Gosford High School
ICT
Software Design & Development – HSC Course
Assessment Task

Module / Unit: Developing a Software Solution (Major Project)

Task Number: 1, 3 **Weighting:** 55% (S 1 = 15% - Task 3 = 40%) **Due Date:** Ongoing (see stages)

Outcomes to be assessed

- H1.2 Differentiates between various methods used to construct software solutions
- H1.3 Describes how the major components of a computer system store and manipulate data
- H2.2 Explains the interrelationship between emerging technologies and software development
- H3.2 Constructs software solutions that address legal, social and ethical issues
- H4.1 Identifies needs to which software solutions are appropriate
- H4.2 Applies appropriate development methods to solve software problems
  H4.3 Applies a modular approach to implement well structured software solutions and evaluates their
  Effectiveness.
- H5.1 Applies project management techniques to maximise the productivity of the software development
- H5.2 Creates and justifies the need for the various types of documentation required for a software solution
- H5.3 Selects and applies appropriate software to facilitate the design and development of software solutions
- H6.1 Assesses the skills required in the software development cycle
- H6.2 Communicates the processes involved in a software solution to an inexperienced user
- H6.3 Uses and describes a collaborative approach during the software development cycle
- H6.4 Develops effective user interfaces, in consultation with appropriate people

Task Description and instructions

You are required to create an independent, properly documented software solution to a problem of your choice. The major project will be submitted in two sections, with each submission comprising a formal assessment task for the subject.

Section 1 (HSC Assessment Task 1)
Major project stages 1 & 2 – 20% (Due: Term 4 – Assessment Week)

- Defining and Understanding the Problem
- Planning and Design

Section 2 (HSC Assessment Task 3)
Major project stages 3 & 4 – 40% (Due: Term 2 – Assessment Week)

- Implementation
- Testing and Evaluation

Marks will be awarded for

- Balance between scope and depth in the completed product
- Depth and clarity of project documentation
- Demonstration of competence in the learning outcomes listed above

NB: Students should frequently refer to the SDD course specification document when completing this project.

Major Project - Sections 1 and 2 (Due: Term 4, Assessment Week)

***Formal progress meetings with teacher will we conducted in Term 4, week 6**

1. Defining and Understanding the Problem

This is the initial stage where data/documents are collated and analysed to identify the overall scope and feasibility of the proposed system.

Defining the problem

- Identifying the problems: needs, objectives, boundaries.
- Determining the feasibility: is it worth solving, constraints (budgetary, operational, technical) scheduling, and project and/or development alternatives, social and ethical considerations.

Design Specifications

- The developer' perspective: data types, variables, software design approach
- The user's perspective: screen design prototypes, appropriate messages & icons, relevant data formats for display, ergonomic issues, relevance to the users' environment and computer configuration.

Modelling

- Representing the system using diagrams:
  IPO, context diagram, data flow diagrams, structure chart, system flowchart, data dictionary, storyboard.

Communication

- Agenda for progress meeting with teacher (work completed to date, problems encountered, questions etc.)
- Minutes of progress meeting with teacher
- Personal log book

2. Planning and Design of Software Solutions

This is the stage where data structures and algorithms are designed for the final solution. Using the information and models collated from stage 1, you can now develop the data structures and logic necessary to develop modules for the system.

Algorithms

- The creation of the algorithms to be used in your project. Algorithms should be in the form of modules comprised of subroutines, with each performing a

single defined task. All algorithms are to be represented in pseudocode. At least one algorithm is to be presented as a flowchart.

Selection of Language to be used

- What programming language is the most appropriate for the system (ask yourself: is the logic user or programmer driven? Is a GUI required? What experience you have as a developer? Does the language provide all the required features for the solution?

Major Project - Sections 3 and 4 (Due: Term 2, Assessment Week)

***Formal progress meetings with teacher to be conducted :**

- **Term 1, week 8**
- **Term 2, week 6**

---

3. Implementation of Software Solutions

During this stage the plans, designs formulated in stage 2 are implemented into a programming language- Development and Testing of source code.

Interface Design

- The design of individual screens: identifying data required, the design of consistent interface, help screens and ensuring it meets the intended audience.

Program Development & Techniques

- Code generation: structured approach (one logical task per subroutine), debugging output statements, elegance of solution, comments, flags, stubs.
- Process of detecting and correcting errors: (syntax, logic, peer checking, desk checking, use of expected output, runtime errors.
- The use of software debugging tools: resetting variable contents, program traces, single line stepping.

Documentation

- User documentation: installation guide, user manual, tutorial.
- Technical documentation: including source code, algorithms, data dictionary and systems documentation.

Communication

- Agenda for progress meeting with teacher (work completed to date, problems encountered, questions etc.)
- Minutes of progress meeting with teacher
- Log Book: Personal progress diary completed throughout the project.

4. Testing and Evaluation of Software Solutions

In this stage we are ensuring the product meets the original objectives and design specifications outlined in stage 1. Also, we are evaluating the product's performance by engaging the users to use and report any faults or recommendations.

Testing the software solution

- Comparison of the solution with the original design specifications.
- Levels of testing (unit or module, program, system)
- The Use of live test data to test the complete solution: larger file sizes, mix of transaction types, response time, interfaces between modules)

Reporting on the testing process

- Documentation and communication of the test data/results and output produced. (Test requirements, test plan, test data and expected results, test results, recommendations).

Major Project Sections 1 and 2 – Marking Guide

Section 1 - Defining and Understanding the problem

Defining the problem

Criteria	Mark	
The problem is clearly defined and feasibility of the proposed system is evaluated thoroughly. All elements outlined in the brief are included in the response together with supporting data/documentation.	17-20	
The problem is clearly defined and feasibility of the proposed system evaluated effectively. Some elements outlined in the brief have been omitted and/or supporting data / documentation is missing.	12-16	
Students collect some data / documentation and provide a basic feasibility report. The problem is adequately defined.	6-11	
Students collect minimal or no data / documentation which restricts the thoroughness of their overall scope and feasibility report. The problem is not clearly identified and feasibility is too brief.	0-5	

Design specifications

Criteria	Mark	
Both perspectives correlate realistically and feasibly with the problem/product. All elements of both the developer's and user's perspective are addressed adequately	17-20	
Both perspectives correlate realistically and feasibly with the problem/product. Most elements of both the developer's and user's perspective are addressed adequately	12-16	
Elements of the developer's and/or user's perspective are missing in the report.	6-11	
Report does not provide an effective overview of the design specifications. Very brief responses and major elements missing.	0-5	

Modelling

Criteria	Mark	
Models provide a clear and concise overview of the proposed system. Demonstrates exceptional understanding of symbols, structure and purpose of the models. All models outlined in the brief are included.	24-30	
Effectively demonstrates the structure and functionality of the proposed system through the use of models. The majority of models outlined in the brief are included and completed correctly	16-23	
Models and diagrams provide a reasonable overview of the proposed system. Response includes some of the models outlined in the brief completed correctly and in context.	8-15	
Basic understanding of modelling tools. Incomplete diagrams and/or a number of errors in each diagram. The diagrams do not effectively represent the system.	0-7	

Communication

Criteria	Mark	
Substantial evidence of progress meetings is present. Agenda is well planned and minutes are clear and accurate. Clear and concise ongoing personal logs reflect the student's progress throughout the SDLC.	8-10	
Clear evidence of progress meetings is present in the form of a detailed meeting agenda and minutes. Personal logs reflect the student's progress through the SDLC.	6-7	
Recorded evidence of progress meetings is present but minimal. Logs are brief and/or do not effectively outline the student's progress to date through the SDLC.	4-5	
Limited evidence of progress meetings/communication with the teacher. Few or no logs were written.	0-3	

Section 2 - Planning and Design of Software Solutions

Standard Algorithms

Criteria	Mark	
Evidence of thorough and strategic creation of standard algorithms. Demonstrates how these algorithms will be used in the completed system.	24-30	
Demonstrates a general understanding of standard algorithms and how they can be used in a completed system. Algorithms created are incomplete or missing some elements.	16-23	
Demonstrates a basic understanding of standard algorithms. Algorithms created are largely incorrect and/major elements are missing.	8-15	
Limited understanding and minimal or no creation of standard algorithms.	0-7	

Selection of programming language(s)

Criteria	Mark	
Identifies, selects and accurately justifies an appropriate programming language that meets the demands of the solution.	8-10	
Selects an appropriate programming language and provides a brief justification.	6-7	
Selects a programming language that may not be suitable for the software solution and/or provides little justification for the choice.	4-5	
Identifies one or more programming languages.	0-3	

Presentation

Criteria	Mark	
A professionally designed and formatted report. Structure uses correct standards and the reader can easily find specific information via use of a dynamic table of contents.	8-10	
An adequately formatted and designed report. Structure uses correct standards and a table of contents is included.	6-7	
Report uses basic formatting and minimal design. Standards are inconsistent and/or table of contents is missing or inaccurate.	4-5	
A rudimentary formatted report. Minimal use of standards and/or lacks presentation. Table of contents is missing or inaccurate.	0-3	

TOTAL

	/130
	%

Section 3 Implementation of Software Solutions

Interface Design

Criteria	Mark	
Demonstrates a comprehensive understanding of development of effective user friendly interfaces. This will include all of the below elements: • Thorough consultation with client • The design targets the intended audience • Takes into account the ergonomic issues of software design (e.g. consistency, alignment, white space, use of colour and font appropriately, correct use of screen elements, use of standards).	17-20	
Demonstrates substantial understanding of development user friendly interfaces by applying the majority of the elements above.	12-16	
Demonstrates clear understanding of how to develop interfaces by including some of the elements.	6-11	
Demonstrates understanding of interface design, applying at least one element.	0-5	

Program Development & Techniques

Criteria	Mark	
Provides a software solution that is substantially correct AND demonstrates a comprehensive understanding of: • Correct code generation • The use of software techniques (debugging output statements…) • The use of software tools (program trace, single line stepping…) • One logical task per subroutine • The use of correct control structures • Intrinsic documentation (comments, logical variable name)	18-20	
Provides a software solution that is mostly correct AND demonstrates substantial understanding by applying the majority of the elements above	10-17	
Provides a software solution that shows limited programming skills AND demonstrates an understanding of the elements above	1-9	

Documentation

Criteria	Mark	
Demonstrates a good understanding of software solutions by developing ALL of the: • User documentation (installation guide, user manual, How to guides) • Technical documentation • Logs (communication, personal reflection/progress)	8-10	
Demonstrates some understanding of software solutions by developing a majority of the elements above.	4-7	
Demonstrates limited understanding of software solution by developing some of the elements above	1-3	

Section 4 Testing and Evaluation of Software Solutions

Testing the software solution

Criteria	Mark	
Demonstrates a good understanding of testing the solution by: • Incorporating a comparison with the original design specifications • Applies a variety of level testing (unit, module, system) • The use of live test data to check response time, interfaces b/t modules, transaction types etc.	8-10	
Demonstrates some understanding of testing the solution by developing a majority of the elements above	4-7	
Demonstrates limited understanding of testing the solution by developing some of the elements above.	1-3	

Reporting on the testing process

Criteria	Mark	
Provides documentation of the testing process that is substantially correct AND demonstrates a good understanding of: • Creating test data • Test plans • Test results • recommendation	8-10	
Provides documentation of the testing process that is substantially correct AND demonstrates substantial understanding of the majority of the elements above.	4-7	
Provides documentation of the testing process that limited AND demonstrates some understanding of elements above	1-3	

TOTAL

	/70
	%