



*UCN, University College of Northern Denmark
IT-Programme
AP Degree in Computer Science
dmai0919*

3rd semester project - System development

Alexandru Stefan Krausz, Martin Benda, Sebastian Labuda,
Simeon Plamenov Kolev
21-12-2020



Title page

UCN, University College of Northern Denmark

IT-programme

AP Degree in Computer Science

Class: dmai0919

Participants:

Alexandru Stefan Krausz

Martin Benda

Sebastian Labuda

Simeon Plamenov Kolev

Supervisor: Mogens Holm Iversen

Abstract/Resume

For this semester project we have decided to take on the task of creating a system where ferry companies can advertise and sell tickets for their trips. As such we researched and didn't find a lot of good solutions available on the market. The software needed to be easily accessible to the wide public as well as fast and secure.

So we created a webapp where anyone can create an account, search and buy tickets depending on their needs as well as for companies to create a business account and post their own trips and tickets up for sale.

Normal pages/characters: 40 425 characters / 16.84 pages

Contents

Introduction.....	4
Problem area	4
Problem statement	4
Method and theory	4
Idea generation and System vision	4
Personas, prototyping and mock ups.....	5
Personas	5
Prototyping.....	6
Exploratory prototyping	6
Experimental prototyping	7
Risk management.....	7
Scrum and Extreme Programming	8
Sprint 0	10
Architecture.....	11
Sprint 1	12
Burn down chart.....	14
Sprint retrospective.....	15
Sprint 2	15
Burndown chart.....	16
Sprint retrospective.....	17
Sprint 3	18
Burndown chart.....	18
Sprint retrospective.....	19
Sprint 4	20
Burndown chart.....	20
Sprint retrospective.....	21

Project retrospective.....	21
UP vs Scrum.....	22
Quality Assurance.....	23
Groupwork	24
Conclusion	24
Literature.....	25
Appendices	25
Appendix 1 Personas.....	25

Introduction

Problem area

In this semester we took a look at the ferry industry and realised there aren't too many software solutions out there for businesses to advertise and sell tickets for rides. This means there is a market gap between provider and buyer so we developed this system as a way to fill this gap and bring buyers and providers closer in a safe environment.

The system needed to be easy to use and secure, but above all else it needed flexibility in order to allow for ferry companies to sell different tickets for different trips and needs.

Problem statement

For the third semester project we decided to create a CMS software that will allow service providers to sell their products namely tickets for predetermined ferries. We feel that there aren't many great software currently on the market that are widely available for use by the general public as well as companies to reach out to as many interested "customers".

We want to harden our skills and knowledge in creating a CMS software, but this time C# will be used, in addition to a web application, which will serve as the client for the customers.

What will the developing process look like? Will we calculate developing time and be prepared correctly?

How will we solve concurrency problems, like multiple users trying to buy the same ticket?

How will we develop a web application using REST?

How will we prepare the tests, so we can ensure as least bugs as possible?

How will we address security issues, like SQL injection attacks, sensitive data exposure etc.?

Method and theory

To answer these questions, we made research about our options, different development techniques and technologies that are accessible to us.

This report will in detail describe the answer for the first question. It will show which processes have we used and how well have we implemented them. It describes the hardships we had, changes we had to make and our opinions on the process we have used.

Idea generation and System vision

Our first task in this project was to come up with an idea for a new system. After that, we created system vision, which is used to keep the project on track, reminding what is the point of this system.

Our product is going to be used by the users who want to buy tickets for a ferry and by companies in order to advertise their ferries. In order to ensure the success, the product needs to be user-friendly, secure, scalable, reliable and robust. There are companies implementing our idea, but the

point of our product is to have better ease of use, so any person can use it to buy a ticket. Sources of revenue can be either as monthly subscription from business account or percentage of each ticket bought on our website. This product needs to be developed in the timeframe from 27.10. until 21.12. by group 4.

Personas, prototyping and mock ups

Personas

A persona is an archetype describing a target user for a product or service. Persona serves as a description of target user's background, goals, and needs that are related towards the product. This also helps us to better understand your user target users, so you can design a better solution to meet their specific expectations. [1]

The main reason behind presenting each user data separately (personas) is that it is easier to empathize with a description of a specific person, rather than a set of statistics summarizing an entire group. Persona may seem as a presentation of a specific person while the reality is that it actually represents an entire group of users with similar characteristics.

For our project we have decided to create 4 personas in total. Each persona was chosen very carefully, each of them from a different user point of view as well as different categories. Identifying types of people and understanding their needs and how they are going to use our product was a very important process for us to gain better knowledge of what our system would require and to create much better general design as well.

Each persona has specific format, mostly consisting of basic information such as: profile (photo, name, type of user) and their overall background)

On the other hand, persona also includes very important information such as: their motivations, frustration and personal needs



Prototyping

After we had finished the process of preparing personas for our system, we had to move to design part of our project. Prototyping is considered as an experimental process, during which the team implements ideas from tangible form (in our case personas) into paper or rather in our case digital form.

As goes for our project, we had in consideration two specific types of prototyping:

Exploratory prototyping

Exploratory prototyping is a systems development method used for developing a computer system, which consists of planning and trying different designs until one of them is suitable enough to continue the development process with. It is important to mention that this way of model development is usually used in situations, when we have very little, or even none of the system requirements known (in detail).

On the other hand, from gathered information and documentations, this way of system development may be cost efficient or may even not deliver less-than optimal systems. One of many mistakes during processes proved to be: lack of finances, lack of courage, lack of experience, incorrect mindset and many others (situational).

Experimental prototyping

Experimental prototype, also known as prototype zero is a process, when the design team experiments with various items, new programs and gains general understanding of what they are about to work with. The main objective of the entire experimental programming is to understand the quality and overall usage of the various programs we are about to work with, to clearly understand what is achievable and if the actual program suits our cause. By doing so, you manage to reduce uncertainty with regard to getting the correct result.

It was exactly experimental prototyping we have decided to use. At the very beginning of this project, we have decided to work with Azure DevOps. For our group, it immediately meant a new environment in which we needed to learn to work. This way we have managed to address possible technical risks of the project as well as to eliminate the majority of the unknowns of our future product.

Risk management

One of the key parts of each successful project is risk management. Its point is to find out all the risks that could potentially negatively affect the project. Naturally, it is impossible to know all the risks before the project starts, but discovering as much as possible and at least the most possible ones results in better preparation and resolution of such problems. Being aware of all the risks guarantees better planning which increases chances of success.

Before our project we discovered multiple risks. Considering current COVID-19 times and worse human immunity during winters, we thought that a member becoming sick has the highest chance and will have the greatest impact on the project. One of our solutions for this risk was using pair programming.

Since we are not experienced developers and we were working with technologies and system development processes we are not familiar with, we realised that we will encounter obstacles in this project. That is why we recognised this risk as one of the most probable ones. We tried to be prepared, either created small spikes or studied a little bit before implementing new features, but we knew that there would be multiple obstacles we will be unprepared for.

Because of this inexperienced and unknown technologies we were not completely sure about the overall architecture of the system and if we understood it properly. That's why we knew that there is a high chance of changes in requirements and took it as one of the most important risks.

ID	Date	Risk	Probability	Effects	Ranking	Mitigatable Y/N
3	28/09/2020	Team member getting ill	8	5	40	Y
7	28/09/2020	Inexperienced team members	8	4	32	Y
5	28/09/2020	Change of requirements	8	3	24	Y
4	28/09/2020	Inadequate technology - problems with version control	6	3	18	Y
2	28/09/2020	Personnel moving places	3	4	12	N
8	28/09/2020	Team member unavailable due to personal reasons	2	4	8	Y
6	28/09/2020	Data loss due to hacking	1	7	7	Y
1	28/09/2020	Member gets hit by a car	1	6	6	N
10	28/09/2020	Data loss due to server down	1	5	5	Y
11	28/09/2020	Holdup because of the continuous integration	3	1	3	Y
9	28/09/2020	Data loss due to mechanical failure	2	1	2	Y

Although the risks should be analyzed before the project starts, they should be also maintained and updated during the project because they and their probability will change. Some risks will happen and will become a fact, others can have greater impact than initially predicted. Not only being prepared but also being aware of changes ensures a successful project.

Scrum and Extreme Programming

Extreme Programming (XP) is an agile software development methodology that aims to improve the quality of software and life of the development team. It is based on the methodologies that work well and their usage to the extreme.

In our project, we used the version 1.0 of Extreme Programming, which consists of 4 values and 12 principles. The values are communication, simplicity, feedback and courage. **Communication** means that everyone should be part of the team and the team should communicate daily face to face, **simplicity** that the code and design should be as simple as possible, **feedback** that it is important to get feedback on all levels from various sources and **courage** that developers should not be afraid to experiment and reject things, to have enough courage to make changes and decisions. The 12 principles are planning game, small releases, metaphor, simple design, define tests first, refactoring, pair programming, collective ownership, continuous integration, 40-hour week, on-site customer and coding standards.

Scrum is an agile framework used for developing software. It uses short periods with predefined length and defined requirements called sprints, at the end of which functional piece of code is created.

Scrum consists of 3 roles, 4 ceremonies and 3 artifacts. The roles are **product owner**, who communicates directly with the customer or it is a customer themselves, **scrum master**, who is responsible for ensuring that Scrum process works and for removing hindrances and a **team**, which is self organizing. Ceremonies are **sprint planning**, where the next sprint is planned, **daily scrum meeting**, where team describes what have they done a day before and what they plan to do in that day, **sprint**

review, where the team demonstrates what have they accomplished in the sprint and **sprint retrospective**, where team discusses what went well in the finished sprint, what went wrong and what should be improved. First artefact is **product backlog**, which is a list consisting of all tasks that needs to be made in the project. The product backlog is maintained and prioritised by the product owner. Next one is the **sprint backlog**, which is a list of all tasks that need to be performed in a specific sprint. Last artifact is a **burndown chart** or **burnup chart**, which is a graph illustrating progress.

In our project, we have used the combination of Extreme Programming and Scrum. Since Scrum describes the project management but says nothing about how the software should be developed and XP focuses mainly on the development of the software, it is easy to combine them in this way. We decided to use an agile development process because it fits our group better (compare Fig.1). We also wanted to learn it more and understand it in practice. We could use Kanban instead, but we thought it's too light on artifacts and principles, so we decided it will be safer to stay with Scrum and XP.

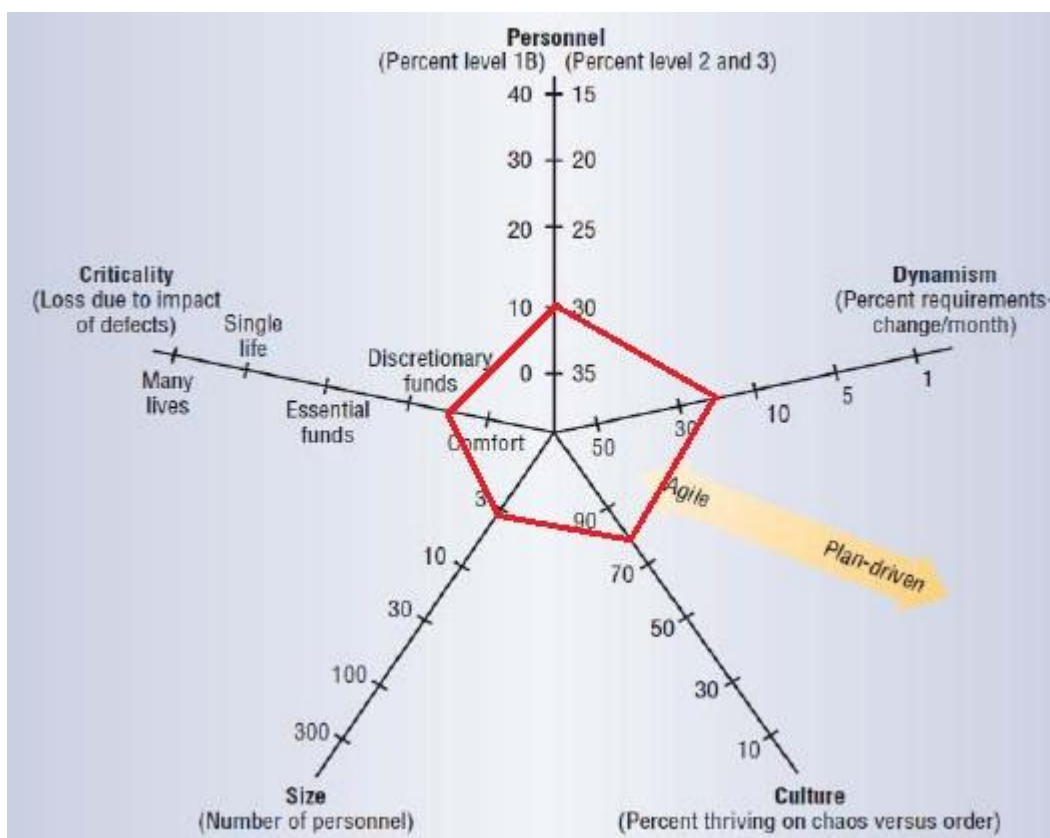


Fig. 1,
Boehm's
Critical
factors [2]

When it comes to development of the software, we used all 4 values and 12 principles with slight changes. Because of the current covid era, we couldn't physically meet together on a daily basis, so communication has suffered a little bit. It created few conflicts and misunderstandings, but we did our best to ensure as good communication as it was possible. We were working with one week sprints which defined small releases and at the end of each week, we presented our results and got feedback. The last change we made in our use of Extreme Programming is in pair programming. Even though we used pair programming most of the time, when we were working on simple tasks with which we already had experience, we decided to omit it and use our time more effectively.

The only thing we changed in Scrum and our project management is not having one person as product owner and one person as scrum master for the entire project, but rather change them at

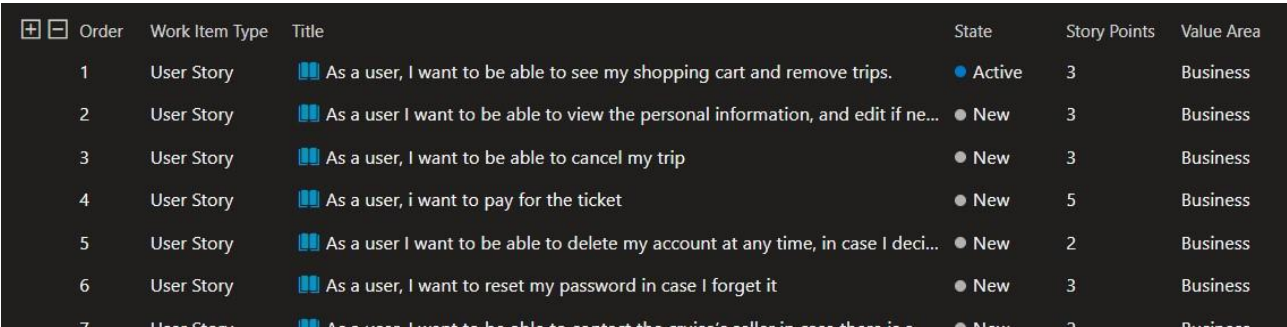
the end of each sprint. By doing this we ensured that every member will be scrum master and product owner during the project. This way, it was easier for us to try and learn their responsibilities.

Sprint 0

Sprint 0 is used for initial preparations of the project. In this sprint we first created a list of the user stories. User story is a small, simple description of a feature that needs to be implemented and its acceptance criteria. We used the INVEST criteria while creating them. This means the user story should be independent, negotiable, valuable, estimable, small and testable. After coming up with all the user stories we could create with current knowledge, we divided them into four based on the MoSCoW principle, must have, should have, could have and won't have.

Having the User stories, we had to predict how long it should take for us to implement them. To estimate it we used story points and planning poker. A story point is a metric used in agile project management and development to estimate the difficulty of implementing a given user story, which is an abstract measure of effort required to implement it. In simple terms, a story point is a number that tells the team about the difficulty level of the story. Difficulty could be related to complexities, risks, and efforts involved. [3] Planning poker is a technique used for assigning story points to user stories. We assigned story points to user stories based on the fibonacci sequence, discussed highest and lowest numbers assigned and after assigning everything we also compared every user story to one 5 story points user story. This comparison is used in order to check if the story points assigned make sense, meaning if another user story is easier to implement it should have less story points and vice versa.

With finished user stories and estimations, the product owner had to prioritize the product backlog. We used Azure DevOps for better visualisation and easier manipulation of user stories and sprints (Fig. 2). Azure DevOps is Microsoft service providing collaboration and development tools. It is clear and provides a lot of features for agile methodologies, it was integral for us to use a tool like this especially because of the current COVID era.



Order	Work Item Type	Title	State	Story Points	Value Area
1	User Story	As a user, I want to be able to see my shopping cart and remove trips.	Active	3	Business
2	User Story	As a user I want to be able to view the personal information, and edit if ne...	New	3	Business
3	User Story	As a user, I want to be able to cancel my trip	New	3	Business
4	User Story	As a user, i want to pay for the ticket	New	5	Business
5	User Story	As a user I want to be able to delete my account at any time, in case I deci...	New	2	Business
6	User Story	As a user, I want to reset my password in case I forget it	New	3	Business
7	User Story	As a user, I want to be able to contact the cruise's seller in case there is a ...	New	2	Business

Fig. 2

Knowing what we would have to implement we already had some ideas where we lacked information or what we were unsure of. For that we created a list of spikes (Fig. 3). Spike is a development method used to gather information about parts unknown for the team. It is time

devoted to new technology and trying it out, in order to know how to use it during the development. Most of the things created during spikes will end up discarded.

ID	Description	Date(finished)
1	Research GIT	28.10.
2	Azure DevOps	30.10.
3	Research front-end	7.11.
4	Check on how to automate tests (and code coverage)	03.11/04.11
5	How can web application autorefresh	18.11.
6	How can we store dates and time(hours/minutes) in db	19.11.

Fig. 3

First spike involves researching version control called git. Until now, all of our members were using only subversion and none of us had any experience with git. We knew that even after researching and trying it out, we still won't know everything and will come to some obstacles, that's why we also included it in our risk management as one of the most important risks.

This was also our first time using Azure DevOps. Since it has a lot of features, thus a lot of ways to get lost, we wanted to spend some time on it, just to ensure we will have everything in the proper place and won't lose any data. One of its features are also automated tests and continuous integration via pipelines. Since this is a bit more complex task we picked it out as a separate task.

Architecture

Finding proper architecture for the project is one of the most important parts. Since each project has different functional and especially non-functional requirements, architecture has to be chosen accordingly. There are many architectural considerations that need to be made like where will our data be stored, where will be parts of the code stored, how to ensure security etc. In our project, architecture was more or less set, so we couldn't make a lot of changes into it.

Our architecture consists of four parts: database, RESTful web service, dedicated client and web application. We are using N-layered architecture, which separates code into different layers depending on its purpose. Our database is accessed by DatabaseAccess classes. Then we are using a repository pattern for dependency injection and easier maintainability. All our business logic is stored in the business layer, which is called by API controllers. API controllers are the only part of the web service which is exposed to the clients. To ensure separation of concerns and high cohesion in both clients we are using Api Helpers, which take care of all calls to the API. For dedicated client, we are using WPF (Windows presentation foundation) windows. For web application we are using MVC (Model-View-Controller), where controllers communicate with Api

Helpers and Razor Views, and those generate browser based client. All the layers have access to their model classes(see Fig. 4).

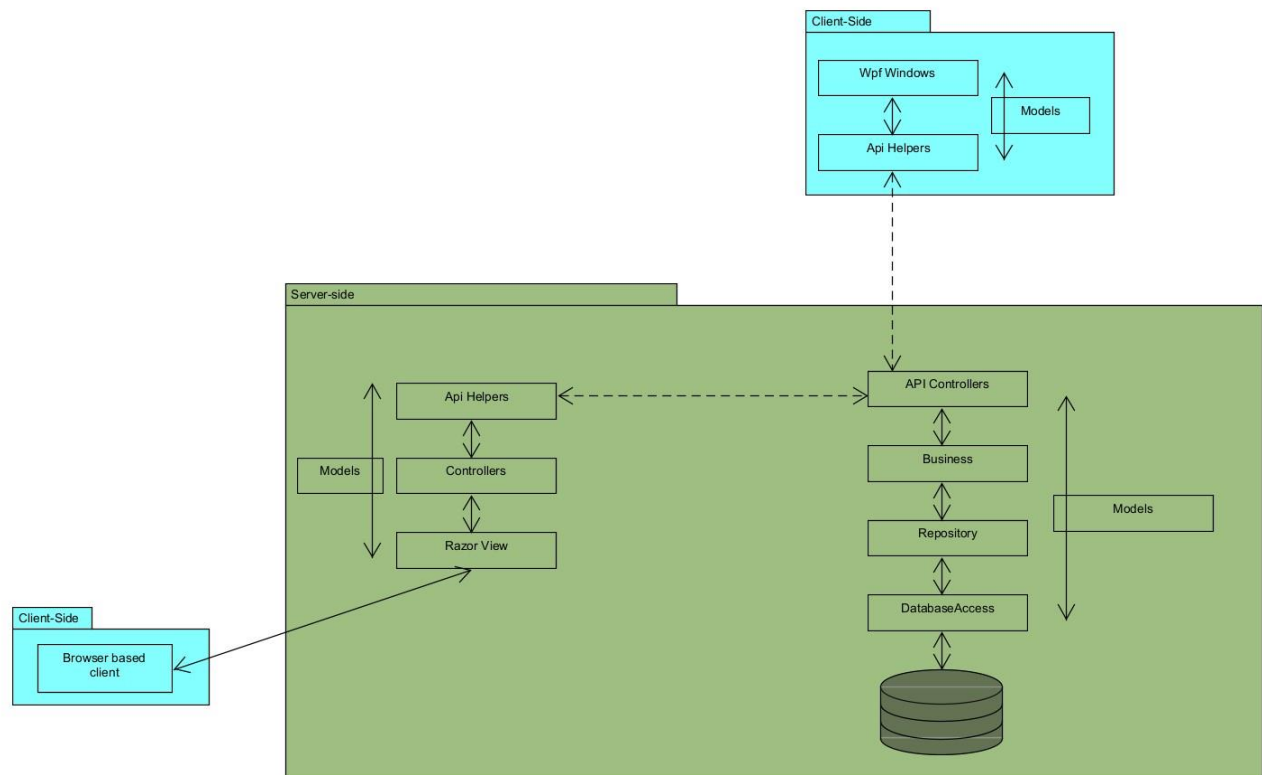


Fig. 4

Sprint 1

First step of each sprint is its planning. In order to know how much a team is capable of implementing in the current sprint is to calculate velocity of the team. Velocity in Scrum is a measuring unit showing how many user stories can a person implement in one man-hour. It is calculated by taking 5 points user story and estimating how long it would take to implement that. Then, dividing 5 story points by estimated time equals the expected velocity for the sprint.

Since it was our first sprint and we were working with many unknowns, we decided to be more careful and estimated our velocity to 0,2. Knowing that and the amount of hours we are able to work in the first sprint, we took 3 top priority user stories into the sprint backlog (Fig. 5). Since part of our work was also to write this report, we added it as a user story as well. Having the stories for the current sprint, we divided user stories into separate tasks (Fig.6), which concluded sprint planning.

	Order	Title	Remaining Work	Story Points	State
+	1	> As a seller, I want to be able to add a new trip.	...	5	New
	2	> As a seller, I want to be able to add a new ticket.		5	New
	3	> As a user I want to be able to create a new personal/profesi...		5	New
	4	Write Report		1	New

Fig. 5

Collapse all

28 As a seller, I want to be able to add a new trip.

SK Simeon Plamenov ...

State New

New

52 Create Tests

SK Simeon Plamen...

State New

43 Create DataBase Table

SL Sebastián Labuda

State New

44 Create DataAccess Class

SK Simeon Plamen...

State New

46 Generate Model Class

SL Sebastián Labuda

State New

45 Create Repository Class

SK Simeon Plamen...

State New

Fig. 6

Beginning of this sprint went really well. We were following all the principles and methods from Scrum and XP. Even though making tests first was a little bit harder at the beginning, it didn't slow us as we managed to learn how to use it properly. After daily scrum meetings, we often discussed how we implemented certain parts to increase communication in our team and consistency in the

code. But short after beginning we encountered many of the risks identified in our risk management. We had multiple problems with the version control and because of our inexperience with it we lost one day only on these problems. Front end of the system also showed us that we were not experienced with it and we encountered many problems we have not encountered before. Luckily we were able to manage to develop the back end faster then we predicted, so we were able to use that time to catch up with that risk. Last risk we encountered was a team member getting sick. We were able to mitigate it, thanks to the XP practice pair programming, but that also meant that one team member cannot use this practice anymore.

Based on these facts we updated our risk analysis table. In normal cases these risks become facts and are erased from the table, but considering their nature (illness, inexperience etc) we knew they can occur again, so we just updated their chances and impact.

Burn down chart

Figure 7 shows a burn down chart, which we used in the first sprint and which is provided by Azure DevOps. Later we realised, it is not a proper burn down chart (compare to figure 8), but we decided to stick with it until the end of the sprint. There are multiple things we can burn down on when using a burn down chart. We can burn down on story points and finished user stories, finished tasks or hours of work assigned to a task. To track the progress of the project, different burn downs of different sizes for sprints and project as well as combination of using burn down and burn up. Since we only had 4 user stories, we decided not to burn down on story points but rather on finished tasks. Since we had 30 tasks in comparison to 4 user stories, it gave a better picture on our progress.

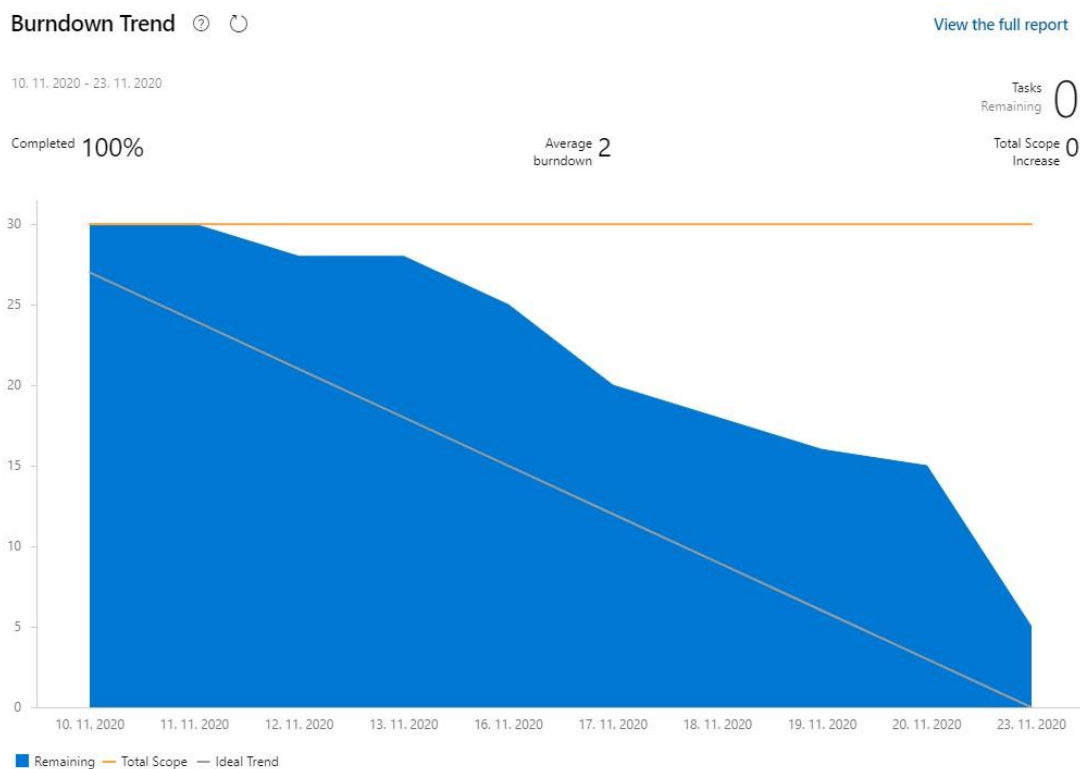
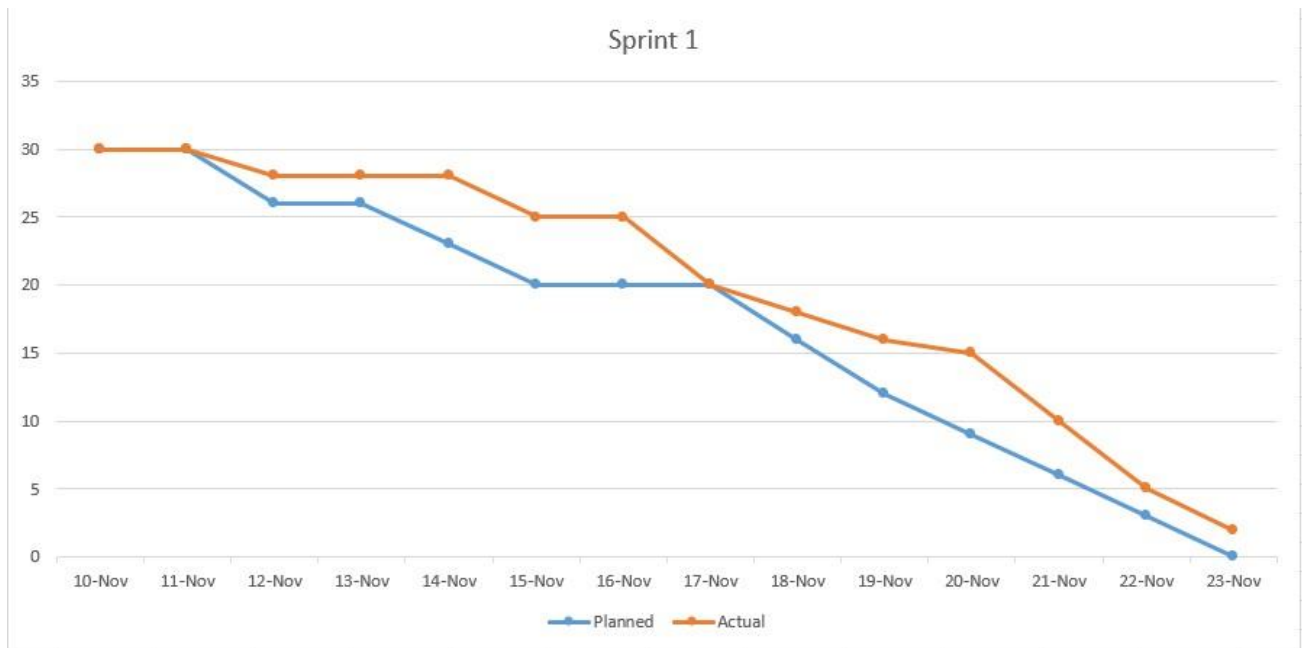


Fig. 7

**Fig. 8**

In this sprint there were days which were still dedicated to learning rather than working on the project, that is where the planning goes straight. Because of our jobs and extracurricular activities, we can't work 8 hours each day on the project so we decided to work during the weekends as well and include them in every burn down. Since we were working for the first time with our architecture and some technologies, it didn't go as well as expected. One of our members also became sick after a few days, which slowed down our progress a little bit. Luckily thanks to the pair programming it wasn't a big problem and it didn't affect us that much.

Sprint retrospective

During our sprint retrospective, we agreed that our biggest problem for sprint one was our lack of experience with technologies we used. Although we have done well considering the amount of risks we encountered, we thought we could have done better if we were more prepared.














But our communication was very good despite our only possibility communicating online. Although we have encountered many risks with high impact, we still managed to implement nearly everything we have decided for the current sprint, so our mitigations worked well. Also pair programming helped us a lot to learn and find many errors and small mistakes.

When it comes to what we wanted to improve for the next sprint, it is a better handling of burndown charts, to have accurate data we can work with and being more enthusiastic at the beginning of the sprint.

Sprint 2

Since we were already a bit more experienced with the overall structure of our project, we knew we would be capable of more for this sprint. Considering the amount of time we had and the fact that one of us is still sick, we calculated that our velocity for this sprint should be 0.3 and took tasks

for this sprint accordingly.

 	Order	Title	Remaining Work	Story Points	State
	1	>  As a seller, I want to be able to update my existing trips.		3	● New
	2	>  As a seller, I want to be able to update existing tickets.		3	● New
	3	>  As a user, I want to be able to buy a ticket for a ride.		5	● New
	4	>  As a seller, I want to be able to delete my trip.		2	● New
	5	>  As a seller, I want to be able to delete my ticket.		3	● New
	6	>  As a user, I want to see all available cruises and their tickets		3	● New
	7	>  As a user, I want to buy multiple tickets for multiple rides.		1	● New
	8	>  As a user, I want to be able to see my shopping cart and re...		3	● New
	9	>  Write Report		2	● New
	10	>  Leftovers from Sprint 1	...	3	● New

In total it was 28 story points including writing the report and things we did not finish in the last sprint. In this sprint we wanted to add manipulation of tickets and trips and tie them together. We also wanted to make buying tickets functioning with a simple structure and shopping cart.

Burndown chart

This time we had more user stories, so burning down on them would be possible, but burning down on tasks worked for us very well so we decided to stick with it. We also managed to split the tasks into relatively the same size. By now we already knew its advantages as well as disadvantages. The progress is more visible then burning on user stories, so sudden drops are not encountered. On the other side user stories have story points with the same estimated time, but tasks can be of different size. Finish tasks also don't show the finished user story. People also have a tendency to start working on multiple parts at the same time so it can happen at the end of the sprint that everything was started but nothing finished, although burndown went well at the beginning.

**Fig.9**

As figure 9 shows, we did not manage to implement everything we planned in this sprint. We encountered various obstacles which resulted in many slowdowns. One of the biggest one was the work at the front end, for which we still lacked information and experience. We also underestimated the user story "Write Report". Although we calculated it as 2 story points, it ended up as 5 story points considering the amount of time needed to finish it.

This sprint also helped us to learn a lot about Scrum, XP and technologies we were using. We were getting more used to the XP principles, especially test driven development, working with burndown charts and weekly sprints. Some of the slowdowns we encountered actually helped us in the long run, for example time spent on UI or report.

Sprint retrospective

Although we have accomplished better handling of burndown charts from last sprint, the enthusiasm at the beginning was still lacking a little bit, but it owed to the fact that many things we experimented with at the beginning did not work as we expected.

Still, our communication went very well and we were helping each other whenever the need arose. Although we did not completely follow the pair programming principle since there were three of us, for this time it was worth it to omit it in some cases.

For the next sprint we agreed that we needed to find a better way of handling UI and passing data within, for which we wanted to experiment at the beginning of next sprint. Again, to be able to track our progress better and have more time we wanted to improve our motivation and enthusiasm at the beginning of the sprint.

Sprint 3

With our sick member getting better, we felt we could get back on track and by understanding the architecture a little bit more, we thought we could get more done. But we didn't want to overestimate ourselves, so we still counted our velocity as 0,3 and took top priority tasks accordingly.

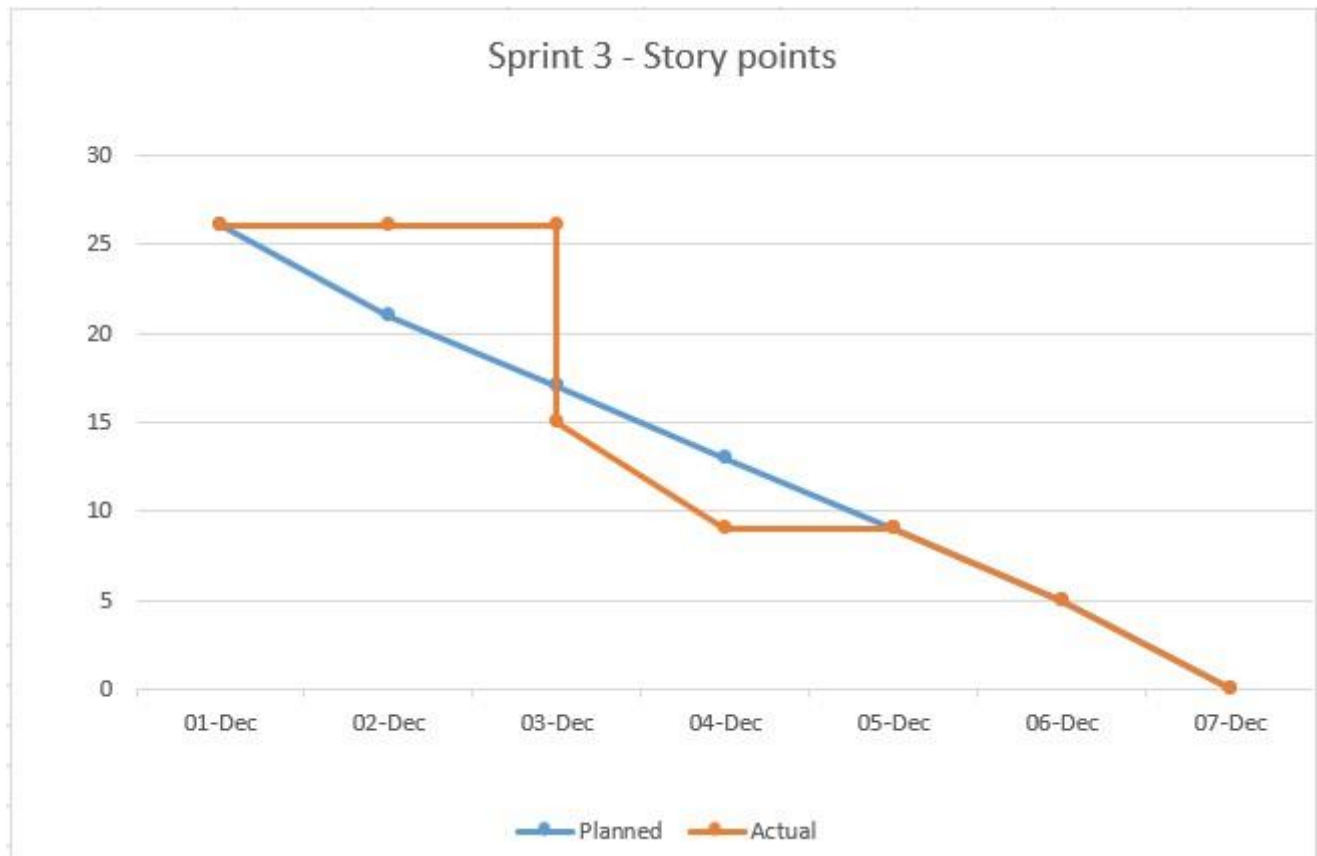
	Order	Title	Remaining Work	Story Point
	1	> As a user, I want to pay for the ticket		3
	2	> As a user, I want to buy multiple tickets for multiple trips.		1
	3	> As a user, I want to be able to filter out or search for specific trips.		5
	4	> As a user I want to be able to view the personal information, and edit if necessary.		3
	5	As a seller, I want to be able to see a list of my tickets.		3
	6	As a seller, I want to be able to see a list of my trips.		2
	7	As a user, I want to be able to cancel my trip.		3
	8	As a user, I want to be able to see my shopping cart and remove trips.		3
	9	> Write report	...	3

With 26 story points including writing a report, in case we would be faster we wanted to take more stories into the sprint. In this sprint we wanted to allow users to be able to search for specific trips by multiple options, buy tickets for trips and pay for them, see and edit personal information, cancel trips and for business users to see their trips/tickets.

After the sprint review from sprint 2 we realised that our buying tickets wasn't completely suitable for the real world and needed more functionality in it, so we needed to remake it in this sprint.

Burndown chart

Since we couldn't create tasks with the same size anymore, we decided to switch for burning down on story points for this sprint and try how this method would work for us.

**Fig. 10**

As can be seen on figure 10, things weren't going according to the plan. We thought that there would be 4 healthy developers working in this sprint, but actually 3 of us ended up sick. At the beginning we thought we would get better soon but on the third day we realised it will take longer, so we had to drop multiple user stories in order to make it. We dropped viewing and editing personal information, canceling trips, removing trips from shopping cart and two-thirds of writing report, counting totally for 11 story points. In the end it was a good call, because otherwise we wouldn't be able to burn down everything.

Sprint retrospective

This sprint naturally couldn't go well with so many people getting sick. Because of that, our enthusiasm also dropped severely. Even though we thought that we already had enough experience with the front end, we still encountered many places where we needed to research for a solution. But we managed to find a better way of passing data in the UI.

But otherwise our work went well and we used pair programming whenever we could. We took proper actions when we encountered the risks and managed to accomplish the plan.

Because of lack of enthusiasm we wanted to increase motivation and positive spirit for the next sprint. We also wanted to increase our knowledge of UI and working with it.

Sprint 4

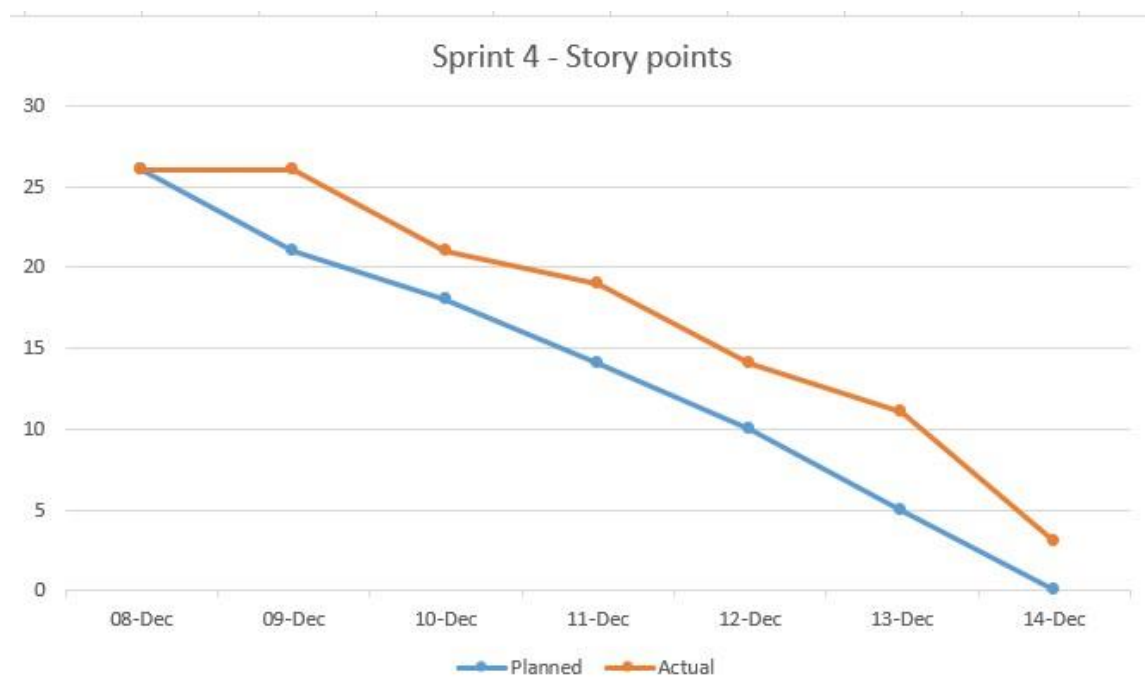
With a lot to catch up, we had to prioritize what will be done in the last sprint and which user stories will not make it to the finished system. By this time we figured out that estimated velocity 0,3 works for our group so we decided to stick with it.

Order	Title	Remaining Work	Story Point
1	> As a user, I want to be able to see my shopping cart and remove trips.		3
2	> As a user, I want to pay for the ticket		5
3	> As a seller, I want to be able to see a list of my tickets.		3
4	> As a seller, I want to be able to see a list of my trips.		2
5	> As a user, I want to be able to cancel my trip		3
6	> Refactor code		2
7	> Beautify UI		5
8	> Write report	...	3

With 26 story points for this sprint, we decided to focus on making the UI more user friendly, refactor the code and allow users to cancel trips and remove trips from the shopping cart. We wanted to add a payment gateway for buying tickets as well. We realised that seeing tickets and trips which created a business account is not working as well as it could, so we needed to remake it.

Burndown chart

As in the last sprint, the tasks we created were not of the similar size and even though we could add time of work to them and burn down on that, we went for burning down on user stories since last sprint didn't go well and we wanted to test if it works for us or not.



In the last sprint we did not manage to finish everything that we planned. Although we were close at the end we haven't finished the user story Write report. One of the reasons is also the fact that in the last days we discussed with the product owner that we have more time for writing the report after the sprint is finished but there are some parts we could improve and few functionalities which would make using our program much easier. This way we violated the contract, but we did it by discussing it with the product owner and if we would have a customer, we would discuss directly with them as well.

Overall, use of burndown charts helped us track our progress. We have tried burning down on tasks at the first two sprints and burning down on finished user stories during the last two sprints. We think that burning down on tasks works for us better, because it gives us extra motivation when we see already at the beginning that we burn something, which helps work more. We will probably use it in the future as well, but we will assign hours of work to the tasks, so we don't need to worry about tasks being of different size.

Sprint retrospective

In this sprint, the only thing that went wrong was UI once again. Although we tried to research it and learn about it a little bit more, we still lacked knowledge and experience.

On the other hand our communication was very good and pair programming helped us quite often. In some cases we switched for individual work but we were helping each other anyway. Communication went very well and this was actually the first sprint where basically no risks were encountered.

Because of a lot of trying, our code wasn't as organised as we wanted, so we needed to work a little bit on that. We also want to improve our knowledge in front end development for future projects.

Project retrospective

Overall the project deeply increased our knowledge especially in system development and use of scrum and XP. Thanks to this project we better understand Scrum, its advantages and disadvantages. The same goes for the Extreme Programming and why it is often used combined.

From Scrum we have used everything for the entire project period. We had all three roles with scrum master and product owner changed for different group members every week. This way each one of us was product owner and scrum master for one week and could better understand their position and duties. We had all four ceremonies and used all 3 artifacts. We had no problem using them and it worked for us very well as a project management tool, probably best for our group so far.

We tried to use as much as possible from XP but sometimes we either couldn't or it didn't work. All the values were in place. Our communication was good (as far as current COVID era allows), we were implementing code as simple as possible and improved it when necessary, thanks to the scrum and tests we were getting feedback on all levels and we were also experimenting a lot especially at the front end. What we couldn't always use from practises is define tests first and pair

programming. Although we tried to write tests first and then program, we had hard time or sometimes even couldn't do it in the front end and we rather live tested the program. But otherwise this practice worked for us, although it was hard at the beginning. When it comes to pair programming, we didn't use it only when somebody was sick or if we were doing something we already knew and it was necessary to catch up with our plan.

During this project we encountered too many risks and work at the front end delayed us drastically. We also had slower starts in each sprint, which can be seen on burn down charts.

But pair programming worked for us much better than we even expected. Communication between team members was really good, so we had no problems there. Even though we encountered many risks, our mitigation of them worked very well and we managed to finish most of what we planned.

One of the solutions for our slow starts are longer sprints. One week is too short if the developer wants to implement a feature and present it to the customer, so at least two week sprints or longer would be perfect for us. In the future projects we also have to research the domain a little bit more, so we don't end up changing already implemented things.

UP vs Scrum

In the last semester project, we have been working with Unified Process (UP). Unified Process is use-case driven. It uses the Use Case Model to define functionalities of the system, divides use cases depending on their priority and starts with implementing the highest one, to tackle the most risky one at the beginning. It also sets architecture of the system early in the development and uses iterations with defined requirements, design, implementation and tests. Unified Process divides development into 4 phases: inception, elaboration, construction and transition. The inception phase focuses on gathering the requirements and ensuring their understanding. Point of the elaboration phase is to implement use cases with highest priority, so it is clear if it would be possible to make the project or not. The longest phase is usually the construction phase, where the rest of the system is developed. Finally the transition phase focuses on releasing the product.

Both scrum and UP have some advantages, which can really help depending on the project. Scrum is very agile, meaning it constantly collaborates with customers and handles changes really well. It gets feedback especially during sprint review. UP on the other hand is very agile in the inception and elaboration phase, but strictly plan driven from the construction phase, thus changes in requirements after reaching the construction phase won't be handled very easily.

UP is also much more artifact heavy. No matter if we used classical UP or lightweight one, it still consists of much more artifacts than Scrum or any other agile method. In this case, agile methods rely on direct communication within the team and with customers. This can be problematic if the development team is too big and we don't often have access to the customer.

Plan driven development relies on the fact that it knows all the requirements and then it goes deep into their development, as for example in waterfall where everything is analysed, then designed, then coded and tested at the end. Agile methodologies on the other hand expect that they don't

know all the requirements and that requirements can be changed. In this case for example XP tells to implement as simple as possible so changes can be made at any time.

Quality Assurance

Software quality assurance consists of multiple activities ensuring the quality of the development process. It is focused on the development process and preventing creation of errors. Basically it consists of planning on how the system will be developed, what precautions will be taken and how the errors and defects will be found and acting according to found defects and errors.

Parts of quality assurance are covered by scrum itself. Talking with the product owner and getting feedback so often ensures that the quality of the product is high enough and the customer is satisfied.

Since we are approaching this project in a test-driven way, that ensures our API will be working errorlessly. We decided to go with integration tests for our simplest user stories - namely the Crud's. In case we encounter something with really complex logic, we will also make sure to create unit tests for it in addition to the integration ones. We try to constantly perform system tests to ensure the whole flow, from the Web Application till the Api, is working correctly. Every time something new was added, we also live tested the system and often live tested every single thing implemented by far to ensure that everything was working. We had to use this approach especially because we had problems rather with the front end than the backend. By that we also make sure that we meet the non-functional requirements, especially the usability. Creating and running these tests makes our program easily maintainable, extensible, scalable and also reliable.

Biggest problem are non-functional requirements, which are hard to measure. That is why we need to compare something, and if it fits into it we met the criteria. These are called metrics. Our most critical non-functional requirements for this project were usability, maintainability, robustness and scalability. Usability was ensured by sprint review and showing the developed system to the product owner to evaluate the result. We tried to use proper colours, buttons on expected places and different parts to look similar so the user can learn how to use our product faster.

Maintainability was achieved by adding proper comments and by architecture, which clearly divides specific parts into layers. We tried to make our product as robust as possible, so no matter what the user input and clicks the program will still work. Lastly, to see if the product is scalable or not we load the tested slowest part of our system, namely buying tickets. We spawned 100 users to buy tickets at exactly the same time, which is rare even during promotions. But it still took in average 0,574 seconds, which is more than acceptable (see Fig 11).

Label	# Samples	Average	Min	Max	Std. Dev.
HTTP Reque...	100	574	370	765	81.56
TOTAL	100	574	370	765	81.56

Fig. 11

Different way to see if the criteria are met is by performing reviews. Even though they are very precise, they are long and heavy on documentation, that's why it isn't used very often.

Groupwork

Our group has developed during the past few semesters. Teamwork as well as fluent work has become truism for our group. Understanding our stronger sides alongside weaker ones helped to correctly divide the work between all group members while reassuring the attaining general knowledge from different areas of work.

Regardless of all of these facts and previous knowledge, we have been once again challenged by the current state of Corona-19 pandemic situation all around the world. As anyone else, even our group in both terms of studying, along with working on our project has been heavily influenced. Our group had to face both already well known problems as well as to face completely new ones. Regarding the project, the biggest challenge proved to be the front-end this time. We have been working in a completely new environment and it understandably took some time to get used to. On the other hand, regardless of working with different programming languages, we could see advancement in personal back-end skills of every group member. The time spent on the back-end was rapidly lowered, while we had to put more focus into the front-end part.

On the other hand, we stepped towards the whole situation with maximal respect, motivation and understanding. Regardless of group members being sick during multiple sprints, partial loss of motivation or interest, each member put as much effort into this project as possible. Each member of the group worked the same way, including any part of our project.

To sum up, pandemic served as a great test. We had a chance to use all our previous as well as newly gained knowledge and studying materials we have learned both semester and even increase our knowledge.

Conclusion

Overall, we have met the requirements and have reached our goal. We implemented most of the user stories we prepared at the beginning. In this project we used and tried loads of things for the first time and even though we were working with many unknowns, we still managed to finish it and learn a lot.

Although we implemented a lot, our system is not completed. There are still some user stories and tweaks especially regarding usability that can be done. We still need to work a little bit more on task and user story estimation, but this will come with experience.

This project helped us to dig deep into new methods and technologies. We learned a lot how scrum and XP look in practice, we learned their advantages, disadvantages and especially if it fits us to work in this environment. We also improved our knowledge about quality assurance and measurement of non-functional requirements.

Literature

[1] docs.idew.org , [Online].

Available: <https://docs.idew.org/principles-and-practices/practices/design-practices/personas>

[Accessed: 28-Nov-2020].

[2] www.pmi.org , [Online].

Available: <https://www.pmi.org/learning/library/select-fitting-project-management-approach-6915>

[Accessed: 02-Nov-2020].

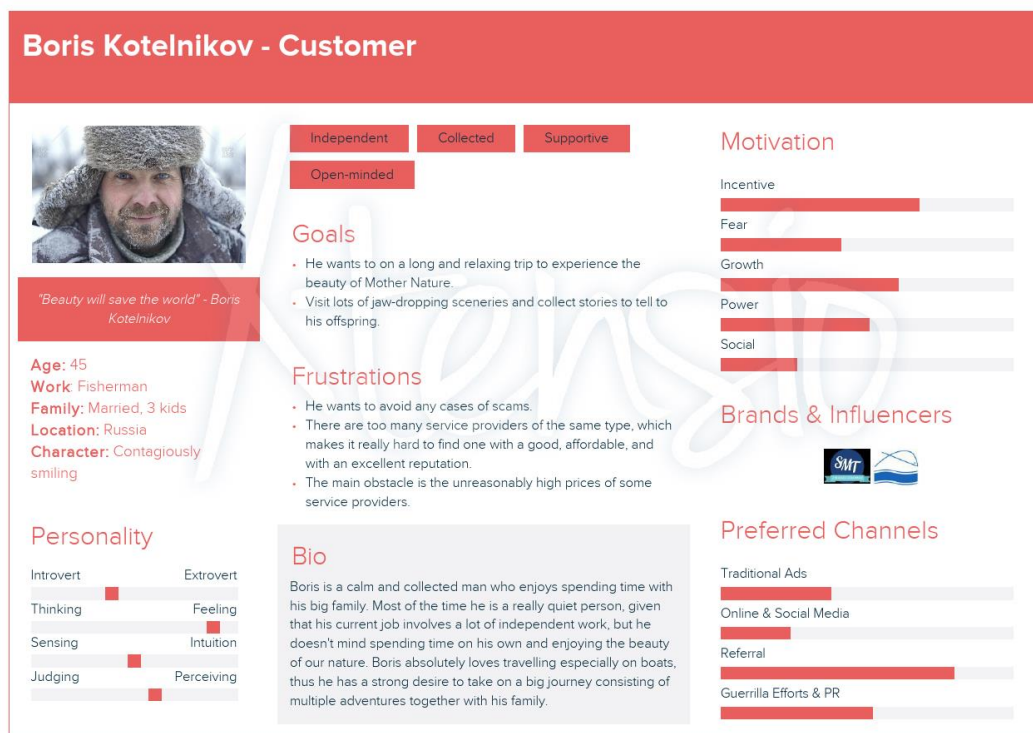
[3] visual-paradigm.com , [Online].

Available: <https://www.visual-paradigm.com/scrum/what-is-story-point-in-agile/>

[Accessed: 29-Nov-2020].

Appendices

Appendix 1 Personas



Jarmila Svobodová - Student



"To travel is to live." - Jarmila Svobodová

Age: 23
Work: Student at university
Family: Single
Location: Brno, Czech Republic
Character: Archetype

Curious **Adventurous** **Autonomous**
Sensible

Goals

- Would like to travel by ferry to see the world
- Searching for consistent and safe provider
- As a student, wants to find trips for affordable prices with basic quality

Frustrations

- Way too many fake or scam websites/providers
- Not oriented in this field, can't find cheaper solutions (as a student)

Bio

Jarmila has always been a very curious child. Exploring everything was here well known characteristic. Her big dream in her early childhood was to explore the world. Now, Jarmila is a student of the university in Brno. She has been saving money during her study and is slowly preparing for her first larger trip. Through university trips connected to her future profession, she already had a chance to explore central Europe. This time, she is aiming higher and that is to explore the cold and windy Northern Europe. However she would like to enrich her experience and that is to travel by ferry. Unfortunately, she is uncertain how to proceed and trying to find her way by finding reliable website to find all types of ferries.

Motivation

Incentive
Fear
Growth
Power
Social

Brands & Influencers

Instagram Twitter Snapchat


Preferred Channels

Traditional Ads
Online & Social Media
Referral
Guerrilla Efforts & PR

Personality

Introvert Extrovert
Thinking Feeling
Sensing Intuition
Judging Perceiving

Marvin Kirby - Administrator



"Java is to JavaScript what car is to Carpet." - Marvin Kirby

Age: 34
Work: Administrator
Family: Married, no kids
Location: Orlando, USA
Character: Grind

Intelligent **Introvert** **Workaholic**

Goals

- Ensure smooth running of existing system
- Satisfy needs of customers and user of the system

Frustrations

- In other websites, he does not have enough administrative power
- Wasted time or lack of progress

Bio

Marvin Kirby has been born as a genius. Studying meant everything to him during his student times. Bullying was his daily bread as others could not understand his talent. He excelled so much that he skipped several classes just to be studying with students of same tier.

He quickly found his talent and love for programming. The more he learned, the more he wanted to know. He quickly became irreplaceable and had the chance to work for world known companies.

Motivation

Incentive
Fear
Growth
Power
Social

Brands & Influencers

Google YouTube Instagram Facebook Twitter Apple

Preferred Channels

Traditional Ads
Online & Social Media
Referral
Guerrilla Efforts & PR

Personality

Introvert Extrovert
Thinking Feeling
Sensing Intuition
Judging Perceiving