Software Engineering

IT Department

Lecture 1: Course Overview & Introduction to Software Engineering

Halal Abdulrahman Ahmed

Agenda

- Course Overview & Objectives
- Assessment & Grading Policy
- Presentation Guidelines & Topics
- Submission Policy
- Classroom Rules & Expectations
- Introduction to Software Engineering

Course Overview

- Software Engineering is about **designing, building, testing, and maintaining software** in an organized way.
- It helps create reliable, secure, and high-quality systems that meet user needs.
- In this course, you will learn how software is developed from planning to delivery and how teams work together to build real-world systems like university portals or hospital apps.

Course Objectives

By the end of this course, you will be able to:

•Understand what Software Engineering is and why it is important.

•Describe the **Software Development Life Cycle (SDLC)** and its main phases.

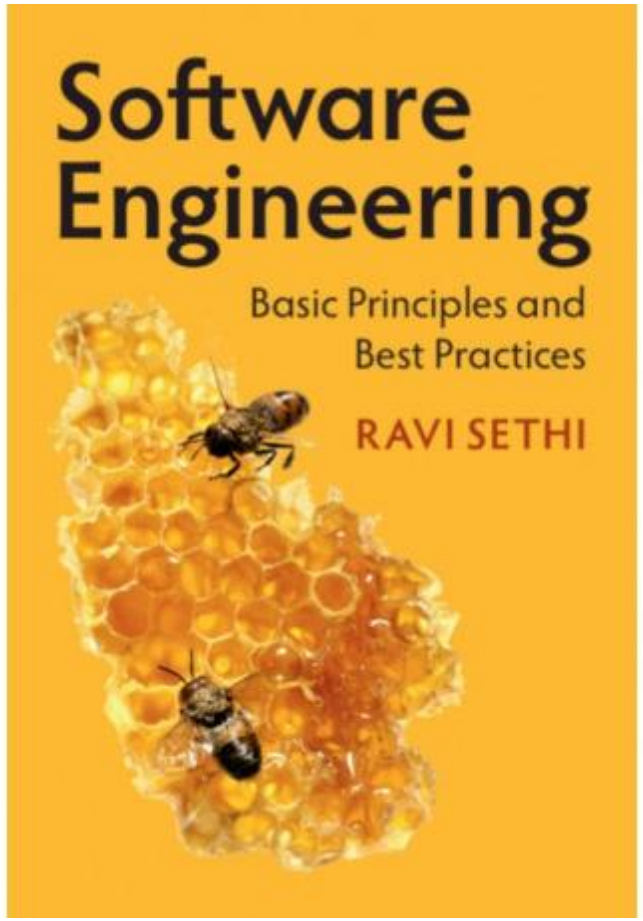
•Explain and compare different **software process models** (Waterfall, Incremental, Spiral, Agile).

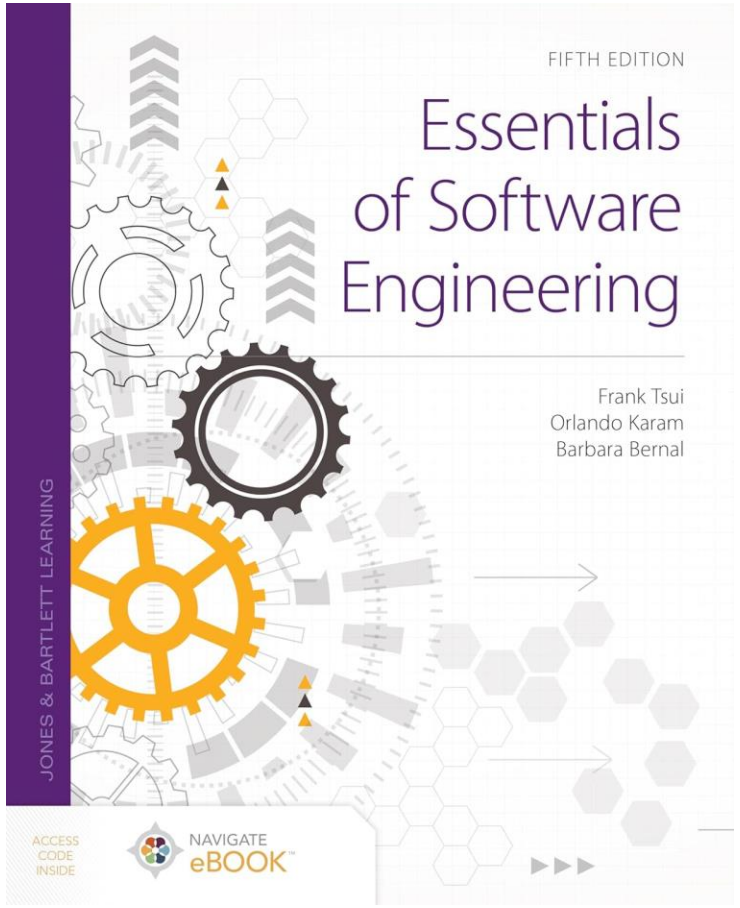
•Understand the roles of **software engineers, developers, and testers** in a project.

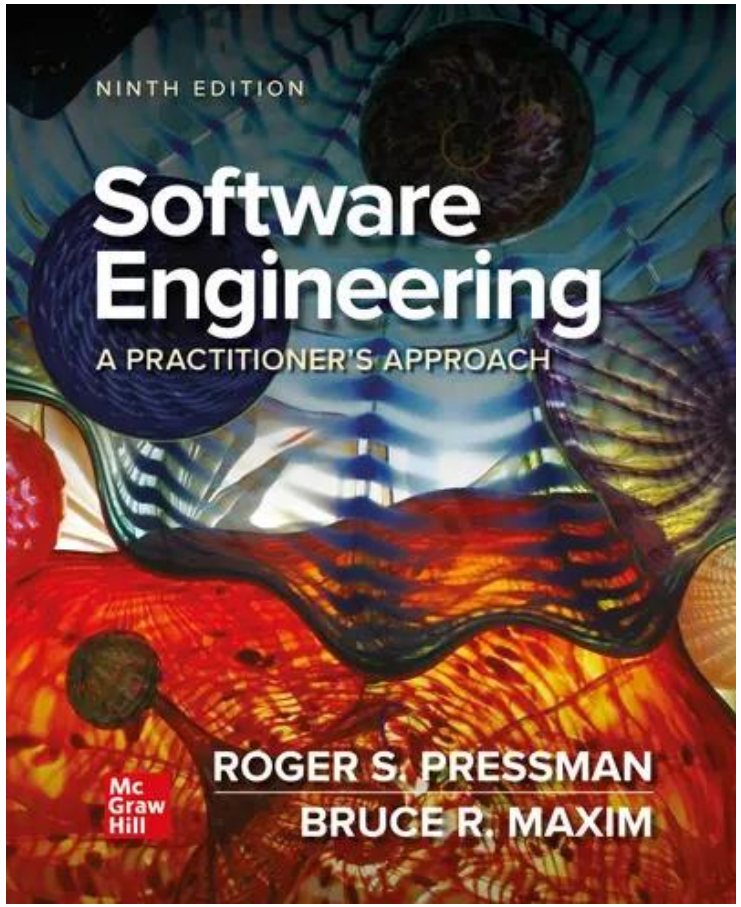
•Recognize the importance of **project planning, documentation, and quality assurance**.

•Relate Software Engineering concepts to **real-world systems** such as hospital, education, or e-commerce software.

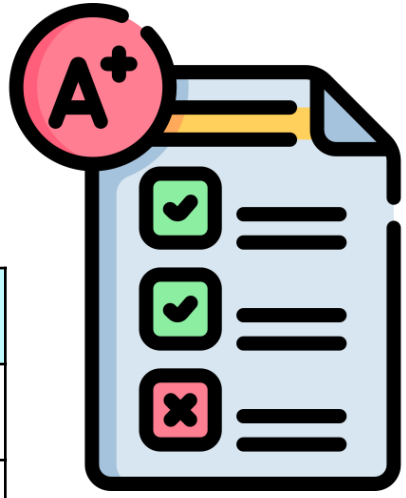
Course Materials







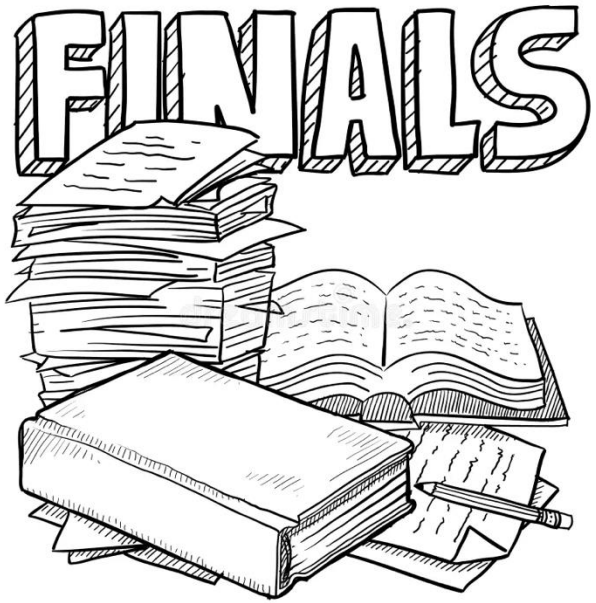
Assessment and Grading



Assessment Type	Weight	Quantity
Midterm Exam	20%	1
Quiz	10%	2–3
Presentation	10%	1
Mini Project	20%	4-5
Final Exam	40%	1

Exam Format (Midterm & Final)

- Covers all **theoretical materials/topics** they are paper based exams.



Exam questions may include:

- Short answers
- Definitions & Explanations
- Scenario-based questions
- Multiple-choice questions
- True or False
- Problem-solving tasks

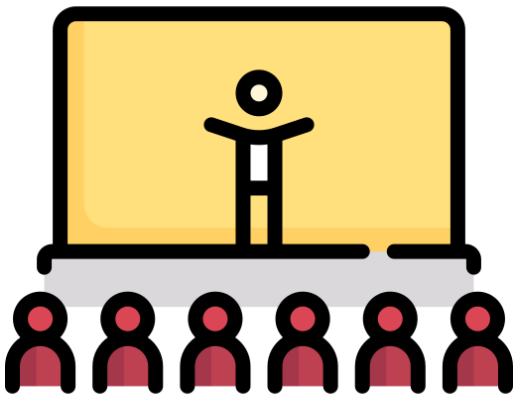
Quiz Format

- Short and simple
- 10–15 minutes only
- Small tasks
- Checks weekly understanding

Mini Project

- Work in groups of 3–4
- Every couple of weeks we will have a mini project.
- Submit it by the deadline. **No extensions allowed.**

Presentation Guidelines



- **Form groups of 6 students.**
- **Choose ONE topic** within your group (first come, first served).
- **Inform the Class Representative**:
  - Class Rep writes **all groups**, **all member names**, and the **chosen topic, and sends full list in one single email by 19 October 2025**.
- **Time:** 20 minutes per group → **15 min** presentation + **5 min** Q&A.
- **Submit slides via Google Classroom only.**
- **Deadline: 27 November 2025.**
- **Everyone presents.**

Submission Policy

- Submit via **Google Classroom** only.
- **No late** submissions, and **no extensions** allowed.
- **Not** more than **25% AI-generated content** is allowed.
- Marks deducted for plagiarism.

Classroom Rules

- **Arrive on Time**: Students arriving **more than 5 minutes late** will not be allowed to enter and will be marked absent.
- **Attendance Matters**: Absences are recorded and may affect grades.
- **Respectful Environment**: Maintain silence, avoid distractions, and respect classmates and the instructor.
- **Devices Policy**: Phones and laptops should only be used for class activities.
- **Assignments & Exams**: Must be submitted/completed on time. No late submissions.
- **Academic Integrity:** Cheating or plagiarism is strictly prohibited.
- **Privacy Rule**: **Recording videos or taking photos without permission is not allowed.**

Contact Information

Email: halal.abdulrahman@tiu.edu.iq

Office Hours: Main Building – Room 321

What is Software?

- Software is a collection of programs, data, and instructions that tell the computer how to work.
- It allows the hardware to perform useful tasks. Without software, a computer is just a machine with no purpose.

Types of software:

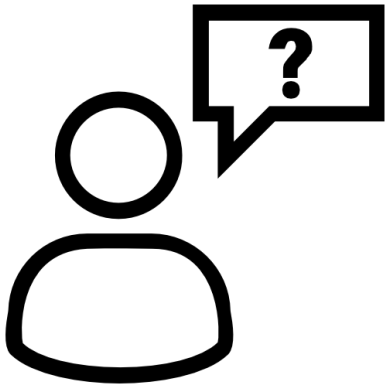
- System Software (Windows, macOS)
- Application Software (Word, Excel)
- Utility Software (Antivirus, Backup Tools)

What is Software Engineering?

- Software Engineering is the systematic and organized process of designing, developing, testing, and maintaining software systems.
- It applies engineering principles to ensure software is reliable, efficient, secure, and maintainable.
- In simple words, it means creating software carefully and professionally, not just coding randomly.
- Example: A software engineer plans, designs, tests, and maintains software just like a civil engineer plans, builds, and maintains a bridge.

Why Do We Need Software Engineering?

- In the early days, programmers wrote code without planning. Projects often failed or were full of errors. This was known as the 'Software Crisis.'



Software Engineering solves this by introducing:

- Planning and design
- Quality control
- Testing and teamwork

Now, large systems like hospitals, banks, and universities are built reliably and on time.

Importance of Software Engineering

Software Engineering is important because it:

- Reduces time and cost.
- Produces high-quality, reliable systems.
- Improves teamwork and documentation.
- Helps manage complex projects.
- Ensures customer satisfaction.

Characteristics of Good Software

Good software should be:

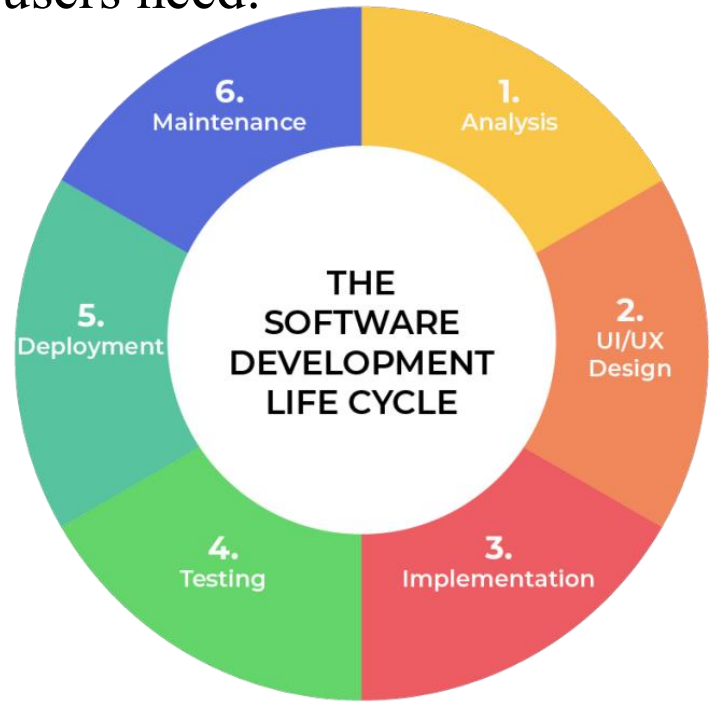
- Correct – performs all tasks accurately
- Reliable – works without frequent crashes
- Efficient – uses system resources well
- Maintainable – easy to update and fix
- Secure – protects data
- User-friendly – simple and easy to use
- Portable – works on different systems

Software Development Life Cycle (SDLC)

SDLC stands for *Software Development Life Cycle.*

It has six main phases:

- Requirement Analysis (Planning and Analyzing): find out what users need.
- Design: decide how the system will work.
- Implementation: write the code.
- Testing: check for errors.
- Deployment: deliver the system.
- Maintenance: fix bugs and add new features.



Phase 1 – Requirement Analysis

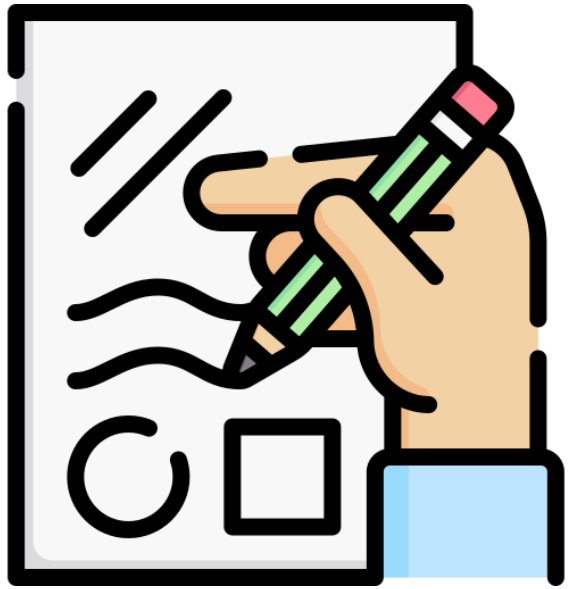
- The team communicates with clients to understand their needs.
- Define the project goals, scope, and timeline.
- Estimate cost, schedule, and resources.
- Decide what problem needs solving and how the project will proceed. Example: Planning how the new university system will work, who will use it, and when it should be ready.
- All requirements are written in a document called **SRS (Software Requirement Specification).**

Example: University registration system – login, course selection, grades view.

Phase 2 – System Design

- In this phase, engineers plan how the system will work.
- They create diagrams, database designs, and user interfaces.

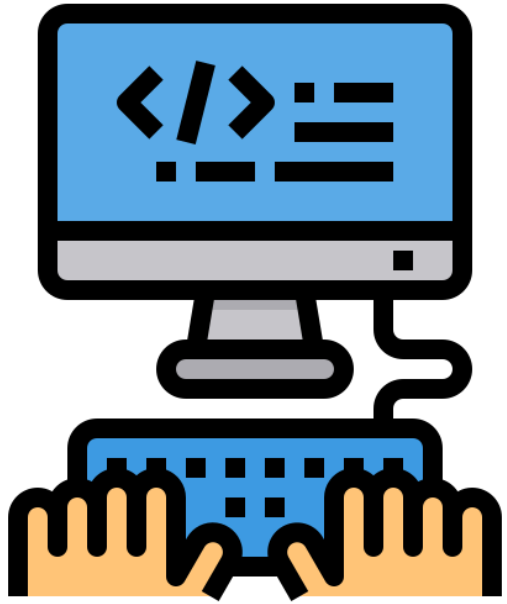
Example: Designing the interface and database for student data.



Phase 3 – Implementation (Coding)

- Developers write the code according to the design.
- They follow coding standards to ensure quality and teamwork.

Example: Building the login and dashboard modules.



Phase 4 – Testing

- Testing ensures the system works properly and meets requirements.

Types of testing:

•**Unit Testing (by Developers) :** Testing small parts of the program.

This is the first step of testing. Each *unit* a small piece of code such as a function, method, or class is tested separately by the developer. The goal is to make sure each part works correctly before combining them.

Example: Test the ‘Add Patient’ form in the hospital system — does it save the patient’s details properly?”

Phase 4 – Testing

•**Integration Testing** (by Developers or Testers): Testing how modules work together. Once all small parts are tested, we connect them and test how they work together. This makes sure data flows correctly between modules.

Example: After testing ‘Add Patient’ and ‘Search Patient’ separately, we check if a newly added patient can be found through the search module.

Phase 4 – Testing

•**System Testing** (by QA Team or Testers): Testing the whole system.

Now we test the entire software system as a whole. This includes checking all modules, user interfaces, and system performance. It's done by a **Quality Assurance (QA)** team, not by developers.

Example: Test login, booking, billing, and reports all working together, as a real user would.

Phase 4 – Testing

•**User Acceptance Testing (UAT) (by End Users):** Final testing by real users

This is the final test, done by the *real users* (like doctors, students, or customers) to see if the system meets their needs and is easy to use. If the users approve, we can officially release the software.

Example: Doctors try the hospital system to check if it’s simple and accurate for scheduling patients.”

Phase 5 – Deployment

Deployment is the phase where the software is **installed, delivered, and made available for real use.**

• The goal is to move the completed system from the testing environment to the **live (production) environment.** Example: Installing the university system on real servers for students and staff to access.

• Sometimes a **pilot version** (trial release) is launched first for a small group of users before the full rollout.

• Deployment includes installation, configuration, data migration, and user training.

• After deployment, users start using the system officially.

Phase 6 – Maintenance

- After deployment, software must be updated and fixed regularly.

- Example: Adding mobile support or new features later.


Any
Question