Gymea Technology High School
INSPIRE. LEARN. SUCCEED.

12 Software Engineering

Task 13: Software Engineering Project

Due Date: 27 Jun 2025

Task Distributed: 13 Mar 2025

Unit: Software Engineering Project

Task Type: Practical Project

Task Weighting: 30 %

Outcomes:

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

Students identify a real-world problem or opportunity that can be addressed through the development of a software engineering project. Students develop project documentation and engineer a software solution that addresses this real-world problem or opportunity. The software engineering project is presented to the class by simulating a client handover.

Task Description:

You are to develop a software solution for a client or you may use Scenario 1 or Scenario 2 (listed below) to help guide you in the development of a software solution.

The software solution:

- demonstrates the design, development and construction of algorithms
- addresses all the clients functional and performance requirements.

The program should also:

- use a language-dependent code optimisation technique
- be configurable in terms of the interface layout, colour scheme, accessibility requirements, legislative and technical requirements as per client needs
- incorporate security elements such as usernames, passwords, basic encryption and decryption as per client requirements.
- include a proposal for an additional innovative solution using a prototype and user interface (UI) design.

Task

The task is broken into three parts:

PART A - Documentation

You are required to complete a project portfolio which addresses the following:

- requirements definition (boundaries, scheduling and financial feasibility, need(s) or opportunities of the user, requirements including functionality and performance, and defining data structures and data types)
- a journal documenting the progress and pitfalls
- a data dictionary
- a gantt chart
- a data flow diagram (level 0 and 1)
- structure chart
- class diagram
- decision tree
- desk check of the algorithm
- evaluation of the software solution

Part B - Practical

Develop, code and upload a software solution for the problem as indicated in the project documentation. You are required to submit both the original code and compiled versions.

In consultation with your teacher, choose the programming language with which their solution is developed. Examples of languages include the VS suite, Python, Java and C.

Your programs should (where relevant) incorporate combinations of the following features: control structures, global and local variables, use of simple and structured data types, classes, objects, attributes and methods, functions, modules and libraries and file handling.

Other examples of tools could include web page creation tools, front-end web development frameworks and SQL databases.

The software engineering project should solve a real-world problem or meet a real-world need or opportunity as determined through negotiation with a client. These clients could be associated with health, education, business, entertainment or social. Note that this list is not exhaustive.

Part C - Presentation

You are to develop a 4-minute presentation in the appropriate format (for example, PowerPoint) that demonstrates their software solution to a client (simulated by peers) using a professional language, style and format.

The presentation will include screenshots of the development and documentation of the process. During a 3-minute question and answer session students will answer questions about their project and receive feedback from the audience.

The presentation must have:

- **Presentation slides:** A set of slides in the appropriate format (for example PowerPoint) that highlights your solutions features, benefits and challenges using appropriate visual aids such as slides, diagrams and screenshots.
- **Software demonstration**: a live demonstration of your system that allows your teacher and peers to see how you interact with the system and provide feedback.

NESA Glossary of Key Words

Understand the verb associated with the task. The verb will provide an understanding of the detail needed to successfully answer the question.

- **Define** - State meaning and identify essential qualities
- **Describe** - Provide characteristics and features
- **Evaluate** - Make a judgement based on criteria determine the value of
- **Identify** - Recognise and name.

Check the NESA Glossary of Key Words for further guidance
https://educationstandards.nsw.edu.au/wps/portal/nesa/11-12/hsc/hsc-student-guide/glossary-keywords

Details of Submission

You are to submit your documentation digitally as a Google Doc file and the practical component as a zip file or downloadable Git repository.

Teacher Feedback and Student Self-Reflection

The task will typically be returned to students within **14 days** of the due date.

Information on how to improve will be provided through written teacher feedback and the marking criteria. Students can clarify or seek further feedback by speaking with their teacher.

Upon return of the task and teacher feedback, students will also be expected to complete the following self-reflection form, to provide them with the opportunity to reflect on the strength of their performance, as well as areas that have been identified to strengthen in future tasks - https://forms.gle/oBnPJ8EsGLTQZm7Z8

How does this link to my learning?

This task will allow students to demonstrate their understanding of theoretical concepts, providing students with the opportunity to showcase their knowledge, understanding and skills in

- Effective use of project management techniques including documentation and communication
- Demonstrates the ability to uses appropriate resources and tools to effectively develop, document and manage their project

Assessment Procedures

All students should be fully aware of the School Assessment Procedures for their year group. These were provided at the beginning of the school year and are available off the school website under the Learning Tab for each year group.

Scenarios

Scenarios could include:

- online prefect voting system
- canteen ordering app
- market day business studies website
- school musical booking system
- a biometric roll call attendance system (controversial)
- school jersey and name design app
- sports carnival administration system
- event management system
- an interactive school map and timetable.

These should be expanded into detailed scenarios to provide guidance for students to:

- extract functional requirements
- model with tools
- develop software solutions and innovative prototypes.

Scenario 1 – parking roster

Warami High School has an exceedingly small staff car park. They have a total of 30 spaces allocated for 120 staff members (including Student Administration and Support), split across the 10 faculties of:

- English – 15 teachers
- Mathematics – 13 teachers
- Science – 10 teachers
- Technological and Applied Studies (TAS) – 12 teachers
- Creative and Performing Arts (CAPA) – 7 teachers
- Personal Development, Health and Physical Education (PD/H/PE) – 7 teachers
- History and Languages – 7 teachers
- Human Society and Its Environment (HSIE) – 9 teachers
- Learning Support – 6 teachers
- Student Administration and Support – 34 staff members

The number of teachers for each faculty is given above. There are a total of 10 files provided, one representing each faculty, that need to be read into the program. In each file, there are the names of each of the teachers.

A timetable of how many spaces each faculty gets per day as well as who gets what spot each day for a 2-week cycle needs to be generated.

The program **must**:

- present a form for the user to select the files containing the names for each faculty
- read the names of each of the teachers in each of the faculties from the files
- display which faculty has what spaces on particular days linked to the teachers' names, using a graphical method such as a block of colour or a picture of a car in a car park.
- enable the user to click on 'Monday' and get the allocations for Monday, or 'Tuesday' for Tuesday's allocations and so on.

Note: Teachers do not have to be allocated the same days in a row, nor do they have to have the same parking spots each time.

The program **should** also:

- ensure that 50% of each faculty is allocated a parking spot in the fortnight
- give the user the option to either read the names of each teacher from the files, or to enter each teacher's name manually with the number of days each teacher is in per fortnight
- have a menu system that allows the user to switch between the default mode as specified in the 'must' section, and this optional mode.

The program **could** also:

- ensure that 70% of each faculty is allocated a parking spot in the fortnight
- allow the user to change the number of staff members per faculty so long as the total of 120 staff members is reached. This means that the user should be presented with a form with each faculty name and an associated input box with how many staff members are in each faculty
- increase or decrease the number of total staff members in the school car parking timetable. This should in turn check to see that the total entered does not exceed the number of total staff members.

Scenario 2 – typing tutor game

There is a lot of vocabulary in the Software Engineering course. There are also a lot of large, complex assessment tasks like this one which require a lot of typing. A typing game which builds up the vocabulary of the students studying Software Engineering, as well as keyboard skills, is required to bridge this gap. Technical vocabulary for this typing program is to be derived from the topics of:

- Programming fundamentals – 10 pieces of technical vocabulary
- Object-oriented programming – 10 pieces of technical vocabulary
- Programming mechatronics – 10 pieces of technical vocabulary
- Secure software architecture – 10 pieces of technical vocabulary
- Programming for the web – 10 pieces of technical vocabulary
- Software automation – 10 pieces of technical vocabulary

In addition, definitions and exemplar paragraphs for each of the below NESA key terms need to be included in the typing tutor program.

- Identify
- Define
- Describe
- Explain
- Analyse
- Justify
- Evaluate

These exemplar paragraphs are to be read from data files provided.

Additional exercises that train students on how to place their hands on the home row, feel for the bumps on F and J, punctuation marks and so on must also be included. Some examples of typing programs are given here:

- Speed Typing Online (https://www.speedtypingonline.com/typing-tutor)
- Typing Trainer (https://www.typingtest.com/trainer/)
- Typing Academy (https://www.typing.academy/).

The program **must**:

- read the provided data file for the NESA key term exemplar paragraphs
- progress the students, from training them how to place their hands on the home row to typing in key words, then whole sentences and paragraphs
- calculate their words per minute and percentage (%) of accuracy
- contain at least 5 levels, from absolute beginner at Level 1 to typing in whole sentences and paragraphs at Level 5.

The program **should** also:

- give the administrator of the typing tutor the option of entering the technical vocabulary for each of the given Software Engineering topics into a form, which then gets written into at least one file
- read the technical vocabulary for each topic from the data file generated by the program.

The program **could** also:

- contain up to 10 levels, slowly progressing from absolute beginner at Level 1 to expert at Level 10, where the students are typing in exemplars of code that perform various tasks from Object-oriented programming such as
    - — class definition and object instantiation
    - — polymorphism
    - — inheritance and/or Secure software architecture to create MD5 hashes such as those in Python or in JavaScript
    - — AES encryption and decryption ciphers in Python, or PGP encryption and decryption in Python and OpenPGP in JavaScript as typing practice
- read these code exemplars from data files made for the program.

Marking Rubric

CRITERIA	0 -2	3 - 4	5 - 6	7 -8	9 -10
Requirements Definition	Student provides a limited understanding of the requirements of the software solution being developed.	Student outlines and identifies a basic number of features of the purpose of the software solution being developed ((boundaries or scheduling and/or financial feasibility, need(s) or opportunities of the user, or requirements including functionality and/or performance, and/or defining data structures and/or data types)	Student outlines and identifies a number of features of the purpose of the software solution being developed ((boundaries, scheduling and financial feasibility, need(s) or opportunities of the user, requirements including functionality and performance, and defining data structures and data types)	Student describes the purpose of the software solution being developed addressing most areas (boundaries, scheduling and financial feasibility, need(s) or opportunities of the user, requirements including functionality and performance, and defining data structures and data types)	Student clearly describes and explains the purpose of the software solution being developed (boundaries, scheduling and financial feasibility, need(s) or opportunities of the user, requirements including functionality and performance, and defining data structures and data types)

CRITERIA	0 - 1	2	3	4	5
Journal	Student provides a limited entries or no understanding of a journal.	Student provides basic entries identifying problems or what was achieved and/or limited entries with basic details.	Student provides some entries outlining problems or what was achieved or how the problems were solved.	Student provides regular entries describing a number of problems and what was achieved and how the problems were solved.	Student provides extensive entries explaining a wide range of problems and what was achieved and how the problems were solved.
Data Dictionary	Student provides a limited or no understanding of a data dictionary.	Student creates a basic data dictionary with a few of the required components or column headings.	Student creates a data dictionary showing some of the required components.	Student creates a data dictionary showing most of the required components and follows the course specifications.	Student creates a comprehensive data dictionary that includes all data types and follows the course specifications.

Gantt Chart	Student has attempted to organise tasks into a Gantt chart or no understanding of Gantt Charts.	Student has most of the tasks ordered and assigned into a plausible sequence.	Student has created a Gantt chart that has all the tasks ordered and assigned into a logical and realistic timeframe.		
Data Flow Diagram (Level 0)	Student provides a Data Flow Diagram that shows limited or no process, or external entities and does not relate to the software project being developed or no understanding of data flow diagrams.	Student provides a basic Data Flow Diagram that shows no or multiple processes, some data flows, external entities and loosely relates to the software project being developed.	Student provides a Data Flow Diagram that shows one or multiple processes, some data flows and external entities and somewhat relates to the software project being developed. NESA symbols used for some of the symbols.	Student provides a Data Flow Diagram that shows one process and most data flows and external entities representing the software project being developed. NESA symbols used throughout most of the diagram.	Student provides a substantially correct Data Flow Diagram that shows one process, all data flows and external entities representing the software project being developed. NESA symbols used throughout the diagram.
Data Flow Diagram (Level 1)	Student provides a Data Flow Diagram that shows limited processes, datastores, external entities and does not relate to the software project being developed or no understanding of data flow diagrams.	Student provides a basic Data Flow Diagram that shows some processes, datastores, data flows, external entities and loosely relates to the software project being developed.	Student provides a Data Flow Diagram that shows some processes, datastores, data flows, external entities and somewhat relates to the software project being developed. NESA symbols used for some of the symbols.	Student provides a Data Flow Diagram that shows most processes, datastores, data flows and external entities representing the software project being developed. NESA symbols used throughout most of the diagram.	Student provides a substantially correct Data Flow Diagram that shows all processes, datastores, data flows and external entities representing the software project being developed. NESA symbols used throughout the diagram.

Structure Chart	Student provides a structure chart that shows limited understanding of the problem definition and does not relate to the software project being developed.	Student provides a structure chart which attempts to show a basic representation of the system by showing a basic number of the subroutines that make up the system or a basic number of relationships to each other.	Student provides a structure chart which shows a representation of the system by showing some of the subroutines that make up the system and/or some of their relationship to each other. NESA symbols used for some of the symbols.	Student provides a mostly correct structure chart which shows a representation of the system by showing most of the separate subroutines that make up the system and their relationship to each other. NESA symbols used for some of the symbols.	Student provides a substantially correct structure chart which shows a representation of the system by showing the separate subroutines that make up the system and their relationship to each other. NESA symbols used for some of the symbols.
Class Diagram	Student provides a class diagram that shows limited understanding of the problem definition and does not relate to the software project being developed.	Student provides a class diagram which attempts to provide a visual representation of some of systems being implemented using the object-oriented paradigm. The model shows a basic number of the classes, and/or their attributes and/or methods, and/or the relationships between classes.	Student provides a class diagram which provides a visual representation of some of systems being implemented using the object-oriented paradigm. The model shows some of the classes, their attributes and methods, and the relationships between classes. NESA symbols used for some of the symbols.	Student provides a mostly correct class diagram which provides a visual representation of systems being implemented using the object-oriented paradigm. The model shows most of the classes, their attributes and methods, and the relationships between classes. NESA symbols used for some of the symbols.	Student provides a substantially correct class diagram which provides a visual representation of systems being implemented using the object-oriented paradigm. The model clearly shows the classes, their attributes and methods, and the relationships between classes. NESA symbols used for some of the symbols.
Decision Tree	Student provides a decision tree that shows limited understanding of the problem definition and does not relate to the software project being developed.	Student provides a basic decision tree which represents a basic number of possible combinations of decisions and their resulting actions. Branches are shown to describe a basic number of the eventual action depending on the condition at the time. Most paths lead to either no decision or no final action.	Student provides a somewhat correct decision tree which represents some of the possible combinations of decisions and their resulting actions. Branches are shown to describe some of the eventual action depending on the condition at the time. Some decision paths lead to either another decision or a final action.	Student provides a mostly correct decision tree which represents most possible combinations of decisions and their resulting actions. Branches are shown to describe most of the eventual action depending on the condition at the time. Most decision paths lead to either another decision or a final action.	Student provides a substantially correct decision tree which represents all possible combinations of decisions and their resulting actions. Branches are shown to describe the eventual action depending on the condition at the time. Each decision path leads to either another decision or a final action.

Desk Checking (Testing)	Student documents test data or no understanding of the desk checking process.	Student attempts to use test data to check the algorithm created or shows basic set of test data.	Student correctly desks checks the algorithm and test some of the conditions with some appropriate data.	Student correctly uses test data to check the algorithm and tests most conditions/expected input against actual output and follows the course specifications.	Student correctly tests all reasonable conditions and data input using a highly appropriate desk checking table that follows the course specifications. Clear evidence of analysis and response to feedback.
CRITERIA	**0 -2**	**3 - 4**	**5 - 6**	**7 -8**	**9 -10**
Evaluation	Student provides no understanding of how to evaluate the final software solution or identifies successes or failures of the design of the solution or testing method or maintenance options	Identifies successes and/or failures of the design of the solution and/or testing method and/or maintenance options. Identifies implementation method but is not appropriate for the solution.	Outlines successes and/or failures of the design of the solution and/or testing method and/or maintenance option. Outlines an appropriate implementation method.	Describes successes and failures of the design of the solution and testing method and maintenance option. Describes an appropriate implementation method.	Clearly describes and evaluates the successes and failures of the design of the solution and testing method and maintenance option and highly appropriate implementation method with justification.
Presentation	Minimal to no description of the software's features, benefits or challenges.The student's class presentation attempts an explanation and demonstration of some of the project concepts used in the project.	Basic description of the software's features, benefits and challenges. Provides a simple outline of how this was achieved. The student demonstrates limited components of the project.	Sound overview of the software's features, benefits and challenges with examples. Some explanation of how this was achieved. The student is able to answer some of the questions raised during Q&A. The student demonstrates some components of the project.	Detailed presentation of the software's features, benefits and challenges. Detailed explanation of how this was achieved. The student is able to confidently answer questions raised during Q&A about the development process. The student demonstrates most components of the project.	Comprehensive and insightful presentation of the software's features, benefits, and challenges with clear links to improvement and future development. Thorough explanation of how this was achieved. The student provides informative and educational answers to questions raised during Q&A. The student demonstrates all components of the project.

CRITERIA	0 - 4	5 - 8	9 - 12	13 -16	17 - 20
Developing solutions with code	Student provides limited or no comments throughout the code and shows a limited understanding of intrinsic documentation.	Student partially outlines the function of the code to solve the task description and attempts to use somewhat appropriate intrinsic documentation.	Student creates, documents and outlines the function of the code to solve the task description through the use of somewhat appropriate intrinsic documentation and comments.	Student creates, documents and describes the function of the code to solve the task description through the use of appropriate intrinsic documentation and comments.	Student creates, documents and explains efficient code to solve the task description through the use of highly appropriate intrinsic documentation and comments.
CRITERIA	**0 - 1**	**2**	**3**	**4**	**5**
Degree of difficulty	Project of limited difficulty with minimal data structures.	A project of minimal difficulty which uses some data structures.	A project of moderate difficulty which uses a range of simple data structures.	A project of substantial difficulty which uses a range of data structures.	A highly demanding project which uses a range of complex data structures.
User Interface and User Experience	Student provides a limited user interface and a poor user experience or no understanding of appropriate screen design, error messages and instructions. Includes a limited number of required features.	Student provides somewhat appropriate error messages. GUI implemented is appropriate for the program. Instructions provided to the user. Includes most of the required features.	Student provides highly appropriate error messages to aid the user. GUI implemented is highly appropriate and aids in the user experience of the program. Instructions provided are clear and unambiguous. Includes all required features.		
Accessibility and Inclusivity	The student has attempted to address accessibility and inclusivity.	The student has shown some consideration of accessibility and inclusivity. The rationale is mentioned in the process diary (journal).	The student has shown good consideration of accessibility and inclusivity. The rationale is outlined in the process diary (journal).		

CRITERIA	0 - 1	2	3		
Directory Structure	Student has attempted to organise the files with folders or no understanding of file structures evident.	Some files and folders are generated routinely. Files are grouped into a logical structure, using nested folders.	Files are generated routinely and grouped into a logical structure. The rationale for the structure is included in the logbook.		

Total: /107