StudyTasks
Process Documentation

Project Management, Time Tracking, Reflection

Course: Software Project
Authors: Serdar Toluay & Ersan Kavak
Date: 2026-01-15

Focus of this submission: PROCESS (project management)

Table of Contents

- 1. Project Overview
- 2. Process Model and Workflow
- 3. Project Management Setup
- 4. Backlog, Task Management and Sprint Reviews
- 5. Time Tracking Summary
- 6. Collaboration and Communication
- 7. Quality Assurance (Testing and Edge Cases)
- 8. Evidence for Process Grading Criteria
- 9. Reflection
- 10. Personal Contribution
- 11. Personal Development
- Appendix A: Detailed Time Log (Work Packages 1-40)

1. Project Overview

StudyTasks is a web-based platform that supports students in organizing study-related tasks within groups. The platform combines group management, shared task lists with responsibilities and status workflow, a calendar view for deadlines, and basic statistics for progress tracking.

This document covers the PROCESS part of the course grading: how we planned, managed, tracked, and reflected on the project work.

2. Process Model and Workflow

We used an agile-inspired approach with an explicit backlog and a Kanban board. Work was delivered incrementally in functional slices (authentication -> groups -> tasks -> calendar -> statistics), while continuously testing and refining edge cases.

Key workflow principles:

- Single source of truth: backlog and tasks maintained in a Kanban board (To Do / In Progress / Done).
- Iterative delivery: features implemented in phases and reviewed after each milestone.
- Definition of Done: feature is only moved to Done when tested end-to-end and integrated with the rest of the system.
- Security by default: endpoints protected via session-based authentication and group membership checks.

3. Project Management Setup

Tools and organization used during the project:

- Version control: Git (repository structure separated into Frontend/Backend).
- Development environment: Visual Studio / VS Code, local PHP + MySQL stack.
- Task tracking: Kanban board with clearly defined work packages and responsibilities.
- Communication: regular coordination calls and chat-based quick decisions (e.g., Teams/WhatsApp).

Milestones followed the feature phases documented in the time log (planning, architecture, auth, groups, tasks, calendar, statistics, testing/closure). After each milestone, we reviewed what worked, fixed integration issues, and refined the backlog for the next phase.

4. Backlog, Task Management and Sprint Reviews

Backlog items were phrased as small deliverables (e.g., 'Implement group creation endpoint' or 'Add calendar day view'). Each item had a clear acceptance criterion and a rough time estimate. Work progressed through the Kanban states:

- To Do (planned, not started)
- In Progress (currently being implemented)
- Done (implemented, tested, integrated)

Sprint review protocol (used for milestone reviews):

- Demo: show the implemented feature in the UI and validate the main user flow.
- API check: verify endpoints with realistic data and permission boundaries (logged-in vs. not logged-in, member vs. non-member).
- Bug/edge-case list: collect issues discovered during the review and add them back to the backlog.
- Decision log: capture decisions that affect architecture or data model (e.g., task responsibility, status workflow).

5. Time Tracking Summary

Time was tracked per work package (1-40). Each work package includes total effort and the split between Serdar and Ersan. The full detailed log is included in Appendix A.

Total project effort: 176 hours (Serdar: 89 h, Ersan: 87 h).

Effort by phase:

Phase	Total (h)	Serdar (h)	Ersan (h)
Project Start & Planning	14	7	7
Architecture & Conception	27	12	15
Authentication & Users	31	12	19
Groups (Core Feature)	25	11	14
Task Management	33	16	17
Calendar	22	17	5
Statistics	8	5	3
Testing & Closure	16	9	7

6. Collaboration and Communication

Work was split by strengths while keeping integration tight. We coordinated frequently to avoid diverging implementations between frontend, backend, and database.

- Clear interface contracts: endpoints and payloads were agreed before implementing UI integration.
- Pair debugging for integration issues (e.g., auth/session problems or permission checks).
- Regular merges and smoke tests to keep the application always runnable.

7. Quality Assurance (Testing and Edge Cases)

Quality assurance focused on functional end-to-end tests of the user flows: registration/login, group creation/joining, task CRUD, status workflow, calendar rendering, and statistics. We specifically tested edge cases and invalid inputs.

- Authentication checks: endpoints reject requests when not logged in (session required).
- Authorization checks: group endpoints and tasks require group membership.

- Input validation: required fields, valid IDs, and safe handling of empty or malformed payloads.
- Error handling: user-friendly messages in the UI and consistent JSON error responses from the API.

8. Evidence for Process Grading Criteria

The following table maps the required PROCESS elements (as communicated in the course meeting) to concrete evidence contained in this submission.

Process requirement	Evidence in this document / project
Time tracking	Section 5 and Appendix A (work packages 1-40 with per-person split).
Backlog and task management	Section 4 (Kanban board workflow, acceptance criteria, Definition of Done).
Sprint review protocols	Section 4 (review agenda: demo, API check, bugs/edge cases, decision log).
Reflection	Section 9 (what worked, challenges, improvements).
Personal contribution	Section 10 (split of responsibilities with examples).
Personal development	Section 11 (skills gained and lessons learned).

9. Reflection

What went well:

- Incremental delivery reduced risk: core flows were implemented early (auth + groups), then extended step-by-step.
- Clear separation of concerns (Frontend / Backend / DB) helped parallelize work.
- Regular integration prevented late surprises and kept the system stable.

Challenges and how we handled them:

- Integration issues (e.g., session/cookie behavior): solved through joint debugging and consistent API wrapper usage.
- Permission boundaries (member vs. non-member): addressed by central auth checks and group membership verification.
- Keeping UI and API in sync: mitigated by agreeing payload formats before implementation and revisiting them in reviews.

What we would improve next time:

- Introduce automated tests earlier (e.g., API tests) to reduce manual regression testing.
- More consistent sprint cadence with fixed review dates.
- Add a lightweight changelog to document decisions and changes per milestone.

10. Personal Contribution

The work split is also visible in the time log. In addition, responsibilities were distributed as follows (with overlap for integration and testing):

Serdar - primary contributions:

- Frontend-heavy implementation (app pages, UI logic, auth checks on pages).
- Calendar UI and interaction patterns (rendering, linking tasks to dates).
- UI for groups and tasks (lists, forms, modals) and general UX improvements.
- Documentation and submission preparation support.

Ersan - primary contributions:

- Backend structure and API endpoints (auth, groups, tasks), including session handling.
- Database model and query implementation (PDO), plus security/permission checks.
- Backend validation, error handling, and role-based admin functionality where applicable.
- Integration support and bug fixing across modules.

11. Personal Development

Serdar - learning outcomes:

- Improved structuring of a vanilla-JS frontend (modules, API wrapper, page protection).
- Practical experience implementing calendar-based UI and date handling.
- Better understanding of iterative UI development and usability-driven refinements.

Ersan - learning outcomes:

- Deeper experience with PHP backend design (endpoint structure, bootstrap pattern, reusable helpers).
- Hands-on practice with session-based authentication and authorization for group-scoped resources.
- Improved SQL/PDO usage for join-heavy queries (tasks, members, tags) and robust error handling.

Appendix A: Detailed Time Log (Work Packages 1-40)

The table below reproduces the detailed time tracking as recorded during the project. Each work package includes the total effort and the split between both team members.

#	Work package	Phase	Total (h)	Serdar (h)	Ersan (h)
1	Project idea defined	Project Start & Planning	2	1	1
2	Target group analysis	Project Start & Planning	2	1	1
3	Project goals defined	Project Start & Planning	2	1	1

4	Project scope defined	Project Start & Planning	2	1	1
5	Project plan created	Project Start & Planning	4	2	2
6	Kanban board initialized	Project Start & Planning	2	1	1
7	System architecture designed	Architecture & Conception	6	3	3
8	Frontend structure defined	Architecture & Conception	4	3	1
9	Backend structure defined	Architecture & Conception	4	1	3
10	API communication concept	Architecture & Conception	4	2	2
11	Database model designed	Architecture & Conception	5	2	3
12	Security concept defined	Architecture & Conception	4	1	3
13	User table created	Authentication & Users	3	1	2
14	User registration implemented	Authentication & Users	6	2	4
15	User login implemented	Authentication & Users	6	2	4
16	Session handling implemented	Authentication & Users	5	2	3
17	Backend auth checks implemented	Authentication & Users	4	1	3
18	Frontend auth check integrated	Authentication & Users	4	3	1
19	Logout implemented	Authentication & Users	3	1	2
20	Group creation	Groups (Core Feature)	6	2	4
21	Auto-add creator to group	Groups (Core Feature)	4	1	3
22	Fetch user's groups	Groups (Core Feature)	5	2	3
23	Groups overview UI	Groups (Core Feature)	5	4	1

24	Secure group access	Groups (Core Feature)	5	2	3
25	Task table created	Task Management	4	1	3
26	Create task (backend)	Task Management	5	1	4
27	Create task (frontend)	Task Management	5	4	1
28	Fetch tasks by group	Task Management	5	2	3
29	Task list UI	Task Management	5	4	1
30	Change task status workflow	Task Management	5	3	2
31	Task access protection	Task Management	4	1	3
32	Calendar month view	Calendar	6	5	1
33	Mark tasks in calendar	Calendar	6	4	2
34	Day view implemented	Calendar	6	5	1
35	Calendar-task linkage	Calendar	4	3	1
36	Compute task statistics	Statistics	4	2	2
37	Display statistics UI	Statistics	4	3	1
38	Functional testing	Testing & Closure	6	3	3
39	Error handling & edge cases	Testing & Closure	4	2	2
40	Documentation & submission prep	Testing & Closure	6	4	2