Automated Software Engineering
Course Project
2025/2026

1. Introduction

In this project, you will develop a book/movie recommendation system. Think of it like building your own personal librarian or film curator who learns your taste and makes suggestions. The twist is that you will create this system using two approaches: the traditional software development workflow and an AI-assisted workflow using Large Language Models (LLMs).

Beyond software implementation, you will also apply DevOps practices by setting up a continuous integration and continuous delivery/deployment (CI/CDE/CD) pipeline. This will allow you to automate testing, deployment, and quality assurance, giving you hands-on experience with modern software engineering practices.

2. Objectives

The project is designed to help you experience the **end-to-end software development lifecycle (SDLC)**, while comparing traditional and AI-assisted workflows, and integrating DevOps practices. By the end of the project, you should:

- Experience the full SDLC, from requirements to deployment;
- Compare the effectiveness of traditional development versus AI-assisted development using LLMs;
- Apply DevOps principles to automate parts of the workflow;
- Configure and use CI/CD pipelines to support testing, integration, and deployment;
- Develop autonomy and self-learning skills in applying modern software engineering practices.

3. Project Structure

The project will be completed in teams of 5 students. You will:

- Define requirements and design the system;
- Implement the system traditionally (manual coding, manual testing);
- Implement the system using LLMs for coding, testing, documentation, and deployment;
- Set up a CI/CDE/CD pipeline to automate builds, tests, and deployments;

- Deploy the system to a cloud or container-based environment, using the most adequate deployment pattern for your system;
- Compare and reflect on the two approaches.

4. Project Description: Book/Movie Recommendation System

The goal of the project is to design and implement a recommendation system that helps users discover books or movies they might enjoy based on their preferences and past interactions. This system should simulate a real-world application where users can register, browse items, rate them, and receive personalized recommendations.

Key Features to Consider

- User management: registration, login, and profile management.
- Catalog browsing: ability to view a list of books/movies with details (title, genre, description, rating).
- Search functionality: search items by title, author/director, or genre.
- User ratings: allow users to rate books/movies (e.g., 1–5 stars).
- Recommendation engine: suggest items based on user history, ratings, or popularity.

Technical Requirements

The system should include the following technical aspects:

- Backend service (*e.g.*, REST API or GraphQL) to manage data and user requests;
- Database (SQL or NoSQL) to store users, items, ratings, and recommendations;
- Frontend (web or mobile) to allow user interaction with the system;
- CI/CDE/CD pipeline for automated building, testing, and deployment;
- Deployment in a containerized or cloud-based environment.

Constraints and Expectations

The project should remain manageable within the semester. You are encouraged to start simple and incrementally add features. The comparison between traditional and AI-assisted development and the configuration of a complete CI/CDE/CD pipeline are the central goals, so ensure that both aspects are applied to significant parts of the system.

5. Project Phases

Requirements & Design

Traditional: user stories, diagrams, architecture documents.
AI-assisted: prompt an LLM to refine requirements and propose designs.

Implementation

Traditional: manual coding following best practices.
AI-assisted: code generation, refactoring, and bug fixing with LLMs.

Testing & Verification

Traditional: manually write unit and integration tests.
AI-assisted: generate test cases with LLMs and compare coverage.

CI/CD Pipeline

Set up automated builds, testing, and deployment using GitHub Actions, GitLab CI, Jenkins, or Azure DevOps.

Deployment & Monitoring

Deploy the system in containers or cloud environment with monitoring enabled.

Reflection & Comparison

Document lessons learned, advantages and disadvantages of both approaches.

6. Deliverables

Source code repository with both traditional and AI-assisted codebases.
CI/CD pipeline configuration and logs.
Deployed system (link or container image).
Final report including a comparison between approaches.
Presentation and demonstration.

7. Evaluation Criteria

- Process & Autonomy (30%) – Evidence of teamwork, self-learning, and proper use of version control.
- Technical Implementation (30%) – System functionality, quality of code, CI/CD correctness, and deployment.
- Comparison & Analysis (30%) – Depth of analysis between traditional and AI-assisted approaches, supported by data and examples.
- Presentation & Reporting (10%) – Clarity of the final report and presentation, including reflection.

This is a group-project, but the grades are individually assigned.

8. Grading Rubric

The table below summarizes the evaluation criteria, their weight, and what is expected for each component.

Criterion	Weight	Description
Process & Autonomy	30%	Teamwork, task division, evidence of self-learning, and effective use of version control.
Technical Implementation	30%	System functionality, code quality, working CI/CD pipeline, and successful deployment.

Comparison & Analysis	30%	Depth of analysis between traditional and AI-assisted workflows, supported by data, metrics, and examples.
Presentation & Reporting	10%	Clarity, structure, and professionalism of the final report and presentation, including reflection on lessons learned.

9. Milestones and Deliveries

Milestone 1 - Requirements (5%) – October 17th

Milestone 2 – Code created as part of a CI/CD pipeline (5%) – November 14th

Final Delivery (90%) – December 12th

- **Source code and Scripts:**
  - Include the source code, scripts, files, and libraries necessary for compiling and running the software
- **Presentation** (e.g., PowerPoint) with the following information:
  - Name of the project
  - Team members and contacts
  - Describe the structure of the software
  - Describe tests and quality evaluation
  - Development plan: make sure you specify which tasks were done by each team member and the effort involved (e.g., hours)
- **Self- and Peer-evaluations**
  - You will be given a form to evaluate your performance and that of your team members.

Defenses

On the final delivery milestones, the group must make a brief presentation of the project highlighting the main characteristics. More precisely:

- The presentation must include the topics previously described for inclusion in each submission
- Do a live demonstration of your software
- Prepare yourself to answer (individual) questions regarding all deliverables and implementation details
- Sign up for any available time slot for the defense in Inforestudante until the final delivery due date (the list of available slots will be released before the deadline for the final delivery)
- Presentation and demo should not exceed 15 minutes