UNSW College

Course Outline

Jan 2025

STEM Diploma

DPST1093 / CPTG1393

Software Engineering Fundamentals

1. Staff

Position	Name	Email
Course Convenor & Lecturer	Dr Harshana Randeni	H.Randeni@unswcollege.edu.au

2. Course information

Units of credit (UOC): 6

Pre-requisite(s): DPST1091 / CPTG1391

Total course contact hours: 96

2.1 Course summary

This course teaches students about software engineering principles via exposure to the important practice of building correct products in effectively functioning teams.

You will be exposed to agile software practices, team collaboration and effective communication through implementing a group project based on agile software methodologies that requires you to analyse, design, build and deploy a web-based application. This course provides essential background for the teamwork and project management required in many later courses.

2.2 Course aims

This course aims to provide students with a strong foundation in the fundamental principles and practices of software engineering that will prepare them for the advanced software engineering workshops. As such, a broad range of key software engineering topics will be taught and reinforced through a group project, that will enable students to apply the theoretical concepts acquired to solve a practical software engineering problem. An agile software delivery style has been chosen for the implementation of the group project, to make students familiar with modern agile development methodologies.

2.3 Course learning outcomes (CLO)

At the successful completion of this course you (the student) should be able to:

1 Demonstrate effective use of applying software development to build full-stack end-user applications.

2 Demonstrate effective use of static testing, dynamic testing, and user testing to validate and verify software systems.

3 Understand key characteristics of a functioning team in terms of understanding professional expectations, maintaining healthy relationships, and managing conflict.

4 Demonstrate an ability to analyse complex software systems in terms of their data model, state model, and more.

5 Understand the software engineering life cycle in the context of modern and iterative software development practices in order to elicit requirements, design systems thoughtfully, and implement software correctly.

6 Demonstrate an understanding of how to use version control, continuous integration, and deployment tooling to sustainably integrate code from multiple parties.

2.4 Relationship between course and program learning outcomes and assessments

Course Learning Outcome (CLO)	**Program Learning Outcome (PLO)**	**Related Tasks & Assessment**
CLO 1	Understanding of underpinnings (PLO1, PLO3, PLO7, PLO11)	Major Group Project, Labs
CLO 2	Understanding of specialist bodies of engineering knowledge (PLO3, PLO7)	Major Group Project, Labs
CLO 3	Understanding of specialist bodies of engineering knowledge (PLO2, PLO5)	Major Group Project
CLO 4	Application of established engineering practice (PLO1, PLO11)	Major Group Project
CLO 5	Conceptual understanding of computer underpinnings (PLO3)	Major Group Project
CLO 6	Understanding of specialist bodies of engineering knowledge (PLO5)	Major Group Project, Labs

3. Strategies and approaches to learning

3.1 Learning and teaching activities

This course involves a number of teaching activities:

Lectures – 4 hours per week (weeks 1-6 and 8-12)

Lectures present theory and concepts, by way of case studies and practical examples. Lecture notes will be provided in advance of each class. There will be 4 hours of timetabled face to face lectures each week.

TutLabs – 2 hours, 2 times per week (weeks 1-6 and 8-12)

We have combined tutorial lab classes for this course. The first combined tutorial lab each week focuses on covering lecture topics and software project management tasks.

Software Project Management tasks will focus on Major Project delivery, progress reviews, team meetings and tutor feedback.

The second combined tutorial lab each week will focus on lab questions and coding for the Major Project as well as project code demonstrations when relevant.

Labs tasks are individual tasks where students build systems that illustrate the ideas covered in lectures.

To obtain a mark for a lab question you should submit it using GitLab.

You cannot obtain marks by e-mailing lab work to tutors.

During the lab, your tutor will give you guidance and can provide feedback on your approach to the problem and on the style of your solution.

There are 15 assessable labs to submit throughout the course.

Each question is worth 1 mark. There are 15 marks attainable across the labs, but you only need to score 10 out of 15 to achieve the full 10% for the lab marking component. This means you can skip a couple of lab questions throughout the term. Most of the lab exercises are designed to teach students the relevant coding and development practises to solve the current or upcoming milestones of the Major Project.

Major Project – split into 4 Milestones (labelled Iteration 0, 1, 2, 3)

A majority of the learning for this course is assessed via the Major Project. In total, it is worth 60% of the total assessment for this course. This is a group-based activity but your individual contributions will also be assessed both via the system (GitLab) and your tutor/lab demonstrator.

Typically, large software projects are broken down into smaller components with clearly defined goals called milestones. These milestones are considered key progress markers for project completion. In this course, we will take a similar approach by dividing the Major Project assessment into four key milestones. Each of these milestones will be indicated by the completion of a particular iteration of the project.

Iteration 0 (Milestone 1) will focus on getting started with teamwork, learning the basics of the git version control system and developing an initial prototype. This iteration is associated with 3% of the total marks for this course.

Iteration 1 (Milestone 2) will focus on basic functionality for the project and creating test cases. This iteration is associated with 27% of the total marks for this course.

Iteration 2 (Milestone 3) will extend the basic functionality with new requirements. This will include the implementation of a web server and additional test cases to check whether the new requirements have been met. This iteration is associated with 30% of the total marks for this course.

Iteration 3 (Milestone 4) is the final project submission. Here you will adapt to a change in requirements, write new test cases and prepare the project for deployment. This iteration is associated with 30% of the total marks for this course.

Online Forum (Teams)

An online forum on teams allows students to ask and answer questions on the tutorial, lab and assignment exercises, and on lecture material.

3.2 Expectations of students

Students are expected to:

- attend all lectures, and ask questions
- attend all tutorials and actively participate in the discussions
- attend all lab classes and work diligently on the exercises
- participate in the group work for the major project – both in terms of software project management practices as well as coding contributions
- On the course forum (Teams), students should:
    - use relevant/meaningful message titles on all posts
    - ask questions clearly and provide sufficient background information that the question can be reasonably answered
    - not post significant pieces of code, especially code for major project or individual project

4. Course schedule and structure

This course consists of 8 hours of class contact hours per week. You are expected to take an additional 5 hours outside classes to complete assessments, readings, and exam preparation.

Week	Lectures	Tutorial and Labs	Assessment	Related CLO
Week 1	Course Introduction, JavaScript intro, Git intro	Welcome, C to JavaScript conversion, Git intro		1,5,6
Week 2	Packages, importing files, dynamic verification	Arrays in JavaScript, Code Review, Teamwork, JavaScript +Git	Team formation and Iteration 0 Released and due	1,2,3
Week 3	Data interchange, Continuous Integration, Static verification	Package Management, Testing Procedures	Iteration 1 released	2,4,5,6
Week 4	Linting, Advanced Functions, HTTP Servers	Agile approach, Typing and Typescript, Linting	Iteration 1 due, Peer Review 01 due	2
Week 5	Persistence, Authorization and Authentication, Software Development Lifecycle - Requirements	Intro to APIs, HTTP servers – express server, http tests, Swagger API definitions, First class functions	Iteration 2 Released Iteration 1 demo	1,2,3,5,6
Week 6	SDLC – Use Cases and User Stories, SDLC – Validation, Conceptual modelling, Code coverage	Conceptual modelling using JSON / YAML, Code Refactoring, Using Server Routes		3,4,6
Week 7				
Week 8	SDLC – Maintainability, SDLC – Design Complexity Exception handling, Deployment	Good Software, Code Coverage, System Modelling	Peer Review 02 due Iteration 02 due	1,3,5,6
Week 9	Iteration 3 Focus	Functional vs non-Functional requirements, User Stories and Use	Iteration 2 demo Iteration 3 released	1,2,3,5,6

		Cases, Creating Server Routes		
Week 10	Iteration 3 Focus	Complexity Analysis		2,4
Week 11	Full-Stack – Front end development, Full-Stack – Building a Minimal Viable Product	Project completion focus		3,5,6
Week 12		Course Review Deployment	Iteration 03 due, Iteration 03 demo, Project Exhibition Peer Review 03 due	1,3,5,6

5. Assessment

5.1 Assessment tasks

Assessment task	Length	Weight	Due	CLOs
Major Project (Milestone 1)	2 weeks	3%	Week 3	1,2,3,4,5,6
Major Project (Milestone 2)	3 weeks	27%	Week 6	1,2,3,4,5,6
Major Project (Milestone 3)	3 weeks	30%	Week 9	1,2,3,4,5,6
Major Project (Milestone 4)	3 Weeks	30%	Week 12	1,2,3,4,5,6
Labs	Throughout Semester	10%	Weekly	1,2,6

The Major Group Project is assessed on three factors. The primary factor is code correctness, the second is code quality, the third is software project management practice. Code correctness is handled by auto-marking. Code quality is hand marked by tutors. Software project management practice is assessed throughout the course by tutors via tutor and lab participation, Iteration demos, code reviews and hand marking of your assessment submissions.

Lab questions are auto-marked primarily on their correctness, lab tutors and demonstrators will also give feedback on code quality for some lab questions.

5.2 Submission of assessment tasks

All assessments for this course will be submitted via the GitLab system.

For lab tasks, given extended timeframes, there are not late submissions except for minor fixes.

As the Major Group project is group work, it is up to the team members to organise themselves and distribute tasks. If a late submission occurs, the team members will be asked if they wish to have the late submission or an earlier submission assessed. If a late submission is assessed, it will follow the standard late penalty.

If a team member is not participating, the tutor should be alerted as early as possible and actions can be taken address the situation.

If a team member can not participate due to misadventure, that team member should submit a special consideration form and contact the lecturer as soon as possible – preferably well before the assessment deadline. Extensions are normally not granted for group work projects, but contribution criteria can be reassessed if there is a determination that you have been adversely affected by your situation.

5.3 Feedback on assessment

Lab questions will be auto-marked and a score provided. You may discuss the outcome with your tutor if you have any questions.

The Major Group Project will be auto-marked after the submission deadline and annotated with comments by the tutor. There will also be demo sessions where your tutor will assess your contribution and software project management practices. You can discuss the tutor's comments in a lab class after you have received the feedback.

6. Readings and Resources

There is no single textbook that covers all the material in this course at the right level of detail and is using the same technology base as we are. The lectures should provide sufficient detail to introduce topics – upon which you will do further in-depth study in the tutorials, labs and group projects. For some lectures, further reading material may be given to students who wish to gain a deeper understanding.

7. Use of AI Tools

This subject prohibits the user of AI tools for submissions. This means that any code or documents that a student submits is assumed to be their own work.

It is understandable that students may use AI tools to assist in the explanation of code and for potentially checking code for bugs. This use is consistent with industry practices, so it is permitted as long as the final submission of work is the students own work.

The authenticity of your code submissions are verified during the demos you do after the end of each Major Project Iteration 1,2,3. If you can not explain your code, or if it suspected of being AI generated, you may receive an adjustment to your marks and a warning. If it happens more than once or it happens in a substantial way, then you will be referred to the UNSW College Academic Intregity unit.