# AI-Driven Software Development

**Summer Semester 2025**

Fakultät für Informatik und Mathematik
Prof. Dr. G. Fraser,
Benedikt Fein,
Philipp Straubinger

## Final Project: Calendar Application (Part I)

**Publication**: 2025-06-23
**Requirements update**: 2025-07-07
**Final submission**: 2025-07-28 23:59 CEST

## 1 Introduction

For your final project, you will develop a calendar web application. Similar to the ToDo app project, this will be a full-stack web application. The frontend must be implemented using a JavaScript or TypeScript framework of your choice. The backend should be developed in a JVM-based language and include both an API layer and a connection to a database. You can choose any database implementation as long as it can run locally (e.g., H2, SQLite, PostgreSQL, MySQL, MongoDB).

Before starting your implementation, carefully consider the overall architecture of your application. While mockups are not mandatory, it is highly recommended to engage in design thinking and create prototypes to plan your user interface and user experience effectively.

## 2 Application Requirements

The application must fulfil the requirements below. In two weeks (2025-07-07), we will publish some updated and additional requirements.

### 2.1 Functional Requirements

- **User Management:** Users must be able to register, log in, log out, and change their password.
- **Calendar View:** The calendar should support both monthly and daily views. All events scheduled for the selected time period must be displayed accordingly.
- **Add Events:** Users must be able to create events. Each event must include at least a title, description, start date/time, and end date/time.
- **Edit and Delete Events:** Users must be able to modify all event details and delete events.
- **Reminders:** Users should receive browser notifications for upcoming events (only if the calendar is open).
- **Recurring Events:** Events may be set to repeat (e.g., daily, weekly, monthly).

- **Event Tags:** Events must be taggable with user-defined tags (e.g., *work, personal, university*).
- **Data Persistence:** All event data must be stored in a database and remain accessible after leaving the page and logging back in later.

### 2.2 Non-Functional Requirements

- **Usability:** The user interface should be clear, intuitive, and modern. The application must be responsive and function well on both desktop and mobile devices.
- **Security:** The application must validate user input, hash passwords securely, and use safe authentication methods (e.g., sessions or tokens).
- **Performance:** The application should load within one second on any device with a stable internet connection.

## 3 Organisational Requirements and Grading

The project is an *individual* assignment. You are *not* allowed to work in groups. You are explicitly allowed and even encouraged to use suitable AI-/LLM-based tools for all development steps. The tool choice is *not* limited to tools we presented in the lecture. Instead, you are free to choose any tools you deem suitable.

### 3.1 Development Process

While designing and implementing the features of the application, you *must* follow a simulated development process. Concretely:

- You *must* track the requirements as issues. These issues should when suitable follow the user story format with proper description, tasks, and acceptance criteria.
- You should also open new issues for other project-related tasks, e.g. for found bugs or planned refactorings.
- You should not commit directly to the `main/master` branch. Instead, create separate pull requests for self-contained changes and at least one per issue. I.e., create pull requests not only for new features, but also for example for refactorings, bugfixes, or additional tests.
- Link pull requests to relevant issues if applicable.[1] This helps you to reconstruct the development process when writing the report and us when grading.
- Since this is an individual project, you will of course not find other developers that can review your code before merging the pull request. You may use the automatic AI-based review tools instead.

[1] `https://docs.github.com/en/issues/tracking-your-work-with-issues/using-issues/linking-a-pull-request-to-an-issue`

Your application should be tested by automatic test suites including unit, integration, and system tests. You should have a line coverage of at least 90 % for both the backend and frontend parts of the system. Keep in mind that successful tests that cover the acceptance criteria of a user story are usually part of the 'definition of done'.

To ensure the coding guidelines you defined are upheld, you should define a continuous integration pipeline. This pipeline should run the test suites, linters, formatting checkers, and other automatic tools you want to use. You should aim for a 'green' pipeline on the default branch and only merge pull requests with passing pipelines.

## 3.2 Written Project Report

In addition to the application itself, you *must* submit a written report. The recommended structure of the report is:

1. System architecture and design decisions
   Give a brief overview over the high-level design decision you made while designing and implementing the application requirements. For example, you can explain technical decisions (e.g. system architecture, frontend framework choice), or in case there were requirements for which the specification allowed for multiple different approaches, explain why you chose the one you implemented. You can also explain the decisions that went into the overall design of the user interface.

2. Used LLMs and AI-based tools
   Summarise which LLMs and AI-based tools you used and how you used them. Explain for which development scenarios you used which tools.
   E.g. prompting model *a* via the web-chat for requirements elicitation, agent tool *b* with models *c* and *d* for testing, …

3. Where did the AI-tools work well?
   You should describe concrete examples, scenarios, and/or development phases where the LLM/AI-tool you used performed well.

4. Where did the AI-tools not work as intended?
   Similar as above for the case where your chosen tools worked well, also describe cases where the tool did not work as well as expected. Please describe how you adapted your usage of the LLMs to get more useful responses. Give concrete examples of issues you faced and how you adapted e.g. the prompt to resolve it.

5. Missing AI-tools (optional)
   Are there scenarios where you would have wanted to use another (AI-based) tool that can (partially) automate stuff for you, but the tool does not exist yet? If so, please describe the scenario and how the envisioned tool could help you. You can also mention tools that exist already, but you were not able to use, e.g. due to a complicated setup and/or the tool's pricing model.

There are no fixed word count or formatting requirements for this report. You can submit it for example as Markdown, AsciiDoc, or LaTeX[2] file directly in the repository. For all explanations and examples, please link relevant issues, pull requests, or commits, if applicable, to help us to better understand the report in context of your implementation.

## 3.3 Submission

The whole project *must* be developed and submitted via the repository provided via GitHub classroom:

`https://classroom.github.com/a/6hBsRUe3`

The submission deadline is Monday, 2025-07-28 23:59 CEST. You will no longer have write access to the repository afterwards. Ensure you pushed all of your code by the deadline. All artefacts (i.e., source code) including the report *must* be part of the repository itself. Only the last state of the default branch of your repository (i.e., the default state GitHub shows when looking at a repository) will be considered for grading. Issues and pull requests (open and closed) in the GitHub repository count as part of the repository.

You *must* add a file `WHOAMI.txt` at the top-most level of the repository in the following format:

```
GitHub username: XXX
Name: XXX
Matriculation number: XXX
```

with your name as it appears on Stud.IP above the profile picture on your profile page.[3] Without this file we cannot match your username to your real name and the matriculation number and cannot enter the grade in the campus portal. Submissions without this file will not be graded.

Please create a `.gitignore` file[4] in your repository that prevents common build, test, and IDE artefacts (e.g. `node_modules/`, `build/`, `target/`, `dist/`, coverage reports) from being committed and pushed.

## 3.4 Grading Environment

To be able to grade your submission, we have to check whether the required functionality of the application works as expected. For that, we need to be able to run the application and its test suite on our machine. We will not be able to grade non-executable submissions. Please provide a `README.md` file in your repository that explains the required tools, SDKs, and setup steps to be able to do run the application and its test suite(s).

You can assume the following tools are installed on our systems:

- Java JDK 17, 21, or newer
- NodeJS LTS version 18 or newer

[2]For LaTeX please also add the compiled PDF either in the repo or as CI artefact.

[3]`https://studip.uni-passau.de/studip/dispatch.php/profile`

[4]`https://git-scm.com/docs/gitignore`

- Standard build tools/package managers: Maven, Gradle, NPM, Yarn (and other build tools like Vite, Webpack, ... that are automatically pulled in as dependencies of the project using the primary build tool/package manager)
- Chrome & Firefox browsers
- Docker/Podman with docker-compose to build and run containers
- Standard Unix commandline tools (e.g. bash, grep, sed, ...)

If we need tools not mentioned here to build and run your application, you *must* ask us *before* the submission deadline about that, preferably via a *non*-anonymous question in the StudIP forum or in one of the Q&A sessions. Only then can we figure out if using the tool is okay, or if that would block us from positively grading your submission.

### 3.5 Grading

To decide on the final grade for this assignment and by extension the whole course, we will take into account the following points, including but not limited to: successful submission of the weekly assignments for the ToDo-app, fulfilment of the requirements outlined in Section 2, a development process following the requirements from Section 3.1, your implementation (e.g., it following a sensible architecture that matches the one described in your report), sufficient tests for the application (cf. Section 3.1), and content and quality of the written report (cf. Section 3.2).