UNIVERSITY OF GOTHENBURG

9th of January 2026

Experience Report

Inventory

Project Supervisor: Richard Berntsson Svensson

Scrum Team 3:

Lovisa Olsson - gusolsloy@student.gu.se
Vera Ek Lundqvist - guseklunve@student.gu.se
Natalija Prošić - gusprosna@student.gu.se
Sophie Vervoort - gusvervso@student.gu.se - Product Owner
Saga Rennemo - gusrennesa@student.gu.se
Sayara Salsabil - gussalssa@student.gu.se
Milica Mandic - gusmanmii@student.gu.se - Scrum Master

1. Project Context and Report Purpose

1.1 Project Background and Vision

Problems: Small-scale retailers do not usually have reliable inventory control systems, often relying on spreadsheets or even handwritten ledgers instead. This leads to inconsistent data, delayed stock replenishing and potential loss of capital due to overstocking or understocking. As a result, managers have to spend extra time to organise stock updates and control their employees' access to it.

Stakeholders: are Employees**,** since they require stock visibility, transactions and return processing and Managers**,** they require additional administrative control over the system, stock ordering and user access management.

Expected Product Value

- Support business growth with an accessible and scalable inventory management system.
- Automated role based access handling that frees up managers time.
- Offer an intuitive and easy to use interface to reduce training time.
- Reduce errors compared to manual recordkeeping by using a centralised data structure.

1.2 Aim and Scope of the Experience Report

Report Purpose

This report features an analysis of how we applied the Agile methodology [1] and Scrum framework [2] during the development of our inventory management system, how our team adapted to Scrum roles, and how effective these methodologies were for this project.

Report Scope

This report focuses on: Scrum framework implementation and execution of Scrum roles, selection, application and effectiveness of Agile practices, Sprint based project management outcomes and team adaptation strategies, Comparison of traditional and Agile project management approaches, Challenges and team learning progression.

Report Exclusion

The analysis will focus on Agile application and team evolution, therefore the details of technical implementation, system architecture specifications and technical documentation will be excluded from this document. These can be found in the user manual.

2. Agile Way of Working & Scrum Framework

2.1 Agile Values

We embedded the four core values defined in the Agile Manifesto [3] into our ways of working throughout this project.

Individuals and interactions over processes and tools: We prioritised direct communication through WhatsApp and in-person meetings over formal task management systems. Our direct communication allowed us to have valuable technical discussions and address blockers in a timely manner. Additionally, we have focused on creating an open space, both in our WhatsApp group chat, and in person meetings, where every member was welcome to suggest new ideas, ask for help or initiate a discussion.

Working software over comprehensive documentation: In every sprint, we prioritised software functionality for the weekly demo submission, and limited code documentation to code comments.

This approach enabled a fast feature-first development. However, it has also created technical debt in knowledge sharing and maintainability that had to be addressed in later sprints.

Customer collaboration over contract negotiation: Weekly TA reviews on Mondays and team sprint reviews on Thursdays created a good feedback cycle, which made it easy to keep the Product Backlog flexible and responsive to potential changing requirements.

Responding to change over following a plan: Early sprints review showed late code merge patterns. Code was merged too close to the demo deadlines which lead to bugs and last-minute fixes. In sprint 3, the team introduced a Wednesday task completion deadline and defined communication hours (09:00-17:00). This gave us a two-day buffer for additional help, testing and fixes. This change reduced Friday deployment stress and improved increment quality.

2.2 Roles and Responsibilities within the Scrum Team

2.2.1 Scrum Master and Product Owner

Our main focus in the initial project phase was to ensure that the Scrum Master and Product Owner understand their responsibilities and together establish an effective workflow. The Scrum Master led and ran Scrum events and introduced Notion as the main planning tool, which improved transparency and provided a shared overview of tasks, sprint goals and deadlines. By focusing on time-framed, pre-planned and structured meetings with clear goals, the Scrum Master supported team efficiency and a smooth sprint execution. During Sprint 1, one of the issues the team faced was unclear and undersized backlog items. The Scrum Master has identified this impediment, and by assigning a section of the sprint planning meeting to that issue, encouraged the team to refine and more closely define our backlog items. [8]

The Product Owner defined and explained the product goal and in collaboration with the developers, created and maintained user stories and product backlog items. Product Owner's impact on team work was noticeable the most in our sprint planning meetings, where she helped the team prioritize the tasks, and made sure that every team member is aware and aligned regarding the product goal. This was best reflected during sprint 1, where the Product Owner refined the backlog so that the team is starting the project by completing the basis of the system functionality tasks, and therefore, even by the end of the first sprint, delivering a base of an inventory system. [8]

Collaboration between the Scrum Master and Product Owner helped keep the clarity, focus and a shared understanding of the project across the team. Upon encountering any minor issues, where, for example, upon planning Sprint 5, the team has fulfilled all the initially planned functionalities, thanks to their close collaboration, we have quickly discovered a need for a delivery and product filtering functionalities that were successfully implemented during that week. [8]

2.2.3 Scrum Implementation and Team Collaboration

The team organized its work using the Scrum framework with one week sprints. Sprint planning was held on Mondays, sprint reviews and retrospectives on Thursdays with iterations on Fridays. Daily standups were replaced by weekly Wednesday progress check-ins and the expectation of every member to update the team upon completing the assigned task via the Whatsapp group, supported by in-person meetings and individual assistance if needed. Key scrum artifacts, including the product backlog, sprint backlog and relevant progress tracking charts were maintained and updated continuously in Notion. [2]

Decisions were made through democratic voting, with the Product Owner having final responsibility on new features and their priority. The team was always open to new ideas, such as implementing a seasonal Christmas mode in sprint 4 and a delivery feature in Sprint 5, evaluating each suggestion based on time cost and prioritizing within our project scope. Collaboration challenges

primarily involved workload balance. Over time, through individual assistance and each member's commitment, collaboration evolved into cross-functionality, where each member contributed to frontend, backend and report writing tasks, which has shown to increase the overall teamwork and flexibility. [3] [4]

2.3 Reflection on Scrum Implementation

Several Scrum practices worked quite well for the team. The introduction of Wednesday progress check-ins from Sprint 3 onward proved effective in identifying issues, especially code-related challenges early in the sprint. This gave us more time for adjustments before sprint completion. Instead of daily stand-ups our team opted for a less formal way, with set working hours (9:00 - 17:00) every weekday where each member was welcome to ask for help and update on progress. This method of communication was shown to be more frequent and of higher value, hence we continued with that approach.

Initial struggles included accurate task estimation and breaking down user stories into smaller tasks, leading to underestimated complexity and uneven workload distribution. Technical challenges related to UI and integration in Flask and the use of GitLab also affected early progress. Communication issues arose during later sprints but were addressed through open discussion and guidance from the project supervisor.

To mitigate these challenges, several Scrum practices were adapted. Sprint retrospectives were moved from Fridays to Thursdays starting in Sprint 2, allowing time for refinements, bug fixes and documentation before the deadline. Meetings were given a clear structure that was available for everyone in Notion, which improved overall focus and efficiency. Scheduling meetings in advance during retrospectives increased participation and accountability. Breaking down features into small tasks, and welcoming honesty on whether a member can handle the task, in addition to assigning a task to multiple developers improved our workload balance.

3. Agile Practices Tools, and Techniques Used

3.1 Core Agile Practices Implemented

The Agile practices we chose to implement include iterative and incremental development of the software, time-boxed sprints, frequent delivery of working increments, regular feedback and improvement, backlog refinement, mid-week check-ins, sprint planning, sprint evaluation and improvement through sprint reviews and retrospectives, making decisions together as a team, and adaptation based on customer feedback [1].

The practices listed above helped improve the adaptability of the team, collaboration, planning skills, our efficiency as a team, as well as the product and development process.

3.2 Agile Tools and Techniques Used

To implement the agile practices described in section 3.1, the team used Notion, WhatsApp, Discord and Gitlab. The Agile tools we selected to help us reach both our sprint goals and product goal were all digital and included a task board, product backlog, sprint backlog, burn-down chart, burn-up chart, velocity chart, agile earned value chart (Notion), version control system (GitLab), collaboration tools (Notion, Figma, Gitlab) and communication tools (Whatsapp, Discord).

The Agile techniques used are user stories, Scrum, task decomposition, time-boxing, weekly check-ins, sprint planning, review and retrospective techniques, and a shared definition of done.

3.3 Motivation and Criteria for Chosen Practices and Tools

We implemented iterative and incremental development together with time-boxed sprints and frequent delivery of working increments. In line with the 3rd Agile principle [4], they allowed us to, upon completing a sprint, have presentable working software, accordingly adjust our priorities, and, by fixing discovered bugs early on, eliminate the risk of resurfacing technical problems.

Backlog refinement and continuous adaptation based on the customer feedback were crucial to prioritizing features and separating them into tasks. The team has implemented this way of working, related to the 2nd Agile principle [4], by having refinement sessions during every sprint planning meeting, including the Product Owner into decisions, and splitting the user stories by workflow steps.

Sprint planning, reviews and retrospectives are based on the 12th principle. These practices were crucial as the simply documented sprints allowed us to have an organized board, go back to the documents when needed and adjust current workflow accordingly .[4].

Our mid-week check-in and use of communication tools, aligned with the 6th principle [4], were chosen based on ease of usage. WhatsApp was the main communication medium, as it allowed us to easily search within the previous messages, save any needed documents and data in the chat and be available quickly.

User stories, aligned with the 1st and 2nd agile principle[4], were distinguished and created as they helped keep the team focused, ultimately, they encouraged collaboration and were chosen as a very simple text, easily revisable when needed. Creating the user stories before the start of sprint 1 helped us get a clear overview of the project, and adapt in later sprints.

Task decomposition and time-boxing, connected with the 3rd and 8th principle, were chosen to help comply with the deadline and working budget we were provided with. A clear Definition of Done correlated with the 7th principle, was used as a way to measure our progress with the software[4]. Ultimately, it helped us reduce misunderstandings during the Sprint Review sessions.

Lastly Scrum, Gitlab and velocity charts are mainly consistent with the 4th, 7th, 11th and 12th principle where the scrum framework encourages daily communication and individual accountability within the team, where Gitlab enables collaboration regarding code tasks and velocity charts a tool to reflect on effectiveness[4]. Our motivation for choosing these are that Scrum for example is regarded as a successful framework that helps teams deliver value incrementally[2]. GitLab was chosen as we were in need of a communal platform for combining each team member's code contribution. Lastly our motivation for choosing a velocity chart to measure effectiveness was the quick overview you get.

4. Application of Agile Principles

Going into this Agile project, the team had a rough idea of what the finished product was going to look like and what its main features would be. However, some of the ideas that presented at the start were lost throughout the project, and many new ideas were also introduced along the way. What was found important by the team was to be open to changing requirements and goals. As late as in sprint five, the team started developing a new core feature that was found important, but which was earlier not even thought of [4].

For each of the sprints, the team agreed on either one big feature or multiple smaller ones that would deliver value to the final product, and therefore making sure that by the end of the sprint, the customer would receive a functional product, with an additional feature. This was also important, since it is an accessible way of measuring the progress of the project [4].

To ensure the efficiency of the work being put in, the team met up and communicated continuously throughout every week of the project to make sure everyone was always on the same page about which direction the project was heading. Not only was the joint planning crucial, but so

were the check ins and compilations of every sprints completed work. In addition to two to three face to face meetings each week, the team also kept in touch via virtual meetings and frequent online communication. We found that meeting face to face was a lot more efficient and that it was easier to keep everyone engaged. However, the option to meet up in person was not always available [4].

There are minor improvements that could be made within the team regarding efficiency reflection and diversity of thoughts. For every sprint retrospective, the team reflected on what went well and what could be improved. Efficiency was only really touched upon once, explicitly. During sprint 3, the team set up guidelines for coding and merging code along with a common coding deadline, which we set for Wednesdays. Having this mid-sprint deadline from sprint 3 onwards ensured that the core functionalities of every sprint were completed with good margins, and any contribution made by another team member was easy to follow. To improve in future, the guidelines would be set up earlier to increase efficiency and avoid do-overs. In terms of diversity of thoughts, no team members had ever worked on an agile project before, and very few on any coding project at all. Regarding most parts there was diversity of thought, which created clashes within the team. The team should have been clearer for the functionality and execution for every task to make sure each developer could complete them without consulting the rest of the team several times [4]. We have worked on this issue, by further explaining and adding both visual (using Figma) and text-based explanations of what the task is within our sprint backlog board, making it easily accessible.

There could also be improvements regarding sustainable development. The team purposely chose to handle as much of the core functionalities as possible, as early as possible. This was to make sure the final program included what the team agreed upon.

Prioritising core functionalities early on also freed up more time towards the end for post project work, e.g. the documentation of the program and fixing up any possible errors that might have occurred during the sprints. However the work pace during the first sprint could not be sustained for a longer period of time if the work balance was intended to be healthy [4].

5. Project Management

5.1 Planning and Prioritization during Sprints

Over time, the team learned to keep sprint goals concise and achievable, particularly after introducing the mid-week checkpoint, which encouraged simpler sprint goals aimed at focusing on functionality over quantity. Backlog prioritization was handled by structuring backlog items by status (“not started”, “done”, “in progress”), sprint ownership and estimated effort using story points and hours. Visual labeling and color coding improved transparency. Complex tasks were further clarified through detailed descriptions, ensuring shared understanding of scope and expectations.

Unlike upfront planning, sprint planning evolved to become more iterative, where we focused on slowly improving every sprint. Lessons from sprint reviews and retrospectives led to improved risk management through the introduction of a risk identification table and more accurate capacity planning using a workload allocation table. These adaptations increased estimation accuracy and overall sprint predictability. [1] [2] [3]

5.2 Monitor and control

The team agreed that we had to oversee work during the sprints, in order to ensure progress and delivery of the high valued increments. We used mid-point check-ins, daily WhatsApp communication and sprint backlog board statuses (“done” , “in progress”, “not started”) to keep track of task progress, remove any kind of blockers and change priorities as needed. Burndown charts were used to plot sprint

progress which eventually provided the team with the ability to instantly access where bottlenecks or product backlog items in danger of not being completed are located. [2] [5]

Changes in requirements were tackled immediately, meaning when new adjustments were identified, they were added to the backlog with clear descriptions and arrangements right away. The team used sprint reviews, Agile-earned value charts and retrospectives to estimate the impact of the changes and decide what could be implemented in the current sprint versus deferred to the future ones, ensuring flexibility without risking commitments. [3] [4]

Feedback from users and stakeholders played an important role in shaping development decisions. Reflections gathered during sprint reviews and user testing helped to perform refinement of features as well as design adjustments. Constant loop of feedback guaranteed that the delivered product aligned with user expectations and project goals [3] [4]

However, mid-sprint checkpoints emerged as an important monitoring aid offering a chance to readjust goals and balance effort across contributors as necessary. Continuous Integration (CI) and automated testing made it possible to catch bugs early on, so there were fewer last minute reworks. Visual aids and structured feedback documents with proactive adjustments of workloads allowed the team to maintain control over both timeline and scope while improving predictability of the outcome by lowering task slippage.

Lastly, closely monitoring risks that were defined during the sprint planning, has shown to be a crucial part of our project's success. With creating a risk identification board, and including a variety of mitigation strategies for the risks identified, our team has successfully managed to mitigate every risk that showed up during the project. Key to this was constant communication and team work,where members jumped in and helped with the tasks of a member who was either sick or unable to attend the meeting or complete that sprint task.

5.3 Progress Tracking and Earned Value Analysis

Throughout the project the team maintained several Agile progress metrics to track progress and identify potential improvement areas. These can be found on the project WiKi page [5].

Velocity and Estimation Accuracy: Velocity charts show a clear learning curve and initial overcommitment. Sprint 1-2 had unrealistic estimation of story points which is reflecting the teams inexperience. Sprints 3-5 show that the team became more comfortable with achievable estimations and working together. Sprints 6-8 lower velocity reflects a shift toward course documentation and refinement while also factoring in holidays. [5]

Burndown Chart Patterns: From sprint 3 the burndown charts show a more consistent task completion rate achieved by the Wednesday deadline. With the occasional Thursday and Friday task completion confirming that the additional buffer days were very beneficial to the team processes. [5]

Scope Management: Burnup chart shows the teams initial struggle with scope estimation and the need to increase it mid-project. The steep sprint 1-3 incline shows that the team was ahead of schedule due to optimistic estimates. From sprint 4, a more consistent trend line demonstrates the team adjustment and better scope understanding. [5]

Earned value Management: The earned value chart tracked the projects overall health using three key metrics: PV (Planned Value), EV (Earned Value) and AC (Actual Cost). Initially the EV is significantly higher than the PV in early sprints with the team delivering more value than planned, confirming the struggles with estimations and early front loading. However a dip and reduction of EV from sprint 4 shows the teams adjustment and learning in estimation.

The AC tracked below the EV, shows that the effort made by the team is smaller than expected and budgeted for, especially towards the later sprints. AC is compared towards the budget and not the hours actually planned each sprint, therefore it does not accommodate for holidays taken by the

developers, nor the fact that developers from time to time have been out sick or for any other reason absent. However, since the EV remained above PV throughout the whole project, it shows the team has worked efficiently, but also that the team could have achieved even more during these eight sprints.

The conclusion then states that, while the team remained efficient in effort usage, the scope planning and estimation accuracy would need improvement. [5]

5.4 Success Factors and Constraints

Since the team had no previous experience with working with Agile or in Scrum, it quickly became clear that the first two sprints hindered rather than promoted a sustainable project over time. In the beginning, the team tackled too much work, making the hours before the demo submissions stressful.

In the third sprint, the team agreed to introduce internal midweek deadlines. This gave us more time to test the system and check the quality of the software before submission.

During the sprint planning the team assessed the risks for each sprint in a risk identification table, where we evaluated how likely each risk was, how high the impact would be if the risk were to happen and what actions the team should take if the risk happened.

The risks that we found and had to mitigate were mostly about members getting sick or not being present, we could mitigate the identified risks by following the action plan that was made in the sprint plan and always having an active communication about it.

6. Agile vs Traditional Project Management

6.1 How the Project Would Differ Under a Traditional Approach

If the project had been managed using a traditional project management approach like Waterfall, then the process would have been more linear and sequential. Requirements would have been defined more widely at the beginning of the project, followed by fixed phases such as designing, implementation, testing and placement. A detailed project plan and schedule would have been established before starting the main phase, with changes controlled through formal change request procedure.

In our project, the requirements changed over time and as all the team members started getting a better understanding of problem scope description and continuous feedback from the stakeholders during sprint reviews. Using the traditional method, these changes could have been much harder and time consuming to accommodate, possibly resulting in rework or delays, scope creep. Testing and user feedback would tend to happen later in the project lifecycle, raising the chance for disappointment of the final product relative to what they envisioned. [7]

6.2 What Worked Better with Agile

Agile project management worked particularly well in our project due to its flexibility and continuous feedback sessions. Events such as sprint reviews and retrospectives facilitated improvement in terms of both the product development and the team workflow. Use of user stories helped focusing more on the value for the user rather than focusing only on technical work. Although the team didn’t execute the daily stand ups as mentioned in Scrum, the progress of the work still passed continuously through a group on Whatsapp and polls for tracking progress. The team members regularly posted updates, raised issues, and responded to questions, which ensured transparency.

These experiences are consistent with Agile literature, where customer collaboration and agility are important advantages of agile methodologies, such as Scrum and Extreme Programming.

In general, the agile approach made it possible for us to react more effectively to changing situations, engage stakeholders more efficiently and build ownership at the team level. [3] [7]

6.3 What Traditional Methods Might Have Handled Better

In spite of the advantages of Agile, there are still some areas in which traditional methodologies might have handled the project better. For instance, most traditional methodologies emphasize the importance of upfront documentation, elaborate planning, well defined roles and responsibilities. On projects with stable requirements that are well understood upfront, this will provide clarity and predictability. [7]

In our project, there were times when uncertainty about long term scope, deadlines, or deliverables created challenges. A traditional approach might have provided closer control over scope, cost as well as schedule via detailed plans and formal progress tracking. In addition, the stakeholders who prefer fixed timelines and well defined outputs, traditional methods may provide easier reporting.

6.4 Differences Between Agile and Traditional PM in Literature

Project management literature generally suggests that Agile and Traditional project management styles differ fundamentally. Traditional project management is commonly viewed as a plan driven, predictive and control paradigm, focusing on upfront requirements that stick to a fixed plan (as described in Waterfall and PMBOK models). Success is typically measured by adherence to scope, time, and cost constraints.

More specifically, while traditional project management techniques are defined as adaptive, iterative and relationship oriented, Agile methods of project management are defined in literature such as the Agile Manifesto or Scrum Guide as adaptive, iterative and relationship oriented. Agile measures in success in customer satisfaction, value delivery, and flexibility of changing requirements. [3] [7]

6.5 Reflection: Experience VS Theory

Most of our experiences from the project prove the views expressed in the literature to be correct. The advantages of the Agile approach that can be found in the literature, such as flexibility, feedback and enhanced cooperation have proved to be the same in the project.

However, our experience also contradicts the idea that Agile requires minimal planning and documentation. Although it suggests lean documentation, our study showed that a certain amount of structure, defined objectives and shared process are needed in order to avoid confusion as well as inefficiencies. This implies that there has to be a harmonious blend between flexibility provided by Agile as well as certain aspects from a traditional disciplined environment for effective project management to take place. Even though the choice of the project methodologies will depend on the complexity of the project and the stability of the project requirements.

7. Challenges, Missteps and Key Learnings

7.1 Obstacles Encountered During Development

The team faced issues with lack of communication from some team members. This meant that the status of some tasks was not always known to the rest of the team. This made it harder to keep track of the sprint progress and keep everyone on the same page.

The lack of communication also made the sprint planning more difficult since it was not clear what tasks the team could assign the members that were not present since the rest of the team did not know for sure that the assigned tasks would get finished. This led to the workload being unbalanced for some members of the team.

The check-ins were dependent on each team member taking personal responsibility and answering the messages and polls in the used communication channels to update about their progress. Even though this was working for most of the team members, it did not work for everyone.

The team handled this problem by sending messages to the addressed team members. When that did not give the desired effect, we had further discussions during TA meetings and also had a meeting with the project supervisor, as well as sending an email with our supervisor as CC.

7.2 Lessons for Future Agile Projects

In future agile projects the team members might aim to have a clearer mutual view of the final product. Since, in this case, we knew what the program should fulfill, the functionalities were clear enough for the team to work efficiently around it. However, the details and the design of the system were never communicated until the Figma board was created. To make the workflow more efficient, the team would aim to have a design structure, before going into the project, and also a more detailed structure of the basic features. Not having agreed upon something ahead never really created clashes, but it did slow down some parts of the project, from having to decide fonts and colors that everyone was happy with, to what buttons should be on certain pages. Depending on what type of project the team would come across next, there is a chance that there is an actual customer with their own requirements and wishes, which would eliminate some of these problems [6].

On top of this, and as mentioned before, the team came to the conclusion, time after time, that meeting in person, as agile principle 6 says, really is the most efficient way of communicating [4]. It could have been done more than it was during this project, but the team had to keep accommodating everyone's needs, whether it was being out of the country or outside of working hours. Since today's online communication is fairly effective, that option was kept active throughout the project, but it might have slowed down the process. The optimal way of communication would have been if everyone was always in the same place during working hours, since it would keep all communication tight and more efficient. For this project that was never possible, but for future ones, hopefully that is a prerequisite.

8. References

[1] Agile Alliance. “What is Agile?”, https://agilealliance.org/agile101/. Accessed 13 Dec. 2025.
[2] Scrum.org, “What is scrum?”, https://www.scrum.org/resources/what-scrum-module. Accessed 13 Dec. 2025.
[3] Agile Alliance. “What is the Agile Manifesto?”, https://agilealliance.org/agile101/the-agile-manifesto/. Accessed 21 Dec. 2025.
[4] Agile Alliance. “The 12 principles behind the Agile Manifesto”, https://agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/. Accessed 28 Dec. 2025.
[5] Scrum Team 3, “Project Sprint Charts and Metrics”, Store Manager Project WiKi https://git.chalmers.se/prosic/store_manager/-/wikis/home. Accessed 8 Jan. 2026.
[6] LaunchNotes. “The Comprehensive Guide to Agile Planning Onion”, https://www.launchnotes.com/blog/the-comprehensive-guide-to-agile-planning-onion. Accessed 8 Jan 2026.
[7] Atlassian, “Project management intro: Agile vs. waterfall methodologies”, https://www.atlassian.com/agile/project-management/project-management-intro. Accessed 7 Jan 2026.
[8] ScrumAlliance, “The Scrum Team Roles and Accountabilities”, https://resources.scrumalliance.org/Article/scrum-team. Accessed 28 Dec 2026