Zewail City of Science and Technology — مدينة زويل للعلوم والتكنولوجيا

Course Project: CSAI 203, Introduction to Software Engineering

Project Objective

The purpose of this course project is to provide students with hands-on experience in applying fundamental software engineering practices. By working in teams, students will learn how to define, design, implement, and deliver a software product that meets real-world requirements. The project also aims to improve collaboration, communication, and documentation skills, preparing students for future software development work.

By completing this project, students will be able to:

- Apply the software development life cycle (SDLC) phases in practice.
- Formulate a problem statement and define the requirements of the software.
- Prepare proposal and design documents using appropriate notations.
- Practice team collaboration and version control (e.g., Git, GitHub).
- Implement a software system incrementally, following good coding practices.
- Deliver a functional software application with partial features (mid-project).
- Practice project management skills, including phase planning and progress tracking.
- Prepare technical and user documentation to support the deployment and use of the software.
- Demonstrate the ability to present and showcase a working software.

Project Phases Timeline

Project Phases	Deliverables	Grade (%) of 20	Deadline
Project Announcement & Group Formation	Team list, Chosen project topic/title, two project ideas, short descriptions, git repo url …etc	**5% (1 Grade)**	Saturday 11/Oct/2025: 11:59 PM
Requirements Specification Document	SRS documents describing what the software will do before moving to design and implementation.	**20% (4 Grades)**	Saturday 1/Nov/2025: 11:59 PM
Design Document	Design document including the different diagrams defining how the system will be built…etc.	**20% (4 Grades)**	Saturday 22/Nov/2025 : 11:59 PM
Core Functionality Prototype	Core features working - about 50% of its intended functionality..etc.	**25% (5 Grades)**	Saturday 6/12/2025: 11:59 PM
Final Delivery & Presentation	Fully functional application with 100% of features implemented, deployment, presentation..etc.	**30% (6 Grades)**	Saturday 20/12/2025: 11:59 PM

Project Phases Details:

Phase 1: Project Announcement & Group Formation

- **Deliverable:**
  - Team list
  - Chosen project topic/title
  - Each group submits **two project ideas** (Primary and backup ideas).
  - Short Description of the project idea and what it is supposed to do.
  - A form will be created for students to submit their group members and ideas.
  - Members are highly encouraged to be from the same lab group.
  - Create a GitHub repository and share access to it with your assigned TA.

Phase 2: Software Requirements Specification (SRS)

- **Deliverable: SRS Document** including:
  - Introduction (problem, objectives, scope)
  - Stakeholders and target users
  - Functional requirements (feature list, use cases, scenarios)
  - Non-functional requirements (usability, performance, constraints)
  - Assumptions and dependencies
- **Purpose:** Define **what the software will do** before moving to design and implementation.

Phase 3: Design Document

- **Deliverable:**
  - Use case diagram(s)
  - System architecture diagram
  - UI wireframes/mockups (You may use: PowerPoint, Draw.io)
  - Database schema (if applicable) or File schema
  - **The Model–View–Controller (MVC) pattern is mandatory. Explanation of how the Model–View–Controller (MVC) design pattern is applied in the project.** Teams must clearly identify the responsibilities of each component (Model, View, Controller) in their design.
- **Purpose:** Define **how the system will be built**.

Phase 4: Core Functionality Prototype

- **Deliverable:**
  - Running prototype (core features working - about 50% of its intended functionality).
  - Progress report (linked to SRS).
- **Purpose:** Ensure steady progress and early testing.

Phase 5: Final Delivery & Presentation

- **Deliverables:**
  - Fully functional application with 100% of features implemented
  - Deployment
  - Project documentation (user + technical)
  - Testing evidence: (At least a ~3-4 **unit tests** for major functions/features)
  - A short description of how testing was done (manual + automated if possible)
  - Final presentation & demo in class
- **Purpose:** Showcase the completed product and reflect on the development process.

Team Formation Phase:

- The number of students per team is **3 to 4 students**.
- A group of 5 is allowed, but extra work will be required as described in project scope.
- Team members are encouraged to be from the same lab. Team members can be from different LABS. But NOTE that: It will be your FULL RESPONSIBILITY to attend the discussion. Any student who misses the scheduled discussion will automatically receive ZERO.
- Each group will assign a representative student as the primary point of contact for communication with TAs.
- The deliverables discussion will take place in the lab after the online submission deadlines.

Note: Workload distribution should be vertical. A student is expected to work across all layers in the software engineering life cycle.

Project Idea Selection:

Students are free to propose their own project ideas, but the project must be:

- **Feasible** within the course timeline (about 10–12 weeks of actual work).
- **Larger teams (5 students):** Must add more features as explained in project scope.
- **Relevant** to Software Engineering principles (so the project involves clear requirements, design, implementation, and testing).
- Creative ideas are welcome, but they must remain achievable within the course timeframe.
- Should address a practical problem or simulate a real application.
- **Two proposals required:**
  - Each student/team must submit **two project ideas**:
  - **Primary candidate** (the main project they want to pursue).
  - **Backup candidate** (an alternative, in case the primary project is rejected or judged infeasible).

Project Scope:

- Functional Requirement Features (10 required):

- These are the visible, user-facing features that the software must perform (e.g., login, search, checkout, reports).
- Each team must implement at least **10** functional requirement features.
- **Team of 5, must add four more functional requirements features.**
- Features should cover both front-end and back-end logic (data storage, processing, rules).

- Non-Functional Requirement Features (2 required):

- These define quality attributes of the software rather than direct functionality.
- Each project must address at least **2** non-functional features, such as:
  - Performance (e.g., system responds within 2 seconds).

    - Security (e.g., password hashing, role-based access).
    - Usability (e.g., mobile responsiveness, intuitive navigation).
    - Reliability (e.g., error handling, stable under concurrent users).
    - Maintainability (e.g., code is modular, documented).
  - **Team of 5, must add <u>two</u> more non-functional requirements features.**

Project Example (E-commerce Software)

- **Functional Requirements:**
  - User login/signup
  - Product catalog browsing
  - Search/filter products
  - Shopping cart
  - Checkout with payment simulation
  - Order history
  - Admin product management
  - User profile management
- **Non-functional Requirements:**
  - **Usability:** Mobile-responsive interface
  - **Security:** Passwords stored securely (hashed)

Project Complexity: Must include both:

- **Simple Front-end:** based on **HTML**, e.g., forms, navigation, and user interaction.
- **Back-end logic:** based on **Flask- Python**, e.g., data management, processing, business rules.

Suggested Project Ideas

To help you get started, here are some suitable project examples:

- **Library Management Software** (book borrowing, returns, search, student accounts).
- **Student Club/Event Management Software** (event scheduling, announcements, registrations).
- **Simple E-commerce Software** (products, shopping cart, checkout, user accounts).
- **Online Polling/Voting Software** (secure login, vote submission, results).
- **Simple Inventory/Point-of-Sale Software** (stock management, sales transactions).

GitHub Repository Structure

- Each team must maintain a single GitHub repository for their project.
- The repository must follow a clear structure, with at least the following folders:
  - **/docs** → Contains all submitted documents (SRS, Design, Reports, Testing Evidence, etc.).

- **/src** → Contains all source code (Flask backend + HTML frontend).
- **/tests** → Contains test cases (unit tests, manual test reports if applicable).
- **/deployment** → Contains deployment instructions, configuration files, and environment setup scripts (if any).

- A **README.md** file must be present in the root of the repository, clearly describing:
  - Project title and team members
  - Short project description
  - Instructions for setting up and running the application
  - Links to relevant documents in /docs
- Commits must be meaningful (e.g., "Added login feature" instead of "fix") and reflect contributions from all members.

Important Rules:

1. The number of students per team is **3 to 4 students**. A group of 5 is allowed, but extra work is required.
2. Teams are encouraged to be from the same lab. Team members can be from different LABS, but NOTE that: It will be your FULL RESPONSIBILITY to attend the evaluation session. Any student who misses the scheduled evaluation session will automatically receive ZERO.
3. Projects are not allowed to be a reuse of another course project.
4. If a student is retaking this course, they must select a new project idea different from their previous one.
5. For each phase, **if a late submission is permitted, a 25% penalty will be incurred** for each day after the deadline. After two overdue days, no submissions will be accepted.
6. Each phase's deliverables must be uploaded to the **Google Classroom** assigned assignment and pushed to the repository **before the submission deadline**.
7. The deliverable version uploaded to the **Google Classroom** assignment is the one that will be evaluated. The GitHub version will just be a reference to track progress.
8. **Submissions that do not follow the cover page and naming convention may receive up to a 10% penalty for that phase.**
9. **Version Control:** All projects must use GitHub to track progress. Commits should reflect contributions from all members. **GitHub is mandatory in the evaluation process starting Phase-2** and must follow the "GitHub Repository Structure" mentioned. Failing to maintain an updated GitHub repository for your projects may result in a penalty of **up to 20%** for the phase under evaluation.
10. All team members are expected to contribute fairly across the project. Unequal contribution may result in individual grade deductions.
11. **Although this is a team project, every member is expected to fully understand all submitted work. TAs may ask any student about any part of the deliverable** (diagrams, code, documentation, etc). Any student unable to explain their project/code may lose marks individually, even if the deliverable was submitted.
12. **Cheating** could result in a negative grade (up to - 10 Course Grades).

13. If your project idea includes Artificial Intelligence (AI), you are NOT required to train models from scratch. You are only allowed to use pre-trained models to implement the intended feature(s). Efforts spent on training models will not be considered in the evaluation.
14. Each phase submission (when needed) must include **clear documentation** as required (requirements, design diagrams, progress reports, testing evidence, etc.).
15. A **final presentation and live demo** are mandatory (Failing to do so may lead to a big % penalty).
16. The project must be implemented using the **Python Flask framework (Bankend) + HTML (Frontend).**
17. The requirements for the frontend UI are **basic** (HTML), using UI design tools such as Figma is not allowed in this course. Please ensure that your non-functional requirements fall within these limitations.
18. Auto-code generation tools (such as UI Design to Code or System Design to Code, or Diagrams to Code) are not permitted for use and will be considered **cheating**.
19. Grading focuses on **software engineering practices** (requirements, design, implementation, design patterns, testing, documentation), not just mastering a specific coding technology.
20. The Model–View–Controller (MVC) pattern is **mandatory** and must be clearly reflected in both the Design Document and the code structure of the project. Teams are required to describe how MVC is applied (Model, View, and Controller responsibilities) as part of their Design Document.

Bonus (Total 3 Grades)

1- Modularization across the project: Split a large Flask app into modules. (Bonus 5% ~ **1 Grade**)
2. Factory Pattern across the project (App Factory). (Bonus 2.5% ~ **0.5 Grade**).
3. Repository Pattern (for Database Access or File Access). Isolate all database queries in a "repository" layer (Bonus 2.5% ~ **0.5 Grade**).
4. Singleton Pattern for Database Connection or File Connection. (Bonus 2.5% ~ **0.5 Grade**).
5- Use of CSS to improve UI across the project (clean layout, responsive design, consistent theme) (Bonus 2.5% ~ **0.5 Grade**).

Document Submission, Cover Page & Naming Convention

- All submitted documents (SRS, Design, Reports, etc.) must have a cover page and follow a strict naming convention:
  CourseCode_ProjectPhase_TeamID_RepresentativeID.pdf

- **Example:**
  CSAI203_SRS_Team05_20231234.pdf
  CSAI203_Design_Team05_20231234.pdf

- TeamID will be assigned at the time of project announcement.
- RepresentativeID is the University ID of the team representative.

Project Deliverable Documents

All project deliverables should have the following cover page:

Zewail City of Science, Technology and Innovation
University of Science and Technology
School of Computational Sciences and Artificial Intelligence

CSAI 203 - Fall 2025

Introduction to Software Engineering
<Project Name>

<Report Title (Deliverable Description)>
Team Number: #
Team Members:
<Name> <ID>

Representative Contact info:
Write one email to be able to contact your team.
<Date>

Evaluation Criteria by Phase

Phase 1: Project Announcement & Group Formation (5%)

- Team list complete and submitted on time (2%)
- Two project ideas submitted (primary + backup) with short descriptions (2%)
- GitHub repository created and shared with TA (1%)

Phase 2: Software Requirements Specification (SRS) (20 %)

- **Clarity & Completeness of Document (5%)**
  - Clear problem definition, objectives, and scope.
  - Identification of stakeholders and target users.
- **Functional Requirements (9%)**
  - At least 10 well-defined functional requirements with use cases/scenarios.
- **Non-functional Requirements (4%)**
  - At least 2 well-defined quality attributes (e.g., security, performance).
- **Organization & Presentation (2%)**
  - The document is well-formatted, and its naming conventions are followed.

Phase 3: Design Document (20%)

- **Use Case Diagram(s) (5%)**
  - Correct and covers all major features.
- **System Architecture (5%)**
  - Logical architecture diagram with clear modules/components.
  - Must clearly illustrate the **Model–View–Controller (**MVC**)** architecture and explain how it is applied in the project.
- **UI Wireframes/Mockups (5%)**
  - At least 3–4 screens, clear flow, aligned with requirements.
  - Provide **simple sketches** or **tool-generated mockups** that show the main screens or pages of your application (3–4 key screens minimum).
  - Focus on **functionality and layout**, not artistic design.
  - You may use **any of the following**: PowerPoint, Draw.io
- **Database/ or File Schema (3%)**
  - Entities, relationships, primary keys, etc.
- **Document Organization (2%)**
  - Well-structured, neat, consistent naming conventions.

Phase 4: Core Functionality Prototype (25%)

- **Feature Implementation (14%)**
  - At least 50% of core features are functional.

- **Consistency with SRS & Design (4%)**
  - Prototype aligns with earlier requirements and design docs.
- **Code Quality & Version Control (4%)**
  - Proper use of GitHub (commits by all members, clear commit messages).
- **Progress Report (3%)**
  - Linked back to SRS, a short explanation of completed vs pending features.

Phase 5: Final Delivery & Presentation (30%)

- **Full Functionality (10%)**
  - 100% of functional and non-functional requirements completed.
- **Deployment (5%)**
  - Application deployed and accessible.
- **Documentation (5%)**
  - Technical + User documentation included.
- **Testing Evidence (5%)**
  - Require at least three automated unit tests (using unittest or pytest).
  - Require at least 1 integration test (e.g., testing a route end-to-end).
  - Require test report (pass/fail summary).
- **Presentation & Demo (5%)**
  - Clear, engaging, and each member presents.
  - Working demo in front of class.