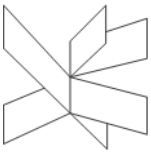


[Online Shop with Distributed Services] – Process Report

[Dawid Faruga(354886), Haoran Li(355420)]

[Jan Munch Pedersen , Troels Mortensen]

[4004]

[Software Technology and Engineering]

[Semester ,3 ,2025]

[06,12,2025]

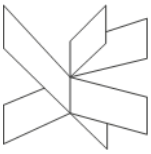
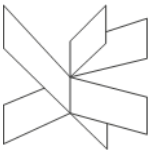


Table of content



1. Introduction

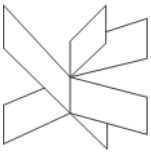
The Online Shop with Distributed Services project was developed within a Problem-Based Learning (PBL) environment and is motivated by the increasing relevance of modern e-commerce systems in both industry and academic study. Online retail platforms represent a complex technical domain involving user interaction, product management, distributed communication, payment processing, and secure data handling. These characteristics make the domain well-suited for exploring system architecture, software engineering methods, and iterative development processes.

The central motivation for choosing this technical theme lies in its combination of practical relevance and educational value. E-commerce platforms must handle real-time data, ensure reliable communication between components, and maintain a user-friendly interface — requirements that challenge students to integrate backend development, frontend design, database modelling, and distributed communication in one coherent system. The project therefore provides an opportunity to apply theoretical knowledge to a realistic scenario, while also developing critical engineering competencies such as modularity, scalability, usability, and structured planning.

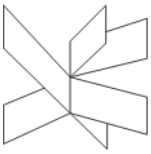
The project was carried out using a PBL approach, which emphasises self-directed learning, teamwork, and reflection. Within this framework, the group was responsible for analysing the problem domain, defining the system boundaries, selecting appropriate technologies, and organising the development workflow. The PBL environment also encouraged continuous evaluation of decisions, fostering a deeper understanding of development methodologies such as the Unified Process (UP) and Scrum.

This introduction is based on factual documentation gathered throughout the project, including logbooks and meeting minutes. These sources provide an evidence-based overview of the project's progress, from requirement clarification and modelling to implementation, integration, and testing. Logbook entries show how the work followed the UP phases — Inception, Elaboration, Construction, and Transition — while Scrum was applied during the Construction phase to support iterative progress, daily coordination, and incremental delivery. Meeting minutes further document team communication, decision-making processes, division of responsibilities, and the resolution of challenges throughout the project lifecycle.

Overall, the combination of a relevant technical theme and the PBL learning model created a rich environment for applying and strengthening software engineering practices. The introduction of distributed communication between services, the integration of a web-based frontend, and the modelling of a complete order-payment



workflow provided valuable insight into real-world system development while remaining manageable within the project's academic constraints.



2. Group Work

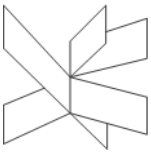
The project group consisted of two members, **Li** and **Dawid**, each contributing distinct competencies shaped by their personal profiles, academic strengths, and cultural backgrounds. The division of responsibilities and the collaboration model followed principles of PBL, enabling both members to take active and independent roles in the development of the Online Shop with Distributed Services system.

2.1 Personal Profiles and Cultural Backgrounds

I (Li) bring strengths in structured thinking, clear communication, and attention to detail in documentation. I naturally take a systematic approach when organizing project information, defining requirements, and maintaining consistency across different parts of the system. These strengths led me to take responsibility for writing the project documentation, preparing architectural descriptions, and developing the web frontend. I tend to view tasks from a holistic perspective, ensuring that the project maintains a clear direction and that each phase produces well-defined and coherent outputs.

I (Dawid) bring strong technical capabilities and a focus on practical implementation. I am effective at understanding system logic, solving complex programming challenges, and transforming design concepts into functional backend components. My responsibilities included implementing the Java and C# services, handling distributed communication, integrating the PostgreSQL database, and ensuring the overall reliability of the system. I approach problems with a solution-oriented mindset and take the lead in debugging, testing, and maintaining backend functionality.

The complementary strengths of both members create a balanced team dynamic, with one focusing on organization, documentation, and frontend development, and the other emphasizing backend implementation, system integration, and



technical robustness. This combination significantly contributed to the progress and success of the project.

2.2 Contribution to the Project

The contributions were divided as follows:

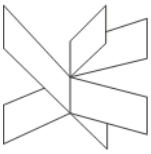
Li

- Lead author of all major documents, including the Project Report, Process Report, architecture descriptions, and diagrams.
- Responsible for the development of the **web frontend**, ensuring usability and alignment with system requirements.
- Assisted in system testing, validation, and preparing the final deliverables.
- Coordinated meetings, organized schedules, and ensured effective communication within the team.

Dawid

- Developed the **core backend system**, including services, distributed communication logic, API integration, and database connectivity.
- Implemented key functionalities across the Java and C# services.
- Responsible for software integration, debugging, and ensuring system reliability.
- Prepared technical explanations and supported architectural decision-making.

The combined contributions resulted in a complete system where documentation, frontend, backend, and distributed components were successfully integrated.



2.3 Team Dynamics and Collaboration

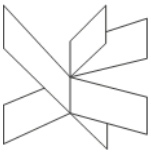
Given the small team size, communication was direct and frequent, enabling rapid decision-making and efficient task allocation. The team maintained a cooperative and supportive dynamic, where each member respected the other's expertise and responsibilities. Regular meetings and shared planning sessions ensured alignment of goals and progress.

The team demonstrated characteristics of a **complementary team**, where members possess different but synergistic skill sets. Li took on the role of **Co-ordinator and Completer-Finisher**, ensuring high-quality documentation and organization, while Dawid adopted roles akin to **Implementer and Specialist**, focusing on building and maintaining the technical system.

2.4 Conflict Resolution and Team Roles(Cosy Team)

Although no major conflicts occurred during the project, minor disagreements related to prioritization and technical choices were resolved through discussion and evaluation of alternatives. The team relied on objective reasoning, system requirements, and feasibility to reach consensus. This demonstrated the ability to handle professional disagreements constructively.

Both members actively fulfilled their responsibilities, **but a little bit social loafing was observed**. Because our group has experienced changes in personnel twice and ended up with only the two of us, our overall project progress has been a bit slow, and our communication and interaction are not as good as other groups. But tasks were transparently divided and tracked, and each member remained accountable for their assigned work. Progress was consistently documented in logbooks and meeting minutes, which also reflected moments where members assisted each other across disciplinary boundaries.



Team Roles: Cosy team！

We are a very laid-back team and don’t impose strict requirements on each other. As long as the project results are ensured, we emphasize a relaxed and enjoyable atmosphere. We value our relationships and maintain team friendships. In terms of project efficiency, compared to other groups, we are relatively slower.

2.5 Professional and Interdisciplinary Collaboration

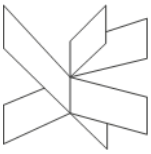
Throughout the project, both group members demonstrated the ability to participate independently and effectively in professional collaboration. Li combined technical knowledge with communication and organizational skills to bridge documentation and system requirements. Dawid contributed domain-specific expertise in backend development and system integration, enabling efficient interdisciplinary cooperation.

The team’s ability to work across different competencies — documentation, frontend development, backend development, distributed architecture, and database integration — illustrates strong interdisciplinary collaboration, a key learning outcome of PBL-based engineering education.

2.6 Team Collaboration within the Unified Process (UP)

The UP framework, with its phase-based structure, provided a clear foundation for how collaboration unfolded across the project lifecycle.

Inception Phase



- Li led the preparation of documentation, the initial problem analysis, and early requirement descriptions.
- Dawid contributed technical insights regarding feasibility and architecture.
  Together, we established the project scope and the high-level structure of the system.

Elaboration Phase

- Li worked on refining use cases, system descriptions, and preliminary UI ideas.
- Dawid designed the distributed architecture, validated communication methods, and prepared technical prototypes.
  This phase allowed the group to converge on concrete design and technology choices.

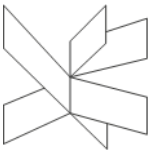
Construction Phase

- Li developed the web frontend and refined system documentation.
- Dawid implemented backend logic, Java/C# communication, and database operations.
  We collaborated continuously during integration to ensure the system functioned as a unified whole.

Transition Phase

- Li finalized documentation and system explanations.
- Dawid completed testing, performance adjustments, and deployment-related tasks.
  This phase ensured the system was stable and ready for delivery.

UP helped structure the collaboration by assigning clear objectives to each phase, guiding both task distribution and expectations.



2.7 Scrum-Based Collaboration and Iterative Workflow

Although the team was small, Scrum principles supported a flexible and iterative workflow.

Sprint Planning

At the beginning of each iteration, we identified priority tasks such as:

- Frontend development (Li)
- Socket implementation or REST API integration (Dawid)
- Database updates (shared responsibilities)

Daily-Style Check-ins

Through frequent short discussions — often informal — we aligned on:

- What was completed
- What was planned next
- Any challenges encountered

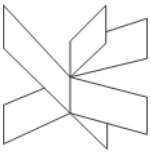
This kept the workflow transparent and prevented task delays.

Sprint Reviews

After each functional component was completed, we validated:

- Whether the frontend interacted correctly with backend services
- Whether logs, API calls, and internal communication behaved as expected

Retrospectives

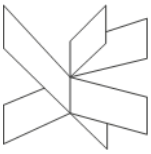


Whenever difficulties arose, we reflected on:

- Whether the technical approach needed adjustment
- Whether our division of tasks remained efficient
- How to avoid repeated blockers

This allowed us to refine our collaboration continuously.

Scrum’s iterative nature helped maintain momentum and adaptability throughout the project.



3. Project Initiation

The initiation phase of the project focused on identifying a suitable problem domain, defining a clear problem statement, establishing the project scope, and preparing a realistic time schedule. These foundational activities shaped the direction of the entire development process and ensured that the project aligned with both academic expectations and the learning objectives of the PBL framework.

3.1 Problem Domain Choice

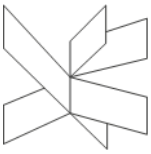
The group selected the domain of online shopping systems due to its strong relevance to modern software engineering and its ability to illustrate multiple architectural and technical concepts in a cohesive environment. E-commerce platforms involve a range of real-world functionalities, such as user interaction, product browsing, data management, distributed service communication, and secure transaction handling.

Choosing this domain provided a meaningful context in which to explore backend development, frontend interfaces, database integration, distributed communication mechanisms, and system scalability. Additionally, the domain allowed the group to incorporate a web-based interface, enabling a more tangible demonstration of system behavior.

The decision was also influenced by the desire to work on a domain complex enough to require thoughtful architectural planning, while still being feasible to implement within the project timeframe.

3.2 Problem Statement

The problem addressed in this project is how to design and implement an online shopping application that integrates multiple backend services using different



technologies while ensuring reliable communication, proper data handling, and a functional user interface.

More specifically, the project investigates:

- How a system can combine a **web frontend**, a **Java-based service**, and a **C# service** in a distributed architecture.
- How communication between heterogeneous services can be established using protocols such as REST and sockets.
- How persistent data can be stored and accessed reliably through a database system.
- How such a system can operate cohesively while maintaining maintainability, modularity, and usability.

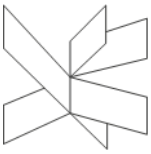
This problem statement guided the design of the architecture and the decisions regarding technology selection.

3.3 Time Schedule and Planning

During the initiation phase, the project group prepared an initial time schedule inspired by both the Unified Process (UP) and iterative principles used in Scrum.

The planning included:

- **Inception Phase:** Defining the problem domain, project scope, use cases, and high-level architecture.
- **Elaboration Phase:** Refining requirements, selecting technologies, and establishing the system structure.



- **Construction Phase:** Implementing services, backend logic, frontend components, and the distributed communication.
- **Transition Phase:** System testing, integration, bug fixing, documentation, and final delivery.

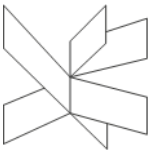
Meeting minutes and logbook entries document how tasks were distributed across these stages and how the schedule was adjusted when technical or organizational challenges arose. While some deviations from the initial plan occurred, the iterative approach made the schedule flexible and adaptable.

3.4 Ethical Considerations During Project Initiation

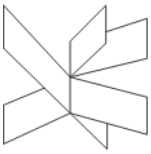
Although the system does not involve actual financial transactions or real user data, ethical considerations were part of the initiation phase.

Relevant aspects included:

- **Data responsibility:** Designing the system to avoid storing sensitive information unnecessarily and ensuring that any test data used did not contain personal details.
- **Security awareness:** Acknowledging that online shopping applications must protect user information, prevent unauthorized access, and safeguard interactions between distributed services.
- **User experience:** Considering accessibility, clarity, and fairness in the design of the frontend interface.
- **Transparency in system behavior:** Ensuring that system logging, internal monitoring, and distributed communication follow responsible engineering practices.



While the academic environment limits the ethical impact, integrating ethical reflection early in the project helped ensure that design decisions were aligned with professional standards.



4. Project Execution

The execution of the project followed an iterative and structured approach shaped by the Unified Process (UP) and supplemented by Scrum practices. Throughout the development, the team continuously reflected on theoretical concepts, methodological choices, and practical implementation outcomes. This section describes how the project progressed and how the chosen methods supported the delivery of the final system.

- **4.1 Execution of the Unified Process**

Inception Phase

During the inception phase, the group clarified the problem domain, identified the system scope, and established the major functional requirements of the online shop. This stage helped align expectations and provided a foundation for later design decisions.
Reflection: The UP's early emphasis on defining goals proved helpful in preventing scope creep and ensuring a realistic development plan.

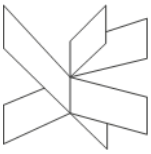
Elaboration Phase

In this phase, architectural risks were explored through prototypes. The team validated:

- communication between Java and C# services,
- database connectivity,
- frontend integration plans.

Reflection: The elaboration phase highlighted the importance of addressing technical uncertainty early. By testing critical components before full development, the team gained confidence in technology choices and reduced integration risks.

Construction Phase



This phase included the majority of implementation:

- Java and C# distributed services
- Database integration
- Frontend development
- API communication
- Error handling and logging

The team alternated between implementation, integration, and testing cycles.
Reflection: The iterative nature of the construction phase demonstrated the practical benefits of evolving the system through increments rather than attempting full implementation at once.

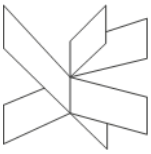
Transition Phase

The final phase focused on:

- Bug fixes
- Integration testing
- Documentation
- Final system adjustments

Reflection: Transition activities reinforced the importance of evaluation and system stability, especially in distributed systems where multiple components must coordinate effectively.

- **4.2 Scrum-Based Iterative Progress**



While UP structured the phases, Scrum informed the team's week-to-week workflow.

Task Planning

At the start of each iteration, tasks were prioritized based on system dependencies. Reflection: Scrum planning made the workload manageable and enhanced clarity regarding weekly objectives.

Short Daily Alignments

Frequent communication through short check-ins allowed the group to quickly identify blockers and redistribute tasks when necessary.
Reflection: This practice strengthened the team's adaptability and supported continuous progress.

Incremental Reviews

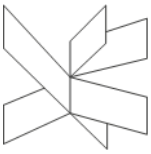
Each completed module—such as user login, product display, or distributed service communication—was reviewed before integration.
Reflection: These reviews improved quality control and ensured each part met functional expectations before moving to the next stage.

Retrospectives

After encountering technical or coordination challenges, the group reflected on how to improve workflow efficiency.
Reflection: This iterative feedback loop enhanced the team's ability to learn from experience, a key goal of PBL.

- **4.3 Reflections on Methods**

The combined use of UP and Scrum provided a balanced and effective structure:



- **UP** offered clear guidance in defining and refining system requirements, mitigating risks, and structuring the lifecycle of the project.
- **Scrum** contributed flexibility, adaptability, and responsiveness to challenges encountered during development.

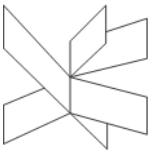
Reflection: The team learned that no single method is universally applicable, and hybrid approaches are often necessary in real-world software engineering. The project reinforced the value of choosing methods based on project complexity, team size, and time constraints.

- **4.4 Project Results and Reflections**

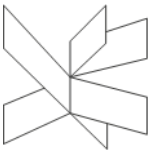
The project successfully delivered a functioning online shop system with a distributed backend composed of Java and C# services. Key project outcomes include:

- A working frontend that interacts with backend services
- Reliable communication between heterogeneous services
- Correct database operations
- Modular architecture aligned with best practices
- Documentation supporting system understanding and maintenance

Reflection:
The results demonstrate the team’s ability to apply theoretical knowledge—such as distributed system principles, communication protocols, and software architecture—to a practical implementation. Challenges in integration highlighted the complexity of distributed systems, but also emphasized the importance of prototyping, testing, and iterative refinement.



The project execution ultimately strengthened the group's understanding of how theory, method, and practice interact in software engineering. The experience showed that practical development often requires adjusting theoretical models to fit real-world constraints, which is a crucial learning outcome of PBL.



5. Personal Reflections

5.1 Li — Personal Reflection

In this project, I was primarily responsible for documentation, system description, web frontend development, and coordinating the structuring of project materials. Through these responsibilities, I gained a deeper understanding of how theoretical methods, collaborative processes, and practical development interact in software engineering.

Working with the Unified Process (UP) helped me recognize the importance of early-phase activities. During the Inception and Elaboration phases, I organized requirements, structured system descriptions, and prepared the documentation framework. These tasks demonstrated the value of UP's emphasis on clear goals, risk identification, and progressive refinement. Reflecting on this, I realized that my strengths in planning and structuring were essential for shaping the project foundation.

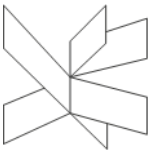
The integration of Scrum enhanced my ability to organize my own learning and adapt to changes. Through iterative planning, weekly alignments, and reviews, I learned how to adjust priorities, manage tasks effectively, and maintain steady progress in a dynamic environment. This experience aligns closely with PBL principles, where self-directed learning and continuous reflection play a central role.

On the technical side, developing the web frontend and integrating it with backend services strengthened my understanding of API communication, user-interface design, and system interaction. Collaboration with Dawid also helped me expand my perspective beyond frontend development, as I gained insights into how distributed backend components operate and interact.

5.2 Dawid — Personal Reflection

My main responsibilities in this project were implementing the Java and C# backend services, managing distributed communication, connecting the database, and conducting system integration and debugging. These tasks contributed significantly to my understanding of system architecture, development workflows, and the practical challenges of distributed systems.

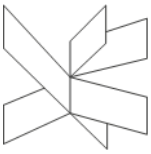
The Unified Process (UP) provided a useful structure for managing complexity. During the Elaboration phase, I conducted technical prototyping to evaluate



communication mechanisms and database integrations. This experience demonstrated the effectiveness of UP's risk-driven approach, where potential issues are addressed early through targeted exploration. Reflecting on this, I realized that establishing technical feasibility before full development greatly reduces integration problems later.

Scrum supported my work by creating a consistent development rhythm. Through iterative tasks, regular check-ins, and reviews, I learned how to break down large development goals into manageable increments. The collaborative discussions during reviews helped maintain alignment between frontend and backend development and reinforced the importance of communication in a distributed system project.

Technically, this project significantly improved my understanding of heterogeneous systems, multi-language development, REST and Socket communication, and database integration. Building services in both Java and C# required me to adopt a more holistic view of system design rather than focusing solely on one language or framework.



6. Reflect on Supervision

This chapter reflects on how to establish a productive relationship with the supervisor and how the supervision process influenced the development of our project. It also outlines what we plan to do differently in future projects to make better use of supervision and how our engagement in the supervision process can be documented.

6.1 Establishing a Successful Supervisor–Student Relationship

From a student perspective, a successful supervision relationship requires clear communication, openness to feedback, and consistent preparation before meetings. Throughout this project, we aimed to approach supervision with professionalism by:

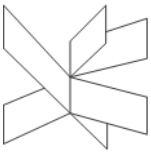
Try to keep up with the advisor's pace and communicate with them promptly if you encounter any difficulties.

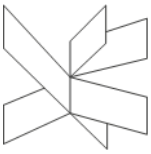
6.3 Future Improvements in Using Supervision

Reflecting on this experience, there are several actions we will apply in future projects to benefit more from supervision:

Do more of:

- **Keep track of meetings with your advisor and attend every meeting without absence.**
- **If you have any questions, communicate more with your advisor.**





7. Conclusion

This chapter summarises the group's key reflections on effective teamwork throughout the project and provides recommendations for future collaborative work. It also highlights how the project demonstrates our ability to manage complex and development-oriented situations in academic and professional contexts.

7.1 What to Do and Not to Do in Group Work

(1) What to Do

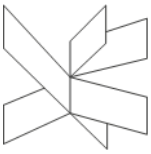
1. Maintain open and continuous communication
Regular updates on progress, challenges, and expectations help avoid misunderstandings and ensure alignment throughout the project.

2. Define clear roles while remaining flexible
Although our two-person group benefited from distinct task boundaries, we also adjusted our responsibilities when necessary to avoid bottlenecks.

3. Invest time in early planning and modelling
Conducting thorough requirement discussions, modelling, and architecture planning early on significantly reduced later rework and improved clarity.

4. Actively seek help and feedback — especially through constructive interaction with the supervisor
Engaging with the supervisor by presenting draft work, asking focused questions, and being open to critique helped us correct mistakes early and refine our approach.
A professional and proactive attitude toward supervision is essential for keeping the project on track.

5. Document decisions, meetings, and action points
Meeting minutes and logbooks improved transparency and accountability, allowing us to track progress and follow through on commitments.



(2) What Not to Do

1. Do not overlook early requirement clarification
Skipping or rushing this stage often leads to misunderstandings and misalignment later in the project.

2. Do not postpone essential tasks until the final stages
In a two-person team, delays can quickly accumulate and impact the entire schedule.

3. Do not avoid addressing disagreements
Open, fact-based discussion is more effective and prevents unresolved issues from hindering progress.

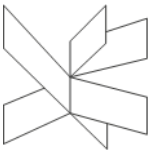
4. Do not rely on only one team member to handle critical tasks
Balanced workload distribution prevents burnout and reduces the risk of technical or scheduling problems.

5. Do not neglect documentation or updates to models and diagrams
Outdated documents lead to confusion during integration and testing.

7.2 Recommendations for Future Group Work

Based on our experience, we propose the following recommendations:

1. Establish a shared vision and clearly defined project scope at the beginning.
2. Use visual tools (UML, prototypes, Kanban boards) to support communication.
3. Adjust priorities dynamically according to Sprint outcomes.
4. Conduct systematic reflections and Retrospectives at each phase.



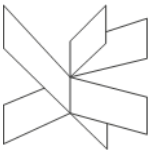
5. Keep documentation, code, and design artefacts synchronised.
6. Increase communication frequency during high-pressure periods.
7. Integrate continuous risk identification and management.
8. Maintain proactive and constructive communication with the supervisor.

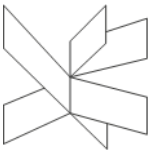
7.3 Demonstrating the Ability to Manage Complex and Development-Oriented Tasks

Throughout this project, we demonstrated key competences required to manage complex tasks in study and work contexts:

- **Structured planning:** Using UP phases and a detailed time schedule allowed us to break a complex project into manageable activities.
- **Adaptability:** Scrum's iterative approach enabled continuous improvement and responsiveness to emerging challenges.
- **Interdisciplinary collaboration:** Coordinating frontend and backend roles mirrored real-world teamwork dynamics.
- **Constructive engagement in supervision:** Implementing supervisory feedback strengthened both the system design and the written documentation.
- **Reflective practice:** Through logbooks, meeting minutes, and Sprint Retrospectives, we continuously analysed our work and improved our methods.

These abilities show that we can navigate uncertainty, collaborate effectively, and apply professional practices in development-oriented situations.





8. References

1. Astah. (n.d.). *Astah UML modeling tool documentation*. https://astah.net
2. Figma. (n.d.). *Figma design tool documentation*. https://www.figma.com/resources
3. Fowler, M. (2004). *UML distilled: A brief guide to the standard object modeling language* (3rd ed.). Addison-Wesley.
4. GitHub. (n.d.). *Git version control documentation*. https://docs.github.com
5. Kolb, D. A. (1984). *Experiential learning: Experience as the source of learning and development*. Prentice Hall.
6. MySQL. (n.d.). *MySQL documentation*. https://dev.mysql.com/doc
7. React. (n.d.). *React documentation*. https://react.dev
8. Schunk, D. H., & Zimmerman, B. J. (Eds.). (2012). *Self-regulated learning: Theories, measures, and outcomes*. Springer.
9. Schwaber, K., & Sutherland, J. (2020). *The Scrum guide: The definitive guide to Scrum – The rules of the game*. Scrum.org. https://scrumguides.org
10. Selenium. (n.d.). *Selenium documentation*. https://www.selenium.dev
11. Sommerville, I. (2016). *Software engineering* (10th ed.). Pearson.
12. Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes*. Harvard University Press.
13. JUnit. (n.d.). *JUnit testing framework documentation*. https://junit.org