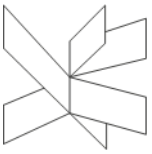


HireFire – Process Report

Hamsa Abdullah Sheikhdon (354974)

Jakub Maciej Bączek (354814)

Tymoteusz Krzysztof Żydkiewicz (355413)

Caranfil Cristian (331164)

Damian Michał Choina - (354789)

Supervisors:

Jakob Trigger Knop

Joseph Chukwudi Okika

Number of characters: 61238

Software Technology Engineering

3rd Semester

19.12.2025

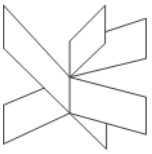
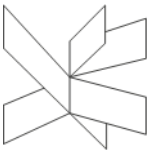
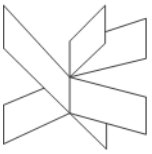
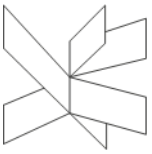


Table of content







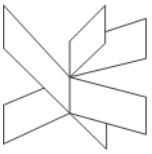
1. Introduction

This process report documents the development process and learning experience of our SEP3 group during the creation of HireFIre, a recruitment platform developed during the 3rd semester of the Software Technology Engineering programme at VIA University College. The project was focused on designing and implementing a distributed system that includes job searching, recruitment and communication between an applicant and a recruiter. The topic of the project was chosen based on the relevance to the student environment in which we activate daily and the ability to challenge our technical skills and the ability to work effectively in a problem-based learning environment.

The project was developed in a newly formed group, created based on the previous semester experience, classmates' interaction and strong values such as responsibility and open communication. From the start the group agreed on maintaining an environment based on cooperation, mutual support and constructive working sessions. These aspects had a high influence on the distribution of tasks, roles and the way decisions were made during the semester.

It was decided to follow two methodologies during the project work. The first one was Unified Process, which represented an overall project framework that provided a clear structure through its phases. The second methodology used was Kanban, which offered flexibility and continuous progress in day-to-day development. Work was carried out iteratively, with progress documented through Kanban boards, meetings and ongoing reflection on both technical and organizational decisions.

The purpose of this process report is to reflect on the project process, group collaboration, applied methods and individual learning outcomes during the semester. Compared to the project report, which focuses on the technical solution and results, this report emphasizes how the work was carried out, how the group functioned and what was learned in this period.



2. Group Work

2.1 Group Description

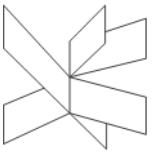
Our group this semester consists of 5 people from 3 different countries, Hamsa from the United Kingdom, Cristian from Moldova and 3 guys Jakub, Damian and Tymoteusz from Poland. Tymoteusz and Jakub have been together throughout all the semester projects since SEP1. In the second semester (SEP2), they successfully integrated Damian into the team. The positive team dynamics led them to continue their collaboration for SEP3, this time adding other new members Hamsa and Cristian. The group formation process was fast, given the fact that it was already our third semester, and we agreed on the group composition even before the semester started. Hamsa's and Cristian's previous group dissolved because its members left the university; since we had interacted with them before and got along well, we proposed collaboration, leaving it to the project process to determine if the updated team formation would be a good match after all.

Cultural map

Our group's cultural composition combines members from Poland, Moldova, and the United Kingdom. To better understand how these backgrounds may influence our cooperation, we created a cultural map comparing the three countries across eight cultural dimensions. The map highlights differences especially in feedback, disagreement, and scheduling, helping us anticipate potential friction points and align our working practices early in the project. (Meyer, 2016, ch. 1-3)

Figure 1 Cultural Mapping

The cultural mapping study focuses on our group's communication, leadership, decision-making, trust-building, and conflict-handling characteristics, as well as those typically associated with Poland, Moldova, and the United Kingdom. It indicates that all three cultures are positioned rather toward low-context communication, with the United Kingdom and Moldova slightly higher than Poland. The map also suggests differences in feedback and disagreement styles, where Poland tends to be more indirect and avoidant, while the United Kingdom is more direct and more comfortable with open confrontation. In addition, the leadership and decision-making dimensions lean moderately toward hierarchical and top-down tendencies across the compared cultures, with our group broadly aligning with this direction. (Meyer, 2016, ch.1-3)



Our group's cultural diversity brings both advantages and disadvantages. The different approaches support broader problem-solving and encourage a wider range of ideas, while also improving our ability to listen and adapt to different communication preferences. At the same time, the contrast between more direct and more indirect feedback styles can create misunderstandings if expectations are not aligned. The scheduling dimension stands out as a potential friction point as well, since our group's working rhythm is noticeably more flexible than the national profiles, making it important to agree on clear milestones and follow-up routines.

Our team has made it a priority to identify and openly discuss cultural preferences to prevent and reduce conflicts. By actively adjusting how we communicate feedback, how we raise concerns, and how we plan deadlines, we aim to maintain an effective and cooperative working environment. This approach supports stronger collaboration within the group and contributes to more consistent progress throughout the project.

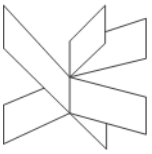
16 Personalities Traits

For analysing, describing and in general a better understanding of our group behaviour and strategic choices, each of the team members took the "16 personality traits" test. We will start by describing each member's results and personality type. (Free Personality Test, n.d.)

Personal profiles impact on the group

Hamsa

Hamsa, as a Commander type (ENTJ-A), naturally tended to bring structure into the team's discussions and steer them toward clear outcomes. His slightly higher extraversion, together with his thinking and judging preferences, supported a direct, goal-focused approach, especially when the team needed alignment and steady progress.



Considering his assertive profile, Hamsa also remained confident under pressure, which helped the group stay focused and keep priorities clear during implementation.

Jakub

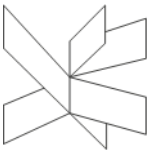
Jakub's Entrepreneur profile (ESTP-A) was reflected in his energetic presence and practical, action-oriented way of working. He often focused on what could be done immediately, which helped the team resolve issues fast and maintain momentum throughout the development process.

His prospecting preference also made him comfortable with adjusting plans when needed, which proved useful during phases where flexibility and quick iteration were required.

Cristian

Cristian, as a Logistician (ISTJ-A), contributed through a calm and systematic working style. With strong thinking and judging preferences, he supported the team by being reliable, detail-oriented, and focused on correctness, which had a positive impact on overall quality.

His assertive identity meant that once expectations were clear, he tended to work confidently and consistently, helping ensure tasks were completed in a stable and well-organised way.



Damian

Damian's Logistician personality (ISTJ-T) was visible in his reserved and careful approach to work. His introversion and observant preference supported precise execution and a strong focus on concrete requirements, which made his contributions thorough and dependable.

At the same time, his turbulent identity suggests higher self-criticism, which often led to additional checking and strong responsibility for outcomes, especially valuable in tasks where accuracy mattered.

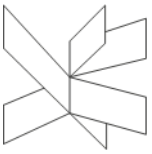
Tymoteusz

Tymoteusz, as a Campaigner type (ENFP-T), brought openness and energy to the team dynamic. His extraverted and intuitive traits supported idea generation and active discussion, while his feeling preference made him attentive to the group atmosphere and cooperation.

His strong prospecting preference also meant he was highly flexible and willing to explore alternatives, which supported the team when adapting solutions and responding to changes during the project.

Personalities impact on the group

After reviewing the personality profiles, it became apparent that our team represents a well-balanced mix of complementary traits, rather than one dominant pattern. While Cristian and Damian, both Logisticians, approached the work in a similarly structured and detail-oriented manner, the remaining members introduced contrasts that helped stabilize the overall dynamics. Hamsa's Commander profile supported direction-setting and decision-making, Jakub's Entrepreneur traits contributed speed and a practical



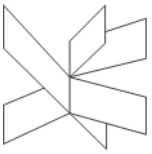
focus on implementation, and Tymoteusz's Campaigner personality strengthened communication and the general team atmosphere. Although minor misunderstandings occurred at the beginning, primarily due to differences in working pace and feedback styles, these were gradually resolved as we established a clearer routine and shared expectations

Due to this diversity, we were able to build an efficient and comfortable working environment. The more extraverted members helped sustain engagement during meetings and encouraged active participation, while the more introverted and structured personalities ensured consistency, quality control, and careful execution. As a result, the group found it easier to generate ideas while still delivering reliable outcomes.

In terms of organising our work this semester, we decided to use Kanban rather than SCRUM because it gives us more flexibility. During dividing tasks, we considered personality profiles and matched responsibilities to individual strengths and preferred working styles. This made the workflow clearer and improved overall efficiency, as each team member could contribute to where they performed best, while still supporting others whenever needed.

3. Project Initiation

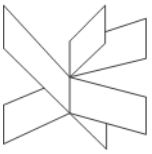
The project initiation goal was to set up our group and come up with a project proposal. Knowing that the third semester would be harder than the previous ones, it was decided to create a group contract in order to establish all the details needed to work well together. In our contract, we added the goals that needed to be accomplished during the semester and a list of rules that had to be followed by each member of the group in order to mitigate possible future conflicts and maintain a professional environment. We included details about meetings and sanctions in case someone did not confirm the plan. We ended up with a flexible schedule that took into account each member's activities. The contract also included the tools that we knew for sure we would use. In this way, every member was informed from the start and could set up



their workspace in advance. Every member signed the contract and took full responsibility for it.

Once the contract was signed, it was time to move to the second phase: the project proposal. In this aspect, it was similar to the second semester. We had to come up with our own topic that would meet the requirements for the third semester. Our first group meeting focused on finding a suitable solution that would be able to fulfill the criteria of a multi-tiered system architecture, the use of two technologies, two servers, and one database, and implementation using Java and C# with a Blazor UI. In this way, our brainstorming session started. At the end of the meeting, we came up with two final ideas from which the supervisor had to choose. One of the ideas was to create a marketplace where people could donate or give away things they no longer used or needed. The second idea was to create a platform that would offer the possibility to connect people looking for jobs with companies. Both ideas were relevant, and we showed interest in both. The recruitment platform was prioritized, as it was more relevant to our personal field as students.

Our decision was supported and confirmed by the supervisor, and in this way, we started working on HireFire.and confirmed by the supervisor and in this way, we started working on HireFire.



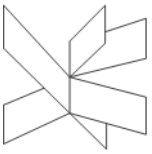
4. Project Description

At the beginning of our semester, our group documented a full Project Description of HireFire, during which we focused on gathering information. We did not use any initial customer questionnaires to gather information, instead we resorted to using online second or tertiary type sources, this is similar to the previous semester (SEP2).

To maintain a practicable scope, we defined a set of delimitations for features, through which was inspired from the numerous sources that we have accumulated. The Project Description also steered our choice of method. We selected Unified Process as the development framework and followed its three phases - Inception, Elaboration, Construction - to organise the work. In parallel we used Kanban to keep the task board visible plus the work moving - Kanban later turned into the core of our Scrumban blend. The description listed the required technologies - C# (.NET and Blazor), CSS, Java, PostgreSQL, CSS besides Trello for task tracking. Those choices kept the technical plan inside the boundaries set by the semester plan and the project brief.

During the first weeks we returned to the Project Description repeatedly. The first idea had included extra features for user engagement, but we cut them so the work would fit the semester length. The Project Description also stayed fixed and gave us a single point of reference while we wrote the use cases, ranked the requirements as well as drew the system architecture.

In conclusion, the Project Description established the cornerstone for our SEP3 project by defining the project's purpose, direction, scope and constraints. It provided a mutual understanding, and it propelled us to make sensible and logical decision-making when designing, implementing and testing our project.



5. Project Execution

5.1 Iteration structure and Timeline

The project was executed using an iterative approach - in line with Unified Process *(Jacobson et al., 1999)*. Work was organized into iterations, each of them had clearly defined goals and deliverables. This allowed the team to incrementally develop fully functional components throughout most of the iterations *(Larman & Basili, 2003)*.

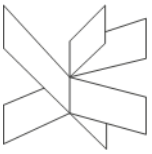
5.1.1 Schedule

The choice of iteration length was important and had to be made early on in the process. The main issue the team encountered was the uneven time availability of team members during the semester. From September till December the group members estimated to have approximately 8 hours per week available for project work, while from the 1st till the 19th of December it increased to 40 hours/week. Due to this discrepancy it was decided for iterations prior to the project period to last 2 weeks each as 1 week was not sufficient for sufficient progress to be accomplished. During the project period however, these iterations were to be shortened to 1 week in order to increase the pacing of work.

The structure mentioned above results in the following schedule, divided into the appropriate UP stages:

Table 1 Schedule for UP stages

Iteration/UP Phase	**Scheduled time**
Inception	Sept 4 - Sept 17
Iteration 1	Sept 4 - Sept 17
Elaboration	Sept 18 - Oct 15
Iteration 2	Sept 18 - Oct 1



Iteration 3	Oct 2 - Oct 15
Construction	Oct 16 - Dec 14
Iteration 4	Oct 16 - Oct 29
Iteration 5	Oct 30 - Nov 12
Iteration 6	Nov 13 - Nov 26
Iteration 7	Dec 1 - Dec 7
Iteration 8	Dec 8 - Dec 14
Transition	Dec 15 - Dec 19
Iteration 9	Dec 15 - Dec 19

The phase lengths were chosen to match the effort and focus required at each stage of the Unified Process. The team chose to cut the Transition period short, due to the fact that the project didn't require deployment which would take up a major part of the time dedicated to the phase.

5.1.2 Detailed Iteration Overview

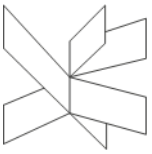
This section is dedicated to outlining the exact goals for each of the Unified Process iterations. *(Jacobson et al., 1999)*

Inception:

Iteration 1: Sept 4 - Sept 17

Primary Objective: Establish scope, establish and validate architecture, reduce high-risk unknowns.

Functional Goals:



- Deliver initial system use cases and requirements.
- Identify core workflows for all actors (Applicant, Recruiter, Company Representative, Admin).
- Prioritize features to implement first in the Elaboration phase.

Architectural Goals:

- Define the system architecture.
- Set up initial repositories and development environments for all tiers.
- Establish basic communication protocols (REST endpoints, DTO formats, authentication strategy).

Risk-Reduction Goals:

- Validate cross-tier connectivity with a simple prototype request/response.
- Confirm database connectivity via Spring Boot and PostgreSQL CRUD tests.

Outcome: Clear system scope, initial architecture diagrams, high-risk technical feasibility validated.

Elaboration:

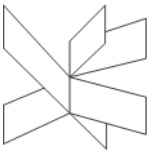
Iteration 2: Sept 18 - Oct 1

Primary Objective: Build core backend domain entities and implement the first CRUD workflows for the applicant account.

Functional Goals:

- Implement core domain entities: Company, Company Representative, Recruiter, Applicant, Job Listing.
- Implement backend CRUD logic for Use Case 5 – Manage Applicant Account.

Architectural Goals:



- Implement Spring Boot → PostgreSQL persistence layer for the Applicant entity.
- Define ASP.NET API endpoints for applicant management.
- Set up initial Blazor UI forms for applicant creation and editing.

Risk-Reduction Goals:

- Verify basic data validation works.

Outcome: A functioning backend foundation with initial CRUD operations and early UI integration for applicant management.

Iteration 3: Oct 2 - Oct 15

Primary Objective: Complete all functionality required for full applicant account management, implement session management and integrate authentication/authorization.

Functional Goals:

- Implement Use Case 5 – Manage Applicant Account fully (CRUD backend done in Iteration 2).
- Implement Use Case 10 - Manage session (Log in/Log out).

Architectural Goals:

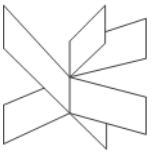
- Integrate authentication/authorization across layers.

Risk-Reduction Goals:

- Validate that authentication and authorization workflows function securely and cannot be bypassed.

Outcome: Applicant account features fully implemented and secured through authentication/authorization. Logging in and out is completely functional.

Construction



Iteration 4: Oct 16 - Oct 29

Primary Objective: Deliver complete company management functionality and implement the review process for company registration requests.

Functional Goals:

- Implement Use Case 2 – Manage Company fully (CRUD backend done in Iteration 3).
- Implement Use Case 8 – Review Company Registration Request.

Architectural Goals:

- Define ASP.NET API endpoints for company management.
- Implement Spring Boot → PostgreSQL persistence layer for the Company entity.

Risk-Reduction Goals:

- Verify that company creation with validation is stable across all tiers.

Outcome: All company-related features fully implemented, including registration review and validated cross-tier workflows.

Iteration 5: Oct 30 - Nov 12

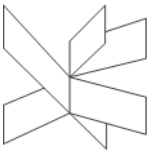
Primary Objective: Implement complete recruiter account management and introduce a full role/privilege system across the application.

Functional Goals:

- Implement Use Case 1 – Manage Recruiter Accounts end-to-end.
- Implement backend CRUD logic for Use Case 9 – Manage Company Representative Account.

Architectural Goals:

- Implement role/privilege system: Applicant, Recruiter, Company Representative, Admin.



Risk-Reduction Goals:

- Validate performance of multi-role routing in Blazor.

Outcome: Recruiter management and multi-role system fully functional across the platform. CRUD operations prepared on the backend for Use Case 10.

Iteration 6: Nov 13 - Nov 26

Primary Objective: Introduce job listing management and implement the recommendation generation logic.

Functional Goals:

- Implement Use Case 4 - Manage Job Listing.

Architectural Goals:

- Add recommendation generation logic.
- Implement application submission business logic and persistence.

Risk-Reduction Goals:

- Validate efficiency of recommendation algorithm.

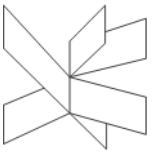
Outcome: Job listing creation and management operational, with a functioning recommendation algorithm and application submission support.

Iteration 7: Dec 1 - Dec 7

Primary Objective: Complete the full job application process for both applicants and recruiters. Finish implementation of Use Case 9

Functional Goals:

- Implement Use Case 7 - Manage Applications.



- Implement Use Case 3 – Browse & Apply for Jobs.
- Implement Use Case 9 – Manage Company Representative Account fully (CRUD backend done in Iteration 5).

Architectural Goals:

- Integrate matching into backend services.

Risk-Reduction Goals:

- None

Outcome: Applicants can browse/apply for jobs, and recruiters can manage applications end-to-end. Company representative account management fully functional.

Iteration 8: Dec 8 - Dec 14

Primary Objective: Implement live chat and provide applicants with status feedback on submitted applications.

Functional Goals:

- Implement Use Case 6 - Recruiter-Applicant Communication.
- Implement viewing of applied positions with status feedback.

Architectural Goals:

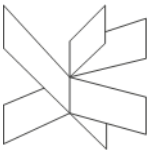
- Implement messaging model.

Risk-Reduction Goals:

- Ensure chat persistence works with minimal latency.

Outcome: Applicants and recruiters can exchange messages, and applicants can view their application statuses.

Transition



Iteration 9: Dec 15 - Dec 19

Primary Objective: Stabilize the system, resolve outstanding issues, and prepare the project for submission.

Functional Goals:

- Fix priority bugs.
- Polish UI.
- Perform final testing.

Architectural Goals:

- None

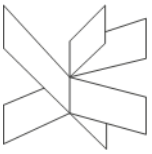
Risk-Reduction Goals:

- Verify system stability.
- Verify system security.

Outcome: A stable, polished, and fully tested system ready for hand-in.

5.2 Kanban workflow configuration

This section describes how the project team organized and managed development work using a Kanban workflow. The goal was to maintain a clear vision of the tasks at hand and better coordinate work throughout the Unified Process iterations. (Anderson, 2010)



5.2.1 Board structure

The team used a digital Kanban board with five columns:

- **Backlog** – Approved tasks not yet ready for development were stored in this column.
- **Ready** – Tasks that met the Definition of Ready.
- **In Progress** – Tasks being implemented by the team at present.
- **Review/Testing** – Tasks ready for peer review and testing.
- **Done** – Completed tasks verified against the acceptance criteria. The coding tasks in this section have been merged to the main branch.

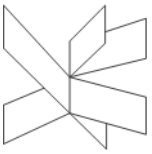
No swimlanes were used, as our team did not deem them necessary. With only 5 members the team thought that they would likely have introduced unnecessary complexity into the taskboard.

5.2.2 Work-In-Progress Limits

WIP limits were applied to improve focus and reduce multitasking *(Anderson, 2010)*:

- **Ready**: 10 items
- **In Progress**: 5 items
- **Review/Testing**: 2 items

The limits were based on the team size of five students and worked well enough not to require changes throughout the project.



5.2.3 Work Item Types

Work items were classified as:

- **Feature Tasks** – Derived from use cases and semester project requirements.
- **Bug Fixes** – Fixing issues found by team members or during testing.
- **Technical Tasks** – Refactoring and small improvements to the code.
- **Documentation Tasks** – Updates to the documentation and diagrams.

5.2.4 Card design

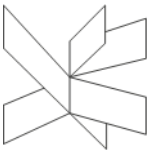
Each Kanban card contained:

- Task title and short description
- Related use case or requirement
- Task type
- Acceptance criteria
- Assigned team member(if there was any)

5.2.5 Pull Policies

Team members pulled new tasks from the *Ready* column only when:

1. They had less than 2 tasks in the In Progress column.
2. The WIP limit for *In Progress* had not been reached.



5.2.6 Definition of Ready and Definition of Done

Definition of Ready (DoR):

- Requirement understood by the team

Definition of Done (DoD):

- Task implemented and merged into the main branch
- Peer-reviewed by at least one team member
- Unit and integration tests passing
- Documentation updated (if applicable)
- Task marked as complete on the Kanban board

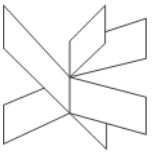
5.2.7 Board Review Cadence

In order to stay consistent throughout the entirety of the project work, the team decided to agree on specific intervals at which the board was reviewed.

Reviews occured:

- **Daily** to check progress and address pressing issues(Only during the project period)
- **Weekly** to refine backlog and tasks
- **At the end of each iteration**, to evaluate WIP limits, summarize the completed work, and identify potential bottlenecks

During the project the team deemed no changes to the process to be necessary. Though occasional bottlenecks were encountered in the **Review/Testing** section the team decided to stay with the original structure as it made sure that the features were delivered completely as soon as possible.



5.3 Work item lifecycle

The work item lifecycle describes how tasks moved through the Kanban system from their initial creation to final completion. For this project, the lifecycle was created to support clarity, improve work organization and grant all of the team members a clear overview of what is being worked on.

5.3.1 Definition of Ready and Definition of Done

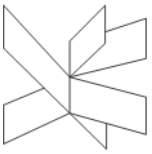
Work items were mainly created during weekly meetings. Most items came from:

- Use case analysis
- Bugs identified during testing
- Technical needs discovered during development
- Documentation updates required for deliverables

During refinement, each task was given a short description, assigned a work item type, and provided with acceptance criteria. Tasks were placed into the **Backlog** column until they adhered to the Definition of Ready.

5.3.2 Moving items to “Ready”

A task was moved into the **Ready** column once the team agreed that:



- The idea behind the task was fully understood
- Acceptance criteria were defined clearly
- No external factors were blocking it
- The task size was appropriate for a maximum of 2 developers to complete within a reasonable time frame

This ensured that team members always had well-scoped work available when they finished ongoing tasks.

5.3.3 Active Development in “In Progress”

When a team member became available, they pulled a task from the **Ready** column into **In Progress**, respecting the pull policies:

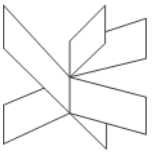
- No more than 2 tasks assigned to the same person
- The In Progress WIP limit (5 items) was not exceeded

Tasks in this stage were actively implemented. If a task required teamwork, it was assigned to all of the actively participating developers simultaneously.

5.3.4 Review and Testing

Completed development tasks were moved into the **Review/Testing** column. This stage meant the tasks were ready for both:

- **Peer Code Review** – another team member ensured code quality, correctness, and adherence to the project’s standards.
- **Functional and Integration Testing** – verification that the implementation worked as intended and did not introduce new issues.



This column was chosen to have an especially low WIP limit in order to act as a throttle and force the completed tasks to be fully tested and merged back into main as soon as possible.

5.3.5 Completion and “Done”

A task was entered into the **Done** column only after fulfilling the Definition of Done:

- Task implemented and merged into the main branch (only after review)
- Peer-reviewed by at least one team member
- Unit and integration tests passing
- Documentation updated (if applicable)
- Task marked as complete on the Kanban board

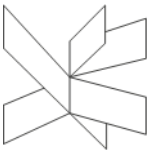
These requirements made sure that every task in the **Done** column was fully complete.

5.3.6 Summary of Lifecycle Flow

In summary, each work item passed through the following lifecycle stages:

Backlog → Ready → In Progress → Review/Testing → Done

This flow provided a clear view into the project’s work status at all stages and ensured that progress could be assessed with ease at any point during the semester.
(Anderson, 2010).



5.4 Iteration-by-Iteration Progress

This section gives a plain, step-by-step description of how the project progressed in each of the nine Unified Process iterations, each iteration summary goes over:

- Planned vs. completed work
- Issues and risks
- Process adaptations (section is only present when necessary)
- Delivered user functionality and artifacts

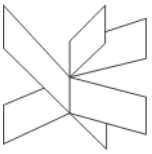
5.4.1 Iteration 1 – Sept 4 to Sept 17 (Inception Phase)

Primary Objective: Establish system scope, architecture, and feasibility.

Planned Work

Tasks moved from **Backlog** to **To-Do** included:

- Write down the first list of Project Requirements (Documentation Task).
- List who will use the system and write the first draft of each use case (Documentation Task).
- Draw a high level visualisation that shows Blazor communicating with ASP.NET, ASP.NET communicating with Spring Boot and Spring Boot communicating with PostgreSQL (Technical Task).
- Create a new git repository and put the basic folder structure in place (Technical Task).
- Create a prototype communication test between Logic tier and Frontend via HTTP which lets the Logic layer send a plain HTTP message to the Frontend and receive an answer (Technical Task).
- Write a small test that lets the Database layer create, read, update and delete a record in PostgreSQL through JDBC, using JPA (Technical Task).



All tasks met the DoR because the scope, the acceptance criteria and the architectural expectations were spelled out in detail.

Actual Work Completed

All above tasks reached **Done**:

- Architecture was validated through two prototype integrations.
- All initial use cases were drafted.
- The three-tier architecture (Client → Logic → Data Access) was documented.

Issues & Risks

- Initial uncertainty about gRPC communication required additional research.
- The checklist for the architecture documentation was not clear and led to small mix ups.

Deliverables

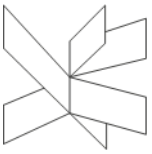
- Initial architecture diagrams
- Use case drafts
- Working prototypes of HTTP and database communication
- Project repository structure

5.4.2 Iteration 2 – Sept 18 to Oct 1 (Elaboration Phase)

Primary Objective: Implement early backend domain entities and first CRUD flow for Applicant (Use Case 5).

Planned Work

Moved to **Ready**:



- Describe what an Applicant entity looks like and how it will be stored (Feature Task)
- Add code to create, read, update plus delete Applicant records in the Data Tier (Feature Task)
- Expose Applicant endpoints through an ASP.NET REST service (Feature Task)
- Build the first Blazor prototypes for pages that concern Applicant Management (Feature Task)
- Agree on and set up one shared DTO structure (Technical Task)

Actual Work Completed

- Applicant entity fully implemented on the Data Tier.
- REST endpoints for Applicant Management fully functional in the Logic Tier.
- Basic Blazor forms created for Create and Edit Applicant flows.

Issues & Risks

- No issues or risks are apparent.

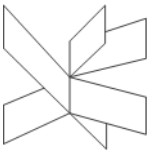
Deliverables

- Working CRUD flow for Applicant
- Test data seeded into the database
- Updated sequence diagrams for Applicant management

5.4.3 Iteration 3 – Oct 2 to Oct 15 (Elaboration Phase)

Primary Objective: Complete Applicant Management and implement Authentication/Authorization (Use Case 10).

Planned Work



Prepared for **Ready**:

- Finish every step a user needs to register and log in (Feature Task)
- Build sign-in plus sign-out that works in the browser, on the server and in the database (Feature Task)
- Implement form validation in the Client (Technical Task)
- Catch and report errors with clear messages (Technical Task)
- Redesign the first views so users see login and logout options (Feature Task)

Actual Work Completed

- All Applicant management functionality implemented.
- Authentication added.
- Log in/Log out features successfully implemented.
- Blazor UI updated to reflect authenticated state.
- Validation added to the frontend.

Issues & Risks

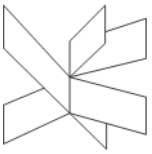
- Additional testing time needed due to security-sensitive code

Deliverables

- Fully functional Applicant account management system
- Authentication and authorization system
- Updated architecture diagrams

5.4.4 Iteration 4 – Oct 16 to Oct 29 (Construction Phase)

Primary Objective: Implement Company Management (Use Case 2) and Review Company Registration Requests (Use Case 8).



Planned Work

- Implement the Company entity and persistence layer (Feature Task)
- Expose REST endpoints that let callers add, change and delete companies. (Feature Task)
- Supply Blazor pages where the company life cycle is managed - create, edit, list and delete (Feature Task)
- Implement the Admin company review flow end-to-end (Feature Task)
- Validation and error handling updates (Technical Task)

Actual Work Completed

- Company CRUD operations implemented and functional.
- Registration request system added with admin approval flow.
- Blazor interface implemented for Company Representatives and Admin roles.

Issues & Risks

- Blazor was posing issues which took a significant amount of time to resolve.

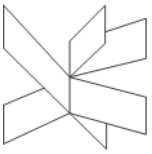
Deliverables

- Complete Company management feature set
- Admin review workflow
- Updated domain model

5.4.5 Iteration 5 – Oct 30 to Nov 12 (Construction Phase)

Primary Objective: Implement Recruiter Management (Use Case 1), CRUD for company representative management (Use Case 9) and Role/Privilege System.

Planned Work



- Recruiter entity implementation (Feature Task)
- Recruiter CRUD flows across all tiers (Feature Task)
- Expose REST endpoints that let callers add, change and delete company representative accounts (Feature Task)
- Implement role-based routing in Blazor (Feature Task)

Actual Work Completed

- Recruiter management system fully implemented.
- Role system validated in Blazor.
- CRUD endpoints working for company representatives

Issues & Risks

- Interactions between Recruiter and Company entities required refactoring.

Process Adaptations

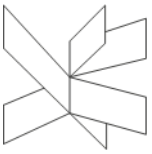
- The team wrote clearer card descriptions and sharper acceptance criteria.
- For every complex backend task, two developers now work side-by-side until the task is complete - this prevents delays.

Deliverables

- Fully functional Recruiter management system.
- Complete role and privilege system.
- Backend for company representative management.

5.4.6 Iteration 6 – Nov 13 to Nov 26 (Construction Phase)

Primary Objective: Implement Job Listing Management (Use Case 4) and Recommendation Logic.



Planned Work

- Job Listing entity implementation (Technical Task)
- Create/Edit/Remove/Close flows for Job Listings (Feature Tasks)
- Recommendation engine design and prototype (Feature Task)
- Database queries for listing filtering (Technical Task)

Actual Work Completed

- Job Listing management completed end-to-end.
- First version of recommendation algorithm implemented and integrated.

Issues & Risks

- Recommendation algorithm needed tuning after slow test results.

Deliverables

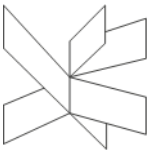
- Job listing management.
- Recommendation engine version 1.
- Updated diagrams.

5.4.7 Iteration 7 – Dec 1 to Dec 7 (Construction Phase)

Primary Objective: Implement job application handling (Use Case 7) and browsing/applying for jobs (Use Case 3), finish implementation of company representative account management (Use Case 9).

Planned Work

- Build application acceptance/decline flow (Feature Task)
- Implement match creation (Feature Task)



- Implement UI for company representative account management (Feature Tasks)
- Connect the browsing logic to the recommendation system (Feature Task).
- Track the status of each application by adding a status tracker (Feature Task)
- Adjust the recommendation algorithm for better results (Feature Task)

Actual Work Completed

- Recruiters can accept/decline applications.
- Company representatives can register, edit and remove their accounts.
- Applicants can browse recommended jobs and apply.
- Matching functional.

Issues & Risks

- Recommendation tuning caused a minor blocker.

Deliverables

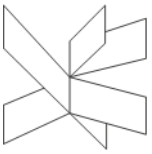
- Fully functional application handling.
- End-to-end job browsing and applying workflows.

5.4.8 Iteration 8 – Dec 8 to Dec 14 (Construction Phase)

Primary Objective: Implement Recruiter–Applicant chat (Use Case 6) and application status feedback.

Planned Work

- Messaging model and persistence (Feature Task)
- Logic Server chat implementation (Feature Task)
- Blazor frontend chat UI (Feature Task)



- Application status UI for applicant (Feature Task)

Actual Work Completed

- Chat system implemented using gRPC between logic and database tiers.
- Blazor chat interface added with live server updates.
- Applicants can view all previously applied positions with the status.

Issues & Risks

- Real-time messaging required using completely different communication between the Client and Logic Server.
- Minor UI synchronization bugs.

Process Adaptations

- The team decided to work in pairs more frequently to speed up the chat debugging process.

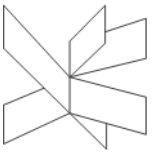
Deliverables

- Chat system
- Application status view
- Updated technical documentation
- Updated architectural diagrams

5.4.9 Iteration 9 – Dec 15 to Dec 19 (Transition Phase)

Primary Objective: Final stabilization, bug fixing, performance checks, and preparation for submission.

Planned Work:



- Fixing bugs from the previous iterations (Bug Fix Tasks)
- UI polishing (Technical Tasks)
- Final testing (Technical Tasks)
- Documentation cleanup (Documentation Tasks)

Actual Work Completed:

- All critical bugs resolved.
- UI standardized and refined.
- Full-system tests performed with no apparent failures.
- Final documentation complete.

Issues & Risks

- No issues/risk were apparent

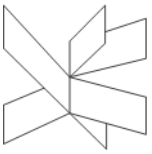
Deliverables

- Final system/code implemented.
- Completed documentation package.

5.5 Metrics and Flow Analysis

This section evaluates how work progressed through the Kanban board throughout the iterations. Metrics were tracked during the whole project in order to expose any necessary process adaptations. *(Reinertsen, 2009)*

The statistics are largely affected by the team's varying availability during the semester. The greater availability during the project period clearly contributed to a larger throughput as showcased below.



5.5.1 Tracked Metrics

Throughout the project we chose to track the following metrics:

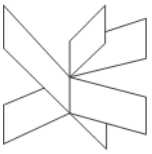
- **Throughput** - the number of tasks fully completed (placed in the **Done** column) within the iteration
- **Cycle Time -** the average time between a task being started (**In Progress**) and completed (**Done**)
- **Lead Time -** the average time between a task being placed in the **Backlog** and reaching **Done**.
- **Work-In-Progress (WIP)** - The number of tasks present in the **In Progress** column.

(Anderson, 2010; Reinertsen, 2009)

5.5.2 Throughput Trends Across Iterations

Throughput followed an upward trend but was strongly influenced by team availability.

- **Iteration 1:** Very low throughput, reflecting architectural focus and minimal implementation.
- **Iterations 2–3:** Modest throughput, caused by limited working hours and lengthy foundational backend work.
- **Iterations 4–6:** Moderate throughput despite an increased number of simple tasks, limited availability still affected how much work was being completed.
- **Iterations 7–8:** Noticeable increase in throughput due to significantly higher team availability, which enabled for faster task completion.
- **Iteration 9:** The highest throughput recorded due to a large number of relatively small tasks combined with high availability.



The increased throughput during the final stages of development shows that Kanban scaled well with the increased team availability.

5.5.3 Cycle Time Analysis

Iterations 1-3

- Average cycle time ranged between 4–6 days per task.
- Tasks often remained in **In Progress** between work sessions.
- Group members were usually taking their time before picking up their tasks.

Iterations 4-6

- Average cycle time decreased to approximately 3–4 days per task.
- Architectural familiarity helped mitigate limited availability.
- However, cycle times remained constrained by fragmented working schedules.

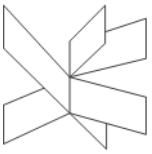
Iterations 7-9

- Average cycle time dropped to 2–3 days per task.
- Tasks moved more continuously through the board due to more frequent work sessions.
- Bug fixes and documentation tasks were being completed quickly (often within 1-2 work days).

These statistics show that work was being completed more quickly towards the end of the project. This was to be expected due to 5 times greater team availability within that period.

5.5.4 Lead Time Analysis

Lead time was strongly affected by how long tasks waited before being actively worked on.



- **Early and Mid Iterations (1–6) -** Feature tasks often remained in **Backlog** or **Ready** longer due to limited availability, resulting in lead times of 7–10 days.

- **Final Iterations (7–9) -** Increased availability allowed faster pulling of ready tasks, reducing lead times to 4–6 days for feature tasks.

Bug-fix tasks consistently exhibited shorter lead times (1–3 days), especially in Iteration 9 where they were prioritized.

5.5.5 WIP and Pull Policy Compliance

WIP limits were respected throughout the project:

- **Ready** never exceeded 10 items.
- **In Progress** adhered to the 5 item limit.
- **Review/Testing** frequently reached its limit of 2 items, particularly during Iterations 7-9.

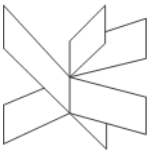
Pull policies were followed without exception. No one broke the work-in-progress limits to offset low stock. Because the limits stayed in place, no one was forced to juggle extra tasks and the progress of every job stayed visible, even when material was scarce.

5.5.6 Bottleneck Identification

The Review/Testing column was the most persistent bottleneck:

Impact on Low Availability Iterations (1-6):

- Reviews were sometimes delayed across multiple days.



- Integration-heavy tasks accumulated temporarily in **Review/Testing**.

Impact on High Availability Iterations (7-9):

- Faster review turnaround due to overlapping work schedules.
- Reduced idle time between development and verification.

(Reinertsen, 2009)

5.5.7 Flow Distribution by Task Type

Flow behaviour changed depending on task types and time availability:

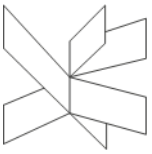
- Feature tasks spent most of their lives in the **In Progress** column, especially during times of limited team availability.
- Bug-fix tasks moved through every stage quickly and did so in every iteration, most of all in Iteration 9.
- Technical tasks moved through the board more quickly than Feature tasks but frequently required additional research.
- Documentation tasks were scarce before Iteration 9, they didn’t generally cause issues and flowed smoothly through the taskboard.

The Kanban board absorbed all those shifts without any change to its structure.

5.5.8 Overall Flow Stability

The flow stability improved progressively within our project, the signs of these were:

- High variability during early low-availability iterations.
- Predictable, steady flow during final iterations with increased availability.
- No accumulation of unfinished work at project end.



5.5.9 Summary

The measurements and the flow study show that the Kanban board kept the Unified Process on track even when team members were not always present.

Key outcomes include:

- Stable flow under constrained capacity
- Improved cycle time as availability increased
- Predictable delivery during the final project phase

To summarize, the Kanban board gave the team one shared view of every job that had to be done and it carried the work through to a finished, reliable system by the project deadline.

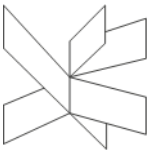
5.6 Risk and Change Management During Execution

This section explains how risks and changes were handled during the nine Unified Process iterations by means of a Kanban-style workflow. Following the Unified Process, the team reviewed risks without pause and let risk steer decisions, while Kanban let them react quickly when new problems or altered requirements appeared.

5.6.1 Risk Identification

Risks were identified continuously during execution through:

- Iteration planning
- Testing and integration activities



- Observation of Kanban board behavior (blocked cards or slow-moving cards)

Key risks identified during the project included:

- Early uncertainty around inter-service communication (Iteration 1)
- Increased testing effort for authentication and authorization (Iteration 3)
- Performance concerns related to the recommendation algorithm (Iterations 6–7)
- Real-time communication challenges during chat implementation (Iteration 8)

5.6.2 Risk Assessment and Prioritization

Identified risks were assessed based on:

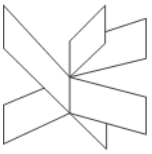
- Impact on core system functionality and performance
- Architectural stability

High-impact risks were tackled first following the Unified Process rule that the biggest dangers drive the order of work *(Jacobson et al., 1999)*. This happened mainly in the Inception and Elaboration phases. The team used a Kanban board - when they had spare capacity, they pulled the risk tasks forward and left the lower priority jobs waiting.

5.6.3 Risk Mitigation and Monitoring

Risk mitigation strategies included:

- Prototype development to validate architectural decisions
- Early end-to-end implementation of core CRUD flows
- Additional testing effort for security-sensitive features
- Performance tuning for the recommendation engine
- Pair programming for complex or error-prone features (Iterations 5-8)



Risks were monitored continuously through:

- Blocked cards
- Repeated bottlenecks in the **Review/Testing** column

5.6.4 Change Identification and Impact Analysis

Changes were primarily driven by:

- Testing outcomes
- Integration feedback
- Newly discovered technical constraints/setbacks

Common changes included:

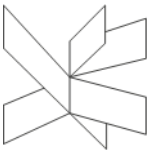
- Refinement of validation and error handling
- Refactoring of entity relationships
- Adjustments to recommendation logic for performance and accuracy

Before implementation, changes were assessed for their impact on:

- Scope and affected components
- Architectural consistency
- Required testing effort

5.6.5 Change Approval, Scheduling, and Validation

- The team agreed on changes together while planning the iteration or when an urgent problem appeared.



- After approval, each change became a Kanban card and was placed in the workflow adhering to the column card limits.
- The **Review/Testing** stage made sure the change and regression tests were run if they applied.

5.6.6 Continuous Improvement

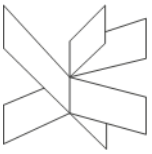
Insights from risk and change management led to:

- Improved card descriptions and acceptance criteria
- Increased use of pair programming for complex tasks
- Better anticipation of testing bottlenecks in later iterations

Overall, the integration of Unified Process principles with the Kanban methodology and execution enabled effective handling of uncertainty while maintaining a steady project progress.

5.7 Collaboration and Communication Practises

This section explains how the team arranged and sustained cooperation and talk during the project. Because the Unified Process repeats its steps besides Kanban keeps tasks moving without fixed phases, the group needed steady contact to align work from one cycle to the next plus to react quickly when new risks or alterations appeared.



5.7.1 Team Coordination and Workflow Communication

The team used a Kanban board every day to keep everyone aligned. The board showed the project's current state and was updated in real time. Task status, priorities and blockers were visible as cards moved from left to right.

Key practices included:

- Using Kanban cards as the main means of communication for different tasks within the project
- Columns were labeled with exact states so that the group knows if the task is finished, not done yet, in progress, and etc.
- When a card stalled, the team discussed it right away.
- When the board hit its work-in-progress limit, whoever was free stepped in informally to clear the jam.

The board spoke for itself, no one had to compile long reports and if any problems surfaced early, the group would be able to pick it up relatively quickly. *(Anderson, 2010)*

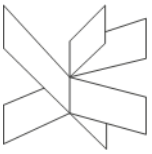
5.7.2 Iteration Planning and Review Communication

Communication around planning and progress often occurs primarily at the iteration boundary. Project ideas were also inquired in non-formal conversations and formal meetings during school hours, provoking thought and allowing team members to come up with their own suggestions and visions for the project’s future.

At the start of each iteration:

- Work items were reviewed and moved to **To-Do** only when scope and acceptance criteria were sufficiently defined
- Iteration goals were aligned with the current Unified Process phase.

At the end of iterations:



- Completed functionality and artifacts were reviewed
- Issues encountered during execution were discussed and fixed
- As more information was gathered (from school lectures), informed adjustments were made in subsequent iterations.

These reviews supported a shared understanding of progress whilst ensuring alignment between planned and delivered outcomes.

5.7.3 Cross-Tier and Technical Collaboration

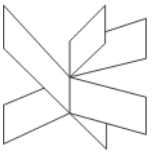
Given the multi tier architecture, effective technical collaboration was necessary to manage system wide concerns. Collaboration practices included:

- Unifying the shape of data that crosses between layers with the shape expected by each tier.
- Work together to find and fix integration problems, first when the architecture is still being validated and again whenever the build gains new features.
- Pier-programming for more complex tasks e.g. recommendation engine or chat.

Pair programming was introduced selectively to reduce blockers, speed up debugging, and improve shared understanding of complex code paths.

5.7.4 Communication During Testing and Review

The Testing section was the main place where the team talked about quality. When work started to slow in the **Review/Testing** section we decided to make these changes:



- Team members focused on testing and reviewing the tasks in the **Review/Testing** column whenever possible.
- The features that were hard to grasp - especially ones that involved security rules, speed targets or references between many kinds of records were discussed in more detail and were often reviewed by more than 1 peer.
- What we learned during testing often turned into new bugfix and refinement tasks - nothing slipped through the cracks.

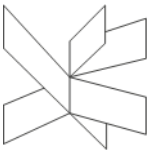
Because information flowed back and forth without pause, every problem found during validation reached the right developer quickly and was fixed as soon as possible.

5.7.5 Adaptation of Collaboration Practices

Our collaboration methods were not static; they evolved as we identified specific workflow challenges. Key adaptations included:

- We improved the clarity of task descriptions and acceptance criteria to minimize mid-task misunderstandings.
- We increased direct collaboration and pair programming for high-risk features likely to cause defects or delays.
- We began anticipating the communication overhead for testing-heavy iterations to prevent bottlenecks before they occurred.

These refinements led to a much smoother execution in the project's later stages and significantly reduced the volume of issues encountered during the Transition phase.



5.7.6 Summary

Our approach to collaboration was intentionally lightweight and continuous, keeping communication tightly linked to the active Kanban workflow. By prioritizing visual management and iterative reviews over formal meetings or heavy documentation, we maintained project alignment without unnecessary overhead. This strategy complemented the iterative nature of the Unified Process, providing the flexibility we needed to navigate technical complexities and evolving requirements as they arose.

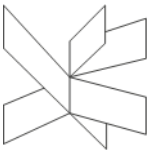
5.8.1 Implementation Approach

Development followed an incremental, vertical-slice approach aligned with the Unified Process. Features were implemented end-to-end across all architectural layers within the same iteration whenever possible, rather than developing isolated components in advance.

Our implementation strategy was characterized by:

- We used Iteration 1 to build prototypes, ensuring our architectural foundations were solid before moving forward.
- We prioritized full CRUD functionality for domain entities to provide a reliable base for more complex features.
- Functionality was expanded incrementally to stay in touch with the specific goals of each Unified Process phase.
- We focused on finishing features across all tiers before starting new work, which helped keep the system integrated.

This vertical slice approach effectively minimized integration risks and allowed us to continuously test our architectural assumptions against working code. *(Larman & Basili, 2003)*



5.8.2 Multi-Tier Integration and Coordination

Technical execution required consistent coordination across:

- Blazor frontend
- ASP.NET logic layer
- Spring Boot backend
- PostgreSQL database

Integration practices included:

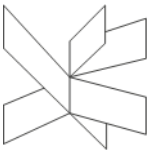
- Coordinated definition of DTOs and request/response structures.
- Early end-to-end request testing during Elaboration.
- Frequent validation of cross-tier communication during feature development.
- Iterative refinement of entity relationships as new use cases were added.

Integration issues were treated as high-priority work items and addressed promptly to prevent delays.

5.8.3 Development and Debugging Practises

Most development tasks were completed individually; however, collaboration intensity varied depending on task complexity.

Practices included:



- Pier programming for complex tasks.
- Joined debugging sessions when performance issues emerged.
- Focused investigation and tuning for performance-sensitive components.

Thanks to more frequent collaborative work, the code quality improved significantly. This measure also ensured that all of the team's members understood the system fully.

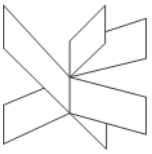
5.8.4 Testing During Execution

Testing happened at the same time as development, not after it. The Review/Testing column on the Kanban board acted as the main quality checkpoint. The team carried out four kinds of testing:

- They walked through every complete user story by hand to confirm that each use case worked from start to finish.
- They paid special attention to parts of the system that handle security, like the login process.
- They verified responsiveness when the recommendation engine or heavy database queries were run.
- Whenever a change touched shared code or data, they reran older tests to be sure nothing else broke.
- Tasks piled up frequently in Review/Testing - that steady pile showed how much testing and verification work the complex features needed and guided later planning.

5.8.5 Handling Technical Challenges

Several technical problems shaped the way the work was done:



- At the start, no one was sure how the services would communicate with each other - the team had to build prototypes to find out
- The recommendation logic ran too slowly and had to be adjusted
- The addition of real time messaging required the use of a different technology for communication between the Logic Server and Client.

The issues were addressed through additional refinement, testing and increased collaboration instead of process changes.

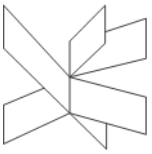
5.8.6 Summary

Technical work was carried out in short, complete steps. Each step resulted in a working part of the system, and the parts were joined together as soon as possible. Every addition was verified and merged into main within a short timeframe. The team followed the Unified Process - they looked for the largest technical dangers first and removed them before they grew (Jacobson et al., 1999). Kanban cards showed what was in progress, what waited and what was done - the plan was adjusted immediately when new problems appeared. Because of those habits, the group moved forward without stagnation and managed to deliver a working system on schedule.

6. Personal Reflections

Tymoteusz

At the beginning of this semester, after the negative experience with my previous SEP group, Jakub, Damian, and I decided that we needed teammates who were more engaged and aligned with our working style. We also knew that Hamsa’s and Crist’s group had split up, so we reached out to them with a proposal to work together. Compared to the previous semester, I immediately felt a much higher level of



engagement from my new SEP teammates, as well as a noticeably better and more relaxed atmosphere within the group.

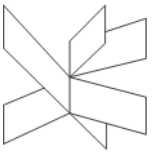
Leaving the group aside, I personally felt that the subjects this semester were much more difficult and demanding, but at the same time far more interesting. Because of that, I had to dedicate significantly more time to studying than in previous semesters. This wasn't easy to balance with working in a warehouse for around fifty hours a month, combined with other responsibilities such as Muay Thai training, maintaining my relationship, and trying to have some free time.

I also began actively searching for a job in the software field, and I quickly realised how challenging this process can be. Many positions require a much broader skill set than I currently have, and the competition is quite strong. Understanding how difficult it is to find a job in this industry made me start questioning whether studying software engineering still made sense for me. Unfortunately, I still haven't found a job in software, but I did manage to secure two interviews, which gave me hope. However, after receiving a rejection with feedback that the other candidate had more relevant skills, I felt overwhelmed and powerless.

I'm still not fully convinced that software is the industry I will work in long-term, but this did not change the fact that we had a project to complete. Luckily, in this case, the great atmosphere in the group and the interesting nature of our project – the HireFire application, which in simple terms works like Tinder for job searching – helped me find motivation and the strength to stay fully active in the project. I kept applying for jobs and continued deepening my knowledge in the field despite my doubts.

To sum up, this semester was a meaningful experience for me. It taught me that perseverance is essential, and that supportive friends around you can give you the motivation to keep going. I hope that next semester I won't face difficulties with finding an internship and that I will regain a stronger sense of purpose in continuing this direction of study.

Hamsa



In the second semester, I was experiencing problems with my semester group. Three of my group members unfortunately left, this caused instability and poor output of the work needed to do well within the project period. As a result, we didn't receive the results we were hoping for that semester unfortunately. This whole experience made me and my other group member Cristi realize how critical commitment and communication were in a group setting and how that propels you toward success.

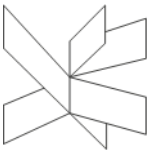
Now in this semester, me and my fellow group member Cristi's situation improved when we decided to partner up with our new group members Damien, Jakub and Tymoteusz. From the start I knew they were highly skilled, communicative people who are excellent at planning and delegating different tasks for different people. Working with teammates who are on the same wavelengths of group ethics, and output reinforces a much more structured and optimized environment. Tasks were instantly distributed whilst deadlines were respected which ensured an effective workflow and expected results.

Specifically in this semester, I wanted to focus on the parts of the project where I thought I could do a good job in like for example, I had a decent understanding of the logic and client tier, so I tried implement some code there, other places could be some documentation in the project report, where I felt like I could paint the narrative of this semester project well. In general, this is a nice turn of events because in my last semester, me and my other group members had to do all of the different areas of the semester project, which burnt us out quite a lot.

Damian

For this semester project, I worked with two group members from the second semester. Since we already knew each other we could start working right away. As a team, we also brought in two additional members with whom we had no prior working experience.

Instead of using SCRUM like we did before, we decided to try Kanban. We felt that SCRUM had too many long meetings that were not really helpful for a small group like



ours. Kanban was much simpler and easier to understand. It let us focus on our actual work without wasting time. It was especially useful when it came to writing code.

The atmosphere in our group was rather informal and relaxed. As we are friends, we do not feel the need to be overly professional or stiff with each other. This made talking about issues straightforward and if someone got stuck or had a different idea, they just said it. During the project period we scheduled a bunch of meetings when we felt that we had to summarize what we have done and move forward by splitting the task.

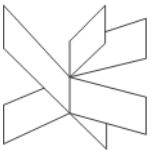
Our main challenge was to build a relatively original app which was not provided as an example for SEP3. Initially we set a very ambitious goal with lots of features. However as the deadline was getting closer and closer, we realized we could not finish everything perfectly. We cut some functionalities from the final app focusing on the most important parts. Despite this, the project was beneficial, as it required us to learn new technologies, especially these which were not covered during classes.

Overall, the process of developing the application and working as a group was smooth and conflict-free, allowing us to focus fully on the technical and practical aspects of the project.

Jakub

During this semester project I have worked with 2 of the same people as the previous semester, but all of the group's members have been friends even before the work began. This was sure to provide for a friendly and supportive atmosphere, which it absolutely did. No one was afraid of speaking out during the meetings and as colleagues we tried to help each other as well as we could.

A big change regarding the process during this project was the use of Kanban as a replacement for SCUM. I found Kanban a bit more relaxed than SCRUM, it allowed me to keep up to date with everything happening in the project by simply looking at the taskboard.



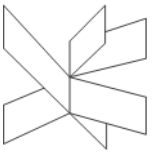
The biggest challenge I've found throughout the project is the implementation of the system itself. The application was significantly more complicated this time around, with a Client and 2 different Servers, the level of complexity introduced many challenges that the team had to solve. This, combined with a good atmosphere provided for a great learning experience.

Cristian

The second semester at Software Technology Engineering at VIA University College was difficult in terms of group work. Out of five members we ended up working and presenting the semester project in two people. Both me and Hamsa decided to keep working together in the third semester, and we were looking for new teammates. Jokub, Tymoteusz and Damian were in search of new members as well, and since we were in a good relationship, it was decided to form a group. We never worked together on a project before, and I was a bit concerned about that area, but I knew the guys were taking it seriously and that I could trust them. I was sure we would deliver a professional project at the end.

The project work started with choosing a topic and making a contract as in the previous semester. We chose a topic that was interesting for everyone, to keep our working motivation high. Also, we thought we could make an application that would help us in future or any other students and people. The difficulty of the product was higher compared to the one from the second semester. Luckily this time everyone was implicated in the process. Based on previous experience, it was decided to use methodologies that would help us reach our main goal. We used UP, learned in the previous semester, and Kanban, from our senior's recommendation. This semester we split all our work into tasks and uploaded them on an application called Trello. After we assigned members of our group and deadlines to the task. This way everybody could see the things that have to be done, and the progress. In my opinion this was a huge improvement in terms of semester project work, compared to previous semesters.

To work effectively, it was decided to have mandatory physical meetings for task distribution, brainstorming and evaluation. We used discord for online meetings, in case



someone needed help with their task or if we wanted to announce updates. This way, we had a flexible schedule and free choice for working preferences. Our group had different types of personality, but we managed to use our strong sides and to cover each other's disadvantages, maintaining a friendly environment.

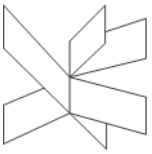
The classes during this semester were also more difficult compared to the previous semester, as expected. Our group tried to implement as much knowledge as possible into the semester project. Even so, not everything could be used in it. The supervisors were very helpful, and we could reach them very easily, which influenced our working efficiency.

7. Reflection on Supervision

We managed to be a bit more organized in terms of work organization this semester than the last one, this unanimously agreed by all the group members, however we faced difficulty with some theoretical aspects that were a bit puzzling to deal with , this is where our supervisor came into play. Throughout the whole SEP3 Project period, we actively inquired help from our SEP3 supervisor Joseph Chudwudi Okika and Jakub Trigger.

Initially, we actively started to consult our supervisors, as soon as we were able to, but before we consulted them, the supervisor who are responsible for this semester's SEP3 project introduced the whole process of the SEP3 Project and gave us time to form our groups (Group Formation) and work on the our Project Proposal

Our first actual supervisor meeting was regarding our Project Proposal, this meeting was with one of our supervisors, Jakub, where he gave us some feedback for our Project Proposal and how we could tweak each proposal to more accurately fit the guidelines of the SEP3 Project.



Our second supervisor meeting was with Joseph and it was centralized around our Project Description, giving us feedback on certain sections of the document, correcting us on any discreet mistakes we've made or whole sections which were incorrect.

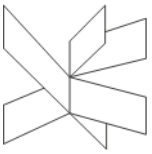
Our third supervisor meeting was regarding our Software Architecture of our project, this meeting was with our supervisor Joseph. We received feedback on how we could improve our architecture and also received feedback urging us to partially revamp our architecture, we followed the advice and we implemented it.

Our last supervisor meeting was our Proof of Concept and Version Control meeting, the meeting was with both our supervisors, Jakub & Joseph. We presented a working Proof of Concept to our supervisor and the methodology of how we are going to be using GitHub to manage and divide work to ensure an efficient workflow, this meeting gave us a lot of insights on how we can improve of Proof of Concept and how we can use Version Control more effectively in a group setting.

In totality, we didn’t have any additional meetings outside of the standard 4 meetings made for us but instead, we solicited many questions from our supervisor irregularly. Even though our supervisor managed to clarify a lot of queries we had, some doubts about our project remained in regards to diagram documentation and specific code/networking implementations.

To summarize, while our supervisor meetings immensely improved our project progress and workflow, they weren’t always sufficient and further group investigation and research was needed to combat particular problems. Our supervisors’ availability and approachability were crucial to ensuring that we fixed all major problems and errors in our SEP3 Project and it was pivotal to how we were able to solve these issues and get our project completed on schedule.

Supervisor Meeting Schedule



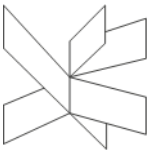
Nr.	Date	Time	Room	Topic	Supervisor
1	10th Sep	9:20 - 9:40	C04.17	Project Proposal Feedback	Joseph Okika, Jakub Trigger
2	24th Sep	9:00 - 9:30	C04.17	Project Description Feedback	Joseph Okika, Jakub Trigger
3	8th Oct	10:20 - 10:50	C03.04	Architecture Overview Feedback	Joseph Okika, Jakub Trigger
4	5th Nov	8:40 - 9:00	A03.02	Proof of Concept Feedback	Joseph Okika, Jakub Trigger

Table 2 Supervisor Meeting Schedule

8. Conclusion

The project's smooth execution demonstrates that combining Kanban with Unified Process can be a strong tool in software development. Though the team had not previously worked using Kanban, the members managed to adopt the methodology with no issues. Kanban allowed for great coordination both during the months leading up to the Project Period and during the Project Period itself despite the varying time availability of team members.

Starting the project early on and keeping up the slow and steady work before December allowed the team to be more relaxed and free of stress nearing the end of the endeavor. The lightweight and continuous communication centered around the Kanban board allowed for the team to keep track of what is happening without the necessity for daily meetings, which saved the team valuable time. The chosen process was proven to be effective for the team as it has allowed for the delivery of the finished product on schedule.



9. References

Jacobson, I., Booch, G., & Rumbaugh, J. (1999). *The unified software development process*. Addison-Wesley.

Anderson, D. J. (2010). *Kanban: Successful evolutionary change for your technology business*. Blue Hole Press.

Reinertsen, D. G. (2009). *The principles of product development flow*. Celeritas Publishing.

Larman, C., & Basili, V. R. (2003). Iterative and incremental development: A brief history. *IEEE Computer, 36*(6), 47–56.

Meyer, E. (2016). The culture map: Breaking through the invisible boundaries of global business. PublicAffairs.

16Personalities. (n.d.). Free personality test. *https://www.16personalities.com*

OWASP Foundation. (2021). OWASP top 10 web application security risks. *https://owasp.org*