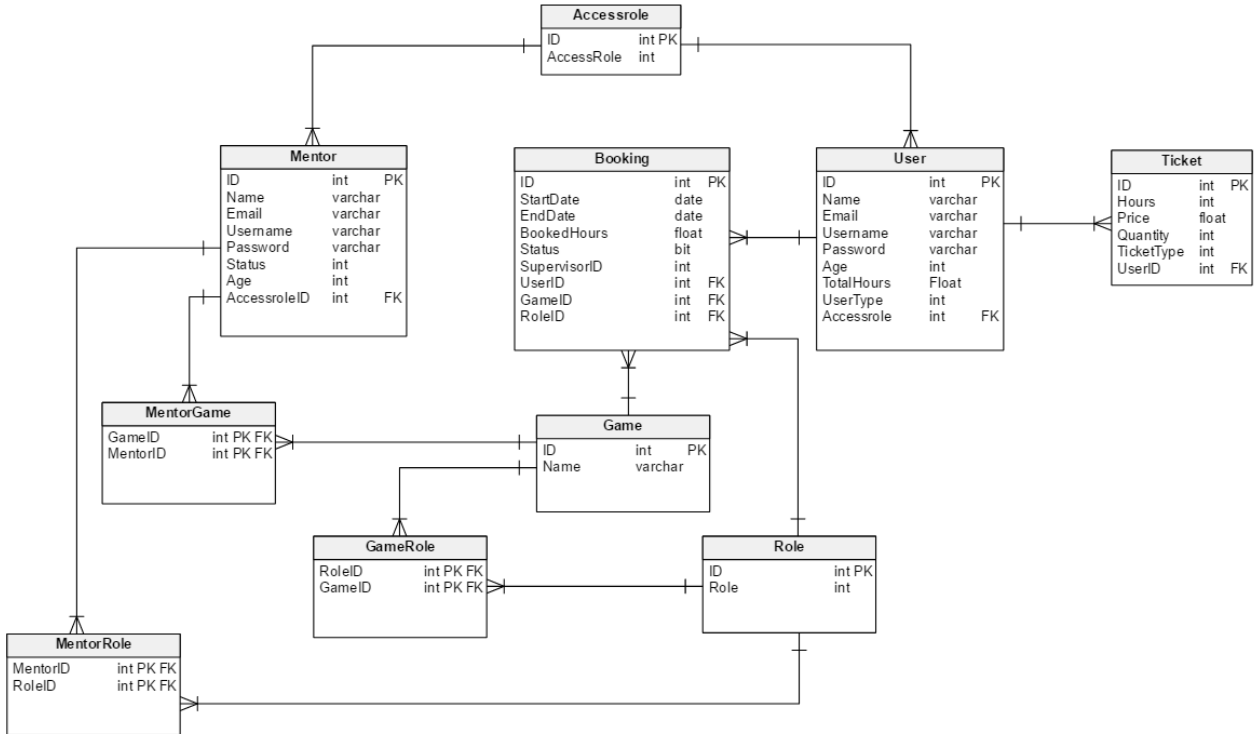


Second Attempt - Conclusion

Because we have a lot of many to many relationships between “Mentor”, “Game”, and “Role”, we decided to combine their relationship into one relation that would represent it – “MentorGameRole”. However, when trying to extract the information from the database, we had some difficulty with our queries that we could not overcome and we had to drop this solution.

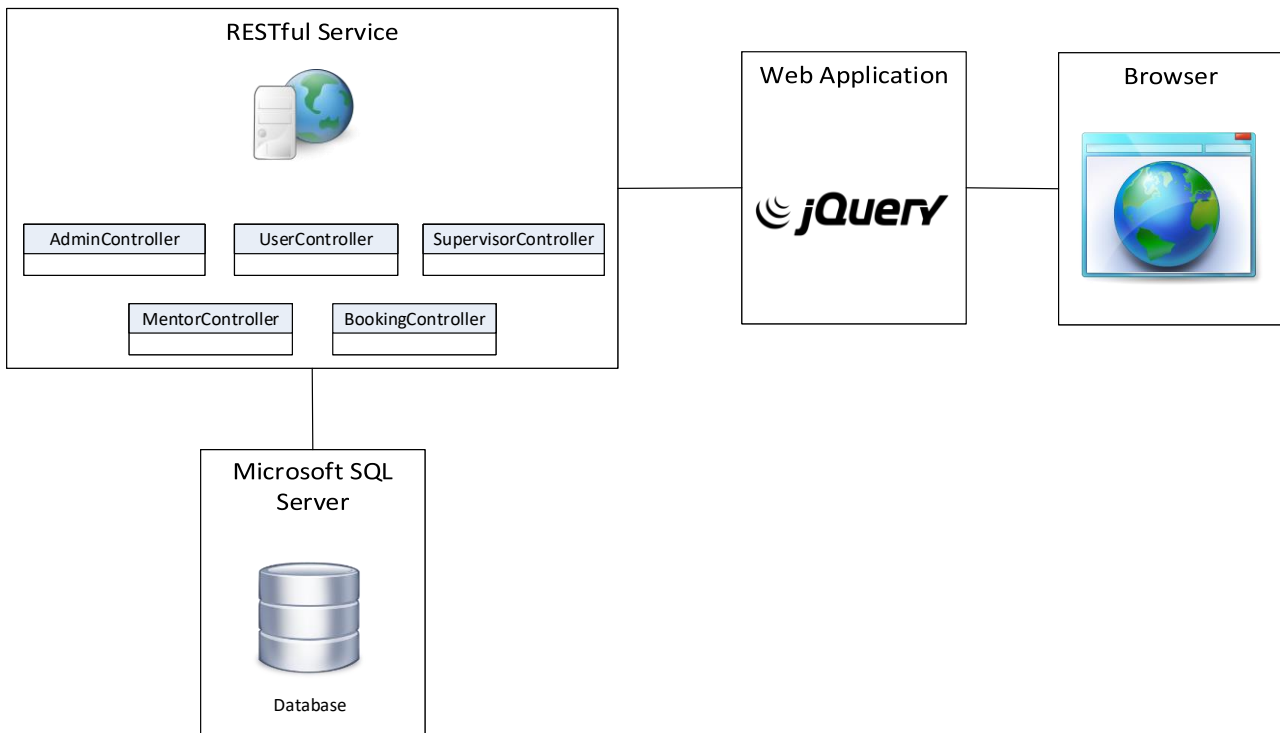
Final Attempt – Current Database Solution

The final solution of our Database Design did not suffer many changes from the Second Attempt, however we expressed each one of the many-to-many relationships in the standard approach. The reason we chose this solution is because we found it as the easiest to work with.



Architecture

The architecture of our system consists of three layers – “Database”, “RESTful Service”, and “Web Application”. The Database is a relational database made with SQL and hosted on “Microsoft SQL server”. The service is a “WCF RESTful” service and the architecture we have chosen for it is modular – for each part of the system we have created separate controllers that will handle the requests from the client side. The client-side is a “Web Application” made in “jQuery” and we are using a new front-end framework for it – “Foundation 6”.



Design Patterns

To ensure that we have fully addressed the issues for the system we are building and ensure improved code readability, we used several Design Patterns that we have been acquainted with during our education.

GRASP

From “GRASP”, we used the design patterns – “Low Coupling”⁴ and “High Cohesion”. We have separated our service into different modules that handles different – “User”, “Booking”, “Admin”, “Mentor”, and “Supervisor” controller. The reason we use “Low Coupling” is because it gives a well-defined structure of our system and better code readability.

To ensure that we achieve good readability and maintainability, we combine Low Coupling with High Cohesion because this will make sure that each “controller” in our system handles their specific request.

⁴ <https://www.cs.colorado.edu/~kena/classes/5448/f12/presentation-materials/rao.pdf>
<https://www.utdallas.edu/~chung/SP/applying-uml-and-patterns.pdf>

Proxy Design Pattern⁵

To ensure that sensitive data is not exposed to the client-side, we use the “Proxy Design Pattern”. More specifically, we are using a “protected proxy control access” that allows us to create a “placeholder” for certain objects.

Code Reuse Pattern

We use this pattern because a lot of our User Stories would have repeatable code and behavior.

Sprint Zero – Review

This part of the process was very crucial for our development, because we had the time to address important issues regarding our product. The most considerable difficulty we faced was reaching a decision and finalizing the Database design some of which carried on into the First Sprint of the development phase and set us back in time.

If we had addressed the Database Design into a more considerable and analyzed way from the very beginning we could have saved ourselves the confusion and crucial time spent fixing the database.

Overall, the other artifacts were not difficult to produce because we had close contact with our Product Owner who gave us constant feedback that we could adapt to.

Sprint One – May 1st – 14th 2017

During the First Sprint of our development process, we will be establishing the architecture for the system and building up the database. We concluded that we will be using a RESTful architecture for our service, a relational database with SQL and new front-end framework – Foundation.

Before we proceed with the implementation of the sprint user stories, the team had to establish a common ground about the architecture we plan to use for our service. We agreed upon using a WCF RESTful service with a Model-view-controller architectural pattern. The reason we chose WCF is because it’s relatively easy to expose and consume.

Sprint Backlog

The Estimation for this sprint will be a total of 25 which is a little more than what we estimated for a sprint. These user stories were picked based on the Product Owner’s desire of prioritization.

As a: Mentor
I want: To be able to apply
So that: I can become a mentor to the system
Business Value: 300
Estimation Points: 5
Responsible: Jonas

As an: Admin
I want: To be able to see all bookings of all the mentors
So that: I can assign a supervisor
Business Value: 200
Estimation Points: 2
Responsible: Jonas

⁵ https://sourcemaking.com/design_patterns/proxy

<p>As a: Mentor I want: To be able to see my schedule So that: I can have an overview of all my booked lessons Business Value: 200 Estimation Points: 5 Responsible: Jonas</p>
<p>As a: User I want: To create an account on the website So that: I can make a booking Business Value: 300 Estimation Points: 5 Responsible: Stefani</p>
<p>As a: User I want: My profile information to be secured So that: I wouldn't have to worry about being hacked Business Value: 300 Estimation Points: 5 Responsible: Stefani</p>
<p>As a: User I want: To see and select the game and role I want So that: I can see what kind of mentors they offer Business Value: 300 Estimation Points: 3 Responsible: Stefani</p>

User Stories

We will only reveal aspects of our user stories that contain interesting points and code for the report and the rest would be placed into the appendix.

User Story 1 – Secure Information

Written by: Stefani

The first user story that was tackled was securing information on “User Creation” or “Password Encryption”. The user story was chosen as first because most of our user stories for Sprint 1 are related to creating an account in the system.

The first approach on solving this problem was by using the “PBKDF2”⁶ Key derivation function and the “Rfc2898DeriveBytes” class which uses “PBKDF2” based on “HMACSHA1”, backed with salt.

The variables “SaltByteSize” contains the size of our salt and “HashByteSize” contains the size of the outputted hash of the password. The variable “HashingIterations”, is needed to tell how many iterations the “PBKDF2” should perform with the hash and the password along with the salt. By default, the iterations are 1000, however 10000 and more are commonly recommended⁶.

The multiple iterations, in case of brute-force attacks, would have an excessive impact on an attacker to guess the password, because he would have to perform 10000 hash iterations for each password guess.

⁶ <https://cryptosense.com/parameter-choice-for-pbkdf2/>

The method "GenerateSalt" is using the "RNGCryptoServiceProvider" which ensures that each user has a unique and random generated salt. In the end, the method will return a byte array that would contain our salt.

```
private const int SaltByteSize = 24;
private const int HashByteSize = 24;
private const int HashingIterations = 10101;
internal static byte[] GenerateSalt()
{
    int saltByteSize = SaltByteSize;
    using (RNGCryptoServiceProvider saltGenerator = new RNGCryptoServiceProvider())
    {
        byte[] salt = new byte[saltByteSize];
        saltGenerator.GetBytes(salt);
        return salt;
    }
}
```

The next method is the one that we will be hashing the password with the salt. The method "ComputeHash" takes the plain password, the salt we just generated, the amount of iterations that it needs to go through and the size of the outputted hash as parameters.

To generate the hash, we will be using "Rfc2898DeriveBytes" class that we inherit from the "System.Security.Cryptography" namespace that takes as parameters the plain password and the salt array. In the end, we will be returning the array of the outputted hash.

```
internal static byte[] ComputeHash(string password, byte[] salt, int iterations =
HashingIterations, int hashByteSize = HashByteSize)
{
    using (Rfc2898DeriveBytes hashGenerator = new Rfc2898DeriveBytes(password, salt))
    {
        hashGenerator.IterationCount = iterations;
        return hashGenerator.GetBytes(hashByteSize);
    }
}
```

To ensure that the password a user has inputted is the correct one, we need to hash and salt the input of the user and compare it with one in the database.

The method "VerifyPassword" takes the inputted password as plain text, the salt generated and the hash from the database. Then it will create a new hash for the inputted password together with the salt and the method "AreHashEqual" will evaluate if the hash of the inputted password is the same as in the database.

```

internal static bool VerifyPassword(string password, byte[] passwordSalt, byte[] dbHash) {
    byte[] inputHash = ComputeHash(password, passwordSalt);
    return AreHashEqual(inputHash, dbHash);
}

private static bool AreHashEqual(byte[] inputHash, byte[] dbHash) {
    int minHashLength = inputHash.Length <= dbHash.Length ? inputHash.Length : dbHash.Length;
    var testValue = inputHash.Length ^ dbHash.Length;
    for (int i = 0; i < minHashLength; i++)
    {
        testValue |= inputHash[i] ^ dbHash[i];
    }
    return 0 == testValue;
}

```

However, because the “Rfc2898DeriveBytes” is based on “HMACSHA1” and there have been recent publications about proved collisions⁷ of “SHA1” which could potentially pose a threat to the security of a user’s password.

The collisions that were recently documented are the possibility of being able to hash two entirely different documents with the same “SHA-1” signature. Because of that we had to research and experiment with different hashing algorithms.

The second approach we chose to solve the User Story, was by simply using “SHA-512” with salt. However, there are lot of mixed feelings about using fast cryptographic hashing functions and most of the sources⁸ recommend using key stretching algorithm such as the “PBKDF2”.

```

internal static byte[] ComputeHash(string password, byte[] salt, int iterations =
    HashingIterations, int hashByteSize = HashByteSize)
{
    //using SHA512 with salt without PBKDF2
    byte[] plainPWWithSalt = new byte[password.Length + salt.Length];
    using (SHA512Managed sha512Hash = new SHA512Managed())
    {
        //Compute hash of plain PW with salt
        byte[] hashedPW = sha512Hash.ComputeHash(plainPWWithSalt);
        //Craete array with hashed password and salt
        byte[] hashWithSaltBytes = new byte[hashedPW.Length + salt.Length];
        //Copy hash into resulting array
        for (int i = 0; i < hashedPW.Length; i++)
        {
            hashWithSaltBytes[i] = hashedPW[i];
        }
        //Append salt bytes to result
        for (int i = 0; i < salt.Length; i++)
        {
            hashWithSaltBytes[hashedPW.Length + i] = salt[i];
        }
        return hashWithSaltBytes;
    }
}

```

⁷ https://www.theregister.co.uk/2017/02/23/google_first_sha1_collision/
<https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html>

⁸ <https://crackstation.net/hashing-security.htm> - Frequently Asked Question paragraph
<https://security.stackexchange.com/questions/16354/whats-the-advantage-of-using-pbkdf2-vs-sha256-to-generate-an-aes-encryption-key>
<https://adamgard.com/blog/3-wrong-ways-to-store-a-password/>

The standard implementation⁹ of “PBKDF2” uses “SHA-1” as a hashing algorithm and because of the collisions, we had to derive from it and find out if it is possible to use another.

After researching, we stumbled upon a Nugget Package that was available for Visual Studio – “Microsoft.AspNetCore.Cryptography.KeyDerivation”¹⁰. It enables you to use “PBKDF2” with “SHA-512” and “SHA-256”.

The implementation of it was straightforward – the method would still take the same parameters as the first approach. However, with this nugget package you can specify the hashing method you wish to use - “PBKDF2”.

```
internal static byte[] ComputeHash(string password, byte[] salt, int iterations =
HashingIterations, int hashByteSize = HashByteSize)
{
    //using PBKDF2 with SHA512
    var keyDerivationPrf = KeyDerivationPrf.HMACSHA512;
    byte[] hash =
    KeyDerivation.Pbkdf2(password, salt, keyDerivationPrf, iterations, hashByteSize);
    return hash;
}
```

User Story 2 – Become a Mentor

Written by Jonas

At the beginning of the sprint we had a meeting with our Product Owner in which we reviewed the Product Backlog again. During the meeting, he addressed some concerns he had with one of the user stories – “Become a Mentor”.

Initially, he wanted to be the one that creates the accounts for each “Mentor Application” he receives; however, he wishes to see all the mentor applications and simply just change their status to Approved or Declined. Because of that we had to change the acceptance criteria of the user story.

Backend

The first step taken to solving the User Story is updating our table “Mentor” in the database with an additional column called “Status” that has a datatype of “Integer”. This column is going to represent the Integer value of an “Enumeration” called “MentorStatus” in our service.

When a user applies to become a mentor, he will initially be given a status of “AwaitingApproval” and when the administrator updates the status to “Approved”, he will be given the authority to log into the system, otherwise he will be unauthorized if his status has been changed to “Declined”.

When a “Mentor” creates an application, a validation is made on the “Username” and “E-mail” to check if a record already exists in the database. Based on the check a custom “HTTP” response is returned to the client by using a “WebOpeartionContext¹¹” helper class. Also, the “AccessroleID” upon creation is hard-coded to 3 which corresponds to “Mentor” in the database.

In addition, we also had to set the configuration of the database context “ProxyCreation” to false, otherwise we were having issues with receiving any response from the service.

The interesting point of this method is the “Many-To-Many” relationship and how to connect this object with the already existing Games and Roles. The first thing we do is that we save the collection of Games

⁹ <https://helpdesk.lastpass.com/account-settings/general/password-iterations-pbkdf2/>

¹⁰ <https://docs.microsoft.com/en-us/aspnet/core/api/microsoft.aspnetcore.cryptography.keyderivation>

¹¹ [https://msdn.microsoft.com/en-us/library/system.servicemodel.web.weboperationcontext\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.servicemodel.web.weboperationcontext(v=vs.110).aspx)

and Roles from the requested object and afterwards we try to find the specific game that Mentor has put into the Application. To ensure that Entity Framework is not going to create a new Game or Role object from the data, we use the method “Attach”¹² which attaches the existing Game and Role. Afterwards, we simply save the changes.

```
mentor.Game = newMentor.Games;
mentor.Role = newMentor.Roles;
var game = mentor.Game.FirstOrDefault(x => newMentor.Games.Any(g => g.ID == x.ID));
var role = mentor.Role.FirstOrDefault(x => newMentor.Roles.Any(g => g.ID == x.ID));
db.Game.Attach(game);
db.Role.Attach(role);
db.Mentor.Add(mentor);
db.SaveChanges();
```

Frontend

After establishing the back-end for this user story, we proceeded with setting up the front-end and learning the basics of coping with the new framework – Foundation 6.

At first glance, working with Foundation 6 did not seem very hard. It resembled Bootstrap a lot – a front-end framework that we have been introduced throughout our education.

However, because Foundation has a lot of different releases and their documentation is not very consistent with which modules work on the newest release – Foundation 6. That was the main issue that caused us a lot of problem when building up the front-end and took time to understand.

The way we built the front-end of this user story is by using a class “Reveal¹³” and adding a tag to the element “data-reveal” that Foundation 6 provides. It is a pop-up dialog that gets revealed on clicking a specified element on the web page.

The element that is going to “reveal” the dialog is a link on the web page called “Become a Mentor”. The link itself contains a tag “data-open” that takes the ID of the div as a parameter.

```
<div class="reveal" id="becomeMentorReveal" data-reveal>
```

```
<li class="show-for-medium"><a data-open="becomeMentorReveal">BECOME A MENTOR</a></li>
```

¹² [https://msdn.microsoft.com/en-us/library/jj592676\(v=vs.113\).aspx](https://msdn.microsoft.com/en-us/library/jj592676(v=vs.113).aspx)

¹³ <http://foundation.zurb.com/sites/docs/reveal.html>

The image shows a web form titled "Mentor Application" with a dark blue header and footer. The header contains navigation links: "MENTOR", "PRICES", "ABOUT US", and "CONTACT". The form fields are as follows:

- Full Name:
- E-mail:
- Username:
- Password:
- Age:
- Game:
- Role:
- Tell us a little about yourself:

At the bottom of the form is a blue "Submit Application" button and a "Nevermind" link.

The way we consume our service in the client is by using “AJAX” calls. The interesting about an “HTTP POST” with “AJAX” is that for it to get through the client to the server, you would have to configure your server to accept Cross-Domain requests.

The way we configured our service to accept these types of requests is by creating a “Global.asax”¹⁴ file – an application file that contains code for responding to application-level events.

The file contained several methods but the one we needed to implement was the “Application_BeginRequest”. By adding the “Access-Control-Allow-Origin¹⁵” header we are telling the server to accept any domain in a cross-site manner.

¹⁴ [https://msdn.microsoft.com/en-us/library/1xaas8a2\(v=vs.71\).aspx](https://msdn.microsoft.com/en-us/library/1xaas8a2(v=vs.71).aspx)

¹⁵ https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS

```
protected void Application_BeginRequest(object sender, EventArgs e)
{
    HttpContext.Current.Response.AddHeader("Access-Control-Allow-Origin", "http://localhost");
    if (HttpContext.Current.Request.HttpMethod == "OPTIONS")
    {
        HttpContext.Current.Response
            .AddHeader("Access-Control-Allow-Methods", "POST, PUT, DELETE");
        HttpContext.Current.Response
            .AddHeader("Access-Control-Allow-Headers", "Content-Type, Accept");
        HttpContext.Current.Response
            .AddHeader("Access-Control-Max-Age", "1728000");
        HttpContext.Current.Response.End();
    }
}
```

In addition to that, we also had to add the “X-Requested-With¹⁶” header in our “AJAX” call which tells the server that the call is made for “AJAX” purposes.

Finally, we include the data that we wish to send to the server by converting it into a “JSON” string and upon a success callback function, we make another reveal that displays the successful application registration or the error that the server has responded with.

```
$.ajax({
    type: 'POST',
    url: "http://localhost:52243/Controllers/MentorController/MentorController.svc/addmentor",
    headers: { 'X-Requested-With': 'XMLHttpRequest' },
    data: JSON.stringify({
        Name: $Name,
        Email: $Email,
        Username: $Username,
        Password: $Password,
        Age: $Age,
        Roles: [{ID: $RoleID}],
        Games: [{ID: $GameID}],
        Description: $Description
    }),
    dataType: 'json',
    contentType: 'application/json'
```

User Story 3 – Create User Account

Written by Stefani

This User Story is much like the “Create Mentor” with just a few differences. The main purpose of this user story is for a basic user to be able to register in the system.

Backend

The back-end implementation of this user story was very simple because we already had established a way to execute a similar function in our service.

The most profound requirement that we received from our Product Owner regarding this user story is that he wanted each “User” that was registering to be able to specify the type of the account – “Single”, “Team” or “Organizations”. The reason is that he mentioned that a lot of his clients come to him and look for

¹⁶ <https://en.wikipedia.org/wiki/XMLHttpRequest>

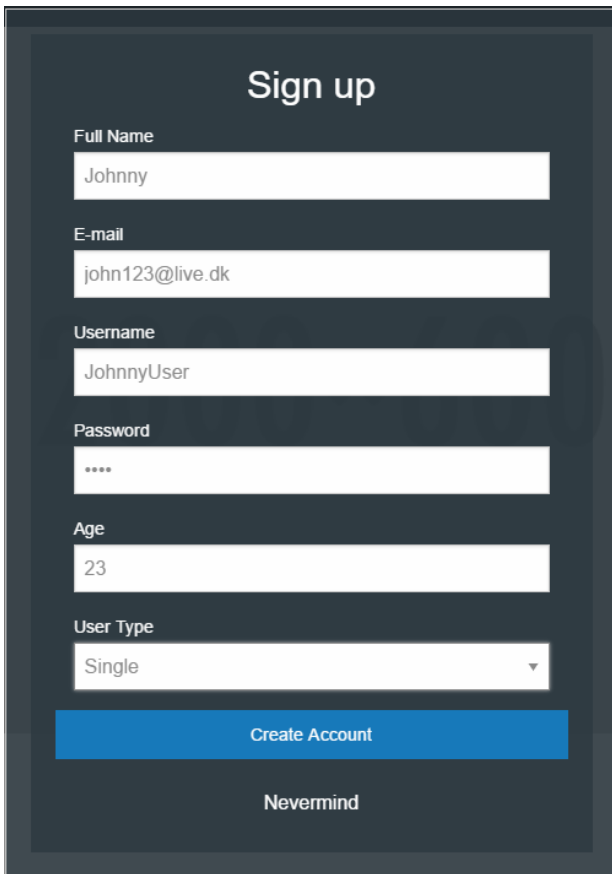
mentors as a team of 4-5 people or organizations with more.

For that we used an “Enumeration” to represent the type of user that is registering.

The implementation of “Create User Account” is almost the same as “Become a Mentor” with the difference that the role the User is assigned is 2 which corresponds to “Basic User” in our database.

Frontend

To create the Frontend of this user story we used the same “Reveal” class provided by Foundation 6 which would be triggered by a link on the web page “Sign Up”. The “JavaScript” for this part of the system is also the same as the one used for “Become a Mentor”.



The image shows a 'Sign up' form with a dark blue background. The form contains the following fields and buttons:

- Full Name:** Text input field containing 'Johnny'.
- E-mail:** Text input field containing 'john123@live.dk'.
- Username:** Text input field containing 'JohnnyUser'.
- Password:** Text input field with masked characters '....'.
- Age:** Text input field containing '23'.
- User Type:** Dropdown menu with 'Single' selected.
- Create Account:** A prominent blue button.
- Nevermind:** A smaller, lighter button located below the 'Create Account' button.

User Story 4 – Select Game and Role to Browse Mentors

Written by Stefani

The purpose of this user story is for the User to be able to select a “Game” of his or her choice from a “List of Games” and by that get the “Roles” available under the specific game. After the user has specified the “Game” and “Role”, they will get a list of “Mentors” that are under these specifications.

Backend

This user story contains three main steps that needs to be implemented – “Get all Games”, “Get all Roles” under specific “Game” and “Get all Mentors” under specific “Game” and “Role”.

To solve the first two “GET” requests, we decided to combine them into one “API” call that would contain an optional parameter. The optional parameter is made with the “GameID” which will return the specific “Game” with all its “Roles” upon being fulfilled.

To specify that a “URL” has an optional parameter, we used the “?” token in the route.

```
[OperationContract]
[WebInvoke(Method = "GET", ResponseFormat = WebMessageFormat.Json, UriTemplate =
"getgames/?gameID={gameID}")]
IList<Game> GetGamesRole(int gameID);
```

Because we have a “Many-To-Many” relationship between “Game” and “Role”, to get the “Roles” under the specific “Game”, we had to use the method “Include”¹⁷, which specifies which objects we want to include in the query. Afterwards, we convert the results into a “List” and add them to a “List of Games” and return them.

However, if it turns out that the optional parameter has not been given, we simply return the list of all the games that are in our database.

Because the relationships between “Mentor”, “Game” and “Role” are “Many-To-Many” and “Entity Framework” represents it as a list, we could not join the tables and extract the data we need. Instead, we had to go through the collection of “Role” and “Game” and find if there are any records that contain the specified “ID” and “Role”. In addition to that, we specify that we only want to get mentors that have the status of “Approved”.

```
public IList<CustomMentor> GetMentors(string gameID, string roleID)
{
    int parseGame = int.Parse(gameID);
    int parseRole = int.Parse(roleID);
    using (EsportsDbContext dbo = new EsportsDbContext())
    {
        dbo.Configuration.ProxyCreationEnabled = false;
        var query = (from m in dbo.Mentor
                    where m.Role.Any(r => r.ID == parseRole)
                    && m.Game.Any(g => g.ID == parseGame && m.Status == 3)
                    select new CustomMentor { MentorName = m.Name,
                    Description = m.Description, MentorID = m.ID }
                    ).ToList();
        _mentors = query;
        return _mentors;
    }
}
```

Frontend

The front-end of this user story was rather simple. Firstly, we call the “API” that would get all the games and then populate a “select” element with them. Afterwards, we make an “on-change” event on the selected element to extract the selected option and based on the “ID” of the selected option, we call the next “API” that would get the “Roles” under the specified “Game”.

Finally, when that is done, we make another “on change” event that would take the “ID” of the selected “Role” and call the method that would get all the “Mentors”. Afterwards, we populate the div with the results.

¹⁷ [https://msdn.microsoft.com/en-us/library/bb738708\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/bb738708(v=vs.110).aspx)

League of Legends ▼

ADC ▼

Eric

Main ADC player.
Challenger, formal LCS
player

User story 5 – Assign a Supervisor to Booking

Written by Jonas

The purpose of this user story is to “Assign a Supervisor” a lesson. A “Supervisor” in the system is someone who would be spectating the lessons and ensuring that the lesson is conforming to the rules and the “Customer” is being handled correctly. The user stories have two acceptance criteria – display all the bookings that don’t have a supervisor and then assign one to a specific booking.

Backend

The first part of the user story was quite easy to implement. The second part of this user story was implementing a “PUT HTTP Operation”. The method “AssignSupervisor” takes two parameters, the “ID” of the booking that we like to update and a User object.

We query the database to find the specified booking by the “ID” and update the “SupervisorID” with the one from the User object parameter. We use the same “WebOperationContext” helper class to create custom HTTP messages and return that.

```
public string AssignSupervisor(string bookingID, User user)
{
    int ParseBookingID = int.Parse(bookingID);
    using (EsportsDbContext db = new EsportsDbContext())
    {
        db.Configuration.ProxyCreationEnabled = false;
        var checkUser = db.User.Any(x => x.ID == user.ID && x.AccessroleID == 4);
        if (checkUser)
        {
            var query = (from b in db.Booking
                        where b.ID == ParseBookingID
                        select b).SingleOrDefault();
            query.SupervisorID = user.ID;
            db.SaveChanges();
            WebOperationContext ctxOK = WebOperationContext.Current;
            ctxOK.OutgoingResponse.StatusCode = System.Net.HttpStatusCode.OK;
            return "Successfully assigned a Supervisor to the Booking";
        }
        WebOperationContext ctx = WebOperationContext.Current;
        ctx.OutgoingResponse.StatusCode = System.Net.HttpStatusCode.BadRequest;
        return "Something went Wrong";
    }
}
```

Frontend

A link named “Supervisor” was created in the administration panel where the admin can access this functionality. On the right side, we will display all the bookings that are available so the admin can have an overview and on the left he will be able to assign them a supervisor by “ID”.

ID	Start	End	Hours	Mentor	Client
16	Eric	24/4/2017 19:30	24/4/2017 20:30	1	Johnny

From our service we use datetime objects which once they get serialized into “JSON”, they do not arrive the same way on the client side and that is because “JSON”¹⁸ does not have syntax for dates – “/Date(1495713600000+0200)”. For that reason, we had to convert them from “JSON” to date objects and cut out the unnecessary parts and extract only the numbers .

```
var $jsonStartDate = item.StartDate;  
var $jsonEndDate = item.EndDate;  
var $normalStartDate = new Date(parseInt($jsonStartDate.replace('/Date(', '')));  
var $normalEndDate = new Date(parseInt($jsonEndDate.replace('/Date(', '')));
```

User Story 6 – Display All Booking Under a Mentor

Written by Jonas

The purpose of this user story is to display all the bookings a mentor has. The implementation was almost the same as “Display all Bookings” with the only change of getting them by the “MentorID”.

Backend

The backend was very smooth and straight forward to implement because we already had a sample of how it should look like. The difference was that we had to query the name of the logged in user. We used the same custom class “MentorBooking” to extract only necessary information.

Frontend

The frontend of this User Story was created by simply using a table that we populate with the response from the service.

Booking No.	Start	End	Hours	Game	Role	Client
15	25/4/2017 14:50	25/4/2017 18:50	4	League of Legends	Jungle	Johnny
16	24/4/2017 18:35	24/4/2017 21:30	10	League of Legends	Jungle	Johnny

¹⁸ <http://www.newtonsoft.com/json/help/html/DatesInJSON.htm#DatesAndJsonNET>

Sprint Review Meeting

Written by The Team

At the end of sprint, we held a meeting with our Product Owner to show him the finished User Stories. We guided him through each User Story and he was a little taken aback from the fact that the system did not look complete just yet but rather only parts of it.

We explained to him that, gradually the system will be start shaping into a finished product.

During the meeting, we asked him about his thoughts on the User Interface and if he had any visions or changes that he would like us to implement. He mentioned that the way the data is being represented could be changed in the future such as the tables that we currently using could be more interactive. However, he advised to keep it primitive until his vision is complete.

The meeting was very helpful and insightful for us because we understood that our focus should not be directed that much on how the design of the interface is like but rather on the functionality of it.

Sprint Retrospective

Written by The Team

1. What went well during the Sprint?

One of the things that went very smoothly during this Sprint was the communication between the team and Product Owner. During the meetings and in-between, the working environment of the team was very positive and motivating - we worked with the same level of ambitiousness. Also, the Product Owner was very active and expressed his visions, concerns, advices very openly which made it easy to work with.

Secondly, keeping a visual representation of the Velocity was very helpful for the team to understand how much progress and effort is needed to finish the Sprint Backlog. In addition to that, throughout the whole sprint we worked with the mindset that each User Story must be completed and ready for release.

Finally, using Pair Programming during the Sprint was very helpful and efficient for the development of the user stories because it leads to less errors and idea generation.

2. What went wrong during the Sprint?

The Database Design was one of the parts that went wrong for us during this Sprint because we had to carry on with the implementation at the start. It set us back in time a great deal and we had to put twice as much effort into completing the Sprint Backlog.

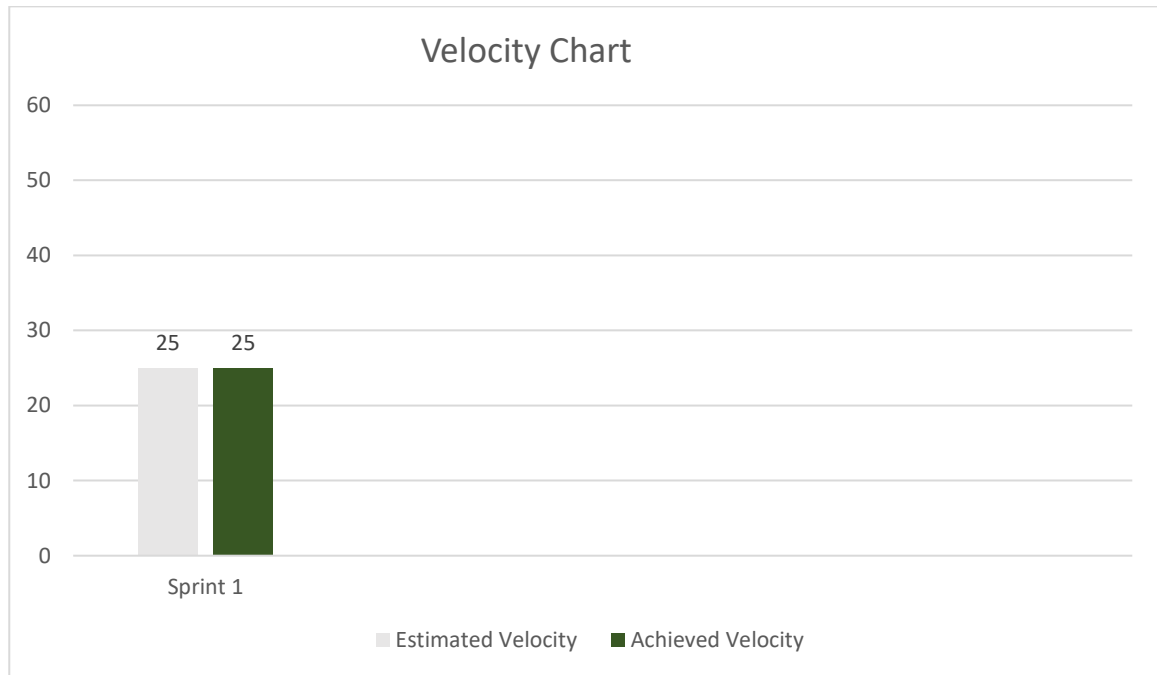
3. What could we do differently to improve?

The activity that we should start performing is keeping logs of everyday work because we had some inconsistent documentation during the Sprint and that made it harder for us to reflect upon.

Velocity Chart

Written by The Team

Due to the difficulty, we experienced with the implementation of the Database, we had to spend the first week of Sprint 1 working and fixing the issue. Because of that we had to work twice as hard to complete the User Stories, which led us to working more than 8 hours a day.



Sprint Conclusion

Written by The Team

Despite that the first part of Sprint 1 was not focused on the implementation of the Sprint Backlog, we managed to complete all the requirements and that gave us an insight of how we cope in stressful situations. Because of that sprint was very hectic and we had to work more than planned.

Sprint 1 was very crucial for establishing the common ground for the system structure, as well as getting introduced to the building blocks of the new framework. We managed to understand the fundamentals and establish the front end despite the difficulty finding the correct documentation for it.

Even though we estimated our effort a little higher, we managed to successfully complete the sprint objective.

Sprint 2 – 15th – 28th May 2017

Written by The Team

Reflecting upon the success we had during Sprint 1, we will be bringing the same strategy and motivation into this sprint to ensure that we achieve the set goals.

We will be starting the cycle with the usual “Sprint Meeting” where we will lay out the “Sprint Backlog”.

Sprint Backlog

Written by The Team

In our previous Sprint, we had to change the “Acceptance Criteria” of the user story “Become Mentor” and that itself generated a new user story that we had to add to our “Product Backlog” and “Sprint Backlog” – “Update Mentor Application Status”. For this Sprint, the total Estimation Points we must complete is 21.

<p>As a: User I want: To be able to log in into the system So that: I can use the services the system provides Business Value: 300 Estimation Points: 8 Responsible: Stefani</p>
<p>As an: Admin I want: To be able to see all supervisors So that: I can have an overview of them Business Value: 200 Estimation Points: 2 Responsible: Jonas</p>
<p>As a: User I want: To be able to give feedback after my lesson So that: Changes or improvements could be made for the future Business Value: 200 Estimation Points: 5 Responsible: Jonas</p>
<p>As an: Admin I want: To be able to change the status of a Mentor So that: I can Accept or Decline their applications Business Value: 300 Estimation Points: 3 Responsible: Jonas</p>
<p>As a: User I want: To be able to make a Booking So that: I can have a lesson with a mentor Business Value: 300 Estimation Points: 3 Responsible: Stefani</p>

User Stories

We will be using the same strategy for revealing the user stories of this sprint.

User Story 7 – Update Mentor Application Status

Written by Jonas

The purpose of this user story is to give the administrator the authority to change the status of a mentor application to “Accepted” or “Declined”. Upon the change of the status an E-mail will be send to the applicant with the corresponding message.

Backend

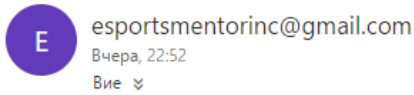
The implementation of this user story was created by making a “PUT HTTP” request. Based on the updated status, we send an email to the queried mentor by using “MailMessage” and “SmtpClient” helper classes that are located within “System.Net.Mail” namespace.

We use the “MailMessage” class to build the content of the e-mail – “Recipient”, “Sender”, “Body”, and “Title”. Afterwards we specify the “SmtpClient” host which in our cause would be “smtp.gmail.com” and we specify the credentials.

```
if(query.Status == (int)MentorStatus.Approved) {
string email = query.Email;
message.To.Add(new MailAddress(email));
message.Subject = "Your Application Status has been updated!";
message.From = new System.Net.Mail.MailAddress("esportsmentorinc@gmail.com");
message.Body = "Dear Applicant, " + "\nWe are happy to announce to you that your
application for becoming mentor has been approved!" + "\nYou can now use your credentials to
log into the system." + "\n\nYours sincerely," + "\nEsportsmentor";
System.Net.Mail.SmtpClient smtp = new System.Net.Mail.SmtpClient("smtp.gmail.com");
smtp.EnableSsl = true;
smtp.Credentials = new NetworkCredential("esportsmentorinc@gmail.com", "test");
smtp.Send(message);
}
else if (query.Status == (int)MentorStatus.Declined) {
string email = query.Email;
message.To.Add(new MailAddress(email));
message.Subject = "Your Application Status has been updated!";
message.From = new System.Net.Mail.MailAddress("esportsmentorinc@gmail.com");
message.Body = "Dear Application," + "\n" +"We are sorry to announce that your application
for becoming a mentor has been declined!" + "\n\nYours sincerely," + "\nEsportsmentor";
System.Net.Mail.SmtpClient smtp = new System.Net.Mail.SmtpClient("smtp.gmail.com");
smtp.EnableSsl = true;
smtp.Credentials = new NetworkCredential("esportsmentorinc@gmail.com", "st9510250491");
smtp.Send(message);
}
```

When the e-mail arrives, it looks like this:

Your Application Status has been updated!



Dear Application, we are sorry to announce to you that your application for becoming mentor has been declined!
Your sincerely,
Esportsmentor

In addition, we had to make an additional function to get all mentor applications with a status of “Awaiting Approval”.

Frontend

The front end was quite simple to make – we made the call to get the awaiting approval applications and an additional form where the admin could change the status based on the ID that has been provided.

ID	Name	Game	Role	Description
23	EricADC2	League of Legends	Middle	Challender player. Can player any lane in League of Legends. Looking to become a mentor

User Story 8 – See all Supervisors

Written by Jonas

The implementation of this user story was very simple, it was almost the same as getting the mentor schedules from the previous sprint. Its purpose is to display all the available supervisors to the admin.

Backend

The way we extract the supervisors is by querying the table “User” and returning all users with the “AccessroleID” of 4 which corresponds to “Supervisor” in our system.

Frontend

The way we show the supervisors is by creating a table with “Supervisor ID” and Name.

ID	Name
26	Stefan Zhapis

User Story 9 – User Booking feedback

The purpose of this user story is to give the users a chance to give feedback on how their previous lessons went.

Backend

The implementation of this user story has two parts.

The first part was getting the bookings of the user that are expired. To do that, we made a method called “GetExpiredBookings” with a return type of “UserBookings”. The “UserBookings” is a custom class that we created to return only necessary data. In the method, we query the database by the ID of the user and the “BookingDate” being less than the current date.

To ensure get the expired bookings we simply get all the bookings under the current “User ID” with status of Completed. To determine whether a booking is completed or not, in our Database we made a “trigger” on the table “Booking” that will initiate upon “Insert” or “Update”. The “trigger” sets the status of “Booking” to completed when “DateTime”, together with the hours, is smaller than the current “Date” and is not “Cancelled”.

```
ALTER trigger [dbo].[Trigger]
on [dbo].[Booking]
AFTER INSERT, UPDATE
AS
BEGIN
UPDATE dbo.Booking
SET Status = 2
WHERE DATEADD(HOUR, BookedHours, BookingDate) < GETDATE() AND Status != 3
END
```

The second part was to make a “PUT” HTTP operation where we update the comment of the booking with the input from the client. The method takes two parameters – The ID of the booking and a Booking object that contains the requested comment. To find the booking that needs to be updated, we query the database by the requested ID and we use “FirstOrDefault” to return the first found element of the query or the default value if nothing was found. Afterwards we update the found item’s comment with the requested one.


```

public Booking UpdateBooking(Booking booking, string bookingID) {
    using (EsportsDbContext db = new EsportsDbContext())
    {
        db.Configuration.ProxyCreationEnabled = false;
        int parseID = int.Parse(bookingID);
        Booking updateBooking = (from b in db.Booking
                                where b.ID == parseID
                                    && b.UserID == booking.UserID
                                select b).SingleOrDefault();
        updateBooking.Comment = booking.Comment;
        db.SaveChanges();
        return updateBooking;
    }
}

```

Frontend

The frontend of the first part of the user story was implemented by using a table to display the information.

Previous lessons

ID	Start	End	Mentor	Game	Role
15	25/4/2017 14:50	25/4/2017 18:50	Stefani	League of Legends	Jungle
16	24/4/2017 19:30	24/4/2017 20:30	Stefani	League of Legends	Jungle

The second part was by creating a form with two input fields – one for the ID of the booking and one for the comment.

Lesson Feedback

Booking ID

Feedback

Submit

User Story 9 – Log in

Written by Stefani

The purpose of this user story is to enable the user to log into the system.

Backend

To solve the backend of this user story, the return type of the method Login is set to “bool” because we want to return if the login was successful or not. In addition, the method takes two parameters – Username and Password.

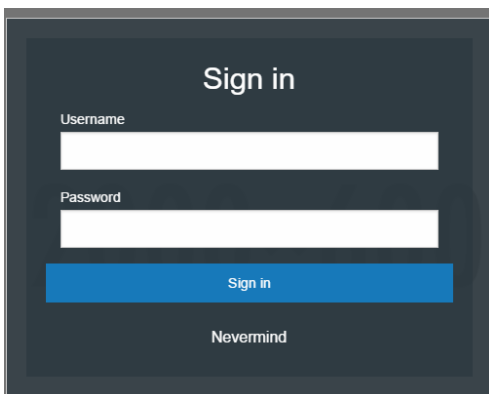
We make a check if the user exists in the database and if he does we take the salt and password of this user. Afterwards, we convert them into byte array so we can compare their length with the password that the user has input.

By calling the method “VerifyPassword” that we created in Sprint 1 we take the plain password from the request body as plain text, hash it and salt it and then compare it with the one in the database. If the method “VerifyPassword” evaluated “true” we return true to the client and vice-versa.

```
public bool Login(string UserName, string Password) {
    using (EsportsDbContext dbo = new EsportsDbContext())
    {
        dbo.Configuration.ProxyCreationEnabled = false;
        var findUser = (from u in dbo.User
                       where u.Username == UserName
                       select u).FirstOrDefault();
        if (findUser != null) {
            var passwordSalt = findUser.PasswordSalt;
            var passwordHash = findUser.Password;
            byte[] pwS = Convert.FromBase64String(passwordSalt);
            byte[] pwH = Convert.FromBase64String(passwordHash);
            var passwordVerification = PermissionController.EncryptionController
                .VerifyPassword(Password, pwS, pwH);
            if (passwordVerification && findUser.Username == UserName) {
                return true;
            }
        }
        return false;
    }
}
```

Frontend

Like the “Sign up”, the front-end of this user story will also be a “reveal” on the page that will be triggered once you click on a link “Sign in”. The “reveal” contains two inputs – Username and Password.



User Story 10 – Create Booking

Written by Stefani

The purpose of this User Story is for a user to create a booking by choosing the Date, Booked Hours, Game, Role, and Mentor.

Backend

In the backend of this User Story, we had to create a Booking Status as part of the acceptance criteria. Upon creation, the booking status should be set to “In Progress”. The method “CreateBooking” is a simple “POST” operation that takes a Booking object as a parameter.

To ensure that the made bookings do not occur before or after work times, we made two constraints on the table that will limit the BookingTimes to be after 8:00AM and before 18:00 AM. The second constraint will make sure that the BookedDate together with the BookedHours do not go over the range.

```
alter table dbo.Booking ADD CONSTRAINT checkStartTime check
(cast(BookingDate as time) >= cast('08:00:00.0000000' as time)
AND cast(BookingDate as time) <= cast('18:00:00.0000000' as time))
```

```
alter table dbo.Booking ADD CONSTRAINT checkTimes check
(cast(DATEADD(HOUR, BookedHours, BookingDate) as time) >= cast('08:00:00.0000000' as time)
AND cast(DATEADD(HOUR, BookedHours, BookingDate) as time) <= cast('18:00:00.0000000' as
time))
```

To make the creation of booking robust, we needed to ensure the user that he/she is booking an actual available time. To do that we need to find open time slots in the desired mentor’s schedule by specified date. We created a for loop that would iterate through all the bookings on a specific date and will find an open slot. We also ensure that the gap that has been found, is more or equal than 1 hour because a lesson is usually more than that.

```
var listOrder = bookedAppointments.OrderBy(x => x.Start).ToArray();
for (int i = 0; i < listOrder.Length - 1; i++)
{
    double span = (listOrder[i + 1].Start - listOrder[i].End).TotalHours;
    if (span >= 1.0)
    {
        freeAppointments.Add(new TimeSlots()
        {
            Start = listOrder[i].End,
            End = listOrder[i + 1].Start,
            Hours = span = (listOrder[i + 1].Start - listOrder[i].End).TotalHours
        });
    }
}
```

Because the ending of each booking will be defined by combining the starting date of the lesson combined with the hours, we had to iterate through each booking in his schedule and calculate the end time.

Frontend

The frontend of the User Story was implemented by having a simple form with three input fields and three drop downs. Firstly, we had to populate the dropdowns with the Games from the database and then when a game has been chosen, populate the second one with the Roles and once both have been specified, we

populate the last dropdown with the Mentors that go under the selected specifications. The three other inputs are to store the specified Date, Time, and Number of hours the user wishes to book.

Book a lesson

Booking date

Booking Time

Booked Hours

In the JavaScript, we extract the value from each of the input fields. To get the selected option from the dropdown we made an “OnChange” event for each of them and stored the values into variables.

The biggest problem that occurred during this is transforming the DateTime object to a format acceptable for JSON. To solve that we created a custom class to handle the conversion.

Sprint Review Meeting

At the end of sprint, we had our usual Sprint Review meeting with our Product Owner where we presented the User Stories that we have made ready for release. The first impression he got from seeing the system is that it looked overall more connected.

However, he had some additional requirements towards some of the user stories.

Firstly, when he saw the User Story “See all Supervisors” he mentioned that he also wants to be able to create accounts for the supervisors.

Secondly, when he saw the user interface of “Create Booking”, he was wondering if whether a booking should have a starting and ending date rather than just a single date. We discussed it and we concluded that indeed, the booking should only have one date. Additionally, he was very pleased with the e-mail notification that a mentor receives once the status of his application is updated.

Finally, he questioned how or where can you see the feedback that has been given to a certain booking and that was something that we hadn’t really thought about but we told him that we will make sure to include it into the backlog of our next sprint.

Sprint Retrospective

1. What went well during the Sprint?

The Sprint Backlog was a success because we managed to implement all the User Stories. And a strategy that continued to ensure this success was Pair Programming. It helped us a lot in resolving issues and finding the best solutions. Furthermore, this also resulted a success at “Continues Integration” because it enabled us to produce ready for release user stories.

Finally, during Sprint One we had major issues with the Database Design that were addressed and fixed. For that reason, coming into Sprint Two we had little to no problems when we had to work with the database.

2. What went wrong during the Sprint?

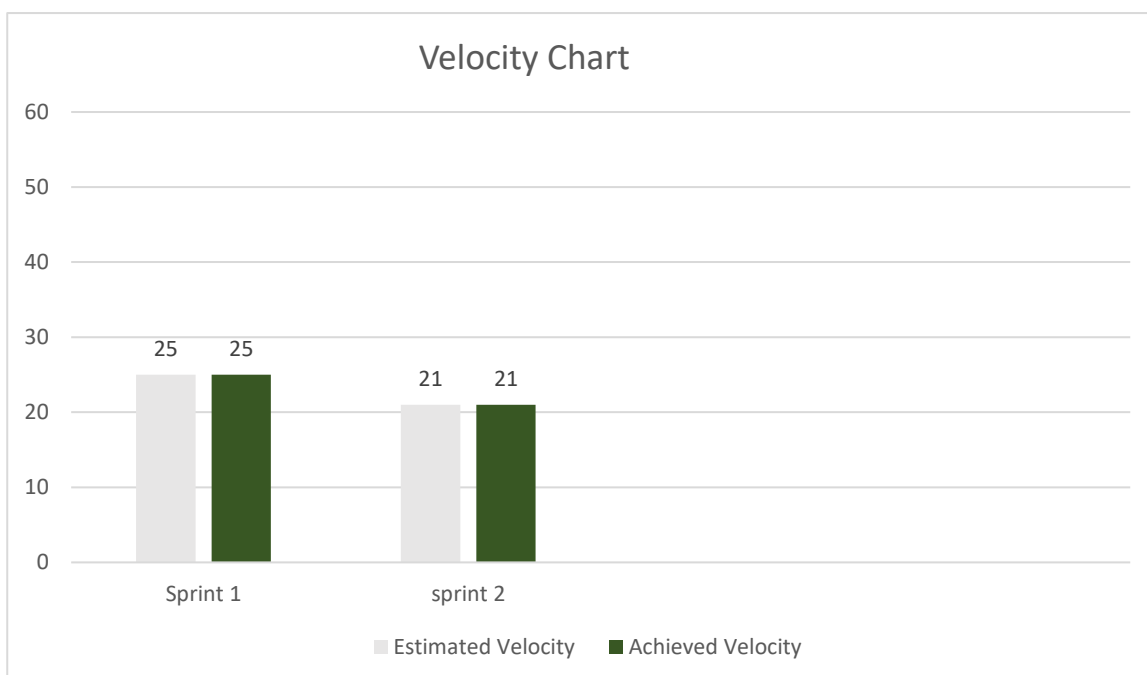
Despite our wishes to improve the track of daily work, we did not seem to fully commit to it. Therefore, this was something that went wrong during Sprint because it resulted in making it harder to reflect upon the work completed during this and last sprint.

3. What could we do differently to improve?

Reflecting on the positive feedback that we have gotten from our Product Owner, we think that the strategies we currently using to ensure the quality of the product are successful and for that reason we do not seem to notice any improvements that we could carry on into our next Sprint.

Velocity Chart

The reason we had higher Estimated Velocity during Sprint 1 was because we were too ambitious and did not expect to encounter so many problems with the Database Design. Taken that into account, we conformed to the total estimation points we agreed upon before starting and that resulted in the Sprint being smoother and less stressful.



Sprint Conclusion

Despite the Sprint being successful, we encountered a lot of programming difficulties with serializing dates and JSON. This problem occurred because the standard JSON serializer in .NET converted the dates into a format that was hard to work with. Mostly, when date object must be sent from the client.

Another issue that we encountered was when using the “Hotmail” as the host of “SmtpClient” because the Hotmail configuration were blocking the calls. For that reason, we had to switch to Gmail to fix this issue. In the future, we plan to investigate and learn why the problem was occurring.

This Sprint was very helpful in creating a more connected overview of the system which our Product Owner was very pleased with. In addition to that because were meeting the estimated velocity each sprint, we had some concerns about our Product Backlog wearing thin and after the Sprint Review Meeting, we managed to gather some more requirements that needed to be established during Sprint 3.

Sprint 3 – 29th May – 9th June 2017

Despite this being our last sprint, we will not be treating it as a “Release Sprint” because the development of this product will continue after “Project Time” is complete.

We will divide this cycle into two parts – the first part we will be completing the Sprint Backlog and the second part – on finalizing our report and finishing our project.

Product Backlog

During the “Sprint Review” meeting in Sprint 2, we received two new requirements that we had to take into consideration during Sprint 3 – “Create Supervisor” and “See all Feedback”. The total Estimation Points of this Sprint is 29, it is a lot higher than what we should be aiming for. However, if the User Stories cannot be completed within the first part of the Sprint, we will not take them into consideration until Project Time is over.

As a: Customer

I want: To be able buy hours

So that: I can book a lesson with a mentor

Business Value: 300

Estimation Points: 5

Responsible: Jonas

As an: User

I want: To be able to cancel a booking

So that: I can notify the mentor if I can't attend the lesson

Business Value: 200

Estimation Points: 8

Responsible: Stefani

As a: User

I want: To be able to reset my password

So that: I can change it if I have forgotten it

Business Value: 300

Estimation Points: 8

Responsible: Stefani

<p>As a: Admin I want: To create supervisor accounts So that: I can have new supervisors Business Value: 300 Estimation Points: 5 Responsible: Jonas</p>
<p>As a: Admin I want: I want to see all the bookings' feedback So that: I can see what improvements could be made Business Value: 300 Estimation Points: 3 Responsible: Jonas</p>

User Stories

User Story 11 – Create Supervisor Accounts

Written by Jonas

This is the second user story that was produced during the sprint meeting in sprint 2. Its purpose is for the admin to be able to create a supervisor account. The reason he wants to be responsible for creating the accounts is because a supervisor is recruited through private channels that he controls.

Backend

The creation of a supervisor account resembles our previous user stories “Create Account” and “Become A Mentor”. However, during the creation the admin will be the one that specifies the password for the account and the password will be encrypted. Once the supervisor gets a hold of his/her account, they will be able to change the password to one of their choice.

The “Accessrole” is set to 4 which is a supervisor in the database and the “Usertype” is set to 1 which is a single user because a supervisor can't be a team or an organization. We make the return type of the method a string and use the same “WebOperationContext” helper class to return a custom HTTP response.

Frontend

The frontend was solved by creating a simple form. Once the button “Submit” is clicked, an AJAX POST request is going to be send to the service.

Create Supervisor

Name

Email

Username

Password

Age

User Story 12 – See All Feedback

Written by Jonas

This is one of the user stories that was produced during the sprint review meeting in sprint 2. Its purpose is to give the ability for an admin to see all the feedback that has been given in the lessons that has been completed.

Backend

The backend of this user story was completed by implementing a simple get method and using a custom class called “BookingFeedback” to extract only the necessary data needed.

Frontend

The solution for the frontend of this user story was completed by having a simple table that shows the bookings with their feedback.

Lesson feedback

ID	Client	Mentor	Date	Comment
15	Johnny	Stefani	25/4/2017 14:50	Super good lesson, everything went as expected.
18	Johnny	Stefani	27/4/2017 0:0	Nothing to complain about.
20	Johnny	Stefani	28/4/2017 16:0	I dont like my current mentor
24	Johnny	Stefani	29/4/2017 16:21	Looking forward to my next lesson, this is perfect
25	Johnny	Stefani	29/4/2017 16:22	I learned alot from the mentoring, definitely going to recommend it to my friends

User Story 12 – Reset Password

Written by Stefani

The purpose of this user story is to enable the user to recover his password in case he/she has forgotten it. I had to spent time researching for I am not that acquainted with what the correct approach for solving the problem is.

Backend

To solve this user story, we created an additional table in our database called “ResetTickets” that will hold the IDs of the users and a token with expiration date. Upon calling the “Reset Password” API, the method will generate a randomized token by using “Guid” – a global unique identifier. The token will be saved into the database and will be given an expiry of 10 minutes.

```
string token = Convert.ToBase64String(Guid.NewGuid().ToArray());
ResetTickets ticket = new ResetTickets();
ticket.UserID = query.ID;
ticket.ExpirationDate = DateTime.Now.AddMinutes(10.0);
ticket.TokenUsed = false;
ticket.Token = Encoding.ASCII.GetBytes(token);
```

Afterwards, we use “SmtpClient” to compose an e-mail with a link to the reset page together with the token.

```
string url = "http://127.0.0.1:3000/resetpassword.html?token=" + token;
MailMessage message = new System.Net.Mail.MailMessage();
message.To.Add(new MailAddress(query.Email));
message.Subject = "Password Reset";
message.From = new System.Net.Mail.MailAddress("esportsmentorinc@gmail.com");
message.Body = "Hey" + query.Username + ", " + "\n"
+ "To reset your password, please click the link below: "
+ "\n" + url +
"\n\nYours sincerely," +
"\nEsportsmentor";
System.Net.Mail.SmtpClient smtp = new System.Net.Mail.SmtpClient("smtp.gmail.com");
smtp.EnableSsl = true;
smtp.Send(message);
```

When the user gets redirected to the “ResetPassword” page, we call “UpdatePassword” that takes the token as a parameter and the password as plain text. To extract the token and include it in the API call of “Update Password”, we extract the url of the page and cut out the token.

```
var url = window.location.href;
var password = $("#newPasswordInput").val();
token = url.substr(url.indexOf("=") + 1);
$("#submit-password").on('click', function(){
$.ajax({
type: 'PUT',
url: 'http://localhost:52243/Controllers/LoginController/LoginController.svc/updatePassword/?
token=' + token,
```

Through the token we find the user that is requesting the password update. During this part, we also check if the token has been used or expired, if not – we hash and salt the newly requested password and store it into the database.

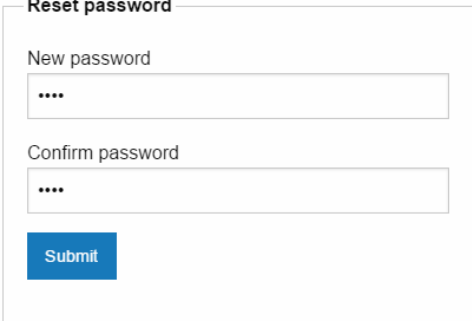
In addition, to ensure the integrity of these tokens we created two triggers on the table – one that will delete the token upon being used and the other one will set the status of token to “Used” if the time has expired.

```
ALTER trigger [dbo].[SetTicket]
on [dbo].[ResetTickets]
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
UPDATE dbo.ResetTickets
Set TokenUsed = 1
WHERE ExpirationDate < GETDATE()
END
```

```
ALTER trigger [dbo].[ResetTicket]
on [dbo].[ResetTickets]
AFTER INSERT, UPDATE
AS
BEGIN
DELETE dbo.ResetTickets
WHERE TokenUsed = 1
END
```

Frontend

The frontend of the user story is a simple form that will only be visible once the user has requested a password reset. In the JavaScript, we cut out the token from the url and pass it into the service through an AJAX call.



The screenshot shows a dark blue header with the text 'E-SPORTMENTOR'. Below the header is a white form titled 'Reset password'. The form contains two text input fields. The first is labeled 'New password' and contains four dots. The second is labeled 'Confirm password' and also contains four dots. Below the input fields is a blue button with the text 'Submit'.

User Story 13 – Cancel Booking

The purpose of this user story is to enable the user to cancel his booking and notify the mentor when that has been done.

Backend

The backend was a simple “PUT” operation which takes the ID of the user and booking. Based on that, it changes the status of the booking to “Cancelled” and notifies the mentor through e-mail by using the same “SmtClient”.

Frontend

Frontend was a simple form that takes the ID of the booking the user wishes to cancel.

Sprint Review Meeting

Even though this is the last sprint of our project time and our system looks complete, there are still some minor issues that we will be addressing in the future development process. During the meeting, we presented the system to the Product Owner where he we guided him through the differences processes. He was pleased with the current state of the functionalities that we presented him however some of the issues we previously mentioned caught his eye.

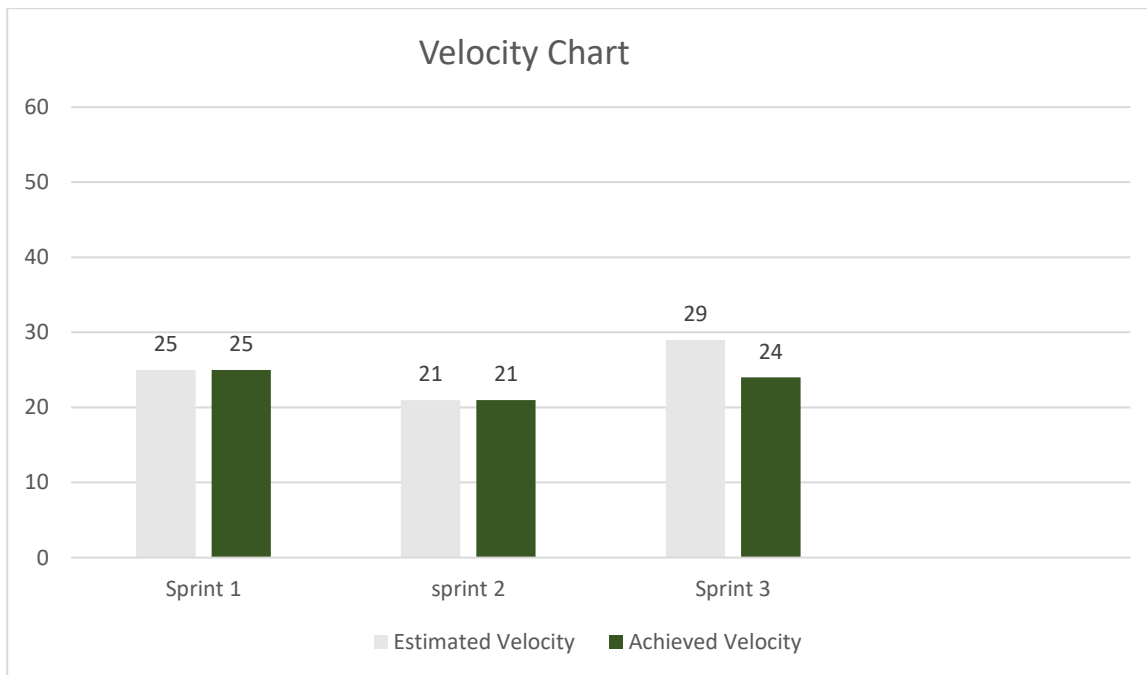
Throughout the whole process, the Sprint Review Meetings were very helpful to involve the Product Owner in the development process and it also made it easier for us to capture his needs and requirements. Both us and the Product plan to carry on with the Sprint Review Meetings in the future development.

Unfinished User Story

Because our sprint was divided into two parts and the first part was all about finishing the Sprint Backlog, we did not have enough time to implement one of the user stories – “Buy Hours”. The reason we did not have enough time to complete this user story is because the subject of one of the others – “Reset Password” was something that took us more time than intended due to the time we had to spend researching in how to approach it.

Velocity Chart

The estimation points for this Sprint were quite high and the amount of time we had to achieve them was only one week for that reason one of the user stories could not be completed.



Sprint Conclusion

The biggest issue during this sprint was finding out the correct strategy of resetting or recovering a user's password. We spent some time researching into different methods of how to achieve this – sending the password to the users by e-mail, making a random generated password and sending it to the user through e-mail with an expiration date, and having a reset link with a token. We concluded that the first two approaches were not secure enough, especially plainly sending the password to the user's e-mail and we decided that we need to figure out how to make a reset link.

Even though, the solution we have is currently working there are some minor issues that we need to take in consideration regarding this user story to make it more optimal and secure. Despite this user story taking a lot of time during this sprint, we gained an understanding of what is the correct strategy of ensuring a secure password recovery.

Marking this as our last sprint during project time, we managed to step back and look at the system as a complete picture and this gave us an understanding of what issues we need to address and what holes needs to be filled to make the system more robust and efficient in the future.

Project Conclusion

To conclude the development process for this project, we went through the previous documented workflow and analyzed on an internal level the strategy that we used to reach the ending point with a success. The Project Conclusion will be divided into different segments where we will reflect thoroughly on the topic at hand.

Process

Overall, the process of completing this project went rather smooth but rather hectic because of the short time-frame we had to complete it.

The communication between the team was easy and we could agree completely with most of the problems we had to solve. To ensure the successful solving of each problem we had to cope with throughout the process, we made sure to take the first important steps and that is discussing of each possible solution and outcome of it. That ensured team integrity and mutual understanding in what state the system currently was.

The communication between the team and the Product Owner was also very easy to establish. When discussing and addressing each of his points of concerns, he was very open-minded and thorough with his requirements. That was of a great help to the development process and successful outcome.

The working environment that we built was very organized and structured and it greatly impacted our motivation. In addition, the meetings we had with our supervisor were of a great help to the production of the report and aspects of the system. The constructive feedback we received on some security issues that could pose threat to certain data integrity was a great revelation. Luckily, we did not encounter the problem of a member absence which also was a positive impact on the working process.

When we were given the opportunity to work on this project, we could see the potential in the idea and the possibility of expanding our knowledge in web development. Furthermore, the problem cores the project was addressing is a topic that both of us have great interest in and that is what sparked the consistent motivation.

In contrast to the positive things, we experienced some problems with one of the user stories – “Log in”. The user story represents the connection link between all parts of the system. The reason the user story raised so many problems was our lack of knowledge in WCF.

The usage of WCF services is still very new to the both of us as well as the topic of identifying different users in the systems and to achieve that we had to learn about custom roles, authentication, and authorization in the context of WCF. The lack of experience we have is what resulted in not fully understanding how to approach and achieve this part. A solution to this would have been to use ASP.NET because of its inbuilt role identifier and our more thorough knowledge about it.

Tools

Because of the lack of knowledge and experience our Product Owner has within the Software Development scope, he did not have any specific requirements towards the tools we should use to create the product. The only requirement he demanded was to have a responsive web page that would look great on any device.

To maintain and store our code, we previously specified that we would be using GitHub to ensure data integrity. We spent a whole day setting it up and encountered numerous problems when trying to create pull requests for that reason we decided to derive from it and use an alternative – Dropbox.

Using “WCF” might not have been the best decision or solution for this product. Even though it was rather easy to establish and consume, our lack of knowledge in the service led to difficulties with limiting the access for each specific user in the system. For future, we might derive from it and use “ASP.NET” instead or “ServiceStack” which was our initial idea however it would have required an immense amount of time understanding how to work with since we have only had a basic knowledge of the technology.

Working with Foundation 6 was rather easy to comprehend because of its close resemblance to Bootstrap. The only issue that occurred was finding the correct documentation and our basic knowledge of user interface design. We believe that as our experience and knowledge of web development increases, our usage of Foundation 6 would be more optimal and would result in more modern and appealing looking product.

To ensure that the user’s experience when using the product is not deprived we used AJAX to allow responsive content with jQuery. The implementation of AJAX was very easy and straightforward and the only problems that ever occurred was resolving the issue with Cross-Domain requests.

To conclude, if we had to re-implement the product with the knowledge we have now, the tools we have chosen would have been different. More specifically the service framework. Even though, we have successfully created a product with the usage of these tools, if we derive from it and find a better solution, we would achieve a much more robust system for our client.

Techniques

Even before the creation of this product, there was no doubt between neither of the team members that Agile and Scrum would be the principles to use for the development process. Both of us have had previous experience within the workflow of Scrum that resulted in success and looking back on the product now and decisions we have made, we believe that it was the right choice.

Pair Programming was an essential part of development process that helped us a lot in improving the team communication, error-proof code, and meeting with the deadline of the Sprints.

An artifact from Scrum that we did not use through our development process was the Scrum Board because to us it did not make sense to implement it within a team that consists of two people. In addition, the communication was very easy to establish as well as finding at what state the user stories each one of us had taken responsibility is.

To conclude, there is no doubt for both of us that for future if can choose, we would always go for Agile because of its flexibility and dynamic workflow.

Personal Evaluation

Stefani Dimitrova

Overall, I believe that the steps and decisions we made through the development process were successful. I started off the project with a lot of questions and confusion on how to address and solve each requirement the Product Owner had towards the product. That was mainly because of my lack of knowledge in Web Development and issues that needs to be considered when making a web application.

I believe that with the consistent motivation I had throughout the whole process, I managed to handle some of the user stories that I thought I would have trouble with. Despite it currently being on a level that I am not fully satisfied with, I believe that if given more time to research and understand, they will become more optimal and robust.

During this time, I managed to learn a lot about team work, communication, SCRUM itself and the limits that as a developer I am currently experiencing. For future, I believe that I need to work on looking at a problem in a simpler perspective because I tend to overcomplicate things which lead me to confusion and frustration. In addition, I believe that if the team was consisting of more people some of the SCRUM artifacts and strategies would have made more sense.

Finally, I think that the biggest problem in the product was not implementing the requirements but creating the connection link between each piece.

Jonas Rytter

My thoughts on the development process are divided in two perspectives – Student and Professional. From a student point of view, I believe that given the short amount of time for this project, the product created is something that I'm satisfied with. However, from a professional point of view, the state of the product right now, is not something that I would release for use to the client. However, this development time allowed me to gain a good understanding of what is the right approach to building a web application.

To me some of the biggest issues in this project was not the programming itself but more the overall understanding of how the different things work. How does the login actually work, what is the process of resetting a password, how should we structure the whole project both in the backend and the frontend.

From previous experience where I have used SCRUM within teams that consists of more members, I could see the purpose of SCRUM and what advantages it brings. However, during this project time we were only two people and for that reason most of the SCRUM artifacts did not make much sense to use.

Future plans

The current system that we managed to build during project time is not on the level that is supposed to be. There a lot of aspects that needs be remade or reconsidered.

In the future, we plan to derive from WCF and instead use ASP.NET with ServiceStack framework. The reason we wish to use "ServiceStack" is because during our internship we got to use it and we could see the potential the framework has. We did not approach "ServiceStack" during project time because it is very vast and it would have required a lot of time to set up and understand. That would have lead us to not being able to have a set us back in time and deprived us from having any kind of product to represent.

Some of the functionalities that our Product Owner talked about in the start were something we will be taking into consideration when taking the product on the next level. Most importantly, we would establish a live chat system through which users can communicate together with a voice and video call. One of the most important aspects in this requirement is that we need to understand how to enable supervisors to spectate anonymously the voice and video calls to ensure the quality of the lesson.

Another thing that we plan to optimize is the Booking system and ensuring that when a user creates a booking he can see the availability of the mentor he wishes to have.

Most of these requirements demand a lot of experience and knowledge in web development and would require some time to understand and learn about. Therefore, we would focus on them in the future to ensure that the product we create would mostly satisfy our Product Owner.