VIA University College

Learnify - Process Report

Semester Project 3

Group 3

Students

Guillermo Sanchez Martinez (355442)

Piotr Wiktor Junosz (355502)

Halil Ibrahim Aygun (355770)

Alexandru Savin (354790)

Eduard Fekete (355323)

Supervisors

Joseph Chukwudi Okika (JOOK)

Jakob Trigger Knop (JKNR)

Character Count: 33438

Word Count: 5378

Software Technology Engineering

3rd Semester

December 18, 2025

Contents

1 Introduction

The project of a Learning Platform was initiated with the goal of creating an accessible and user-friendly online environment for learners. This project focused on developing a distributed system in a team environment, leveraging various distinct technologies and methodologies to ensure effective collaboration and successful delivery.

---

2 Group Work

Many of our group processes stemmed as a continuation from our previous project experience, because our current group consisted of the same five members. For this reason, we continued with improving our collaboration based on E-estimate personal profiles completed in the previous semester. Knowing the fact that our profiles consisted mostly of dominant red and blue working style and our last experiences we found a way of working really efficiently by creating clear todo lists and working mostly in smaller sub-groups. The high number of "Red" profiles unfortunately meant also that we were prone to conflict.

Figure 1: Personal Profiles (Appendix 6.1 Personal Profiles)

Our team was culturally diverse as we come from: Slovakia, Poland, Moldova, Spain and Turkey. The Southern European members of our team exhibited flexible relationship-based communication which proved to be a good balance for the structured task-oriented methods used by Central and Eastern European members. In addition, our earlier experiences together enabled us to handle cultural differences better because we understood each team member's work habits which helped us predict problems and address personal challenges before they could harm the team's performance. For example, knowing who preferred straightforward communication versus who tended to procrastinate with the set tasks allowed us to motivate each other effectively while maintaining a healthy working balance.

We based the group contract (Appendix 1.1 Group Contract) for the project on our previous contract with slight changes, for example to the point system. This group contract outlined our core values, purpose, expectations, conflict resolution strategies, and accountability measures in the team. While social loafing still occurred in the group, the team tried to handle it in a different way compared to having super strict contract with a lot of penalties in it. Unfortunately, this approach did not work well, thus many serious talks took place within the team, including warnings about the necessity of supervisor interventions if performance of the member did not improve.

2.0.1 Professional Collaboration

To document our ability to independently take part in professional collaboration, we adopted industry-standard practices such as using git. We used branching with Pull Requests to review code, ensuring no single person could break the main build. More details about use of version control can be found in the UseOfVersionControl (Appendix A: UseOfVersionControl.pdf). We communicated asynchronously via Discord/Teams to respect deep work time.

An important feature of the project's process was the role of a Product Owner (PO). The PO was responsible for managing the user stories, prioritizing the backlog, and ensuring that the team was aligned with the project goals. This role was crucial in maintaining a clear vision and direction for the project.

Contrary to our previous experience, this role was set to be more settled and less rotating among team members. This was done to ensure consistency and a clear point of contact for the team regarding project requirements and priorities. The role of the PO was assigned democratically at the start of the project.

---

3 Project Initiation

The initial ideas we worked with included a Health Assistant, Bank System, and the Learning Platform. These were chosen as relevant candidates from a brainstorming session based on their potential impact and feasibility. After discussing the vision and scope of each idea, we held a democratic vote to select the final project. The Learning Platform was chosen due to its clarity of purpose and alignment with our skills and interests.

The alignment on the actual vision was more complicated than the initial idea selection. We had to ensure that everyone in the team agreed on the project's goals, target audience, and key features. This involved several discussions and compromises to reach a consensus that satisfied everyone.

One of the problems we faced during the initiation phase was deciding on the actual scope of the project. We were all juggling between thinking about it as an actual ambitious product versus a simpler prototype suitable for the course requirements. This problem was resolved by agreeing on scalable and flexible solutions that allow for both simple approaches and potential complex upgrades.

Another challenge was approaching the problem in a data-driven manner. Considering our mostly European backgrounds, we had to ensure that the project's aims were relevant and applicable on a global scale by researching, and not relying solely on our personal experiences and assumptions. This also included exploring various perspectives and shifting some of the focus on minorities and underrepresented groups in education. This aligned with our mission to improve the education, rather than just having business goals focusing on making profit off of profitable majorities.

3.1 Choice and Alignment on Framework

Shortly after the group formation, a process brainstorm session was hosted, where most details of the desired framework were discussed and the Product Owner was selected. The full export of the session can be found as Appendix 12.1 Process Brainstorm.

---

4 Project Execution

The way of working in the project was a radical shift from our previous experiences. We adopted various aspects of Agile methodologies that fit our desired workflow. This included sprints, regular meetings, pair programming, Kanban but also our own adaptations to these practices.

Our main workflow was initially communicated and written down in the (Appendix A: Project Guidelines). This document served to outline our processes and roles but also aligned us on semantics and definitions of key concepts. Even though this document served as our aim and theoretical perfection we could aim for, perfect adherence was not always possible and advantageous.

One of crucial aspects of our work was embracing asynchronicity, which might contradict with our vision of pair programming but was seen collectively as the means to achieving better productivity and work-life balance. Asynchronicity was also implemented on a level of subgroups - fx. different feature groups would collaborate within the group in real time while asynchronously reviewing and agreeing with the rest of the team on other issues.

4.1 Before Project Period

Because of daily school and work commitments, we mostly worked asynchronously in this period. Our sprints were one week long, starting and ending on Wednesdays. Each sprint would start around lunchtime as we wanted to have time to end the previous sprint collaboratively. Each sprint would start with a planning meeting where we would discuss the goals and tasks for the sprint. At the end of each sprint, we would review all features and tasks, and discuss any blockers or challenges faced during the sprint.

4.2 Project Period

Our work during the project period looked as follows:

4.2.1 Week perspective

We met every working day, weekends voluntary individually.

4.2.2 Day perspective

We preferred to work remotely as this reduced commuting time, allowed for more flexible working hours, but also prepared us for future remote work.

Each day started at 8:20 with a daily standup meeting where we discussed for each: - What was done - What will be done - Any blockers

After the standup, we would plan the day's work, which we considered one sprint in this dense period. Based on our framework, the purpose and focus of a sprint is to merely label and agree on which aspects of the Kanban board we focus on during the sprint. Contrary to Scrum, not all focus was put on putting tasks from TODO to DONE, but rather on deciding how far to push each task (allowing reviews to be done in a different sprint and similar).

Depending on the approach of each sprint, we would either be in a group call or split into channels for pair programming. With more iterations, we realized that collective calls were more effective as they allowed for quicker communication and synchronization on key issues; often problems would reappear for multiple people and having everyone listen to the solution saved time in the future.

Each day the calls lasted until around 16:30 with breaks for lunch and short breaks in between. Depending on everyone's situation, some days the work would continue later into the evening/night for those who preferred that.

4.2.3 Feature perspective

Working on a feature was usually for 1-3 people subgroups. The feature was started by discussing the definition of done, breaking down the tasks, and creating a branch for it. Each feature was practically a vertical slice of the product and the work was also often split into vertical slices (in order to reduce interpersonal dependencies). The aim of each feature team was to deliver a working feature compatible with main (which was evolving in parallel) by the end of the feature work. Each feature would have to be reviewed by someone least biased before merging to main.

Working with the team usually involved calling, sketching ideas, and drawing UML diagrams, while researching domain context and processing data from users. On a practical level, we used Visual Studio Code with Live Share for pair programming, GitHub for version control, Figma/FigJam for brainstorming and unrestricted diagramming, PlantUML inside VSCode for quick UML sketches, and Discord for screen sharing and communication.

4.3 Adherence to the Framework

It must be noted that the adherence was not perfect. One such example is periodical claiming of tasks that often happened with no reason, which resulted in a so-called "Code Ownership Trash Can" where labels were put from tasks that did not need to be reserved by anyone. This can be seen on the figure below:

Figure 2: Code Ownership Trash Can

Another important point is the splitting of the work and the adherence to Kanban in critical situations - like before deadlines. In these situations we resorted to more chaotic working as can be seen on the figure below, depicting a cumulative clump of tasks that arised towards the end of the project:

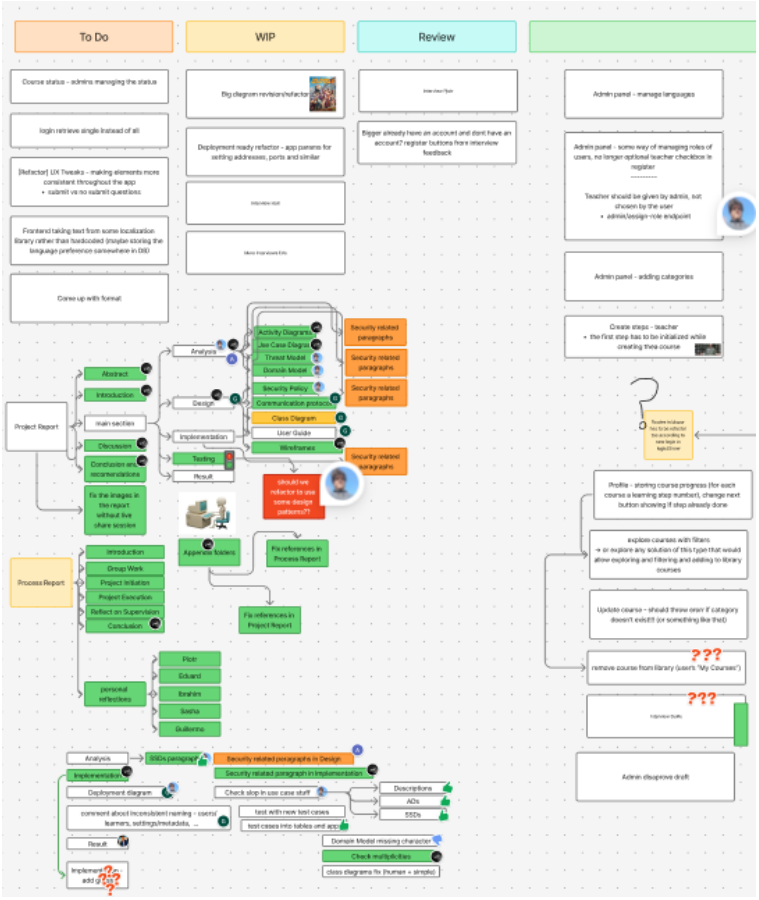

Figure 3: Tasks Before The Deadline

5 Choice of Tools and Methods

5.1 Task Decision and the PO Role

One of the most often used techniques for determining the priority of tasks was utilizing a board with Impact and Effort axes. Despite a lack of public information about such usage - in time restricted situations, the tasks were prioritized from left to right, and in periods with time to spare - top to bottom. Overall, this method proved to be an effective planning technique that allowed the PO to quickly and relatively with no bias to prioritize tasks. A snapshot of a usage of this graph can be seen below:

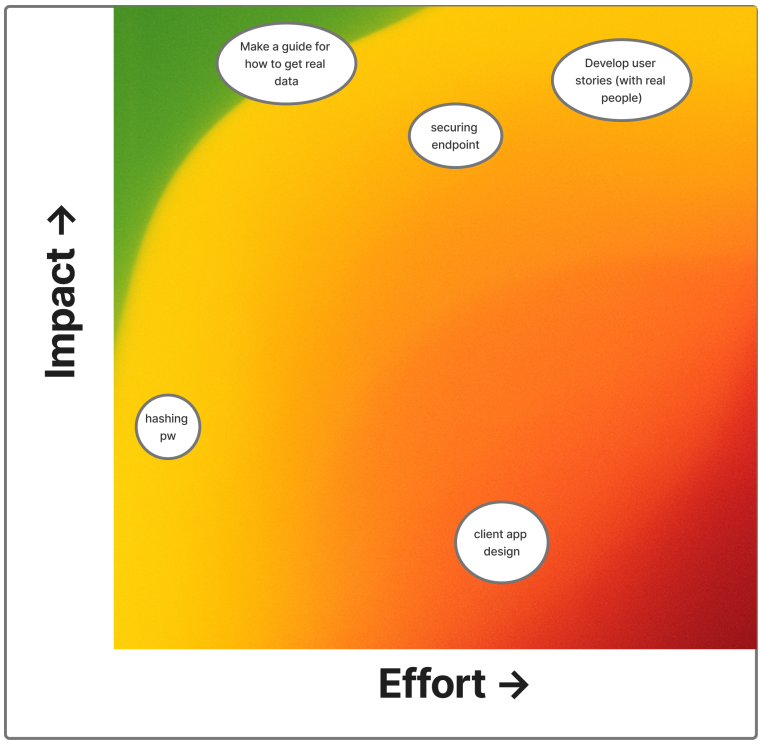

Figure 4: Impact Effort Snapshot

Creating concepts from sources like user stories, non-functional requirements and others also proved to be a powerful idea that arised from the project initiation period. This method allowed great flexibility while keeping the opinionated choices of prioritization - often arising from the data-driven nature of the project.

5.2 Tools

A powerful tool for this project compared to other semester projects proved to be Visual Studio Code by Microsoft. Through the power of extensions, actions, and powershell scripts, the whole team was able to work collaboratively and effectively.

For documentation, a hybrid approach with markdown (Pandoc) and PlantUML was chosen. Markdown allowed for easy collaboration (both Live Share and Git) with not many conflicts while providing a powerful framework in which styling is not a concern during content creating. PlantUML allowed for quick diagramming and sketching of ideas without the need to open a separate application. This allowed for quick iterations and changes to diagrams as the project evolved.

PlantUML was also important for all UML purposes in general - text-based form of diagrams allowed for great collaborative experiences, simple exports and truly dynamic changes. The ease of use and its popularity also resulted in UML being used outside of pure engineering diagramming, for example as shown on the figure below depicting a quick guide we shared on our board in Figma:

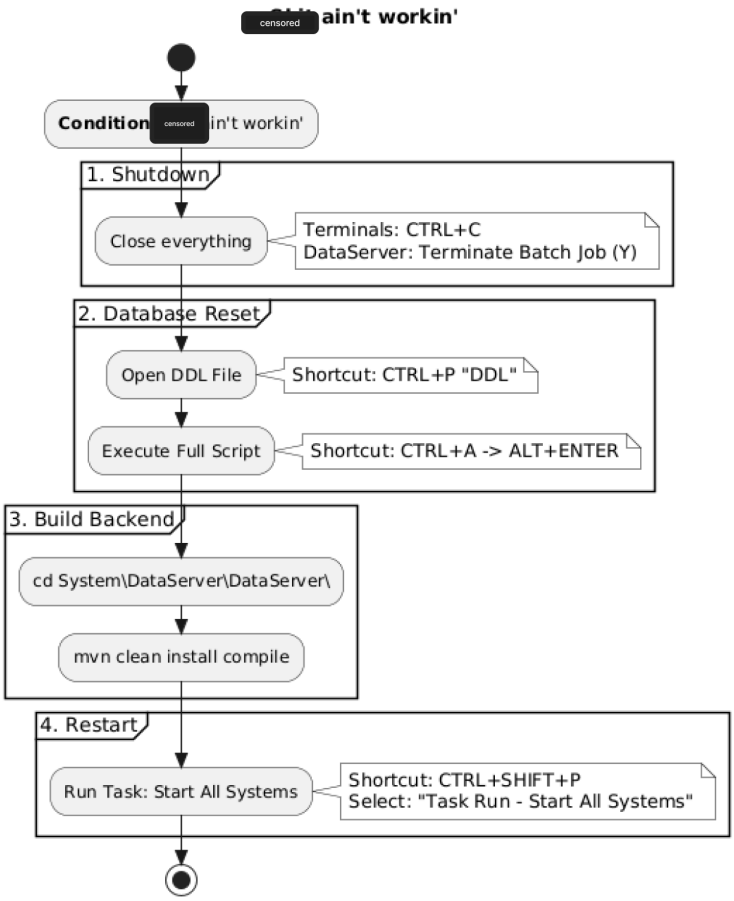

Figure 5: UML Guide

Visual Studio Code also allowed for efficient work with the database - providing the ability to connect to the database and easily execute all queries from the same environment as everything else.

On top of hosting all these features, Visual Studio Code also allowed for automating terminal sessions for running all parts of the system (building documentation, running all systems, zipping everything) and because of the language server support, also provided one IDE for managing the whole project.

Despite that, a significant part of the collaboration still relied on Figma - including a part of documentation (diagrams, user story maps, ...) and general task/iteration management. Figma proved to be a superior tool for brainstorming, quick collaboration sessions, and truly free and easy alignment sessions.

---

6 Personal Reflections

6.1 Guillermo Sánchez Martínez

During this semester we wanted to try a different agile framework since we do not know which one our future company will use ,and we would like to find the most relevant for us in order to work more efficiently. This time we decided to combine Kanban with the role of Product Owner from SCRUM. This allowed us to switch more easily from one task to another without needing to wait for the Scrum Master to assign the next task. Additionally, keeping the role of Product Owner, helped us maintaining an unified and static vision of our final system.

At the beginning of the project I thought this agile method was the best for me, but now that we have finished I would not say so. Because of not having any tasks assigned, I felt less attached to the project, and in some cases that meant taking longer to finish a tasks because it did not feel like mine. This effect of personal disattachment from a task has been studied and it has been proved that clear task ownership is directly linked to increased motivation and project engagement (Masood et al., 2021). I am still getting to know myself and this project has definitely helped with that. If I could reshape our agile framework I would have included a rule saying that a task must be taken by a person, so I could feel more attached to it and therefore work more effectively. To continue discovering myself and how do I perform in different methodologies, I am open to explore other agile frameworks in the next semester in order to find the one that suits best with me. Since in a company I will have to adapt to their prefered working methodology.

In this project there has been differences between members that worked actively and members who worked passively. Meaning that some of us were taking tasks one after another, sacrificing their time for the good of the project, while others spent less time and priorized personal time. I personally think that this group is great, but after collaborating for two projects in a row we might need a change. We already know how each of us reacts to different challenges and because of not having task ownership, we knew that if we didn't finish a task someone could take over our work and continue from there. Maybe expanding the group to 12 members next semester is the change we need.

Regarding next semester project, I am excited to work with such a big team and take the project as a job and not as an assignment. Since it might be the last big project I am going to do before landing an intership, and therefore it will be the last opportunity to learn a lot before getting into the real job market.

Concluding, the Kanban framework has been something useful to work with but I would make some modifications to it in order to perform better. I am happy with the final system we built, and I feel like this has been the first real project I have ever done, which is something i am proud of.

6.2 Piotr Junosz

This semester, our group decided to explore new methods and frameworks so that we can learn more about different types of software development processes. The idea was to maintain working in the iterative, agile approach, transiting from strictly using Scrum with Unified Process (UP) to Kanban workflow while retaining Product Owner role which was found useful in the previous semester. For me, the most significant aspect resulted from changing the way of working, more precisely planning and splitting the tasks within the group. We moved from a push-based planning model, in which the work is assigned upfront in the beginning of each sprint, to a pull-based system using our Kanban board. The efficiency of completing and splitting the tasks improved greatly due to the fact that after each group member finished their work, they could immediately pull next tasks from "To Do" column. Those experiences can be explain by comparing push and pull theories, which confirms the success of our Kanban approach. A Push System relies on predetermined scheduling, where "work items are scheduled based on deterministic planning" (Kanban University, 2022). In comparison, Kanban uses a Pull system, where the process is controlled by available capacity. The Kanban Guide explicitly states that in a pull system, "completed work is regarded as more valuable than starting new work" (Kanban University, 2022). Our new team rule - that everyone has to commit all changes before taking a break - was designed to support the flow and improve efficiency of completing the tasks even more. Because of this policy, unnecessary waiting for a group member to finish they assigned tasks was avoided and everyone could just take over the unfinished work. Having experienced the fluency of the pull system, I now realise that self-organized pulling work based on capacity is far more efficient than the push system in the Scrum + UP setup. This successful transition showed us that there are a lot of different frameworks and methodologies to try out in the future development work. Understanding each new process method while comparing to the ones already learned is essential when it comes to choosing your own best and most efficient approach. As a next step, I would like to know more about defining clear process rules as well as learning any new agile frameworks that can improve group work efficiency even further.

6.3 Alexandru Savin

My thinking about feasible SEP3 ideas and their implementation started long before the actual project. I and my group members shared a common idea that we should develop a product in sync with the current SDGs, and although it had to be scoped and narrowed down, it is good to know it's core idea remains to promote free education possibilities and share human knowledge. During the project I brightened my understanding and got new competencies in Java, C#, and architectural patterns.

I'm think overall it was a correct and fair decision, that we chose kanban board as our main framework. Our app is of small scale and it was comfortable to keep track of where the current development progress was. We transferred the role of Project Owner of Scrum into our project and it played out well for the project, however for group members to feel more responsibility I can imagine we should have implemented the term based switch of the POs as we did in our previous semester.

Unfortunately, I did not make timesteps of our kanban board in order to analyze a cumulative flow infographic. Based on additions over time from our github inside statistics, we see, we've experienced some blockages and our workflow went not as steady as expected, nevertheless the easy pull of tasks mechanism allowed group members to overcome those obstacles. Each of our group members shared ownership of the project and could contribute anytime to adding or completing the tasks from the board.

Github version control and working on different branches felt much more useful compared to last semester. We developed a rule and sticked to it most of the time, claiming that each branch once ready, had to be peer tested locally, reviewed and only then merged into main, which also makes common sense.

I highly value the real feedback I received, because it did in fact once happen that, while one of my implementation methods pull request was being analyzed by the reviewer from security risks perspective, it proved to be a threat, which was shortly mitigated shortly after and before completing the feature and deleting the branch.

The time spend on this project gives me another chance to dive in and retrospect my workflow over the past weeks, my knowledge, my abilities to contribute to a project as an independent individual, evaluating and completing tasks on one side of the hand and as a student, a team member, on the other hand, where to be a good student is to be predictable, someone who can either be keep up with the tasks he choose to solve and be transparent about the difficulties he encounters and just be a group member other peers could rely on. All in all, I'm looking forward to the next semester project and eager to learn and practice methods to help me work and acquire knowledge more efficient.

6.4 Halil Ibrahim Aygun

During this project, I was involved in almost every stage of the system, working across different layers and at different points in time. I often handled features that needed to work end-to-end, meaning I had to understand and connect the database layer, backend logic, and client-side behavior. This gave me a much broader view of the system than focusing on a single isolated component, but it also came with challenges that required constant coordination with the rest of the team.

Our workflow was based on shared responsibility rather than strict code ownership. Multiple people could work on the same branch, and progress was discussed daily during meetings where we talked about what we had done, what we planned to do next, and how we could improve our approach. Having a shared Kanban board helped keep tasks visible and flexible, but it also meant that changes made by one person could quickly affect others. Because of this, I had to deal with merge conflicts several times, especially when my branch was waiting for review while other features were merged into main.

Handling these conflicts taught me how impactful even small changes can be in a shared codebase. I became much more aware that my decisions did not only affect my own work, but could slow down or complicate the work of others. This experience highlighted the importance of communication and timing, especially in a project where many features evolve in parallel.

At the beginning, working with a distributed system that combined multiple technologies and programming languages felt chaotic. However, by applying the principles and structure we learned during the semester, I

gradually understood how the different parts fit together. This helped me move from seeing the system as many separate components to understanding it as one connected whole.

Through this project, I feel more confident working in larger teams and navigating complex systems with multiple branches and contributors. I learned how to collaborate in an environment where changes happen continuously and where coordination is just as important as writing code. If I were to work on a similar project again, I would focus even more on early communication about changes, discussing potential impacts beforehand, and supporting teammates more actively. Helping others and receiving help often led to learning new things myself, making collaboration a clear win-win.

6.5 Eduard Fekete

This semester brought the best and the worst of experiences. To start on the positive side, I believe that the way of our working was an extremely powerful blend of industry standards that was created after our team debates. I dreamt of something like this for a long time, every time listening to Uncle Bob, Kent Beck, Martin Fowler and other great guys, I just thought about how exactly it could work for me, how could I arrive at some satisfactory way of functioning that would be easy to get people into, that would not make VIA complain, and that would actually make sense.

And here it was, the nice chaos of XP with the structure of Kanban and benefits of Scrum. Although, not to idealize - many people absolutely avoided pair programming. Once when I suggested "let's do it together, can you share screen?", the answer was "Nah, it's super easy, I'll just do it and you check it afterwards". And yes, that defeats the purpose, this wasn't 100%. And it wasn't super easy.

It also really got me when someone would "review their own changes", and in an attempt to redeem "refactored the code" and funnily enough, all this without even running or contributing to the tests.

However, claiming that the process failed because of a lack of adherence and expertise puts us back to the waterfall times, where an argument of type "You should have thought about it in the Analysis phase" would be the way to justify why waterfall works rather than addressing its flaws.

And so the question arises - *what was it that failed and what was it that worked*?

In order to look why things worked, let's look at what worked. We were able to efficiently and quickly deliver features, with not much bureaucracy and sweet sweet agility. Disputes of what and how to do were replaced with getting data, shared code ownership, and a bunch of refactoring.

What didn't work? I believe that our bus factor was at most 2, if not 1. We absolutely lacked a shared understanding of crucial aspects despite tons of accessible documentation. And there it was - "tons" of documentation maybe failed us.

The group contract is another chapter of the story itself. Karaoke never happened and was not even a proper punishment to begin with. I never liked the idea of involving supervisors or any such external party in a position of authority in the SEP conflicts because it just felt like cheating to get conflicts resolved by someone else. Now I would not go into a project without signing something saying that if someone decides to not show up absolutely for anything, and if they lie about their work, they will have real consequences besides "singing karaoke". I cannot believe my *naïvety* of thinking that easing the rules from last semester would just work like magic, I think that freedom is something more complicated than just removing rules.

And this connects to the bus factor again - this system worked better than the previous Scrum + UP one, but was it because we were more productive or because suddenly without any fairness/equality concerns of sprint planning, there was full power for a couple of people to do the work while the rest could ease? Did the lack of code ownership mean a lack of responsibility over the system? Is it in any way fair if someone has more responsibility over the system *purely* because they put more effort? And actually, did I bite myself on this because I predicted a lot of this in my head and thought that having no constraint of taking on the work would somehow only improve my efficiency without having a negative impact on the rest of the people?

To bridge the gap between speculative illusions of reality and the more refined reflection of it, let's form some facts:

- the group contract was broken often, consequences were not enforced
- the methods were more of an inspiration rather than a strict framework
- promises were broken, assigned work was not done
- transparency and honesty were often lacking (e.g. AI usage)

And with a weak proof or rather subjective anecdotal evidence, if these facts hold true, then I claim that none of the previous/other processes would have worked. I believe that a team that does not have a good sense of responsibility, honesty, and transparency can not achieve good results with any normal process; with normal including nothing that would break the law, ethics, ideally common sense as well.

Nevertheless, I would not be able to claim that I am data-driven without proper objective measurements. Therefore, I propose that if the process of judgment is fair, then the grade should reflect the quality of my work on semester project, not my subjective talks in this report. And without such judgment, I could merely conclude and reflect that my contribution was not perfect but never expected to be perfect, though of high quality, consistent, and conducted in a professional and ethical manner.

And to reflect further, I would like to say that I haven't felt such pride in a long time as when seeing some of the team members just crushing the whole SEP aspect of the project. Conducting an hour-long interview on a call because we needed data, studying how kanban works, speaking up on big issues, and even going to "prison" just to lock on a task because that's what we agreed to do are in my opinion the moments that will shape this team into something more than regular Software Engineers. I hope that people who find themselves in this understand the transformation that happened during the semester in the same way I see it.

---

7 Reflect on Supervision

In the project team, we were often aligned on ideas and approaches, which made critical analysis and reflections on our work more challenging. The role of supervision thus became a crucial element of our project process. The supervisors often challenged our assumptions and decisions, ensuring that the project was kept improving based on proper processes and data, rather than just group consensus and momentum.

Each of our supervisor meetings had a clear agenda - acquiring external feedback and clarifying doubts. We did not focus on mere presenting of our progress but rather on seeking constructive criticism and guidance. This approach allowed us to identify potential pitfalls and areas for improvement that we might have overlooked without the supervisors.

Another aspect of the supervision was filling the technical gaps in our knowledge and project requirements. The supervisors always helped with long-term planning which could not have been done by the team alone due to the limited knowledge towards the beginning of the project.

The frequency of meetings was resulted mainly from the mandatory supervisor meetings and our own needs. Therefore, every time we were not sure about our decisions related to the project we communicated with one of the supervisor about the need to have a meeting with them. We communicated mainly through ItsLearning and e-mail.

---

8 Conclusion

The project showed how flexible structured team collaboration works through our transition from Scrum to Kanban pull-system workflow. The experience we gained helped us develop a list of best practices and common mistakes which will guide our future group projects.

The team needs open communication and respect as fundamental values together with a Product Owner who manages the vision to achieve successful teamwork. We should keep using standup meetings to exchange information while supervisors need to provide essential feedback and help with technical issues and code submissions must happen regularly to avoid creating bottlenecks.

It would be a good idea to avoid three main mistakes which include making decisions based on assumptions when data exists, enforcing processes that block work and splitting into groups that do not collaborate well with each other. We should focus on achieving fundamental goals before attempting to tackle complex problems.

We need to keep the group contract active while using Kanban pull-based workflow management and feature decisions must be based on data. We should seek outside feedback to confirm the project's direction.

---

9 References

- Kanban University. (2022). The official guide to the Kanban method (Version 2). Mauvius Group Inc. https://kanban.university/wp-content/uploads/2023/04/The-Official-Kanban-Guide_A4.pdf
- Pierce, J. L., Kostova, T., & Dirks, K. T. (2001). Toward a Theory of Psychological Ownership in Organizations. The Academy of Management Review, 26(2), 298–310. https://doi.org/10.2307/259124

10 Appendices

10.1 Appendix 1.1 GroupContract

10.1.1 Group Contract

Can be found as GroupContract.pdf

10.2 Appendix 7.3: Other documents

10.2.1 Project Guidelines

Can be found as ProjectGuidelines.pdf

10.3 Appendix 6.1 Personal Profiles

10.3.1 Personal Profiles

Can be found as PersonalProfiles.png

10.4 Appendix 12.1 Process Brainstorm

10.4.1 Process Brainstorm

Can be found as ProcessBrainstorm.png