Software Engineering & Project Lifecycle for CS Students

From Methodologies to Practical Project Planning

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

Introduction

Software Engineering (SE) is the discipline of designing, building, testing, and maintaining software systems.
Understanding SE principles helps students and developers **deliver high-quality software, manage projects effectively, and reduce development risks**.

---

1. Software Development Life Cycle (SDLC)

SDLC defines the phases for developing software systematically.

Phases:

1. **Requirement Analysis** – Gather and analyze client needs
2. **System Design** – Architecture, database, UI/UX design
3. **Implementation / Coding** – Development of functional modules
4. **Testing** – Unit testing, integration testing, system testing
5. **Deployment** – Release to production
6. **Maintenance** – Bug fixes, updates, enhancements

Mini Project Application:

- For a library management system: start by documenting requirements, then design tables, UI mockups, coding, testing, and final deployment.

---

2. Software Development Methodologies

Waterfall

- Sequential, phase-by-phase
- Suitable for small projects with fixed requirements

Agile

- Iterative and incremental
- Sprints, continuous feedback
- Popular for web, mobile, and large projects

Scrum

- Agile framework
- Roles: Product Owner, Scrum Master, Development Team
- Artifacts: Product Backlog, Sprint Backlog

Kanban

- Visual task tracking
- Continuous workflow
- Focuses on efficiency and reducing bottlenecks

---

3. Project Planning and Management

- **Requirements Documentation** – Clear, unambiguous requirements
- **Work Breakdown Structure (WBS)** – Divide project into smaller tasks
- **Gantt Chart** – Visual timeline of tasks
- **Risk Management** – Identify, assess, and mitigate risks

---

4. Software Design Principles

SOLID Principles

1. **S** – Single Responsibility
2. **O** – Open/Closed
3. **L** – Liskov Substitution
4. **I** – Interface Segregation
5. **D** – Dependency Inversion

DRY & KISS

- **DRY (Don’t Repeat Yourself)** – Reuse code
- **KISS (Keep It Simple, Stupid)** – Avoid unnecessary complexity

Design Patterns

- Singleton, Factory, Observer, MVC (Model-View-Controller)

---

5. Testing Techniques

- **Unit Testing** – Test individual modules
- **Integration Testing** – Ensure modules work together
- **System Testing** – Full system evaluation
- **Acceptance Testing** – Confirm system meets requirements

Tools: JUnit, Selenium, Postman

---

6. Version Control

- **Git Basics** – commit, push, pull, branch
- **Collaboration** – GitHub/GitLab for team projects
- **Best Practices** – Commit frequently, meaningful messages, branch workflows

---

7. Documentation & Reporting

- Maintain clear technical documentation
- Include user manuals, API docs, and project reports
- Essential for team collaboration and future maintenance

---

8. Mini Project Ideas

1. **Library Management System** – Agile, with iterative releases
2. **Inventory Management System** – Implement Scrum sprints
3. **Simple E-Commerce Platform** – Include version control and unit testing
4. **Student Portal** – Apply design patterns, documentation, and testing

---

Key Takeaways

Key Takeaways

- SE principles guide project planning, development, and maintenance
- Agile and iterative methodologies improve project efficiency
- Version control, testing, and documentation are non-negotiable for professional software
- Applying SE concepts ensures **final year projects and real-world applications** are successful

---

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app