Production Optimiser Final report

Authors: Fran Hruza, Danko Čurlin, Georgs Lukass Rozins, Paula Šalković, Shreesh Kumar Jha, Giovanni Orciuolo, Samarth Bhatia

Version: 1

Date: 17.01.2025.

Introduction

Current manufacturing industries always have faced and will face complexity in optimizing production processes, from resource allocation to scheduling. To address these challenges, MITC has developed an AI-driven Production Flow Optimizer. This tool processes data about machines and products to calculate the most efficient production sequences, offering tangible improvements in efficiency and cost-effectiveness.

The Optimizer operates using Excel data inputs with specific guidelines on the file's structure making it easy to use and efficient compared to more complex alternatives. However, to expand its accessibility and scalability, the goal was to make Production Flow Optimizer into a web-based platform, which utilizes MITC's already developed algorithms.

The project was proposed by MITC, main stakeholders include Vladana Celebic, John Moberg, Akshay Goyal, and Nils Erlands.

The requirements were outlined in the Requirements document with appropriate identification, actors, and descriptions. Design was outlined and updated throughout the project lifecycle in the Design Description document, currently in version 3. In this report these two documents will be referenced as requirements document and design document.

The project followed the SCRUM methodology.

Project results

List of deliverables

- Project plan
- Design document
- Requirements document
- Acceptance Test Report document
- Acceptance Test Results document
- User interface – detailed descriptions are provided in design document section 4.3
- Backend API - detailed descriptions are provided in design document sections 4.3 and 6.2
- Python service tool wrapper – detailed descriptions are provided in design document sections 4.3
- Sprint review and retrospective reports every sprint

Requirements breakdown

Requirement ID (refer to requirements document)	Status	Notes
FR1	Implemented	-
FR2	Implemented	
FR3	Implemented	
FR4	Implemented	
FR5	Implemented	
FR6	Implemented	
FR7	Implemented	
FR8	Implemented	
FR9	Implemented	
FR10	Implemented	
FR11	Implemented	
FR12	Implemented	
FR13	Implemented	
FR14	Not Implemented	Low priority requirement left over at the end of the project, with more time it would have been implemented, it was not a must have feature
FR15	Implemented	

NFR1	Partially implemented	Backend and frontend is fully secured, Python service tool wrapper only checks CORS rules, no actual authentication was implemented
NFR2	Implemented	
NFR3	Partially implemented	Point 2. and 4. implemented 3. is partially implemented, 1. is not implemented, with enough time it would have been
NFR4	Implemented	Logging enabled on all activities
NFR5	Implemented	
NFR6	Implemented	
NFR7	Implemented	
NFR8	Implemented	

Project work

Team structure

The team consisted of the following individuals:

- Giovanni Orciuolo
- Danko Čurlin
- Paula Šalković
- Fran Hruza
- Shreesh Kumar Jha
- Samarth Bhatia
- Georgs Lukass Rozins

Fran Hruza was chosen for the Scrum master role and Danko Čurlin for the Prodcut Owner role, and this arrangement was kept for the rest of the project. As far as organization into sub teams go it had the following structure at first:

- Backend team
    - Giovanni Orciuolo
    - Paula Šalković
- DevOps/CI team
    - Fran Hruza
    - Danko Čurlin
- Frontend team
    - Shreesh Kumar Jha
    - Samarth Bhatia
    - Georgs Lukass Rozins

In the first 2 weeks of the project Georgs Lukass Rozins was moved to backend team to implement Python wrapper for service tool provided by MITC and later moved to new documentation team because of extra attention needed for documentation that was lacking. Furthermore, Fran Hruza and Danko Čurlin transitioned into backend development roles because of lack of tasks regarding CI setup and volume of tasks on the backend. This is how the structure looked like at the end of the project:

- Backend team
    - Giovanni Orciuolo
    - Paula Šalković
    - Fran Hruza
    - Danko Čurlin
- DevOps/CI team
    - Danko Čurlin
- Frontend team
    - Shreesh Kumar Jha
    - Samarth Bhatia
- Documentation
    - Georgs Lukass Rozins

Project organization and routines

Communication

During the starting phase of the project, we opted to use Slack for communication but in a matter of days switched to Discord because of Slack messages being deleted after 90 days and other features Discord provided that were preferable. Discord was split into following channels to filter information based on topics, having the following structure:

- Social – channel for non-project related topics
- Meetings – summary of meetings held
- Resources – place for different link and documents or tips on technology, process, coding, and anything else someone might find useful
- Supervisor-links – channel for aggregation of links and resources relevant to the project supervisors for easy access
- Best-practices – rules and coding conventions used in project
- Sprint-hours – hours tracking relevant information and links to Excel sheet containing hours
- Reports – channel for progress reports, every Thursday usually members would give updates on their tasks
- General – general discussion
- Backend
- Frontend
- Python model – related to service tools
- CICD
- Docs
- Jira
- Git
- Bugs

MS Teams was used for meetings

Version control and collaboration

A GitHub repository was set up for code version control and for centralization of code. It contained 3 folders, backend, frontend, and model-services to separate different components of the project. Our branching strategy was as follows:

- Main – stable production branch, the code gets built and deployed using GitHub actions to Danko Čurlin's home server at the end of each sprint after being merging develop branch into it
- Develop – integration branch, all feature branches are merged into it after being tested and the code was reviewed by one of the team members
- Feature Branches – for each feature being developed a new branch was branched out from latest develop, once the feature was done a pull request was created, automatically tested using GitHub actions and upon being reviewed by a team member was able to be merged into develop branch

Task tracking

For task tracking Jira was used. After creating the initial requirements document and design document, product owner created tasks in the product backlog which would later be labeled with IDs of requirements they pertained to in the requirements document. The product backlog was updated with new tasks based on feedback from customers, remaining work, requirement coverage with task and discussions in the team on what was missing. Tasks were ordered in the product backlog based on priority. At the start of each sprint, during the sprint planning meeting 7 – 10 tasks were transferred to sprint backlog. In some sprints each member who took on a particular task would put story point estimate between 1 and 20 but that was abandoned halfway through the project because point would get assigned at random point throughout the sprint and it would not look right on the sprint burndown chart. Each sprint had a kanban board with, at first, 4 lanes TO DO, IN PROGRESS, PULL REQUEST and DONE but later BLOCKED was added to keep track of tasks that were blocked.

SCRUM related practices

Sprints lasted 2 weeks, except the first and last one.

Sprint planning meeting was scheduled on Monday at 11am UTC+1 time every 2 weeks. Product owner would discuss with the team what needed to be done and tasks from product backlog were transferred into sprint backlog. This was also a time for review and retrospective discussion. In the second week of each sprint, we had a meeting with supervisors, which was also a meeting where we would report on the status of our tasks to the rest of the team and supervisors.

At the end of each sprint, sprint retrospective and review reports were published as presentation slides.

Sprint hours were tracked in an Excel spreadsheet.

Product owner had a regular meeting with customers to report on the status of the project and gather potential new requirements and feedback.

Effort breakdown

Per Sprint breakdown for each member

Based on Excel tables used for tracking progress, Sprint 6 is an estimate because of the holidays around Christmas in which we did not track time, and some team members were off at that time.

Members	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6	Sprint 7	Total
Danko Čurlin	10	9	26	4	18	21	11	99
Giovanni Orciuolo	9	8	18	11	16	18	18	98
Paula Šalković	4	20	7	13	0	0	10	54
Fran Hruza	6	8	15	5	11	3	15	63
Shreesh Kumar Jha	15	21	36	33	23	22	18	168
Samarth Bhatia	15	20	35	33	24	21	17	165
Georgs Lukass Rozins	9	10	16	11	10	2	16	74
Total	68	96	153	110	102	87	105	721

As we can see from this table, the working hours are not evenly distributed. This might be due to a couple of factors. First hypothesis is that members of the team have different competence levels, and some might need more time to finish their tasks than others which is due to being in different university programs, different work experiences or lack thereof

and different years of university education. In addition, 7 members for a project of this scope might be on the larger side and the work could have possibly been evenly divided if the team was smaller. Another benefit of a smaller team would also have been easier communication and coordination. The number of hours could also be underestimated or inflated underestimated as there was no verification, and the tracking of hours was left to each individual team member. Also, in some sprints not all members were available due to exams and holidays.

Sprint Task Distribution Breakdown

Note: This does not correspond to actual completion, only the distribution of tasks for each sprint. Tasks also had different difficulty or effort requirements; thus, people who had easier tasks would take more of them. Also, some tasks were transferred into the following sprints, especially in Sprints 5 to 7.

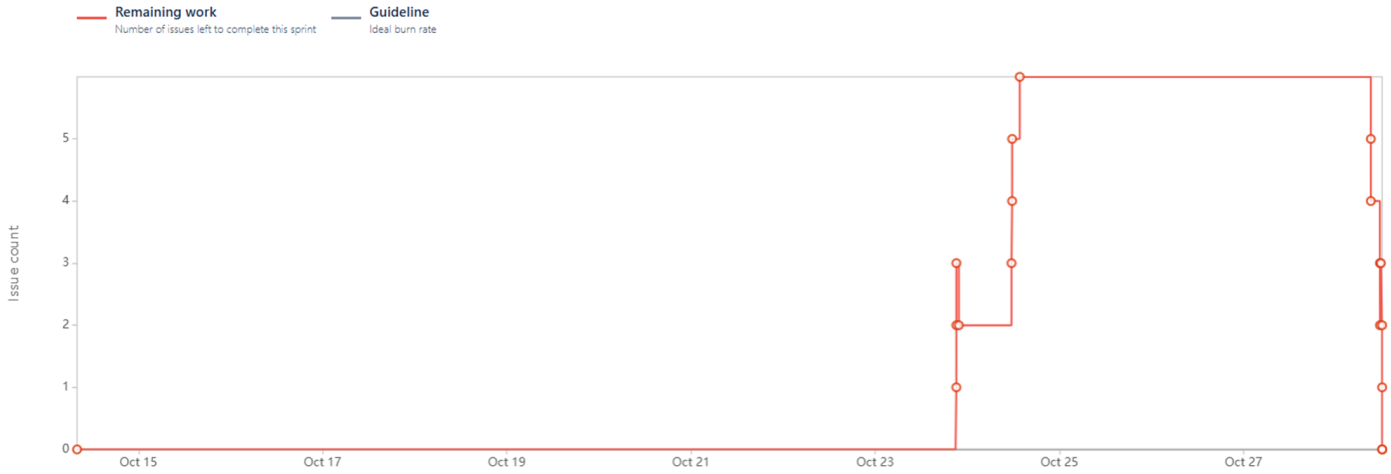
Members	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6	Sprint 7	Average
Danko Čurlin	3	1	4	3	4	5	3	3.29
Giovanni Orciuolo	1	1	2	2	4	2	0	1.71
Paula Šalković	0	1	1	3	1	1	1	1.14
Fran Hruza	1	2	3	2	4	4	2	2.57
Shreesh Kumar Jha	1	1	2	2	1	1	1	1.29
Samarth Bhatia	1	1	1	2	1	1	1	1.14
Georgs Lukass Rozins	1	1	0	1	1	2	1	1

As far as task distribution goes, it was even for the most part.

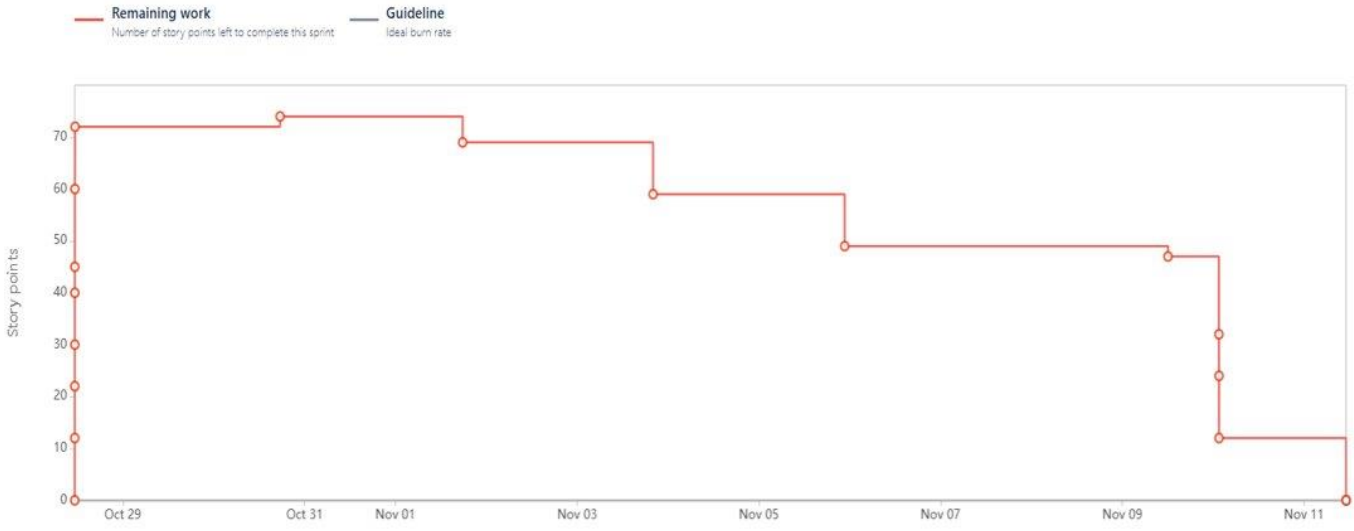
Sprint-by-Sprint burndown charts

Date - October 14th, 2024 - October 28th, 2024

Sprint goal - Requirements and design analysis and project scaffolding



Date - October 28th, 2024 - November 11th, 2024



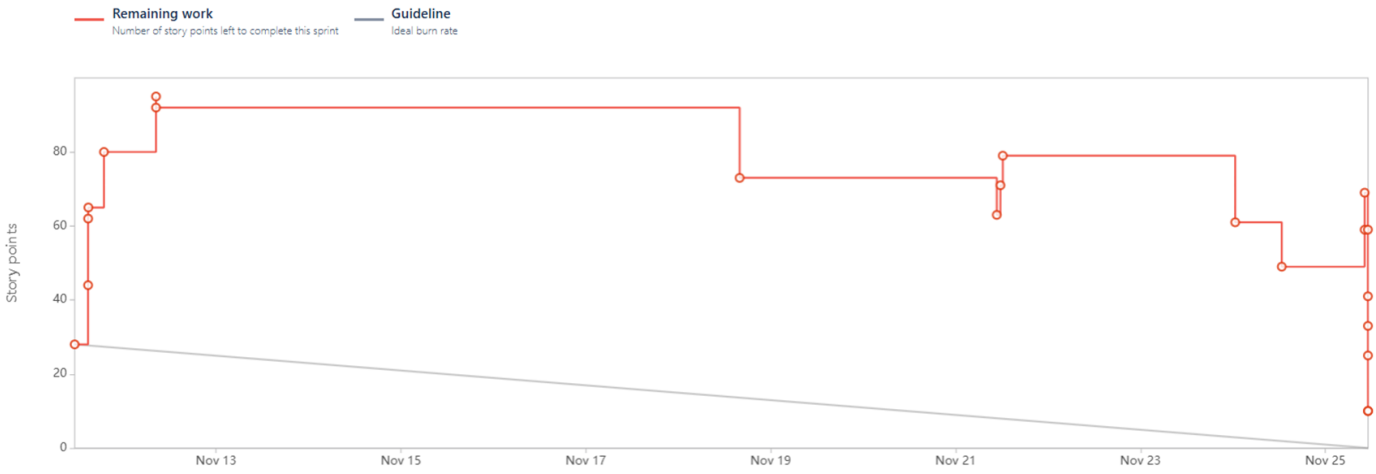
Sprint

SCRUM Sprint 3

Estimation field

Story points

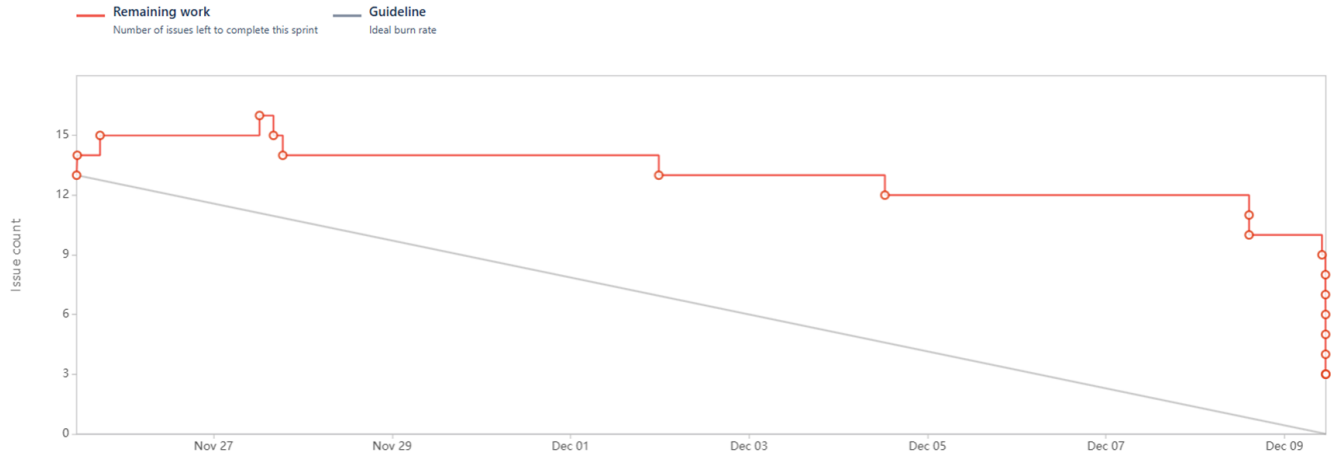
Date - November 11th, 2024 - November 25th, 2024



Sprint: SCRUM Sprint 4

Estimation field: Issue count

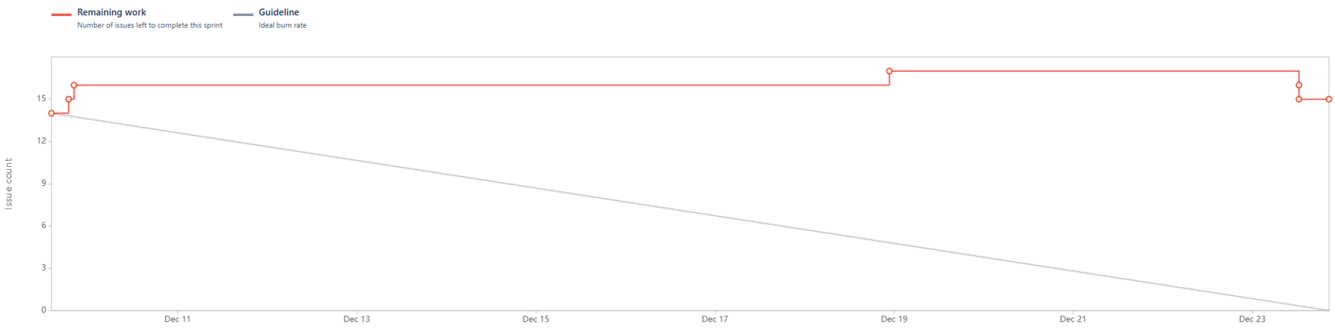
Date - November 25th, 2024 - December 9th, 2024

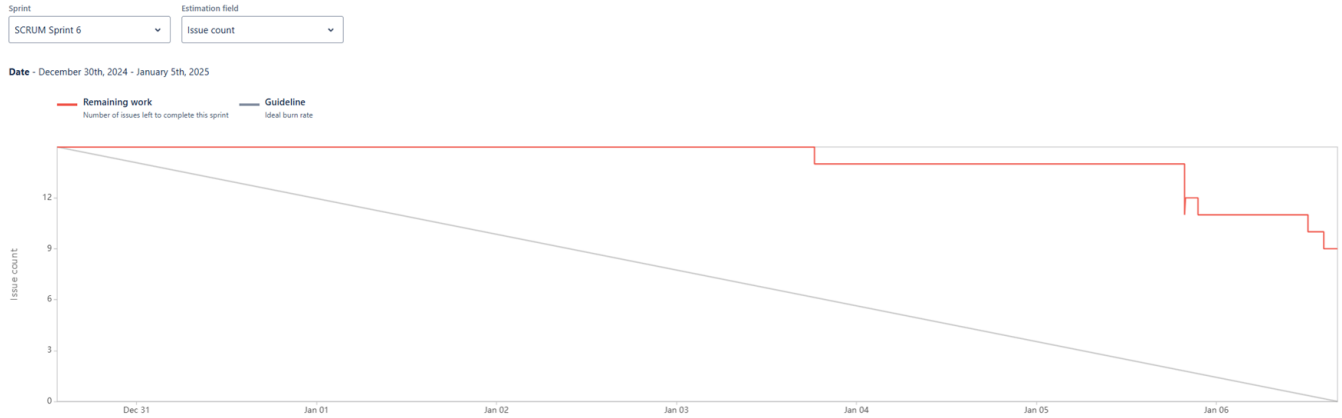


Sprint: SCRUM Sprint 5

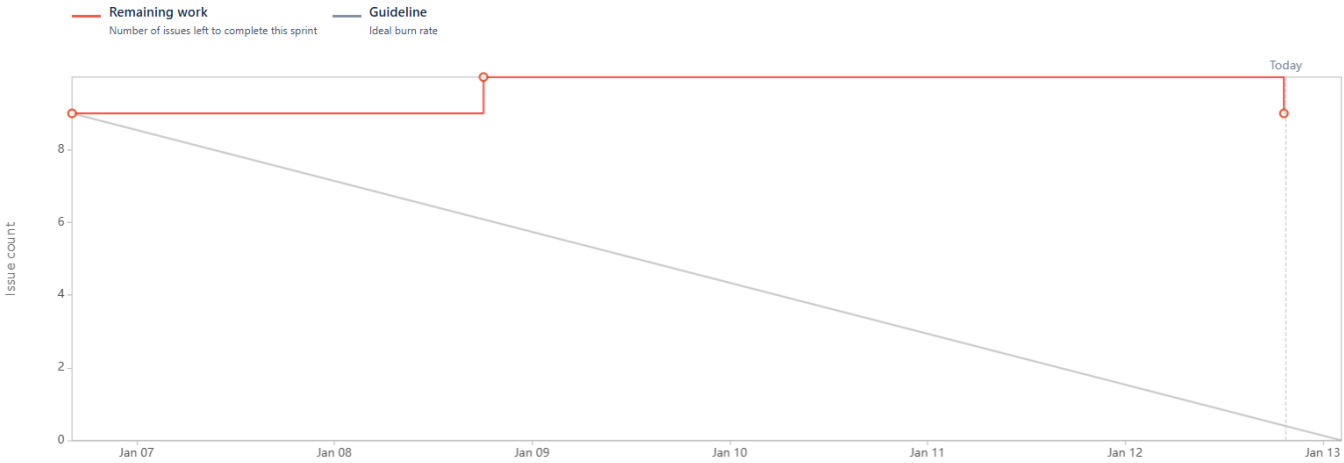
Estimation field: Issue count

Date - December 9th, 2024 - December 23rd, 2024





Date - January 6th, 2025 - January 13th, 2025



Git Stats

Contributors

Preview Give feedback

Period: All Contributions: Commits

Contributions per week to main, excluding merge commits

Commits over time

Weekly from Oct 20, 2024 to Jan 12, 2025

Plan adherence

In our initial plan we set out to develop a user interface, standalone backend API and a Python microservice that would run MITC's production optimization algorithm and connect it with the backend API. During the project however, the requirements changed from the platform being about just algorithms that optimize production sequences, and it shifted to being able to run any MITC developed algorithm regardless of its input, output, or application. In our initial plan, we stated that we would use microservices and service discovery, but because of the changes in requirements, we opted to implement manually adding these services to the platform instead. We also needed to make these services more generic since we first understood that these services would only take an Excel or .csv file as input and have the same output and the formatting of the output would be standardized. Therefore, we opted to enable specifying the type of input the service expects and made it, so the output is always a JSON file and implemented generic displaying of these JSON files.

We also did not end up using unit testing, instead replacing it with Testcointainer. Jenkins was also not used although stated in the initial plan, we opted for GitHub actions instead.

As seen in the requirement breakdown section, we did not manage to implement everything that was planned and with more time and better planning things could have been implemented by the project deadline.

Positive experiences

During the project we got to work as part of a multicultural team of developers with different ideas, ways of working and competences. Furthermore, team members had vastly different schedules, and we needed to coordinate our process with that in mind which we managed to do through asynchronous communication through Jira ticket comments and Discord messages and report. We also had a chance to learn about SCRUM process and try to adhere to it in a real project. Trying to implement the SCRUM process instead of just learning about it gave us an unfamiliar perspective into software development and what goes into an actual project, and we hopefully learned how important the process is. Some members also got to practice their presentation skills a lot through regular project updates we had. The product owner had a chance to see how the interaction with the customers worked and consequently how important it is to verify the requirements they have for the project. We also had the opportunity to work with Jira which is often used in the industry and is notorious for being complex to navigate, of course this being the choice of the team to try using Jira. Still, this choice made it possible for us to learn to navigate Jira. Another experience was writing technical documentation which often gets overlooked but we learned it plays a vital role in keeping the project going and consistent. Furthermore, it helps us understand what we are missing, where we should focus our attention and gives us

In the end we faced a lot of challenges in terms of communication, hitting milestones in a timely manner, task distribution but it was a great learning experience as we tried to deal with these challenges and accomplish all our goals at the same time.

Improvement possibilities

One of the key areas we could improve in future projects is writing documentation parallel to code, which would give us a better understanding of what is being implemented and would also save us time at the end of the project. In addition, it would allow us to keep the documentation consistent with the state of the application. Furthermore, we could have scheduled more meetings or have been more consistent on reports and checkups of members' progress. That would give us a better overview of what are the areas we need to focus on more, allocate resources or redistribute tasks. Sometimes the plans for presentation and responsible people were difficult to coordinate due to some technical limitations, like not possessing a functional way of sharing presentation slides and could

have been solved by more initiative from other members. We also should have aligned our frontend and backend development since the core functionality for the backend was completed in Sprint 4 while frontend was operational at the end of Sprint 6. That way there would have been more time for fixing bugs and integration as well as polishing the design and details of the application. As far as the SCRUM process itself is concerned, our sprint reviews and retrospectives should have been a lot more comprehensive and there should have been a dedicated meeting where we could discuss issues and improvements in real time. That way everyone would have more input, and our process would be improved quicker. In addition, the SCRUM master could have been more hands on with coordinating everything and checking up on progress more often as well as reacting to arising issues more quickly before they became harder to deal with.

References

Requirements and design analysis document v3 - https://www.fer.unizg.hr/_download/repository/RASD%20v3.pdf

Design document v3 - https://www.fer.unizg.hr/_download/repository/Design_Description_v3.pdf

Sprint reports - https://www.fer.unizg.hr/rasip/dsd/productionoptimiser/documents#%23!p_rep_162681!_-235321

Jira Reports - https://productionoptimiser.atlassian.net/jira/software/projects/SCRUM/boards/1/reports/burndown?source=overview

GitHub repository - https://github.com/production-optimiser/production-optimiser