Software Construction and Testing Project Winter 2025

Dr. Ahmed Maghawry
TA. Nadeen Serag
TA. Menna Singergy

1 Project Overview

This course project is a team-based assignment designed to help you apply the principles of software construction and testing in a real-world setting. The project emphasizes good software engineering practices, clean code, testing, and teamwork. **Objective:** Deliver a fully functioning software system that demonstrates construction, testing, and deployment best practices.

1.1 Team Formation

Teams should consist of 5-6 members and must be formed by **04/10/2025** (within one week after this document is released).
Teams must submit their member list via: `<https://forms.gle/WVAVHWa5x5LVtnNX8>`

1.2 Project Ideas

You may choose from the following suggested domains, or propose your own idea:

- **Restaurant Reservation and Management System :** Implement a web application where customers can reserve tables, order food, and provide feedback. The admin side could manage bookings, view feedback, and update menu items.
- **E-commerce Website :** Develop a website that allows users to browse, search, purchase products online, and get notifications while providing a seamless shopping experience and robust administrative features. The
- **Personal Finance Tracker with Budget Recommendations :** Create an app to track expenses and incomes, set budgets, and offer spending recommendations. Include data visualizations and reports.
- **E-Learning Course Management System :** Build an online course platform where instructors can create courses, add lessons, upload resources, and manage students. Students can enroll, view content, and track their progress.
- **Your Own Idea:** You may propose a project idea different from the suggested ones, but you must first discuss it with your TA.
  Note: You may also reuse an idea from another course, as long as it is clearly defined, adds value, and has a well-scoped objective.

1.3 Points To Be Covered In Project

The points that **must** be covered throughout the project:

- **Programming Paradigms:** Demonstrate the use of both declarative and imperative programming styles.
- **Design Patterns:** Apply and justify the implementation of appropriate design patterns.
- **Test-Driven Development (TDD):** Show evidence of test-first development and integration of TDD practices in the project.
- **Testing Techniques and Coverage:** Apply unit testing, integration testing, and end-to-end testing. Ensure test coverage includes both front-end and back-end components.
- **Code Quality:** Adhere to clean code principles, SOLID principles, and separation of concerns. Evaluate code maintainability, readability, and modularity.
- **User Stories:** User stories should be written to capture important functional needs. These will guide development and serve as the basis for designing test cases. A user story usually follows the format:

  As a [role], I want [feature] so that [benefit].

  Ex: *As a customer, I want to reserve a table online so that I can secure a booking before arriving at the restaurant.*

  Note: You do not need to write a user story for every feature. Focus on the key scenarios most relevant to your system.

2 Milestone 1 - Idea & Architecture

Deadline: **by week5** - 11/October/2025 - (10% of the total grade)

- **Task:**
  1. **Project Idea:** Provide a one-page document with objectives, scope, and expected outcomes. Include justification of the chosen idea.
  2. **Requirements:** State the most important Functional and Non-Functional Requirements with User Stories to guide development and form the basis for test case design.
     Each student must cover at least one FR and one NFR to present in the beta milestone.
     Ex: For a team of 6, include at least 6 FRs and 6 NFRs.
  3. **System Architecture:** Select an architecture type (Monolithic, Layered, or Microservices) and explain why it is suitable for your project.
- **Bonus:** Architecture diagrams (Class, Sequence, Entity, etc.) to show system structure and interactions. Diagrams make design easier to understand, flaws easier to detect, and communication clearer.
- **Submission:** A Google Form will be provided to upload your work.
- **Evaluation:**
  - Idea relevance & clarity
  - Architecture justification & structure
  - Requirements quality and completeness

3 Milestone 2 - Working Beta Version

Deadline: **by week10** - 25/November/2025 - (15% of the total grade)

- **Task:**
  1. **Working Beta Code:** Deliver a beta version of the software without syntax errors, demonstrating the happy path scenario (end-to-end working software).
  2. **Code Quality Evidence:** Apply SOLID principles, maintain separation of concerns, and use modular, clean coding practices.
  3. **Programming Paradigms:** Apply both imperative and declarative programming styles.
- **Submission:** Upload your beta code and documents via a Google Form. Bring the working project on your laptop for evaluation (max 30 minutes).
- **Evaluation:**
  - Functional happy path
  - Features aligned with the architecture chosen in M1
  - Code quality
  - Programming paradigms

  Note: Any feedback provided from the evaluator in this milestone should be applied in M3

4 Milestone 3 - Final Delivery

Deadline: **by week13** - 16/December/2025 - (15% of the total grade)

Apply feedback provided from M2 or further enhance the project (e.g., refactoring, code reviews).

- **Deliverables:**
  1. **Completed System:** All planned features implemented with proper error handling.
  2. **Integration:** Front-end and back-end integrated with smooth and correct data flow.
  3. **Design Patterns Application:** At least two design patterns.
  4. **Testing Package:** Unit, Integration, and End-to-End (E2E) tests. Include TDD evidence. Test cases must be traceable back to user stories to ensure full coverage of functional requirements.
- **Bonus:** Deploy your work and show it running.
- **Submission:** Upload your final code and documentation via a Google Form. Bring the working project on your laptop for the 40-minute discussion.
- **Evaluation:**
  - Full functionality
  - Integration between modules
  - Testing & quality
    - Unit Testing
    - End-to-End
    - TDD principles

5 Cheating Cases

- **Duplicate Work Across Teams:** If two teams submit the same code, both will receive zero.
- **Unattributed Copying:** Code or diagrams reused from online sources (e.g., GitHub) without proper attribution will result in zero.