Process Report

Airline Reservation System

Distributed and Heterogeneous System

Names of Students:

Jan Vasilcenko – 293098

Karrtiigehyen Veerappa - 293076

Nicolas Popal - 279190

Patrik Horny – 293112

Supervisors:

Jakob Knop Rasmussen

Ole Ildsgaard Hougaard

VIA University College

Bring ideas to life
VIA University College

Number of characters: 37131 including spaces

Software Technology Engineering

Semester 3

17 December 2020

Table of Contents

List of Figures

List of Tables

1 **Introduction** (Jan Vasilcenko, Karrtiigehyen and Patrik Horny)

We were introduced to the 3rd semester project on 2nd of September where we were given lots of information about how the process of making the project is going to look like. We discussed about the examples of distributed systems, along with skills and knowledge required to create a functional and well-structured project.

On 16th of September, we handed in our project proposal where we proposed to work on Hospital System, Food Tinder, and Airline Reservation System. Our supervisors gave us a feedback about our project proposal and after some discussion we decided that we will develop an airline reservation system. This seemed like the most reasonable option, since none of us had full grasp of a hospital system works, and the idea for the food tinder was half-baked.

During following weeks, we started laying a foundation for our project, where we defined the necessary functionalities and the capabilities, as well as the requirements. The Software Development of Distributed Systems (SDJ3) course provided us with knowledge on how to develop distributed systems. The Internet Technologies, C# and .Net (DNP1) course taught us how to work with C# and .Net, which we would have to use since the airline reservation system was a heterogenous system using C# and Java. The security aspects of the project were learned from the Networking and Security (NES1) course. Knowledge from the previous semesters such as SOLID principles, design patterns, and client/server systems were used extensively in this semester as well.

On the 11th of November, we had to hand in a Proof of Concept of our system, which greatly helped us to realize how to start to implement the distributed system using the three-tier architecture.

In the previous semester, SCRUM (Schwaber & Southerland, 2017) was taught to us, and this is the framework we used to structure our group collaboration. This is done to keep the group more engaged to work on the project. This framework made sure that we did not lose focus on the project and that our time-schedule will be used efficiently.

2 **Group Description** (Karrtiigehyen and Nicolas Popal)

Our group consists of four people:

Jan Vasilcenko – Czech Republic

Karrtiigehyen Veerappa – Malaysia

Nicolas Popal – Czech Republic

Patrik Horny – Slovakia

Four members of the group already worked together during the first and second semester projects. Our group consists of 2 Czechs, 1 Slovak and 1 Malaysian. Even though the Czech-Slovaks, who are very close due to their historical and cultural background make most part of the group, there we no problems in between the group as we get along very well with each other. Even though Czech and Slovak nationalities are quite close, this is not reflected in the Hofstede's Dimensions (as can be seen in the picture below).

Czech Republic, Malaysia, Slovakia

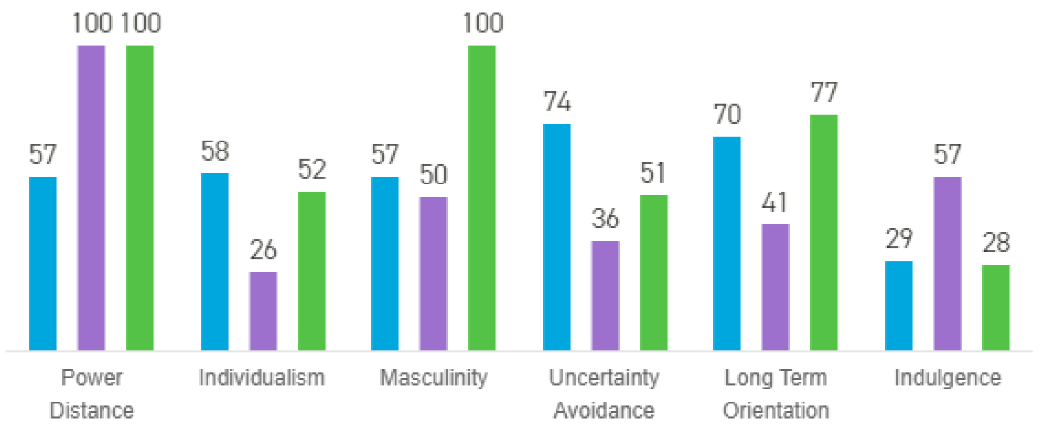


Figure 1: Hofstede's Dimensions

2.1 **Power Distance** (Karrtiigehyen and Nicolas Popal)

Power Distance measures inequalities between lower and higher ranked members of a certain hierarchy. (family elders vs. children, high management vs. employees) However, power distance is hard to measure inside of the project work as there are no big differences in hierarchy structure since all the members are equal with each other. Most probable scenario would be a person responsible for the project acting arrogant over others, distributing work without much contribution etc. However, nothing similar happened in our case, as we were all satisfied with our roles in our group.

2.2 **Individualism/Collectivism** (Karrtiigehyen and Nicolas Popal)

Individualism and Collectivism in project work relates on how people usually work as part of the group. In our case, even though Czechs and Slovaks have mediocre individualism on Hofstede’s Scale, this was not our case as we all are quite social and work well as part of the group, tending more towards Collectivism.

2.3 **Masculinity/Femininity and Indulgence/Restraint** (Karrtiigehyen and Nicolas Popal)

Masculinity and Femininity represents the values of the group and society. Masculine society strives for success, heroism, and assertiveness, while Feminine strives for cooperation, caring and satisfaction in their work-life balance. Indulgence values enjoying life and having fun, while restraint focuses on social norms and following social standards. In the case of our group, we had a great balance between work and free time, so, we tend to lean more towards Femininity and Indulgence, as we value our time more than a potential success, so the Hofstede’s Dimensions are not represented correctly, especially for the member from Slovakia in case of Femininity where Slovakia leans toward Masculinity, and Czechs and Slovaks in case of Indulgency, where both countries lean toward Restraint, as completely opposed to our group’s nature. This work-life balance is probably inherited from Denmark’s nature as Danes (and generally all

Nordics) tend to value their work-life balance more than strive for success, so living in such environment might affect a person to some extent.

2.4 Long/Short Term Orientation (Karrtiigehyen and Nicolas Popal)

Long/Short Term orientation displays if the members of the team are focused on smaller tasks rather than long-term ones. In our case, I would say inexperience causes us to be oriented specifically on smaller tasks, than the long-term ones, so at first glance, it might seem that the Hofstede's Dimensions are not represented correctly. However, this might not be the real case.

2.5 Uncertainty Avoidance (Karrtiigehyen and Nicolas Popal)

Uncertainty Avoidance measures the amount of precautions made to handle unexpected situations, adapting to them, and preventing them. In our case, we did not encounter many unexpected situations, so it is very hard to estimate whether we represent the Hofstede's Dimensions correctly.

3 **Project Initiation** (Karrtiigehyen and Nicolas Popal)

3.1 **Selection of Topic** (Karrtiigehyen and Nicolas Popal)

When the time had come to choose the project topic, we had trouble with deciding what system we wanted to develop. There were some requirements for the project that we were about to plan and construct. For example, it had to be a heterogenous system, using a distributed architecture, and using sockets and web services. Other than those requirements, we were allowed to choose any topic that we wanted to work with.

After a long debate, three project proposals were selected, a Hospital System, Food Tinder, and Airline Reservation System. The Hospital System was a system for keeping medical records about patients. The Food Tinder was a system where people could meet by posting and liking pictures of food. The Airline Reservation System is a system where people can buy tickets for flights and also manage the flights. The Hospital System was very interesting for us all, but only a crude representation would have been made by us since none of us are experienced in how a hospital system functions. The Food Tinder was by far the most unique proposal, but it was a half-baked idea and none of us had a full realization of how the system would function. In the end, the Airline Reservation System was chosen because we felt like we understood how an airline reservation system should function, given that we implemented a Library System where people can reserve and borrow books for SEP2. All of us have also bought tickets online through airline reservation systems before such as Wizz Air, giving us experience on how the interface for the user should be.

These conclusions proved to be true after consulting these proposals with our supervisors, who assured us that the Airline Reservation System is the best option out of the three, and it was consequently selected as our main topic for the third semester project.

3.2 **Formation of Group** (Karrtiigehyen and Nicolas Popal)

All four of us have worked in the previous two semester projects together. In the previous semester project, there was a fifth group member, but we chose to work with only the four of us this semester as we felt like we could work very well within the four of us.

4 Project Description (Karrtiigehyen)

In this phase of the project, the problem domain was established, and ultimately the goal of the project was set. The goal of the project was to create an airline reservation system, where users can book flight tickets, and some users can manage the flights. This was a realistic goal, but some delimitations had to be set in order to not complicate the project too much. For example, one of the delimitations was that no type of payment or money system will be added to the application. This made it easier for us to implement the system without worrying the problems that can arise from implementing a transaction system, which in itself is a complex system to implement.

After entering the initiation phase, we started by making a risk assessment table to pinpoint the riskiest scenarios that might occur during project development and to prevent them and how to react to them. Risk assessment table can be seen below.

Risk	**Probability**	**Impact**	**Cause**	**Prevention/Mitigation**	**Responsible person**	**Response**
Illness (Normal/Corona)	Medium	Medium/ High	Various/Corona epidemic	Healthy lifestyle/Self-prevention	Patrik Horný (SCRUM Master)	Redistributing work
Technical Failure	Low	High	Various	Cloud/Git backups, reliable equipment	Patrik Horný (SCRUM Master)	Replace malfunctioning equipment
Member Sabotage	Low	High	Loss of Motivation, Personal	Team-building, frequent progress checking	Patrik Horný (SCRUM Master)	Redistributing work, Supervisor meeting
Change in User Stories	High	Medium	Inexperience, Inadequate Analysis	Detailed analysis	Karrtiigehyen (Product Owner)	Implement changes
Work Overload	Low	Medium	Inexperience	Careful planning, workload distribution acording to skills and needs	Patrik Horný (SCRUM Master)	Redistributing work

Misunderstanding Customer	High	High	Inappropriate communication with the customer	Communication with customer	Karrtigehyen (Product Owner)	Changing the product according to customer's needs

Table 1: An early version of the Risk Assessment

This is an early version of the risk assessment, where the risks were made to be broad and not system specific. But the risk assessment was later revised, which can be found in the Project Report subsubsection 2.5.2.

5 **Project Execution** (Everybody)

Project execution went better than expected. This semester we tried to be more punctual on what needs to be done in time which helped us to finish the system sooner and focus on documentation and polishing the code and the system. On the other hand, we were able to finish sooner given the fact we did not have a member from last semester who slowed the process down.

5.1 **Project Development and Methods** (Karrtiigehyen, Nicolas Popal and Patrik Horny)

The project was developed during third semester with an extra three weeks reserved for the semester project from end of November and start of December.

Since we did not really spend that much time in school, meetings were held mostly on Discord and occasionally on Zoom. This year it was mandatory to use GitHub for version control of the project. At first it was little bit tricky to set it up and learn the commands, but after some time we got familiar with it and we will be using it for the rest of our upcoming projects since it is the easiest way of managing the whole project.

As our main framework we have used SCRUM together with a web application called ClickUp, which helped us to be on a track with sprints and project backlog. We did discuss on whether we should use SCRUM or Kanban to manage the work for the project. Ultimately, we settled for SCRUM since we learned it the previous semester and have had experience working with SCRUM.

5.2 **SCRUM** (Karrtiigehyen, Nicolas Popal and Patrik Horny)

Scrum proved to be a handy framework during the work on third semester project. At the beginning of each sprint, we held a discussion of what do we want to put into the sprint, what are the main points to put attention to and how much time it is going to take us to do it. In Sprint Retrospective, we realized our mistakes and put into consideration what needs to be fixed and improved.

We formed a group where we set a role for each member of the group with a condition to change roles if somebody does not like his role.

The roles were split as seen below:

- Patrik Horny – Scrum Master – his role was to manage the whole development team and overall making sure that the whole process of making the project is going smoothly. This includes setting up the meetings, managing project timeline and making sure that the team members have no problem with their given User Story.
- Karrtiigehyen – Product owner – he was responsible for setting up a Product Backlog and making sure that it was fully understood.
- All members of the group – Development Team.

It was agreed on that the length of the sprint was set to be 3 days with the amount of work done per day set to be 8 hours / member.

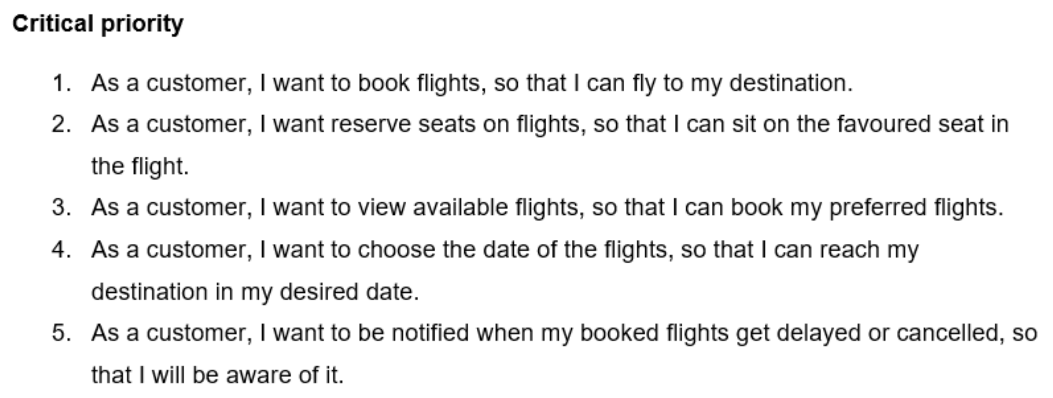
5.2.1 **Product Backlog** (Patrik Horny)

In the product backlog we put all the requirements that product owner wishes to be implemented in this system. After the backlog was established, we split it to priorities and begun working on critical priority requirements.

The process of accomplishing requirements was in the beginning rather slow, but after we got more familiar with Blazor and figuring out how forms work, accomplishing the requirements became easier.

At first, when we looked at all the requirements, we were a little bit sceptical if we are going to be able to accomplish them all. We wanted to make a system that resembles real-world airline reservation system, so we made requirements based on that. We know that this kind of system is a little bit complex, so we did not know what to expect therefore we had to put more thought into the design of the system. But at the end of the day, we were able to accomplish them all.

The product backlog is included in the Appendix O.



Critical priority

1. As a customer, I want to book flights, so that I can fly to my destination.
2. As a customer, I want reserve seats on flights, so that I can sit on the favoured seat in the flight.
3. As a customer, I want to view available flights, so that I can book my preferred flights.
4. As a customer, I want to choose the date of the flights, so that I can reach my destination in my desired date.
5. As a customer, I want to be notified when my booked flights get delayed or cancelled, so that I will be aware of it.

Figure 2: Some examples of product backlog

5.2.2 **Sprint Planning** (Patrik Horny)

Before we begun a new sprint, we did a planning on what we were going to do in the sprint. First sprints were all about implementing core features of the system and if the time permitted, we would focus on low priority requirements in the last sprints.

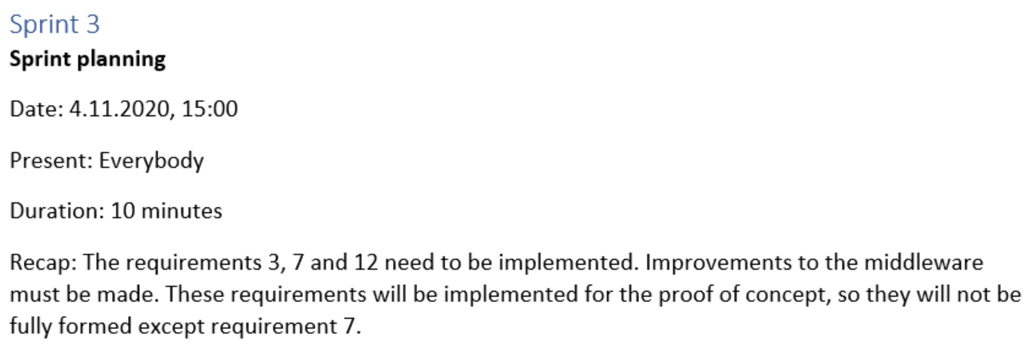
Planning went surprisingly well at the beginning and at the end. The beginning was just setting up the whole project and by the end of it we gained more experience in coding, so we were able to do more work.

Problems occurred in the middle of the November, as you can see in the Burndown chart, where we set way too ambitious goals on which user stories to finish which just resulted in pushing them to next sprints. Also, one of our colleagues was travelling back home at that time so very little was done in one particular sprint.

We wanted to finish the system around 20th of November and started testing at the end of November. This of course did not happen, and things had to be pushed. Even though our estimations we were not precise, we were able to work out most of the things at the end of November and begun the testing around at the beginning of December.

Sprint planning helped us built the system gradually and gave us a good overview on what to focus.

The planning for each sprint can be found in the Appendix P.


Sprint 3

Sprint planning

Date: 4.11.2020, 15:00

Present: Everybody

Duration: 10 minutes

Recap: The requirements 3, 7 and 12 need to be implemented. Improvements to the middleware must be made. These requirements will be implemented for the proof of concept, so they will not be fully formed except requirement 7.

Figure 3: An example of a sprint planning

5.2.3 **Sprint Backlog** (Patrik Horny)

During the sprint planning we picked specific user stories from product backlog and put them into sprint backlog. Then we gave each user story an estimate time of completion. We decided not to split user story into the subtasks since each user story share the similar tasks, so we thought of it as redundant.

The planning for each sprint can be found in the Appendix P.

5.2.4 **Burndown Chart** (Patrik Horny)

Each sprint has its own burndown chart which we then combined and put it into the main chart. At the end, this gave us a perspective of how well we were doing and if the team was on track with the time schedule.

The Burndown Chart is included in the Appendix Q.

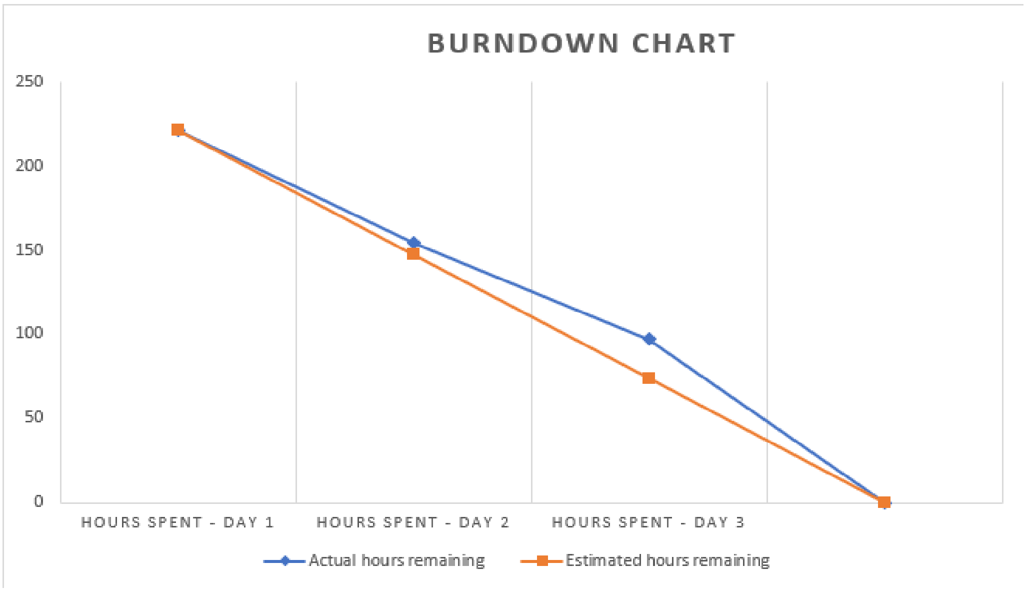
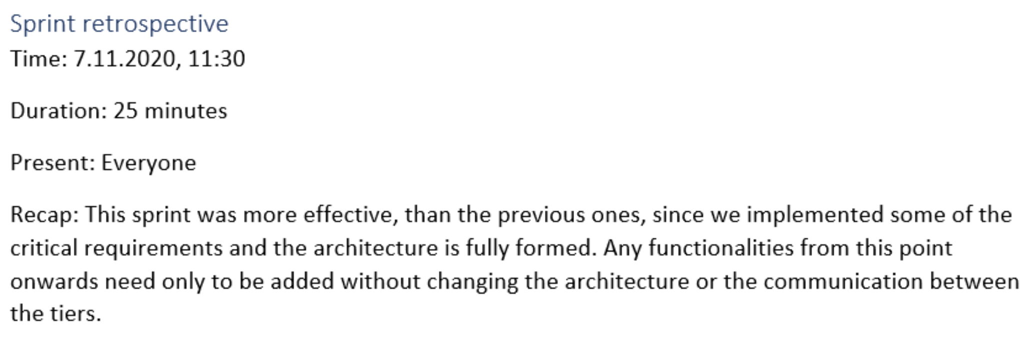


Figure 4: Burndown chart

5.2.5 **Sprint Review and Retrospective** (Patrik Horny)

In sprint retrospective we took look on what we managed to complete at the end of the sprint. There we documented our thoughts on what to focus next, what went well and what not and what to dismiss in the next sprints.

The Sprint Reviews and Retrospectives is included in the Appendix P.


Sprint retrospective

Time: 7.11.2020, 11:30

Duration: 25 minutes

Present: Everyone

Recap: This sprint was more effective, than the previous ones, since we implemented some of the critical requirements and the architecture is fully formed. Any functionalities from this point onwards need only to be added without changing the architecture or the communication between the tiers.

Figure 5: An example of sprint retrospective

5.3 **GitHub** (Patrik Horny)

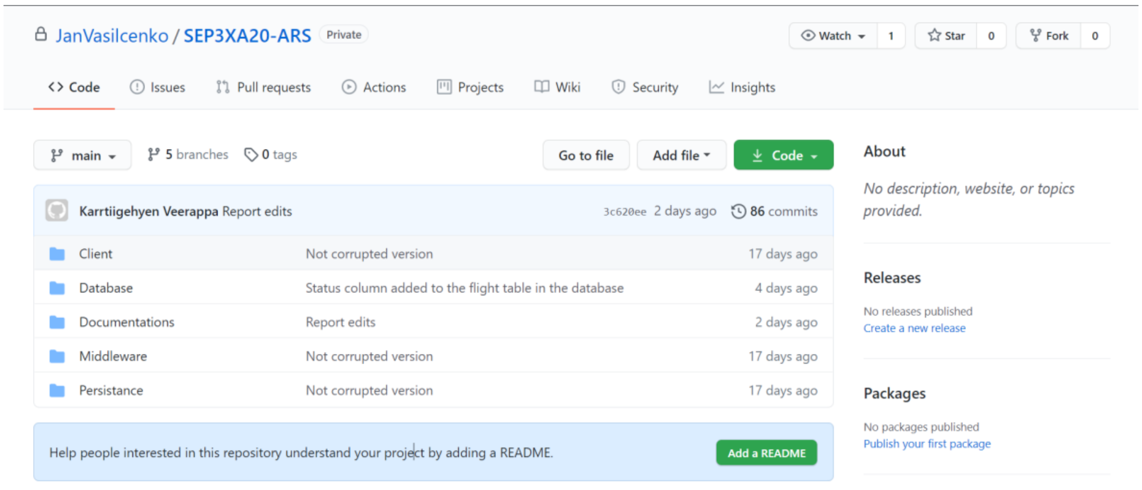
This semester we were asked to use Git as a version control framework for the semester project and using it was very useful. Last semester we were sending files through Discord and e-mails which was inconvenient since every time we had to download the file and save. Git resolves these issues very well.

We set up a shared repository which we all got access to. Then we made a three-tier architecture folder-wise structure. This gave us a good overview on where what changes were made and where were we supposed to upload new files, especially in the documentation folder where we made subfolders based on sections in a project report.

At first, we were making all changes to the main branch, but we ran into some issues when we merged our versions of the system and then we had to fix the bugs. We decided that we will make more branches where each of us would work on their stuff and then when we were finished, we would make an overview of the changes we made and then merge the versions.

Using Git was at first little bit problematic with learning new commands and setting up the repository but after everything was finished, using it was not a problem.

We will definitely include Git in our upcoming projects mainly because of the convenience of the version control of the projects and clear overview who made the changes and where with an option to revert changes if something breaks.


JanVasilcenko / SEP3XA20-ARS Private

Watch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights

main 5 branches 0 tags Go to file Add file Code

Karrtiigehyen Veerappa Report edits 3c620ee 2 days ago 86 commits

Client	Not corrupted version	17 days ago
Database	Status column added to the flight table in the database	4 days ago
Documentations	Report edits	2 days ago
Middleware	Not corrupted version	17 days ago
Persistance	Not corrupted version	17 days ago

Help people interested in this repository understand your project by adding a README. Add a README

About

No description, website, or topics provided.

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Figure 6: How the Github projects looked like

5.4 **Blazor** (Patrik Horny)

Blazor is new web framework developed by Microsoft. It allows to create web apps using C# and HTML. Blazor apps are based on reusable components that can be shared between users.

We were excited to make good looking UI since making UI in JavaFX in the last semesters was not the best way and it was difficult to use, and UI did not look good. Since we worked with Java for the last two semesters, switching to C# was not that much big of an issue since they share some similarities and Blazor uses C#, connecting the code to the front-end was not a problem, at least most of a time.

Of course, there were times where we wanted to use the Javascript, but we did not manage to initialize it, since in Blazor Javascript is replaced by C#. Also, most of the issues we had in the beginning was with figuring out how the format of the forms and the binding with the database. In the end we found out we should had just use the Blazor forms and not to mix them with regular HTML forms and inputs. Also, when we ran into some problems, often it was hard to find answers to the problems since Blazor is beginning to get recognition and there was very little information on the Internet on how to solve them. Therefore, we had to improvise a lot on how to tackle our problems.

Other than that, Blazor provides many interesting features like form validation, routing, binding the functions to the UI, it runs on WebAssembly which provides very fast performance in the browser. Working with Blazor was definitely interesting and it will be interesting what will the development of this framework look like in the future.

5.5 **Unified Process** (Karrtiigehyen, Nicolas Popal and Patrik Horny)

Since SCRUM had no defined development techniques, Unified Process was used. It was used in each sprint, where we would go through the Elaboration, Construction and Transition phases. This gave us a guideline of what should be done in each sprint. Because of the unified process, the functional requirements that were supposed to be done in the sprints were working with no hiccups, and documentation of the implementation of the functional requirements were also done.

5.6 **Critique to The Project** (Jan Vasilcenko, Karrtiigehyen and Patrik Horny)

We think that it is important to talk about things that have not gone as well as we hoped for and to talk about the design flaws of the project. This will hopefully make future projects free from these flaws, even though it is sometimes easier said than done.

5.6.1 **General Assessment and Critique** (Jan Vasilcenko, Karrtiigehyen and Patrik Horny)

This project can be called a success since all of the functional requirements were met, but there have been many obstacles and problems during the development of the airline reservation system. A prime example of the hurdles we faced is *Blazor.* While we were taught how to use *Blazor* in the DNP course, it was difficult to work with it since not many helpful documentations could be found on it, given *Blazor* is a relatively new and evolving technology.

While every user story was implemented and tested, there was not enough time to polish the code to handle all types of exceptions. During the implementation of the application, we encountered several problems due to our lack of experience designing and working with distributed systems and three-tier architecture. An example of this is the implementation of sockets, where it is just a barebones TCP sockets which uses a *Request* class to send objects back and forth. A better implementation of the sockets would implement it with a proxy design pattern, and custom exception handling within the sockets. But ultimately, we decided to keep the design patterns simple to avoid second-system effect (C2 Wiki, 2004), and to focus more on the architecture of the distributed system.

5.6.2 **SOLID Principles** (Jan Vasilcenko, Karrtiigehyen and Patrik Horny)

These principles served as a guideline how to construct the system and make it flexible and reliable.

The only notable violation of the SOLID principles can be found in some of the DAOs. For example, in *FlightDAO* class the method that adds the flights to the database also handled some logic where the ID of the flight is autoincremented, violating the Single Responsibility Principle. As a result of this, cohesion of code is reduced. A possible solution to this is would be to handle the autoincrementing of ID in the middleware instead of in the DAO.

Other than that, other principles were not violated and if they were, they were violated in a less severe manner than the Single Responsibility Principle.

5.6.3 DRY Rule (Jan Vasilcenko, Karrtiigehyen and Patrik Horny)

Applying DRY rule to the semester project was our goal from the start. But applying the DRY rule in implementation was harder than expected. For example, a lot of the DAO concrete classes have a code repetition. Code repetition was tried to be nullified by making some reusable methods that can be reused in the same classes. But this did not entirely eradicate code repetition. A good solution to this could be to use the command design pattern.

5.6.4 **Testing** (Karrtiigehyen)

The white-box test that was chosen for the system was integration testing, as this tests the functionalities. And the airline reservation system is a very user centric system, we thought it would be best to do just integration testing and to exclude unit testing.

Ideally, the integration testing should have took place in the presentation tier, but instead we did it in the application tier. There are several reasons for this. Firstly, our thinking is that the application tier holds the business logic so it should be tested (and in a way, the data tier is also tested since the application tier is dependent on the data tier), and the presentation tier is mainly regarding the GUI. But there was also another reason for not doing the integration testing in the presentation tier. The reason was that none of us are experienced in doing tests in C# or .NET. We did learn how to test Java applications using Junit last semester, and we also learned by ourselves on how to use Mockito for unit testing. This proved to be a useful skill for doing the integration tests in the application tier, but our lack of experience with testing in C# and .NET prevented us from conducting the integration test in the presentation tier.

5.6.5 **SCRUM** (Karrtiigehyen)

SCRUM was a very useful framework for controlling the workflow in our group. Having experience working with SCRUM last semester, it was easier to work with it this semester. While SCRUM was helpful (especially sprint reviews and retrospectives, and burndown charts), we did have one big problem with it. We constantly felt like we had to go out of our way to develop the system with SCRUM. Meaning, if felt like we always had additional work such as documenting the sprints, and also that we were constricted on when and how we develop the system. It was touched upon the discussion we had at the start of the semester whether to use SCRUM or Kanban in subsection 5.1. Kanban combined with SCRUM to make a hybrid framework, where the planning and management of work was controlled with Kanban, but also having sprint events (daily SCRUM, sprint retrospective, sprint reviews) and burndown charts, would have perhaps been a better choice.

6 **Personal Reflections** (Everybody)

In this section we will discuss our opinions about the whole process of making the third semester project as well as other matters.

6.1 Jan Vasilcenko

When the topic of SEP 3 was introduced, I was really interested because heterogenous systems meant more complex program to develop. Also new technology such as Blazor was new to me and I was really excited to start learning it. Then I realized that we can utilize everything that we learned so far in first and second semester and apply it to our current SEP. At first, I did not really get the usage of web services, but when we got every information I saw why they are really good to use. Working on this project was a really strong experience, because of corona. Online meetings were held on daily basis and we managed to keep on working and finally building the project. A lot problems have arisen from this, I was used to meet face to face in school, because I cannot really focus at home. To solve this problem and get accommodated to working from home I developed a schedule and designated certain number of hours each day on project work. This was successful in a way that I did not overwork and that my work progressed each day.

6.2 Karrtiigehyen

This semester, we were asked to do a distributed system which is heterogeneous for the semester project. We were also allowed to choose whichever topic we want to work with, as was the case the previous semester.

The group that I joined for this project consists of most of the same members from the previous SEP, since I felt like I could voice out my opinions and felt that everybody was on the same wavelength. We are short of one person compared to last semester, but I did not feel any additional amount of work despite being short of one person.

Using Git for version control was mandatory this semester. While it was confusing and difficult at first, it quickly became a great benefit to the group. In the previous semester, if any changes were made to the code, diagrams, or reports, we would have to use email to send the updated versions to one another. Git eliminates this cumbersome method. There were some issue using Git though. Sometimes when multiple people are working on the same branch, some work done by a person was lost due to not pushing the working and pulling the work done by other people first. But it was a minor issue since it was rarely done, and it can be reverted.

The implementation of the code also became much easier compared to the last semester, since this time we used Separation of Concerns, which made it easier for multiple people to take part in coding. SOLID principles were also a great way of designing the system since it made testing and other matters simpler.

The main problem that I encountered in SEP3 was the partial closing of the school due to COVID-19. This meant that most of the group meetings had to be done virtually. It was difficult at first, but I got used to it later. This also meant that the supervisions from our supervisors was mostly virtual, but the supervision was not lacking in any way.

In a nutshell, SEP3 had some problems, but overall was quite satisfactory for me. I learned a lot about software development, in terms of programming and group work.

6.3 Nicolas Popal

On the start of the 3rd semester, information about semesterly project were introduced. As in the 2nd semester, we could choose our own theme of the project, which was nice. We continued in the same group formation as in 1st and 2nd semester, because we already know each other, and we work perfectly as a team.

In previous semester, we were studying and attending lectures from home, because of corona. This changed in this semester and we could finally start to study in school. I was happy about this change, as I am not as productive at home as in school.

In our group, communication is a key. We were regularly meeting on mostly discord, where we were discussing and dividing group work on project, but also in school, where we could also ask for help our supervisors. We had some experience from previous semester with online meetings, so it was nothing new for us.

Overall, the 3rd semester was for me interesting experience, where I learned a lot of new things, and where I improved skills from previous semesters. On second side I cannot say that I am looking forward for 4th semester, as I was expecting to continue with AR/VR specialization in Viborg, which did not happen.

6.4 Patrik Horny

Third semester project was much more interesting and complex and elaborate than previous projects. We've learned a lot of new information about how to implement three-tier architecture pattern, establishing security for our system, version control and many more. It was definitely interesting and despite the pandemic situation and lack of physical meetings I enjoyed working on the semester project.

At first, I was a little worried if we are going finish our system since creating a ticket reservation system seems a little too complex. Studying software engineering gave me an insight on how the complexity rises by the time as more features needs to be implemented. I was particularly afraid of the reserving a seat requirement. Not just because of the UI we needed to make working but also of the implementation. But thanks to our good team we were able to get through even through complicated requirements.

With my group it was always a joy to work with since the roles are strictly set, we know who has got more knowledge in what, so the workflow of the team was really good. We tried to be more strict with accomplishing goals on time which helped the morale of the team and getting things done sooner.

Overall, I'm happy with what we accomplished. Even though we did not make the most polished system, we finished what we wanted and I'm looking forward to work on another semester project with my group.

7 **Supervision** (Jan Vasilcenko, Karrtiigehyen and Patrik Horny)

For this semester, our project supervisors were Jakob Knop Rasmussen and Ole Ildsgaard Hourgaard. Their supervision was very good. The questions we asked were answered clearly and supervisors were very helpful when solving our problems. The feedback on some of the assignments, such as Project Description and Architecture, were the most useful as they guided us clearly to our path. The biggest issue in the supervision was the response time, when we sometimes had to wait for number of days to get a reply. However, due to a coronavirus situation and the fact that some of these questions were asked during the semester project work weeks, when teachers usually get large number of questions and requests from other semester project groups, it is understandable that the response time was slower than we expected.

8 **Conclusion** (Karrtiigehyen)

In this semester project, we learned how to construct a distributed system, which also had to be heterogeneous, using the three-tier architecture. From this, we learned how to how to design a three-tier architecture with the use of web services and sockets, how to construct an application using more than one programming language, and how to properly use Git for version control.

SCRUM was a good tool to keep progress of work and to structure the work left to do. It also allowed us to reflect on how work was done in the Sprint Retrospectives, which showed us how to improve our management of work for the future sprints. The burndown chart was used to properly visualize the work done and left, which led us to better time management. While SCRUM is a good framework to manage the workflow, we could not take full advantage of it. This is because of lack of physical meetings due to the pandemic. All of the meetings and development was done remotely through virtual meetings, which made the SCRUM framework more of a hindrance than a useful tool to structure our work.

Unified process was used as the development technique since SCRUM does not have any defined development techniques. The Unified Process is better compared the methods such as the Waterfall method, which was used in the first semester. The Unified Process reflects the real world better since a lot of the phases overlap and previous phases are often revisited and redone.

While developing the software, we constantly kept in mind that the SOLID principles and the DRY rule should be followed, but they were violated in some cases because of our inexperience dealing with these principles.

In a nutshell, the purpose of this semester project was accomplished, with the use of SCRUM and Unified Process, and Git despite having difficulties communicating about the project as a result of the pandemic. Virtual meetings were held constantly to fill in the void of not having physical meetings. This taught us the value of meeting in person to discuss problems and cooperate on our project.

9 Sources of Information

1. Schwaber, K. & Southerland, J., 2017. The Scrum Guide. [pdf] Unknown: Creative Commons. Available at: < https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf> [Accessed 31 May 2020].

2. CatB, 2004. *Second-System Effect*. [online] Available at : <http://www.catb.org/jargon/html/S/second-system-effect.html> [Accessed 08 December 2020].

10 Appendices

O – Product Backlog

P – Sprint Breakdown

Q – Burndown Chart

R – Link to Github

S – Link to Video Demonstration