UNIVERSIDADE DA BEIRA INTERIOR
Faculdade de Engenharia | Departamento de Informática

Engenharia de Software - 2025/26

Project

Overview

The project is an integral part of this course. You will work in teams to design and implement a solution consisting of both back-end and front-end components. Throughout the project, you will apply the knowledge gained in class on software requirements, design, architecture, modelling, coding, testing, and continuous integration and deployment.

Goals

1. To create something useful;

2. To learn about the software engineering development cycle;

3. To practice Agile methods in an immersive academic context;

4. To leverage existing AI-based tools to enhance the development process;

5. To gain valuable experience and enjoy the journey.

Restrictions

There are several **hard** restrictions on the project:

1. The software must include both back-end and front-end implementations. The use of modern frameworks such as **Django**, **React**, or **Node.js** is highly recommended;

2. The software should aim to be useful and may integrate third-party modules or services (e.g., APIs);

3. The code must include the development of **at least four user interfaces** beyond basic database CRUD operations and standard login/registration functionalities;

4. The solution must feature a **well-justified architectural style**, accompanied by **wireframes** and **mockups**;

5. All project activities should be **organized and regularly updated** in the **Jira** tool, with access granted to the instructors;

6. Software requirements must be **elicited using the Character.ai tool**;

7. Each team is required to adopt at least one **AI-based tool** to support programming and code development; the choice of the tool is entirely up to the team, allowing for autonomy and experimentation;

8. In each sprint, a team member must be **nominated as the Team Captain (Scrum Master)** on a rotating basis, ensuring that **each member takes on the role once** during the project.

9. In each sprint, one team member must take on the role of **Product Owner.** The Product Owner is responsible for **managing and prioritizing the Product Backlog**, ensuring that the team focuses on the most valuable tasks. The role must be rotated so that **each team member serves as Product Owner once**, and this must be documented in the team's project deliverables.

In addition, consider the following **soft** restrictions when choosing the project topic:

1. Each team should be composed of **5 students**. Any exceptions must be approved in advance by the professor;

2. All team members are expected to actively participate in team activities, including attending in-person lectures;

3. **Recommendation:** Team members should, whenever possible, be enrolled in the same practical class group of the course.

Team Formation

Deadline: **26.09.2025**

How to apply: https://forms.gle/P2ay1U2FF6JaGdnv7

Project Setup

Deadline: **19.12.2025**

Once your team is formed, you must create a **Jira dashboard** for your project. Suggested boards include: **Product Backlog**, **To Do**, **In Progress**, **Done**, **Artifacts**, **Daily SCRUM,** and **Team Contract**. Additionally, make sure to **grant access to the instructors** using the following emails: pombo.ubi@gmail.com, xxx.

Jira Board Structure:

- Product Backlog: List of all features, enhancements, and requirements to be addressed throughout the project;

- **To Do, In Progress, and Done:** Columns representing the current status of project tasks and activities;

- **Artifacts:** Links to external project-related files, such as **wireframes**, **mockups**, **modeling diagrams**, **GitHub repository**, and other relevant documents;

- **SCRUM:** Evidence of SCRUM compliance — including **daily team meetings**, **sprint planning**, **sprint reviews**, and **retrospectives** — must be provided. This evidence can be textual, image-based, or a combination of both.

- Team Contract: A content outlining key team agreements, including: a) expectations for team meetings, b) team and individual responsibilities, c) strategies for managing challenges and resolving conflicts, and d) any additional considerations important for team functioning.

Project Proposal

Title: Development of a Software Incident Management System

Software incident management is a critical process for ensuring the stability and quality of IT systems. Many organizations currently use generic or manual tools to log, classify, and resolve incidents, which can lead to inefficiencies, lack of traceability, and delays in resolution.

In this project, students are required to follow the complete software development lifecycle for a dedicated incident management system, starting with requirements identification—where functional and non-functional needs are gathered and documented—followed by system design, which includes creating architectural, interface, and data models (such as UML diagrams and wireframes), and culminating in implementation, where the technical stack, development approach, and integration strategies are defined to build a functional prototype or proof-of-concept.

The system should support the following functionalities:

-Logging and categorization of incidents.
-Automatic or manual prioritization of incidents.
-Tracking the status and resolution history.
-Integration with monitoring tools (e.g., Nagios, Datadog).
-Generation of reports and metrics (e.g., MTTR, number of incidents by type).
-Automatic notifications for responsible teams.

The system must be scalable, intuitive, and integrable with other systems used by the organization.

Methodology

The project will follow the SCRUM methodology and will be divided into five sprints. The first four sprints will each last two weeks, while the final sprint will last four weeks. The first sprint **will begin on September 29**.

Sprint #1: from September 29 to October 10

Sprint #2: from October 13 to October 24

Sprint #3: from October 27 to November 7

Sprint #4: from November 10 to November 21

Sprint #5: from November 24 to December 19 (final delivery))

The practical classes held during the **first week of each sprint** will include **cumulative work related to the practical project**. Each team must present a **summary of the work completed** during the **practical classes of the second week of each sprint**.
Each team will be responsible for managing its own **sprint backlog**. In each sprint, it will be possible to introduce **improvements to features addressed in previous sprints**.

Expected Outcomes

By the end of the project, each group is expected to have produced the following artifacts:

1. **Definition of Personas, Scenarios, and User Stories**
2. **Prototype (Wireframes)** – using tools such as Figma, Miro, etc.
3. **System Architecture**
4. **Software Design**
    a. Software modelling, including:
        i. Use case diagrams
        ii. Class diagrams
        iii. Activity diagrams
        iv. Sequence diagrams
        ...
5. **Mockups**
6. **Programming and Refactoring**

Assessment

C1) **Application of the SCRUM Methodology** – 15%

Assessment of the team's ability to apply SCRUM practices consistently and effectively throughout the project, including:

- Clear definition and regular updates of the **Product Backlog** and **Sprint Backlogs**;
- Structured execution of **Sprint Planning**, **Daily Standups**, **Sprint Reviews**, and **Sprint Retrospectives**;
- Visible progress through **incremental delivery** of working software components;
- Documented **evidence of SCRUM rituals**, including text/image-based logs in Jira or equivalent tools;
- Proper role rotation with clear assignment and documentation of the Team Captain and Product Owner roles per sprint.

C2) **In-Class Participation and Intermediate Presentations** – 30%

Evaluation of individual and group engagement during class sessions and check-in presentations:

- Active **participation** in weekly team activities and in-person sessions (attendance and contribution);
- Demonstrated understanding and **domain mastery** through applied tasks and discussions;
- **Coherence** and **consistency** between deliverables (e.g., requirements, design, wireframes, mockups, and implementation);
- Quality of intermediate presentations, assessed based on:
    - Clarity and structure of communication;
    - Correct use of technical terminology and modeling notation;
    - Visual organization and ability to explain design decisions;
    - Objective and critical reflection on project progress and challenges.

C3) **Application of Software Engineering Principles** – 55%

Evaluation of the quality, completeness, and rigor in applying software engineering concepts across all project stages:

- Requirements Engineering: quality of personas, user stories, functional/non-functional requirements, and specification documents;
- Prototyping: effectiveness and usability of wireframes and mockups;
- System Architecture: well-justified architectural style, modularity, scalability, and documentation;
- Software Design: clarity and correctness of diagrams (use case, class, activity, sequence, etc.);
- Implementation and Refactoring: code quality, adherence to best practices, version control usage, incorporation of feedback, and partial system development (as specified);
- Use of AI tools: appropriate and autonomous integration of AI-based tools for requirement elicitation, code support, and productivity enhancement.

NOTE#1: *This project will be graded on a* ***0 to 20 scale****.*

NOTE#2: *Under the* ***continuous assessment (ensino-aprendizagem) model****, this project accounts for* ***8 points of the final course grade*** *(i.e.,* ***40%****). The final contribution of the project to the course grade will be calculated using the formula:* ***Final Score = A1 × 8 / 20***

Under the ***exam-based assessment model****, the practical component related to this project will be* ***individual****, with an* ***assignment defined by the course instructor****.*