UNIVERSITY COLLEGE OF NORTHERN DENMARK

System Development Report

Computer Science AP Degree DMAI2019 Group 6

3rd semester project, 2019

Authors

Jan Kalasnikov
Valentin Yordanov
Denis Kovacek

Teachers

Jesper Strandgard Mortensen

Table of contents

Introduction

Nowadays around the world, each day there are more natural disasters where many people, animals, and nature suffer. Nature often receives huge damages that are often irreversible. On the other hand, if the damage is in one way or another recoverable, it takes decades or even centuries to regain its normal appearance. Most of these disasters are caused directly or indirectly by human negligence. There are many people who want to help, whether it's volunteering or with money support in order to help the affected people. The technology nowadays is one of the tools that can be used to help nature.

Problem Area

The project idea is a system where the user will be able to see many disasters from around the world in one place alongside some information about them, statistics or the status of the current situation. An option for the user to donate money for a disaster of his choice is the main feature. The visitor has the option of creating a profile where he will be able to input his personal information such as name, address, email and bank account information and become a user. The core functionality in the system "SaveTheWorld" is a donation by the user for a particular disaster. When the user handles his donating, he will be able to choose to donate to a specific disaster and to insert the amount of money that he wants to donate. The system will run on both web client (ASP.NET MVC) and desktop client (WPF) connected together with a service (WCF).

Another option the person who wants to donate money each month or the desired period of time, he will be able to use the option subscription and the amount which he will choose it will be taken from his bank account and the money will be distributed equally to each of the world's disasters.

The next functionality in the system will be an online shop where the customer will be able to make an order. The user will be able to choose attributes, such as color, the quantity for each product. The amount of money he will pay will be equally distributed to each disaster.

The system could take part in education becoming a place that everyone can visit and raise awareness about all the disasters around the world.

Permission access will be granted to two types of users. The regular user, who will have permission to shop, donate to disasters, subscribe and manage his profile. The administrative user, who will have the same rights as the regular user. However, he's privileged with manage options, where he can apply CRUD operation for all users, products or disasters.

Problem Statement

How can we implement a software application that will challenge the population to donate to the specific disasters of their choice?

How will we avoid concurrency when two users at the same time want to donate to the same disaster?

Method and Theory

After discussing the problem with other team members, we reached a consensus on the system we will build. In order to manage the workflow in our team, we were presented to two different approaches, Agile and Plan-driven approach. Our learning goal is to get familiar with the Agile approach since it's being used by many companies nowadays. The fact that it is much more flexible and scalable to work with, we decided to pick SCRUM methodology as it is one of the most common agile methodologies. The other options were XP and Kanban. We concluded that XP has a too strict priority order for our needs and Kanban was less suitable for scaling. In a nutshell, we chose SCRUM because it provided us with more freedom in constructing the solution and dividing work into smaller cycles during sprints. That is why we decided to use SCRUM combined with a few artifacts from the Plan-driven approach which will help us to get a much clearer idea of the whole system. In terms of database, we chose SQL Management Studio (SQL Scripts). We decided so because of scalability, security, and reliability in which we have assured ourselves during the previous projects. However, we considered another option which was MongoDB. After giving it a study we figured there is no default transaction support and joins were not supported so we concluded that it is not the right database for our application. We also used recommended literature by our teachers from Sommerville, Troelsen, and Miles to expand our current knowledge and create a work that would correspond with a university level.

System Vision

System vision in UP is usually a document that clarifies the system vision. It was used to maintain the project progress and never lose the main idea of the whole project and goals that have to be achieved along the way. It represents a catalyst for the project. The system vision was one of the first documents to be created for this project, basically right after the group was assembled. It was meant to be put together in an easy and simple understandable way containing a basic project idea.

For this project the system vision was stated as follows: "SaveTheWorld" is a program intended for philanthropists and environmentalists, who either would like to help the world by donating money or getting themselves more aware of ongoing crises in the world. "SaveTheWorld" is different from the other relief-like websites in a way that it obtains information about all the ongoing disasters in the world and the possibility to donate to each one of them. By creating this system vision and putting it together like that it was more than just a future goal set for the project.

SELECTION OF DEVELOPMENT METHODS

Plan driven vs Agile approaches

The difference between plan-driven and agile approach is, as the name tells, the one is more flexible and dynamic (agile). It advocates adaptive planning, evolutionary development, early delivery, and continual improvement. On the other hand Waterfall - Plan driven is more strict and dependent on continuous process documentation with repeatability, predictability and defined the incremental process.

- As you can see **Waterfall** could be called as “bureaucratic” development approach. It has already defined requirements, based on an already pre-defined structured process.

- On the other hand, **Agile** development is expected to change during the process and is very flexible towards changes.

Nevertheless, the final outcome is the same for both approaches. Both deliver top-tier software in an efficient way.

Agile development

Agile software development is based on an incremental, iterative approach. Instead of in-depth planning at the beginning of the project, Agile methodologies are open to changing requirements over time and encourages constant feedback from the end-users. Teams work on iterations of a product over a period, and this work is organized into a backlog that is prioritized based on business or customer value. The goal of each iteration is to produce a working product. In Agile methodologies, leadership encourages teamwork, accountability, and face-to-face communication. Business stakeholders and developers must work together to align the product with customer needs and company goals.

The 12 Principles of Agile Development

Agile development has 12 principles by which it is recognized

1. The highest priority is to satisfy the customer by delivering valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with preference to the shorter timescale.

4. Business people and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6. The most efficient and effective method of conveying information to and within a development team is a face-to-face conversation.

7. Working software is the primary measure of progress.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhances agility.

10. Simplicity - the art of maximizing the amount of work not done is essential.

11. The best architectures, requirements, and designs emerge from self-organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

AGILE development methods

Extreme Programming(XP)

Extreme Programming is a software development methodology designed to improve the quality of software and its ability to properly adapt to the changing needs of the customer or client. The core principles are feedback, simplicity and constant change. One of the main practices used also in XP is pair programming. When two programmers sit together and work on the code, it reduces mistakes and speeds up the development process.

Kanban

Kanban is one of the methods of agile development that emphasizes on continuous delivery of a product without burdening the development team. This means that for example if several tasks need to be done no more code can be deployed to testing until that field has been cleared out. We can describe Kanban as a billboard with attached tasks that need to be done, it has many steps where through which the product needs to get so the product is satisfactorily delivered.

SCRUM

Scrum is one of the most popular frameworks for managing and delivering software products and each project that can be assigned to teamwork these days around the world. Scrum is **Lightweight**, **Easy to understand** and **difficult to master.** In this approach, there are some groups of components that are used for creating the highest possible value for the product and they are - **Roles**, **Events** and **Artifacts.** The most famous term in Scrum is Sprint. Sprint is the time-box of four weeks or less where the development team has Sprint goal which has to be achieved.[1]

[1] "What is Agile Methodology? Tools, Best Practices ... - Stackify." 17 Sep. 2017, https://stackify.com/agile-methodology/. Accessed 10 Dec. 2019.

Plan driven development

The waterfall is usually planned with a **Gantt chart** which shows the deadlines for each phase. Once one of the phases is completed and then the team moves to the next phase. The team cannot go back to the previous phase without starting the whole process from the beginning. Before they move to the next phase the customer needs to review and approve the requirements.

Advantages of plan-driven method

- **Easy to use and manage:** Because waterfall follows the same pattern for each project, it is generally easy to understand and use. The team does not need any training before working on a waterfall project.

- **Documentation required:** Documentation is required in every phase of the project. That means the logic behind the code is well-explained.

- **Discipline:** Every phase of the waterfall method has a start point and end which are known to be very clear to everyone.

Disadvantages of the plan-driven method

- **Changes are bad:** Once a team completes a phase and they figure out something is missing from the requirements phase it can be very expensive to go back and fix it.

- **Accurate requirements:** The first phase of the waterfall is gathering the requirements and trying to identify the requirements early in the project can be a big problem since most customers don't know what they want.

- **Code later:** The project must go through two phases before the coding can begin. Customers won't be able to see MVP soon at the beginning of the project.[2]

[2] "Plan-driven software development - Wikiversity." 1 May. 2019, https://en.wikiversity.org/wiki/Plan-driven_software_development. Accessed 10 Dec. 2019.

Map with critical project factors (Boehm diagram)

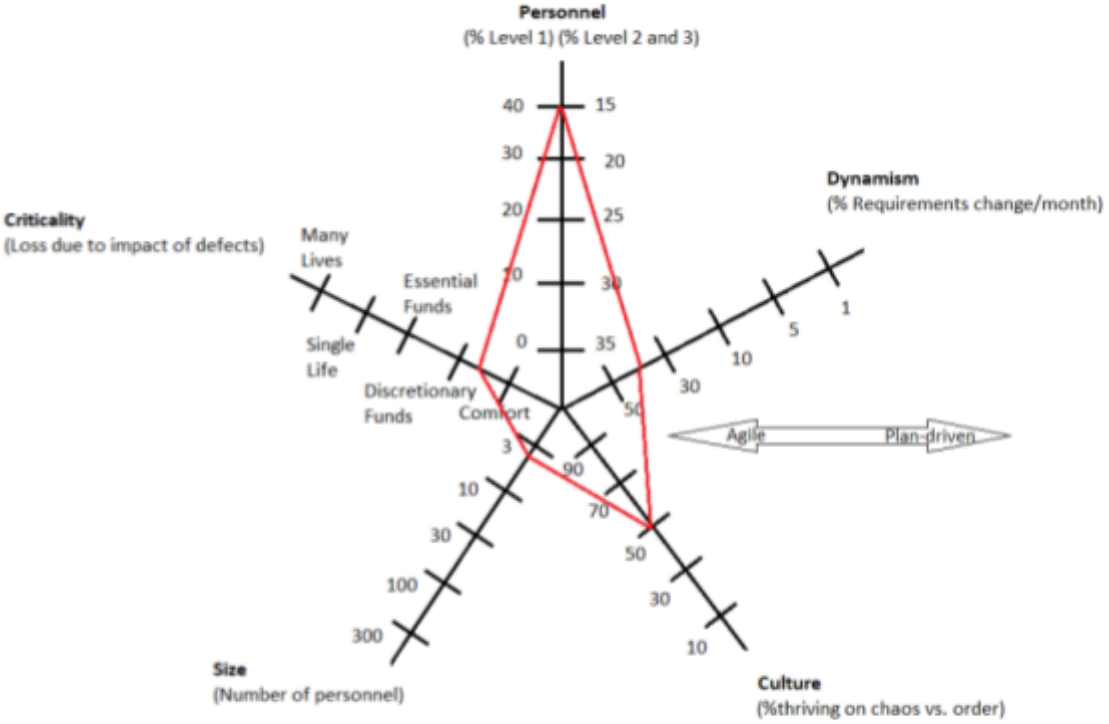

Figure 1 - Boehm Diagram

Before starting the work on the project of course we need to know the steps and the development approach that our team will follow to deliver a valuable product. Our decision is based on a map with critical factors or Boehm diagram where based on five measurable criteria we decide in which development approach we will stick with.

Based on our experience and level of method understanding and use, all members of the group are shifting between 1A or 1B level our Personnel supervision at the highest level, which means that we need to be guided by Plan driven approach.

Since our project is not that critical, our level of criticality is somewhere between Comfort and Discretionary Funds which is not that much plan-driven oriented. Another criterion that guides us to use an Agile approach is the Size of the team. Since we are three members in our group the approach that we should take is Agile.

The culture in our group is not set because of the new members of the group. During the study process that we had already in the current semester and mostly from the task that we did together, for now, our culture is in good control and stability. We expect that we will get to use to work together since we understand each other very well. That's why our level in this criterion is somewhere in the middle of the scale which means that we can use an Agile or Plan-driven approach.

One of the most important criteria that gives us an overview of deciding the development approach is the requirements for the system. If the requirements are expected to be changed in the future it's more Agile approach then Plan-driven and the same in the other way around. Since our requirements are expected to be changed in the future because of the system and the work that we will have with the group the value in Dynamism criteria is close to the center, which means that we should take the Agile approach.

The Boehm diagram gives us the best solution for choosing which development approach our team should use for creating the system "SaveTheWorld". Based on the result from the diagram our team decides to take the Scrum like the main approach that we will follow. But because of our previous experience and the fact that this is the first time that we are working with Agile methodology we also decided to include some practice from XP and UP processes. Another reason to include other practices is that SCRUM has no guidelines for development.

XP practices used

Simple Design

One of the XP practices that we are using for the system is that we keep the design as simple as possible. Our goal for the project is to make first all the functionality and the backend and then if we have time, improve the design. This principle gives us the opportunity to implement a bigger part of the backend and to have worked GUI.

Pair programming

Since we are students and all the efforts that we make to improve ourselves and our learning process, our team takes the option to do pair programming. This kind of work helps us to improve our programming skills because of working together and share our knowledge. The other purpose to do pair programming is because four eyes are better than two or we can save time for resolving different kinds of problems and to avoid them also.

Collective ownership

Our group has an agreement that all the members have the same privileges as the others over the whole project. We are using collective ownership practice from XP which means that everyone can remove, update or to add new information on the reports or to processes the software product.

UP artifacts used

Domain model

Because we already have some previous experience with UP artifacts and we touch the Agile approaches for the first time we need to include some "help" to make sure that our process of working on the project will be well planned and understandable for us. The team decides to include the Domain model because it gives a very good understanding of the communications between all the classes. Also is a really good overview of the entire system.

Relation Diagram

One of the requirements for the current project is to include a database for the system. There are two approaches to creating the project. The first approach is the code first another is the database first, create the database and then the code. Our team decided to take the first database approach. It was necessary to have an overview of all the primary and foreign keys. That's the reason for creating the Relation diagram.

SCRUM practices used

The whole project as we mention, based on the Boehm diagram our group decides to take as the main approach SCRUM development. As a group, we are focused on each different SCRUM team, SCRUM artifacts, and events. In our current project, we don't have real customers that's why all the roles were split differently to each of the members for each sprint. The reason for this was that each of the members can feel the real world and the responsibility of each role. About the artifacts, we use all of them to have a clear view of each sprint and for the whole process of working in the entire project. The last group of the CRUM artifacts is the Events where we made the most of our efforts. Each event is strictly followed by the group which helps us to handle problems and to keep track of our work also to plan each sprint and the whole system work.

PLANNING AND QUALITY ASSURANCE

Agile Planning

Agile planning is a project planning method that estimates work using self-contained work units called iterations or sprints. Sprints are periods of 1-4 weeks in which a team focuses on a small set of work items, and aims to complete them. Agile planning defines which items are done in each sprint, and creates a repeatable process, to help teams learn how much they can achieve. The agile process is focused on the concept of iteration. All sprints are of equal length, and an agile team repeats the same process over and over again in every sprint. Each sprint should result in working features that can be rolled out to end-users.

An **iterative process** allows the team to learn what they are capable of, estimate how many stories they can complete in a given timeframe (the team's velocity) and learn about problems that impede their progress. The team holds a planning meeting at the beginning of each iteration to break down each of the features scheduled for the iteration into specific technical tasks. Many teams set an overall goal for the iteration to help guide the selection of features. At the beginning of the meeting, the highest priority features are typically selected from the release plan. Each task has its own "task estimation" it can be from 1 hour to 2 days. Tasks that would take more than 2 days should be broken down into smaller tasks, as it would make conflicts and could delay the software building process.[3]

[3] "Agile Planning: Step-by-Step Guide | monday.com Blog." https://monday.com/blog/agile-planning/. Accessed 10 Dec. 2019.

Project Planning

At the beginning of planning the project first, we have to wrap up the idea of the software. We chose the idea for the project which will fit our requirements. The system is for donating to disasters around the world where we can display concurrency and a couple of important topics that we learned at school. When we set our final goal we started to create all the functionality for the system in the Product Backlog where we also estimated them. The Agile planning works with a sprint/iterations where we need to include the functionality from the Product Backlog and split them into tasks in Sprint Backlog for each sprint. In our case, we have 4 main sprints, where we have to implement all the functionality for the system. We follow all the rules of Agile Planning, where we saw the power of planning and especially when the team has everything planned and performs step by step.
In the plan below is the schedule in days for all the sprints.

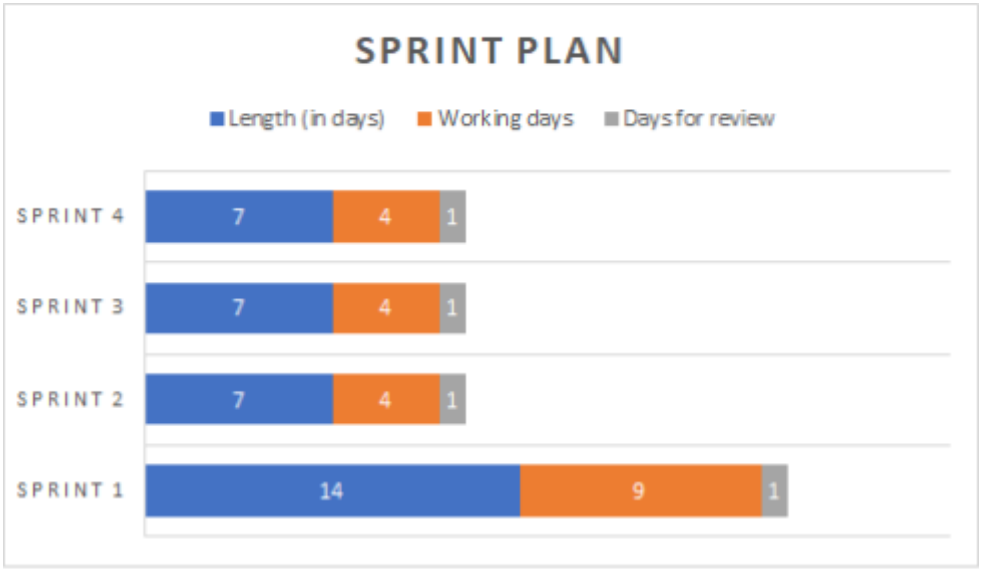

Figure 2 - Sprint plan

Quality Assurance

The quality manifests itself in how flawless the systems functionality is. Good quality is an important necessity these days, since there has been a lot of poorly constructed software as the market is fast paced and require early deployments, that are not well-tested. Development team being highly focused on the quality of their software products means that there will be some costs saved along the development way. Quality also depends on the size of the project and the team. Quality Assurance is based on standards that are typically certified.
Since we are a group of three members the quality assurance of the project is important but in a more informal approach where we have less paperwork, mainly due to our group size. In the development approach of our choice (SCRUM), the team relies on good communication and meetings to discuss problems in this area.

Testing

Testing is one of the most important pieces when it comes to developing an application. We faced an option to do the code first or the test first. We went with the code first approach because we did not want to possibly lose sight of our goal for the sake of passing a method before it was even written. The main intention of using tests was to check the logic behind the code. The tests were decided to be implemented at the end of the last sprint for what the group decided were the main features of the application. We decided to test tasks in the user stories we considered important and not for every task. Tasks that were tested included user login info check, checking if the disaster already exists (for the admin mode) and creating a new product (also for the admin mode). For this project, we used unit tests in Visual Studio. We had the option of choosing between unit and integration tests. The group decided to use unit testing. The reason for that was that we were looking for a faster, less time-consuming way, but still efficient enough. Knowing that integration tests were more complex and take more time to write and execute the choice was obvious. Using the assert statement within the unit test we were able to check whether the methods that were tested return wanted results. The tests were as previously mentioned done in the end to make sure that the methods work flawlessly. We found unit tests to be extremely helpful and reassuring that the work is on the right track.

```
public class CreatingProductTest
{
    [TestMethod]
    0 references
    public void CreatingAProduct()
    {
        ProductCtrB productCtr = new ProductCtrB();
        ProductB prod = new ProductB();
        prod.ProductName = "umbrella";
        prod.ProductDescription = "a very nice umbrella";
        prod.Stock = 10;
        prod.Price = 10.35M;
        prod.ProductId = 6;
        Assert.IsNotNull(productCtr.CreateProduct(prod));
    }
}
```

Figure 3 - Code

For example, when testing a creating a product method, we wanted to make sure when we create a product with its belonging properties, the method returns a new object holding all the relevant info about the new product. For that purpose Assert.IsNotNull method which tested if the specified object that was passed is not null.

Quality criteria and architecture

FURPS+

FURPS+ is a model representing the classification of quality attributes and is used as a checklist for the system requirements to cover. FURPS is a technique of validating the prioritized requirements after understanding the client's needs and necessities. FURPS abbreviation stands for Functionality, Usability, Reliability, Performance, and Supportability where some of them are non - functional requirements (URPS). The "+" in FURPS+ allows us to specify constraints, including design, implementation, interface, and physical constraints. Basically, the functional requirements are planned when the team designs the system and they define a specific task or functionality in the system. The non - functional requirements are implemented in system architecture.[4]

Non-functional requirements

In the system "SaveTheWorld" our non - functional requirements are based on the expected behavior and how the system should respond is some specific request or actions from the user.

Usability

We expect our program to be used by the people enthusiastic in saving the environment and people who want to help prevent disasters. Everyone is more than welcome to use it. The visitor needs to create a profile with his name, email, address, phone number, of course, password and his bank account information like a number of the account, expiry date, and CCV. By doing so, the visitor becomes the user with all belonging features.

Reliability

The system will have different message errors for the specific occurrences. For example, if the user tries to donate or to make an order without logging in first, the system will send a message error like this one: "You have to log in before donation!". The failure frequency, in this case, will be the same as the number of attempts to donate by the user.
On the other hand, when a user is making a profile, the system will hash his password for better security.

Performance

Since our system is using Entity Framework which is built on top of ADO.NET, the response time sometimes can be a little bit longer than normal. Another good performance expectation expected is the accuracy in dividing the total amount donated by the total amount of disasters. By doing so the user can donate the same amount of money to each disaster from the list using simple division.

---

[4] "What is FURPS+? – Business Analyst Training in Hyderabad ...." 5 Aug. 2014, https://businessanalysttraininghyderabad.wordpress.com/2014/08/05/what-is-furps/. Accessed 12 Dec. 2019.

Supportability

The fact that our system is based on Multi-layer architecture, allows developers to make changes to the system easily. It gives good adaptability to future implementations. This also makes the whole software more maintainable.

Architecture

One of the most important parts of developing software systems is their architecture. Architecture is affecting the whole system. If the architecture is designed well the maintainability increases and all the possible fixes in the future become easier to solve and notice in the first place. Another good reason to have well-designed architecture is that the system price and performance will be raised. Also for the developers, it will be easy to split the work between the different architecture layers.

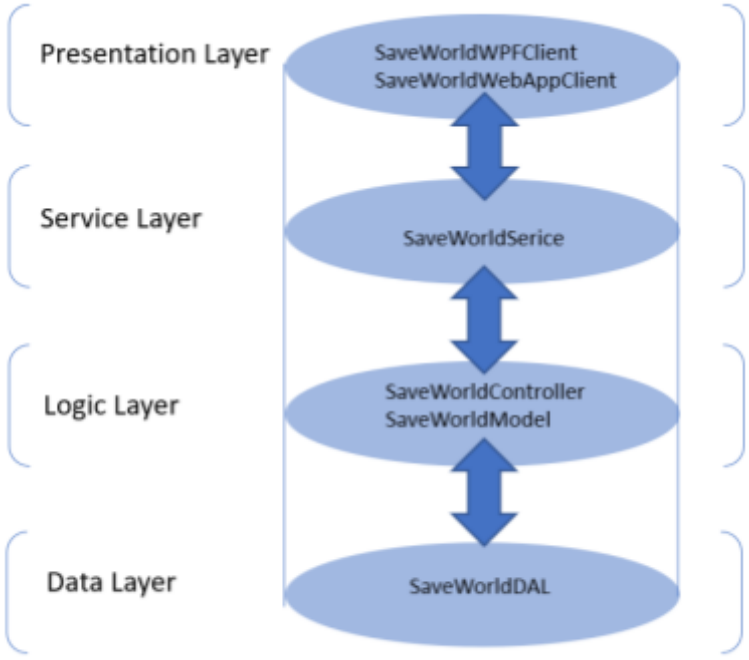

Figure 4 - Architecture

Our architecture has four main layers. This type of architecture supports high cohesion and low coupling. Essentially, if there are changes being made in one of the layers, it would not affect the other layers, making them more independent.

Presentation Layer

Presentation Layer is intended to hold our clients or client projects to be more precise. Both dedicated and web clients will be held in this layer. We have used WPF for a dedicated client and ASP.Net MVC for web clients. The whole point of the Presentation Layer is to make an interface for a user. It is also a place where all of the program's functionality will be graphically presented in a user-friendly way. Our type of system architecture allowed us to make changes in the Presentation Layer without changing the rest of the layers since it only communicates with the Service Layer.

Service Layer

We have been working in Microsoft's Ecosystem, therefore the usage of Windows Communication Foundation has been the logical step for us to take. We used this layer to allow the clients to the business logic for both Desktop and Web clients. The Service Layer only communicates with the Logic Layer. In our service, we are implementing all the necessary service operations from our controller or Logic Layer. For example, the user in the client will be checked first with the service operation "CheckLogin". What it does is checks if the user already exists. In our services, we relate all the data that the user request, with the Logic Layer where we manage and control it.

Logic Layer

Adding the Logic Layer gave us a better understanding of how quality architecture should look like. In this layer, we control and handle the information coming from the Service Layer and send it to the Data Layer, where the whole process with handling data is happening. The Logic Layer in our case helps us control all the methods for the system. For example, in the user controller, we have a method "GetUser". The user controller is sending an id as the user's parameter and returns the object which is found or defined in the user data class where all operations are happening.

Data Layer

One of the layers in our architecture is the Data Layer, where the whole processing part with the different CRUD operations is happening. For this particular project, we are using Entity Framework to interact with the database. This layer in our project is the most critical part of the whole system. The reason that we chose to implement all the manipulations with the database here, is that this is the layer that communicates with our database. Another reason is that all the members in the group have agreed that the controller or the Logic Layer is being used just to navigate the data from the user and to the user, so none of the data interactions happen there.

FUNCTIONALITY/ANALYSIS

Story Cards

One of the highest levels of defining requirements from the customer is the Story Card. Each story card consist of one iteration or sprint and represents the user stories. The requirements, which are written by the customers, have to be explained with short texts including the business terms.
In the front of the story card, the functional requirements of the priority level and estimation level are presented. On the back of the story card, the acceptance test is presented. If the story card is implemented correctly, the acceptance test should execute without errors.
The reason we chose story cards in this format, was that it included all the vital information. It holds the ID of a story, estimated time, all the relevant info about the actions that need to be performed by implementing the user story and by who. One neat feature about story cards in this format is that acceptance tests are written in the back of each one of them, but for the use of this report, they have been put under each story card. Both story cards and acceptance tests are owned by a product owner. And acceptance tests are written to make sure that the story works as a product owner intended it to, that it fulfills all the required functionality.

Story Name: Inserting new item(s)		ID:1
As a	Admin	
I want to	Insert new item(s) to the list of products	
So that	My web shop will have more products	
Estimate: 7		Priority: 3

Acceptance test:
1. Login as admin.
2. Open manage page for products.
3. Insert the data for the new item.
4. Press button "Save".

Story Name: Removing existing item(s)		ID:2
As a	Admin	
I want to	Delete item(s) from the list of products	
So that	We won't have the specific product anymore	
Estimate: 5		Priority:3

Acceptance test:
1. Login as admin.
2. Open manage page for products.
3. Choose the desired product.
4. Press button "Delete".

Figure 5 - Story cards & Acceptance test

These two-story cards show the requirements for inserting new items and remove the existing one. One of the stories above is about inserting a new item(s) which is stated in the name of the story. The action is being performed by the administrator of the shop and he wishes to add more products to the existing list of products. The purpose of it is to make a supply a bit wider than it is. As mentioned the story card contains estimated hours of work and priority. Our group prioritized all the stories from 1 to 5 where 1 is the highest priority and 5 the lower one. For the special story, we have an

acceptance test which after following all the steps the system should return completion.

Product Backlog

PRODUCT BACKLOG				
ID	**Story name**	**Actor**	**Estimation**	**Priority**
9	**Subscription Options**	User	20	1
15	**User choosing to donate to a disaster**	User	40	1
16	**Shopping basket - put**	User	25	1
17	**Remove items from basket**	User	4	1
20	**Pay for the items**	User	10	1
10	**Donate anonymously**	User	6	2
18	**Edit the item(s) in the basket**	User	4	2
1	**Inserting new**	Admin	7	3

Figure 6 - Product Backlog

There are multiple definitions as to what a product backlog is, but the most common and simple one is that it is a list of things to be done within a project. In a way, product backlog revolves around the user stories with their IDs, users, estimated hours of work they require and priority. The items, or stories, in the product backlog are listed by priority. Each story has its name, ID and estimated hours of work projected in addition. Later on in the report, it will be shown how these stories are often divided into tasks when they are being worked on in sprints but for this purpose, it is only story name and named features being listed in the backlog. An important thing to mention is that the owner of the product backlog is a product owner. The product owner is also responsible for the content, availability and the priority of the product to-do list. The product backlog was so to speak a backbone of the project in this particular case and it was used to decide which stories will be dealt with in each particular sprint. The importance of product backlog lays in allocating just the right amount of working hours intended for each sprint and its belonging user stories to follow the ideal streamline. Throughout the sprints and the project, the product backlog was of great help in realizing in which phase the project is and how much work was done or needs to be done. Ultimately it leads to the optimum allocation of resources and time to finish the tasks that were set in the beginning and along the way.

In the beginning, when the team sets all the story cards and their estimation, we were thinking that we will have enough time to implement all of them. During the implementation we fall into some specific problems were we spend more time than we expected. Because of the pressure of the time, three of our stories were not implemented and they are Subscription Option, Anonymously donation and Edit item in the basket.

Risk analysis

Risk	Probability	Effects
Misunderstanding between the group members	Medium-Low	Critical
New technology requirements for implementation	High	Catastrophic
Communication between some of members computers is crashed	Low	Marginal
Unexpected problems with WCF service	High	Catastrophic
Time pressure	Medium-High	Critical
New development processes for implementation the system	Medium-High	Critical
Unexpected errors during coding	Medium-Low	Serious

Risk 7 - Risk Analysis

In each project doesn't matter what is the subject, every time there is a number of risks that the team or the person who will make the creation of this project has to be aware of them. In case that some of the risks happens, it will affect the schedule and the whole process depending on the risk. The Risk Analysis is focusing exactly on what is the probability of the special risk to happen and how it will affect the whole project.

At the beginning of our project the risks that we mention in our risk analysis, some of them happened and affect our implementation. The risks that we expected at the beginning were in close relation with all the new materials that we learn during this semester. The reason that they appeared in our risk analysis is because of the first time that we touch with them. For example, this was our first project where we have to include WCF service in our code and as was expected we had that problem during our process. The action that we use to handle it was by mitigating and accept it. We spend time at the start to prevent this risk and also during the implementation of our system we spent time to handle it. In the end, we understand why this unexpected problem occurs and it was because before that another risk that we expect appeared. The problem and the risk were that the communication between our machine was somewhere crashed. That risk we handle it by action avoid. This gives us the opportunity to continue working on the project. Of course, we found another way to communicate so the problem was gone. Until the end of our work the other risks that we expected at the beginning for our joy, they were not that big deal during the implementation of our system.

Design

Domain Model

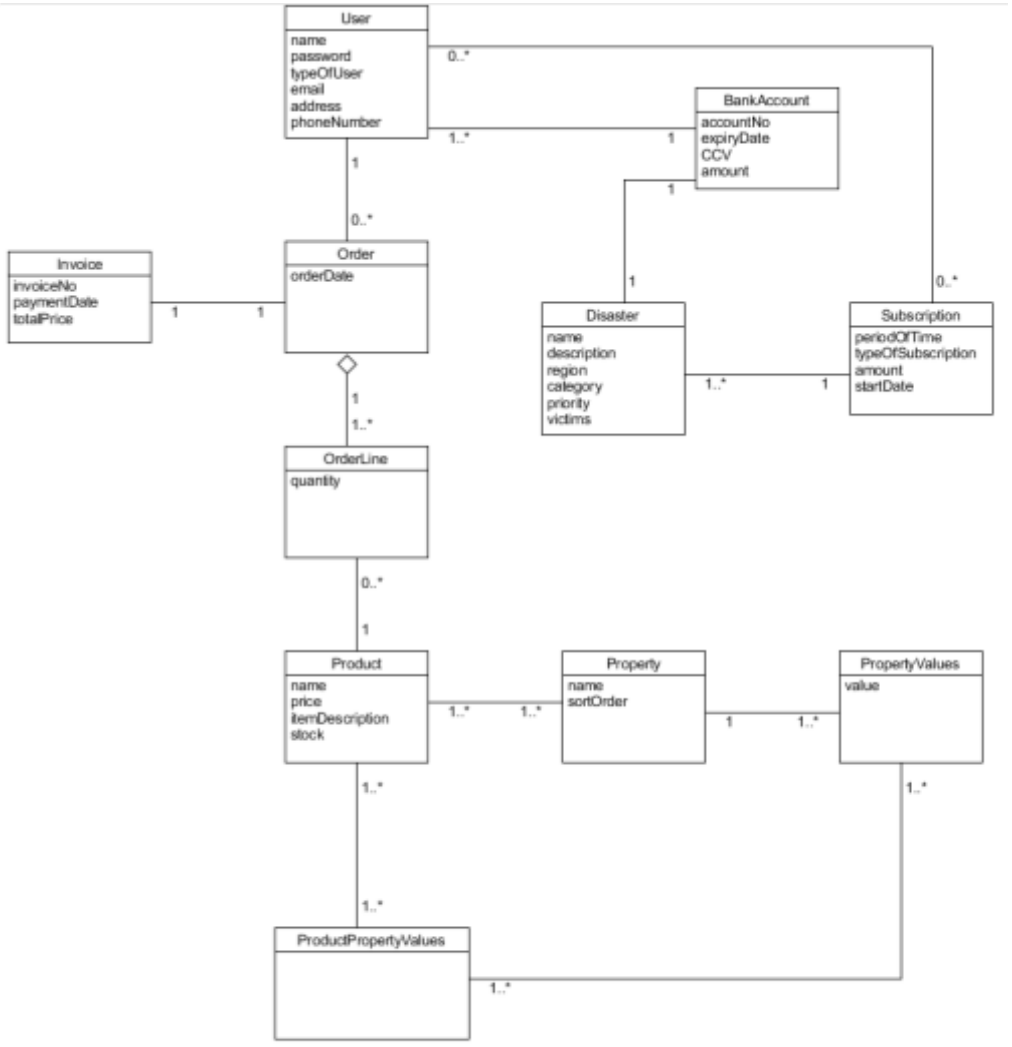

Figure 8 - Domain Model

The domain model as shown above consists of eleven classes in total. The lines between the classes show the dependency of one class to the other(s) and their associations with multiplicity. Each class above is shown with its belonging attributes but no methods as it is meant to be when making a domain model. Empty diamond shapes seen in the model above show an aggregation or relationship between a whole and its parts. The latter is shown between the order and OrderLine class. The model has been made using UML which has all the necessary tools to make the model make more readable and understandable. We decided to use this artifact from UP as it gives us a great overview of the software which is being developed. For our team was important to negotiate upon this and reach the same

consensus after which we can later develop the system. We believe that the Domain Model is a necessary part of the “System development world”.
In this domain model, we use two patterns. The first one is the Agreement pattern. We are using it to improve the communication between our users and the products. The agreement between them is the Order. Each time when the user wants to make the order he is creating a first order line that consists of the product and the desired quantity. Here we include the Relation pattern where whenever a user creates an order line or add a new product to his basket, the order line aggregated to order. Then when he finishes adding the product in the order, he finishes the order and all the order lines that are created to take the order id in the database as a foreign key. This is one of the main functionalities that our system will offer to the user or donate to all disasters. The total amount from the order will be distributed equally to each of the world's disasters.
The system “SaveTheWorld” will be able to give the option for different users to use the same bank account. They will be able to use this functionality in the shop and donation part.
Another functionality similar to the shop is the subscription where instead of choosing and buying products the user will be able to choose the period of time he would like to donate for, with desired amount. The amount he chose will be transferred from his bank account to the disaster bank account automatically according to his subscription type.
The system can give access to the user in two different types of users. The first one is the regular user who will be able to donate to a specific disaster, to make a subscription and to donate through shopping. Also, he will be able to change information about his profile in the system. Another type of user is admin. Alongside all the features regular user has, admin will also be able to manage disasters, users and shop by CRUD operations .
The products will be made from recycling materials and the shop is not going to have many different kinds of t-shirts or many different kinds of umbrellas, and each product will have different properties and different property values. That’s why we include the classes Property, PropertyValues, and ProductPropertyValues where the data for the specific product and its values is going to be stored and managed. However, after careful consideration these features were abandoned by the development team because of their time-consuming nature.

Relational Model

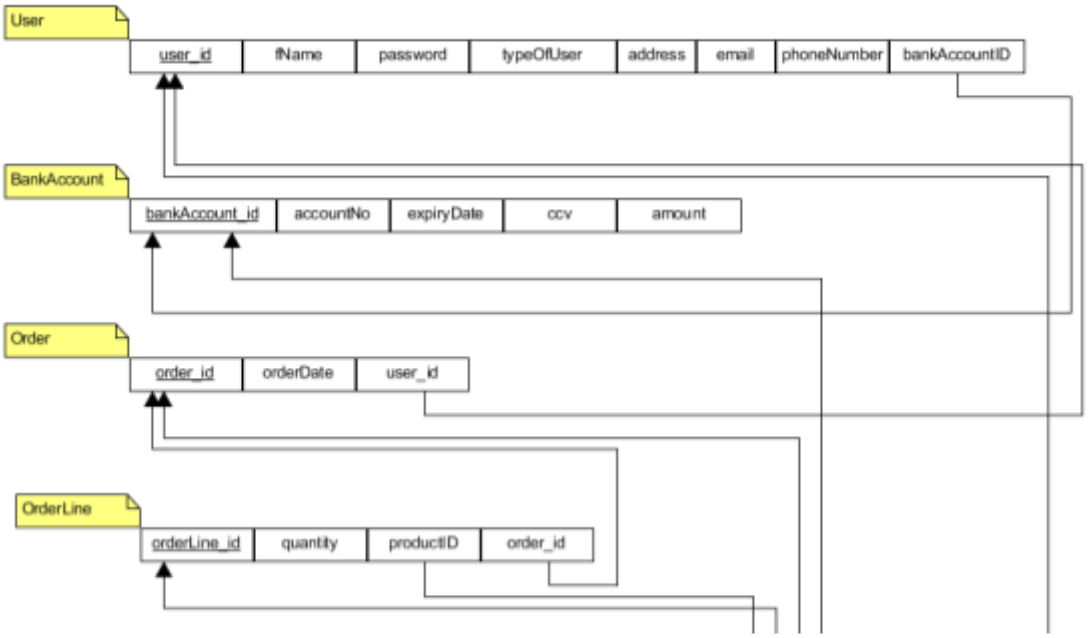

Figure 9 - Relational Diagram

The relational model represents how the data is stored in relational databases. The data is stored in the form of tables. The relational model is derived from the domain model. For this specific purpose domain model consisted of eleven classes (tables) in total.
Each class(table) consists of attributes. One of the key features of the relational model is the usage of keys. To be more concrete primary and foreign keys. Primary keys are unique for each table and they have been stored in the form of id. Foreign keys are important because they show how tables correlate between themselves. While making the relational model an additional three tables(relations) were added.

Sprints

Sprint 0

The purpose of Sprint 0 was to set up all the necessary tools that we will use, all the principles that we'll follow during the project and the most important to set up the idea for the software. Our group used GitHub Desktop for sharing the project and its development stages. The whole idea for the project was set up in the start of the semester but all the necessary properties that are most important were wrapped up in sprint 0.

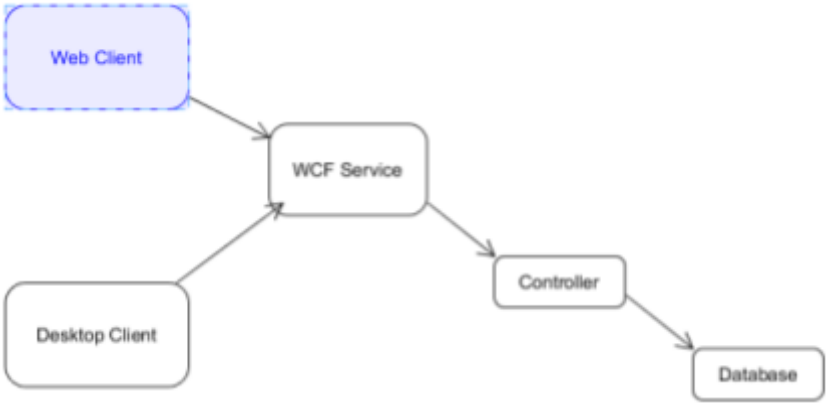

Figure 10 - Candidate architecture

To be more clear the view in front of us, we create candidate architecture which allows having good understanding how the project should be architectured which is one of the most important purposes for our group.

Sprint 1

Sprint one went as planned, especially because we have been able to prepare at Sprint 0, where we managed to set up the version control, Trello and other necessary services. At the beginning of Sprint 1, we have experienced several problems with Visual Studio, as it gives us unsolvable conflicts so we were tempted to create a whole new project. That's why there is a deflection in the burndown chart. Nevertheless, during Sprint 1 we were able to create a login & registration page for the user and design the overall layout of the application. We created a website as well, where we prepared the graphical layout.

Sprint goal

The goal for this sprint is to enable the visitor to become a user. Making it possible for him to create an account, log in to the existing account and log off. These features will later be crucial since most of the program's functionality will be related to users. The other functionalities will include donations, shopping and possibly other features.

Burn Down Chart

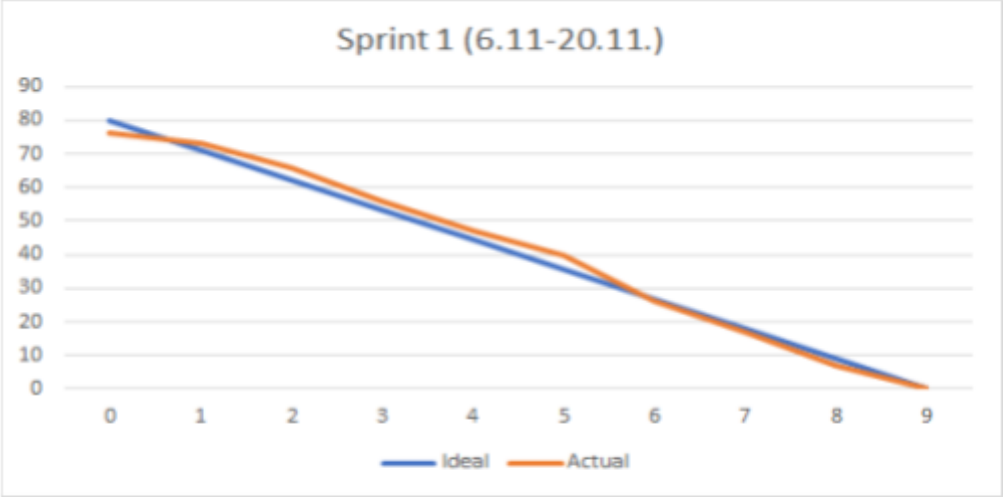

Figure 11 - Burndown chart, Sprint 1

It seems it went pretty smoothly but as it was for our sprint zero, where we used SCRUM methodology we were bit lost in the beginning, but we are confident that next Sprint will be better in terms of management. We had as well bit difficulties with the product backlog and user stories, how is it related so we spend quite some time on that too.

Sprint Backlog

The sprint backlog is a set of items picked from the product backlog. The structure of a sprint backlog, in this case for Sprint 1 concretely, can be seen in the table below. It consists of a user story, task, members of the team responsible for each task, estimated amount of work and how those hours were allocated throughout the sprint. In Sprint 1 there were three user stories to finish with seven tasks in total. This project group is made up of three members so the work was to be divided between them depending on their availability during the sprint. The sprint lasted for 9 working days and was the longest one of the following ones. All the functionalities and features that were planned were also implemented during Sprint 1.

Sprint backlog 1												
User Story	Task	Member	Estimate	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9
Creating an account	Making a registration page	Jan, Denis	10	8	7	5	4	2	0	0	0	0
	Coding to add new account into database	Valentin	12	12	11	10	8	6	4	2	0	0
	Coding to add bank account to the account	Valentin	12	11	10	9	7	6	3	2	0	0
Logging in to the account	Making a login page	Denis, Valentin	10	8	7	5	4	4	3	1	0	0
	Checking if the email and password matches	Denis	12	13	12	10	10	9	7	5	3	0
	Making log in button working	Jan	10	11	10	9	7	6	4	3	1	0
Logging off	Making Log out button working	Jan	10	10	9	8	7	7	5	4	3	0
		Total hours:	**76**	**73**	**66**	**56**	**47**	**40**	**26**	**17**	**7**	**0**

Sprint Length	2 weeks
Workdays During Sprint	9 days

Team Member	Available days during Sprint	Available hours per day	Total available hours during Sprint
Jan	4 days	5 hours	20 hours
Denis	6 days	6 hours	36 hours
Valentin	6 days	4 hours	24 hours
Effective/Ideal hours:			80 hours

Figure 12 - Sprint 1 Backlog

Sprint review

In this sprint, every user story from the product backlog was implemented in the program. The working hours have been aligned almost perfectly to cover the estimated time that chosen user stories from the product backlog. Good allocation of working hours and required work led to finishing all the work in time and making the required functionality work, satisfying the acceptance tests.

Sprint retrospective

Sprint 1 went well and all the user stories were finished as planned with optimum working hours allocated per each story. The fact that this crucial part in making user accounts was finished opened the path to create almost all the other functionalities and features of the program. There were minor setbacks with GitHub and synchronizing work that has been committed through it. Since these problems have been solved during the sprint it leaves room to believe that there is an opportunity to allocate more hours for incoming user stories for the next sprint.

Sprint 2

Sprint goal

In Sprint 2 the goal is to make some features available for the user. The intention is to make the user possible to see the list of the products from the online shop and add products to the basket. The other thing is to make it possible to list all the disasters, choose one and donate to chosen disaster a desired amount of money.

Burn Down Chart

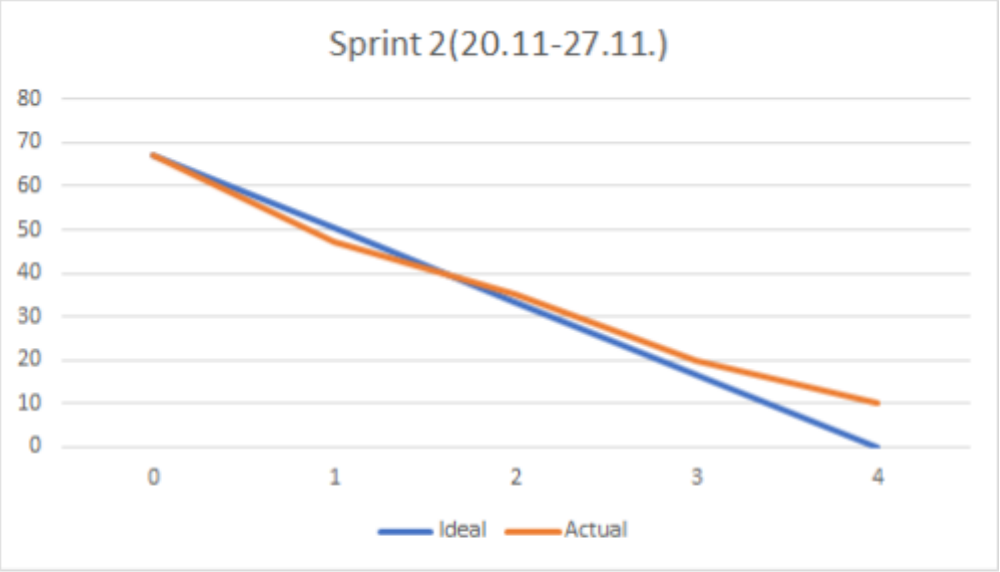

Figure 13 - Sprint 2 Burndown chart

Sprint 2 was trying to follow the almost perfect streamline that was set in the first sprint. As it is visible on the graph above the actual streamline was a bit below an ideal one until midway through the sprint. After that, the work input became higher mostly because it was realized that one of the user stories required more work than it was previously set. In the end, it led to ten estimated hours being postponed to be finished in the next sprint because the task proved to be a slight challenge.

Sprint Backlog

Table for Sprint 2 backlog consisted of two user stories and four tasks for each user story. The sprint itself lasted for four working days and the tasks were allocated and divided so each member of the team would have the time to finish them. At the end of the sprint, all of the planned tasks were finished apart from a single one which proved to have demanded more work than initially planned so it was simply postponed to the next sprint.

Sprint backlog 2							
User Story	Task	Member	Estimate	Day 1	Day 2	Day 3	Day 4
User choosing to donate to a disaster	Making a disaster page	Valentin, Jan	8	5	2	1	0
	Code to donate to a single disaster	Valentin	14	10	7	3	0
	Check if user is log in already	Valentin	8	5	3	1	0
	Check if the amount is less than the balance	Jan	10	7	4	2	0
Shopping basket - put	Making a shop page	Jan	5	4	3	1	0
	Code for adding item in order line	Denis	7	4	3	1	0
	Code for creating an order line	Denis	6	5	4	1	0
	Code for creating an order	Denis	7	7	9	10	10
		Total hours:	**67**	**47**	**35**	**20**	**10**

Sprint Length	1 weeks
Workdays During Sprint	4 days

Team Member	Available days during Sprint	Available hours per day	Total available hours during Sprint
Jan	3 days	5 hours	15 hours
Denis	4 days	6 hours	24 hours
Valentin	4 days	7 hours	28 hours
Effective/Ideal hours:			67 hours

Figure 14 - Sprint 2 Backlog

Sprint review

In Sprint 2 there were two user stories to be implemented which was also one of the biggest features of the program. Making it possible for a user to donate to a single disaster was successfully implemented. Putting desired products to the products basket (adding order lines to the order in other words) went almost as planned. All of the tasks in this user story were finished apart from creating an order. In the end, this resulted in leaving 10 hours of unfinished work which is to be prolonged to a Sprint 3.

Sprint retrospective

Following the same pattern which worked without a problem in the Sprint 1 lead to the same attitude approaching Sprint 2. User stories were nicely and evenly divided into tasks. There were 8 tasks and 7 of them were finished by the end of the Sprint 2. The code for creating order as the last task in the shopping user story required more work than planned so it couldn't be finished as planned. It is important to mention that it is the first time since the sprints have started that there was a case of a task that required more work that couldn't be finished being transferred to the other sprint. For the next Sprint, the plan is to allocate work even better to make the required functionalities in time.

Sprint 3

Sprint goal

The goal for this sprint is to implement features for administrators such as managing shop page, user and disaster page. It would enable him to do all sorts of CRUD operations within the named pages. One of the highest priorities is also to finish the remaining tasks from the previous Sprint 2, which was creating the order. It makes it a high priority because by finishing it would enable us to make a shopping page ready for implementing other features such as paying for the order and removing products from the basket(order lines from the order).

Burn Down Chart

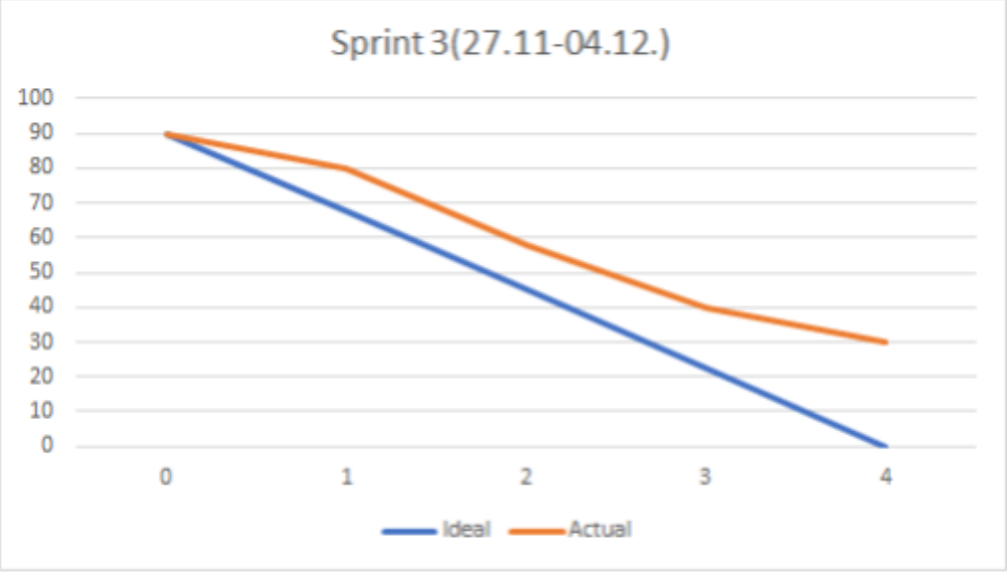

Figure 15 - Sprint 3 Burndown chart

The graph for Sprint 3 looks a bit different than it was the case for the first two sprints. Extra work carried from the Sprint 2 and a will to finish most of the user stories by the end of Sprint 3 took its toll on the shape of the actual streamline. Almost from the start, the work that was put in was higher than the ideal streamline and it kept the same pattern until the end of the sprint with some minor oscillations along the way. By the end of the sprint, there were thirty hours left to be done and which were transferred to the next sprint.

Sprint Backlog

Sprint 3 has officially lasted four working days and consisted of more user stories and tasks than the previous ones. An important thing to mention here is the fact that one task from the previous sprint was set to be done in this one and it affected the planned workflow. There were many tasks here mostly because based on the availability from our group members in this sprint, it was decided to implement all of the administrator functionalities in the program such as managing the shop page and disaster page. Furthermore, all of the features for the users were set to be done by the end of the sprint leaving room for testing after it was done.

Sprint backlog 3							
User Story	Task	Member	Estimate	Day 1	Day 2	Day 3	Day 4
Shopping basket - put	Code for creating an order	Denis	10	9	6	6	0
Remove items from basket	Implement the deleting order line	Denis	2	2	0	0	0
	Code to return the stock to the product	Denis	2	2	2	2	3
Pay for the items	Assign the order id to each order line	Valentin	3	3	2	0	0
	Implement the transfer of the money	Valentin	7	7	7	7	8
Inserting new item(s)	Making a manage page design	Valentin	2	0	0	0	0
	Making the product manage page design	Jan	2	0	0	0	0
	Code to add new item in the database	Jan	3	3	4	3	3
Removing existing item(s)	Implement to delete item from database	Denis	5	5	3	0	0
Updating existing item(s)	Code to update the desired item in database	Jan	7	8	7	5	5
Adding new disaster	Making the disaster manage page design	Jan	2	0	0	0	0
	Code to add new disaster in the database	Jan	6	5	4	2	2
Updating existing disaster	Code to update the desired item in database	Valentin	7	6	5	2	0
Updating account	Making profile page design	Jan, Valentin	2	1	0	0	0
	Code to update the desired fields in the user profile in database	Valentin	5	5	3	3	4
Check the disasters	Implement the permission access	Denis	1	1	0	0	0
Viewing a list of the user's accounts	Code to show all the available users in the system	Jan	2	2	0	0	0
Deleting user's account	Implement to delete user from database	Valentin	5	5	6	6	5
Editing a user's profile	Code to update the desired user in database	Jan	7	7	5	3	0
User viewing the list of disaster	Code to show all the available disasters in the system	Denis	5	5	4	1	0
Checking information about chosen disaster	Implement the permission access	Valentin	2	2	0	0	0
Viewing account	Implement the permission access	Denis	2	2	0	0	0
View my basket	Code to show all the available order lines in the current order to the user	Valentin	1	0	0	0	0
		Total hours:	**96**	**80**	**58**	**40**	**30**

Sprint Length	1 weeks
Workdays During Sprint	4 days

Team Member	Available days during Sprint	Available hours per day	Total available hours during Sprint
Jan	4 days	8 hours	32 hours
Denis	4 days	8 hours	32 hours
Valentin	4 days	8 hours	32 hours
Effective/Ideal hours:			96 hours

Figure 16 - Sprint 3 Backlog

Sprint review

Sprint 3 consisted of twelve user stories, counting the one that was inherited from the previous Sprint 2 and it was finished in this sprint. A total of seven tasks remained unfinished with only two of the tasks proving to require more working hours than planned. Sixteen tasks were finished by the end of the sprint out of the total twenty-three tasks.

Sprint retrospective

Looking back on this sprint it contained the largest amount of user stories and tasks than any of the sprints before. Most of the program's functionalities and features were planned to be implemented by the end of this sprint and, for the most part, it turned out like that. There are only a few unfinished tasks left for the last sprint so the fact that some of the work from this Sprint number 3 is going to be added to the next one does not present any sort of problem or a drawback. This is supported by the fact that a similar thing happened in Sprint 2 when there was a task that needed to be transferred to the next sprint so the group was better prepared for such a scenario.

Sprint 4

Sprint goal

The main goal for Sprint 4 is to finish all the user stories that were started in the previous sprint. What it means that there are three more stories in which the team expects that they won't be implemented because of the pressure of the time. The main reason is that the group started noticing that some tasks might take a longer time to finish than estimated which was anticipated in a way so Sprint 4 was in a way used for just to finish the stories from Sprint 3. As mentioned previously, new technologies were being implemented and it took some time to get used to working with them which created some setbacks. There is a total of thirty hours estimated left to complete tasks in this sprint.

Burn Down Chart

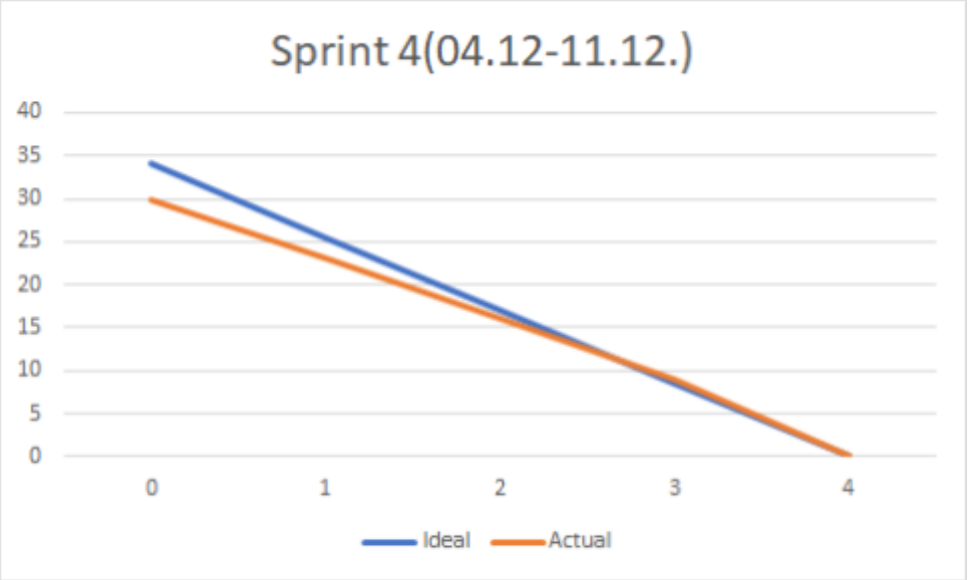

Figure 17 - Sprint 4 Burndown chart

The sprint and the hours were well estimated by the team which was the reason why, for the most part, actual and ideal streamline are almost perfectly matched. Knowing that there is not as much time left it was decided to finish only the user stories that were started in Sprint 3. The decision didn't affect our application much since we have already implemented or were close to finishing the implementation of the most important features of the application. Exactly what was the reason that actual streamline found itself a bit below the ideal one at least for the first part of the sprint. By the end of the sprint, all the desired and started user stories were finished.

Sprint Backlog

Sprint lasted for four working days and was officially the last one for this project. As mentioned there were no new, or unused user stories from product backlog added. Only the user stories that were started in the Sprint 3 were used and finished in this sprint. There was a total of seven user stories and a single task for each user story left. The estimated time was thirty hours for these user stories. The functionalities left to be finished included some of the features for the user on the shop page and some of the features for an administrator.

Sprint backlog 4							
User Story	Task	Member	Estimate	Day 1	Day 2	Day 3	Day 4
Remove items from basket	Code to return the stock to the product	Denis	3	2	2	1	0
Pay for the items	Implement the transfer of the money	Valentin	8	5	3	1	0
Inserting new item(s)	Code to add new item in the database	Jan	3	3	2	1	0
Updating existing item(s)	Code to update the desired item in database	Jan	5	4	2	2	0
Adding new disaster	Code to add new disaster in the database	Jan	2	2	1	0	0
Updating account	Code to update the desired fields in the user profile in database	Valentin	4	3	3	2	0
Deleting user's account	Implement to delete user from database	Valentin, Denis	5	4	3	2	0
		Total hours:	**30**	**23**	**16**	**9**	**0**

Sprint Length	1 weeks
Workdays During Sprint	4 days

Team Member	Available days during Sprint	Available hours per day	Total available hours during Sprint
Jan	3 days	4 hours	12 hours
Denis	2 days	5 hours	10 hours
Valentin	3 days	4 hours	12 hours
Effective/Ideal hours:			34 hours

Figure 18 - Sprint 4 Backlog

Sprint review

Sprint 4 was the last in this project and contained all the stories that were not possible to finish in the Sprint 3. In some ways, this was what Sprint 4 was expected to be used. In short, it was meant to be a fall back in case some of the tasks required more work than planned. User stories regarding CRUD operations for the administrator and a user were done in this sprint.

Sprint retrospective

Considering all things said for this and previous sprints, the work allocated for this sprint was just enough to let the team finish all of the started user stories from the Sprint 3 and allocate the rest of the time where it's needed the most. There were three user stories from the product backlog that we decided not to pursue. The decision was made because we considered there was not enough time left to implement them. It was mostly CRUD operations left for this sprint which didn't require learning about new technologies like some of the tasks before did which required some extra working hours being put in. In case most of the work was done by the end of the previous sprint, this sprint could've been used for implementing additional feature but the requirements of the project were met and this part of the project can be concluded now.

CONCLUSION

In this project, we were introduced to a lot of new approaches to system development. Unlike the first-year project where we have used UP, we decided for a new approach. It was decided that we'll use a map with critical project factors to help us figure out what setup to use for this project. We figured out that we should go more with agile development rather than plan-driven. From plan-driven development, we have kept only a domain model and relational model because we found it useful for our purpose. What agile development allowed us was freedom and more improvisation. It sometimes led to good ideas being born along the way instead of sticking strictly to the plan and just proved to be more adaptive. Also, we used some practices from XP like pair programming, collective ownership, and simple design. These things combined led to optimum planning and working on our application and as a team, we were really satisfied with how it turned out. Another big part of the whole project was sprints. Even though it was a bit hard to grasp the formality of the sprints, in the beginning, it was of great use later. Sprints enabled us to know where we are with the project and how far do we need to go to reach the goal ultimately boosting our efficiency. We firmly believe that with our application we achieved something unique for the market. What the application offers is the platform on which the user can get informed and make a difference without any distractions such as social media content or any redundant content in general.

Reference list

1) "What is Agile Methodology? Tools, Best Practices ... - Stackify." 17 Sep. 2017, https://stackify.com/agile-methodology/.

2) "Plan-driven software development - Wikiversity." 1 May. 2019, https://en.wikiversity.org/wiki/Plan-driven_software_development.

3) "Agile Planning: Step-by-Step Guide | monday.com Blog." https://monday.com/blog/agile-planning/.

4) "What is FURPS+? – Business Analyst Training in Hyderabad ...." 5 Aug. 2014, https://businessanalysttraininghyderabad.wordpress.com/2014/08/05/what-is-furps/.

5) "The Scrum Product Backlog - International Scrum Institute." https://www.scrum-institute.org/The_Scrum_Product_Backlog.php.

6) "12 Principles of Agile Methodology | CGI.com." 18 Jul. 2017, https://www.cgi.com/us/en-us/life-sciences/blog/12-principles-of-agile-methodologies.