# Software Engineering 11–12 (2022): Sample unit of work Year 12

Sample units are provided by NESA to illustrate teaching, learning and assessment of syllabus outcomes and content. Teachers should seek advice from their schools and sectors about local requirements for units of work, including opportunities for reflection and evaluation, and recording evidence of adjustments to meet the needs of individual students with disability.

Unit title: Software Engineering Project

Duration: 30 hours over 3 terms

Description: This unit of work guides students through the development of their Software Engineering Project. The Software Engineering Project content should be delivered throughout the course and integrated with other focus areas to support students to apply deeper knowledge, understanding and skills in an area of personal interest. This unit provides opportunities for students to develop the knowledge, understanding and skills associated with project development, including identifying and defining requirements, research and planning approaches and issues, producing and implementing software solutions, and testing and evaluating code and engineering solutions. Throughout this sequence of learning, students will have opportunities to seek feedback to inform the development of their project. The Software Engineering Project is developed in the classroom under the supervision of the teacher.

| Outcomes |
| --- |
| * **SE-12-01** justifies methods used to plan, develop and engineer software solutions * **SE-12-02** applies structural elements to develop programming code * **SE-12-03** analyses how current hardware, software and emerging technologies influence the development of software engineering solutions * **SE-12-04** evaluates practices to safely and securely collect, use and store data * **SE-12-05** explains the social, ethical and legal implications of software engineering on the individual, society and the environment * **SE-12-06** justifies the selection and use of tools and resources to design, develop, manage and evaluate software * **SE-12-07** designs, develops and implements safe and secure programming solutions * **SE-12-08** tests and evaluates language structures to refine code * **SE-12-09** applies methods to manage and document the development of a software project |

### Unit information

#### Software Engineering Project

| Content |
| --- |
| Identifying and defining  * Define and analyse the requirements of a problem, including:   + demonstrating need(s) or opportunities   + assessing scheduling and financial feasibility   + generating requirements including functionality and performance   + defining data structures and data types   + defining boundaries * Explore tools used to develop ideas and generate solutions, including:   + brainstorming, mind-mapping and storyboards   + data dictionaries, including selecting appropriate data types   + algorithm design   + code generation   + testing and debugging   + installation   + maintenance * Investigate types of software implementation methods, including:   + direct   + phased   + parallel   + pilot |

| Suggested teaching, learning and assessment activities | Suggested resources |
| --- | --- |
| Students are introduced to the development of a large-scale software project. In the Year 11 course students developed knowledge and skills in the processes used by software engineers to develop software solutions.  Students engage in a guided discussion associated with the knowledge, understanding and skills developed in their Year 11 project work. Through this guided discussion they consider how methods and strategies from their Year 11 projects and activities can be applied to their Software Engineering Project.  Questions include:   * How did we analyse the problem to gain an understanding of the requirements? * How did we record these requirements? * What tools did we use to plan the solution from start to finish? * How did we manage the technical requirements of the solution? * How did we develop ideas? * What did we learn from evaluating our solutions?   Students use a checklist (Appendix A) to guide them throughout their project. Students are reminded that the checklist is a personal management tool and may be redesigned to suit their needs. | [Appendix A](#_Appendix_A) |
| Students use knowledge from the discussion to build an approach to **identifying and defining** the elements required for their Software Engineering Project. Students use an appropriate case study to investigate the software development approach used by professionals to develop an app or software:   * What approach was used in the development of the app or software? * How did the professionals identify the need? * What were the requirements?   Students develop their own Software Engineering Project idea(s), using a scaffold (Appendix B). The scaffold can be duplicated for a range of project ideas. Students need to justify their ideas throughout each stage of their project development. The scaffold can be used to support the student to justify their idea(s) and to clarify its feasibility for both the teacher and the student.  Upon completion of the scaffold, students define and analyse the requirements of the problem:   * What problem will the solution solve or address (demonstrating need(s) or opportunities)? * What steps do I need to take to plan my time (scheduling feasibility)? * How much is this project likely to cost (financial feasibility)? * What are the functional and performance related aspects of the solution? * How will I determine any data structures or data types? * What are the boundaries of the solution? What should it do? What does it not need to do?   Students then prepare a pitch to the class on their preferred idea using a presentation method of their choice. The purpose of the pitch is to persuade their audience that their identified problem needs a software engineering solution. Students peer-assess the pitch and provide feedback based on the requirements of the problem.  Note: The pitch connects to work completed in other units throughout Year 11 and Year 12 regarding working with clients.  Students consider the feedback provided on their pitch and complete a project proposal. The project proposal is completed and signed by both student and teacher. Students begin exploring and documenting the tools used to develop ideas and generate solutions. Students use:   * brainstorming, mind-mapping and storyboards: to visualise the idea and communicate the need for the solution * data dictionaries, including selecting appropriate data types: to specify the data structures or related data types * algorithm design: to conceptualise the functionality that the software will need * code generation: to determine what libraries or ‘code banks’ are available that may be incorporated into the solution * testing and debugging: to develop criteria that could be used for testing during development and post development * installation: to determine how users will get access * maintenance: to determine what ongoing support, bug fixes and updates may be required.   The teacher provides students with different project management approaches to investigate types of software implementation methods:   * direct * phased * parallel * pilot.   Through teacher-led discussion, students compare the different methods as applied to their project idea. To compare methods, students use a scaffold on the benefits of each implementation method (Appendix C) and a SWOT analysis (Appendix D, page 2) for each of the software implementation methods.  Students consolidate the direction for the idea or problem for the intended software solution through the development of a feasibility justification, to investigate the types of software implementationmethods. Students select, apply and evaluate a software implementation method following development of the solution.  Students collate the activities and documentation created so far and formalise their presentation to be included in the Identifying and Defining section of the project folio. Students decide on the presentation method and the use of subheadings, diagrams or visualisations. | Teachers guide students towards a case study that illustrates the development of an app where the client goals and the software needs lead to a pre-production description of the needs of the final app solution.  [Appendix B](#_Appendix_B)  [Appendix C](#_Appendix_C)  [Appendix D](#_Appendix_D) |

| Content |
| --- |
| Research and planning  * Research and use the Waterfall software development approach, including:   + logical progression of steps used throughout the life cycle   + stages of ‘falling water’   + advantages and disadvantages   + scale and types of developments * Research and use the Agile software development approach, including:   + rate of developing a final solution   + method tailoring   + iteration workflow   + scale and types of developments * Research the WAgile software development approach, including:   + understanding it is a hybrid model   + analysis of the ‘when’ and ‘how’ intervention is applied during the development life cycle   + scale and types of developments |

| Suggested teaching, learning and assessment activities | Suggested resources |
| --- | --- |
| Students have identified and defined the scale of their software engineering problem, the software solution required, and will now commence researching and planning software development approaches. The teacher provides guided reflection and consolidation of previous learning, including the use of Waterfall, Agile and/or WAgile approachesused for students’ small scale project work in previous Year 11 or Year 12 focus areas. Students reflect on the features of each approach and how these features contributed to the success of past projects. Students conduct further research on each approach to consolidate their understanding.  Students apply their research findings to make decisions about an approach they could apply in the planning and development of their project. Students refer to a case study of their choosing, which examines a software approach taken to develop a solution similar to their proposed project.  Students synthesise their research using a method of their choice. This synthesis is to be incorporated into the student’s Research and Planning section of their project folio. Students decide on the presentation method and the use of subheadings, diagrams or visualisations.  Students include the following in the synthesis:   * steps involved * analysis of advantages and disadvantages * workflow or roadmap. | Teachers guide students towards a case study that illustrates the benefits and limitations of software development approaches. A case study that presents the theories behind applying each to a similar project would be of benefit. |

| Content |
| --- |
| Research and planning  * Apply project management to plan and conduct the development and implementation of a project and software engineering solution, including:   + scheduling and tracking using a software tool, including Gantt charts   + using collaboration tools |

| Suggested teaching, learning and assessment activities | Suggested resources |
| --- | --- |
| Students use the knowledge gathered from the synthesis of software development approaches in the previous activity to document the project planning components in their folio.  Students connect the learning from Year 11 and other Year 12 focus areas to draw on the scheduling and tracking (project management) software they have used and develop a plan for their project. Students have used collaboration tools throughout Year 11 and Year 12 and can use them for this project. Students document their scheduling and project management approach using a method of their choice. Methods include the use of dedicated project management software, bespoke tools developed to present a Gantt chart, management flip cards, financial plans or visualisation management software. This documentation is to be incorporated into the student’s Research and Planning section of their project folio using the subheading of ‘Scheduling and Project Management’. |  |

| Content |
| --- |
| Research and planning  * Explore social and ethical issues associated with project work, including working individually, collaboratively and responding to stakeholders * Explore communication issues associated with project work, including:   + involving and empowering the client   + enabling feedback   + negotiating |

| Suggested teaching, learning and assessment activities | Suggested resources |
| --- | --- |
| Students use the planning they have prepared in previous activities to explore the social and ethical issues associated with project work. Students develop a summary for inclusion in their project folio within the Research and Planning section. Students decide on the presentation method and the use of subheadings, diagrams or visualisations.  The summary is to include the development of the project solution from the perspective of the individual, team and any external stakeholders. Students incorporate a summary of social issues associated with their project including:   * employment * equality * environmental issues * cultural identity * economic impact.   Students incorporate a summary of ethical issues associated with their project including:   * health and safety * accountability * conflicts of interest * bias. |  |

| Content |
| --- |
| Research and planning  * Investigate how software engineering solutions are quality assured, including:   + defining criteria on which quality will be judged   + ensuring requirements are met using a continual checking process   + addressing compliance and legislative requirements * Demonstrate the use of modelling tools |

| Suggested teaching, learning and assessment activities | Suggested resources |
| --- | --- |
| Students will draw on their summary of the social and ethical issues associated with their project created in the previous activity. Students continue their research and planning by **investigating how their software engineering solution will be quality assured**. Students develop a report to be included in the Research and Planning section of their folio. Students decide on the presentation method and the use of subheadings, diagrams or visualisations.  The report developed includes:   * defining criteria on which quality will be judged * ensuring requirements are met using a continual checking process * addressing compliance and legislative requirements.   Students then plan, select, justify and document the modelling tools that will be used as part of the development of the project. To justify the modelling tools, students answer the following questions:   * How and why does the modelling tool support development within the boundaries of the selected approach? * What features of the modelling tool support the illustration or testing of features and/or specifications proposed that will solve the problem? * Why does this modelling tool present the prototypes required to meet the needs of the client?   Students include the report in the Research and Planning section of their folio. |  |

| Content |
| --- |
| Research and planning  * Explain the contribution of back-end engineering to the success and ease of software development, including:   + technology used   + error handling   + interfacing with front end   + security engineering |

| Suggested teaching, learning and assessment activities | Suggested resources |
| --- | --- |
| Students complete their project planning by documenting how back-end engineering could contribute to the success of their project. Students prepare a summary to be included in the Research and Planning section of their project folio. Students decide on the presentation method and the use of subheadings, diagrams or visualisations.  Factors to be included in the summary include:   * technology used * error handling * interfacing with front end * security engineering. |  |

| Content |
| --- |
| Producing and implementing  * Design, construct and implement a solution to a software problem using appropriate development approach(es) * Present a software engineering solution using presentation software * Develop, construct and document algorithms * Allocate resources to support the development of a software engineering solution * Demonstrate the use of programmed data backup * Implement version control when developing a software engineering solution * Explore strategies to respond to difficulties when developing a software engineering solution, including:   + looking for a solution online   + collaboration with peers   + outsourcing * Propose an additional innovative solution using a prototype and user interface (UI) design |

| Suggested teaching, learning and assessment activities | Suggested resources |
| --- | --- |
| Students have now completed their research and planning and will commence the production and implementation of their software engineering solution. In this sequence of learning, the teacher will guide students to the syllabus content, including programming, coding and interface design, which is essential knowledge for the construction of their project. This includes their selected software development approach.  As students develop their solution, they document any ongoing evaluation and decisions made to solve any problems encountered in the development process. Students document strategies to respond to difficulties when developing a software engineering solution. These could include:   * saving time by using code libraries or predeveloped code modules by looking for a solution online * creating stop points and meeting with classmates regarding problems and approaches to solutions * explore outsourcing solutions for specific challenges.   While this is the development phase of the project, students need to document any work completed in the Producing and Implementing section of their project folio. Students decide on the presentation method and the use of subheadings, diagrams or visualisations. This will include:   * documenting algorithms, which may be diagrammatic (flowcharts) or pseudocode * allocation of resources to support development, which may be time, financial, people and any digital/software elements like sprites, images, forms etc * documenting their use of programmed data backup procedures during development * documenting implementation of version control and how it relates to backup.   During the course students will have used prototyping as a method to test and correct ideas, interface design and user experience (UX). Students will have limited time and resources to develop their solution. Students will need to document how they may develop additional features or the next iteration of their solution by prototyping the user interface (UI) of these. This should be documented in the Producing and Implementing section of their project folio. |  |

| Content |
| --- |
| Testing and evaluating  * Apply methodologies to test and evaluate code * Use a language-dependent code optimisation technique * Analyse and respond to feedback * Evaluate the effectiveness of a software engineering solution, including:   + developing a report to synthesise feedback   + developing a test plan   + testing data used/generated based on path and boundary testing   + comparing actual output with expected output |

| Suggested teaching, learning and assessment activities | Suggested resources |
| --- | --- |
| Students document their testing and evaluating components throughout the development of their Software Engineering Project under the guidance of the teacher. This should be documented in the Testing and Evaluating section of their project folio. This section of the project folio is presented using diagrams, visualisations or files, or a combination where appropriate.  The teacher reminds students of the range of methods to test and evaluate code that have been studied throughout the course. The teacher also checks students have selected and documented appropriate methods to test and evaluate their programming code as they develop their software project.  In the development of their solution, students should apply and document a language-dependent code optimisation technique that suits their type of project. The techniques selected could target one or some of the following:   * reducing code complexity * avoiding unnecessary pre-test loops * using familiar data structures for sorting, searching or string concatenation * avoiding unnecessary functions * avoiding dots * initialisation * data aggregation.   The application of code-optimising should be documented in the Testing and Evaluating section of the project folio. Students decide on the presentation method and use the subheading ‘Code Optimisation’*.* This section of the project folio is presented using diagrams, visualisations or files, or a combination. Students use the comparison table in Appendix E to compare appropriate techniques and their contribution towards code optimisation.  When applying code optimisation techniques, a range of test results and data may be available for analysis. These analyses should be documented in the Testing and Evaluating section of students’ project folio.  At the stages of working prototype and final solution, students peer-evaluate a range of aspects, including:   * functionality * ease of use * aesthetics.   Students analyse and respond to feedback from their peers and teacher. These analyses are documented in the Testing and Evaluating section of their project folio.  Students evaluate the effectiveness of their software engineering solution prior to submission.  Students generate a formal evaluation report that includes:   * the test plan * any test data used, including path and boundary data * a functionality comparison between actual output and expected output.   The evaluation report should be included in the Testing and Evaluating section of their project folio. | [Appendix E](#_Appendix_E) |

| Reflection and evaluation (space for teachers to reflect on and evaluate the unit) |
| --- |
|  |

## Appendix A

### Software Engineering Project checklist

**Teaching advice:** This checklist is a minimum set of tasks that Software Engineering students should ensure they complete when developing their Software Engineering Project. This is used to establish milestones and guide teacher/student discussions and feedback.

The following tables contain a description of the tasks with a column to check with a tick (P) when complete.

|  |  |
| --- | --- |
| **Project proposal** | |
| Problem definition |  |
| Research |  |
| Project pitch |  |
|  |  |
| **Identifying and defining** | |
| Project brief – develop the project brief |  |
| Feasibility study – include scheduling and financial feasibility |  |
| Tools, hardware and software required (functional performance, data and boundaries) |  |
| Skills analysis – what skills are needed? What skills (if any) still need to be learnt? |  |
|  |  |
| **Researching and planning** | |
| Research for the problem |  |
| Commercial or other solution that might already solve this problem |  |
| Idea generation – any mind maps, sketches, draft algorithms, code blocks or pre-developed modules/libraries |  |
| Planning Gantt chart |  |
| Finance plan |  |
| Research evidence and selection of software development approach |  |
| Social and ethical implications of the project |  |
| Solution specification development |  |
| Mock-ups, design idea generation, storyboard |  |
| Implementation of software engineering techniques that influence the final design ie reverse engineering |  |
| Documentation of a testing approach, including test data and developing the test plan |  |
|  |  |
| **Producing and implementing** | |
| Develop algorithms |  |
| Develop programming code, backup procedures and version control measures |  |
| Design and develop interface elements |  |
| Run, modify and refine code |  |
| Compile program |  |
|  |  |
| **Testing and evaluating** | |
| Document method to test and evaluate code |  |
| Optimise code |  |
| Apply test plan |  |
| Respond to feedback |  |
| Project evaluation |  |

## Appendix B

### Scaffold for Software Engineering Project ideas

Note: This scaffold is used to explore a single idea or replicated to investigate multiple ideas.

The SWOT analysis (Strengths, Weaknesses, Opportunities, Threats) is used to determine the opportunities and risks associated with the need.

| Idea or software name | Insert your idea name here |
| --- | --- |
| Goal:  The purpose of the project – why is this solution required? |  |
| Context:  People, environment, social problem the software is to be designed for. |  |
| Objectives:  What it wants to achieve from a user, enterprise or social standpoint. |  |
| Scope:  What software will interface with?  What hardware is required?  What services (if any) may be affected?  What are the ongoing support elements of the project?  Who are the key stakeholders? |  |

### SWOT analysis of idea

**S**

**W**

**O**

**T**

*Strengths of the idea*

*Limitations of the idea*

*Opportunities of the idea*

*Risks or threats of the idea*

## Appendix C

### Pitch slide deck – storyboard scaffold

**Teaching advice:** Students may use this scaffold storyboard to develop a pitch slide deck to plan their pitch. This storyboard may contribute towards the development of the slide deck and support either an oral, video, multimodal or other presentation. The pitch slide deck forms part of the overall pitch and may be reduced or extended where required based on assessment criteria. The pitch connects to other work completed throughout Year 11 and Year 12 regarding working with clients.

#### Slide 1

Cover/Title slide: proposed app or software package name and hook image

*Type slide notes here*

#### Slide 2

Problem statement: Why is an app or software solution important and how will it solve the problem?

*Type slide notes here*

#### Slide 3

Initial problem research: What data or evidence supports the reason to develop an app or software solution for the problem?

*Type slide notes here*

#### Slide 4

Motivation: How solving this problem lets me (the student) demonstrate my knowledge, understanding and skills.

*Type slide notes here*

#### Slide 5

Summary/Conclusion: Why this app/software solution will achieve the objectives of my proposal.

*Type slide notes here*

## Appendix D

### Benefits comparison of software implementation methods

In each of the boxes present the benefits of implementing your solution using each method.

Direct

Phased

Parallel

Pilot

*List the benefits of implementing your solution using direct implementation here*

*List the benefits of implementing your solution using phased implementation here*

*List the benefits of implementing your solution using parallel implementation here*

*List the benefits of implementing your solution using pilot implementation here*

### SWOT analysis of [enter method here]

**S**

**W**

**O**

**T**

*Strengths of the method*

*Weaknesses of the method*

*Opportunities of the method*

*Threats or risk of the method*

## Appendix E

### Scaffold for the comparison of code optimisation techniques

The number of optimisation techniques is dependent on the complexity of the project or code. Additional rows can be added to this scaffold as required.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Project name:** | | | | |
| **Optimisation technique** | **Reduced code** | **Improved overall performance** | **Improved maintenance** | **Improved code security** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |
| --- | --- |
| **Efficacy summary** |  |