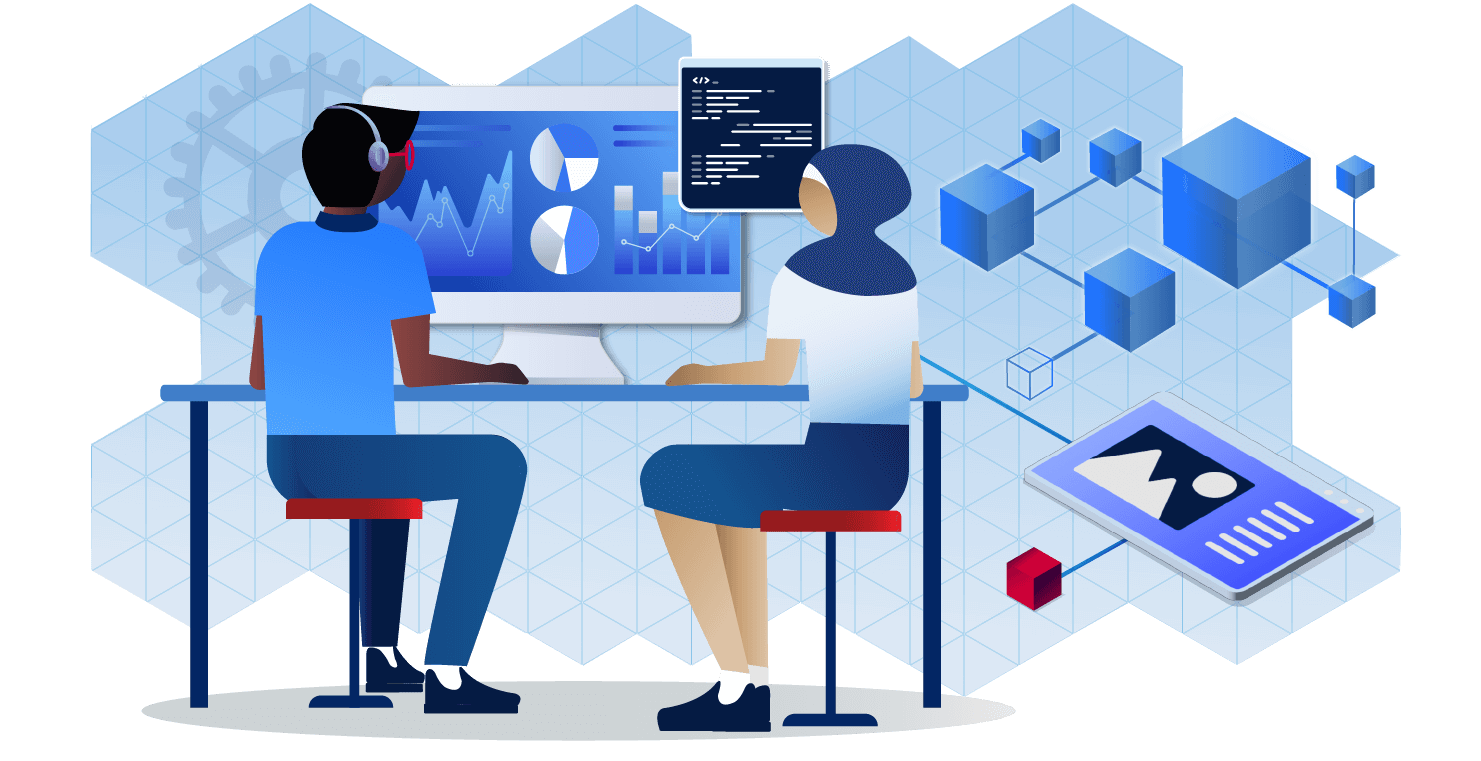
# Software Engineering Stage 6 (Year 12) – Sample Software Engineering Systems Report Template



Contents

[1. Identifying and defining 2](#_Toc196993426)

[1.1. Define and analyse problem requirements 2](#_Toc196993427)

[1.2. Tools to develop ideas and generate solutions 3](#_Toc196993428)

[2. Research and planning 4](#_Toc196993429)

[2.1. Project management 4](#_Toc196993430)

[2.2. Quality assurance 5](#_Toc196993431)

[2.3. Systems modelling 6](#_Toc196993432)

[3. Producing and implementing 12](#_Toc196993433)

[4. Testing and evaluating 13](#_Toc196993434)

[4.1. Evaluation of code 13](#_Toc196993435)

[4.2. Evaluation of solution 14](#_Toc196993436)

[Progress Diary 18](#_Toc196993437)

## 1. Identifying and defining

### 1.1. Define and analyse problem requirements

**Problem context**  
Students **analyse** the problem by **describing** each of its individual components and **explaining** how each of these components contribute to the problem needing resolution.

The current process of running a school athletics carnival is largely inefficient,

**Needs and opportunities**Students **describe** the needs of the new system to be built based on the problem context and using the table given below.

|  |  |
| --- | --- |
| Need | Description |
| 1. Registration system | A system for staff to register students for events, including their name, year, house, and selected events. |
| 2. Score and Time Entry | A digital interface for staff to quickly enter event results (e.g., times or distances) and assign scores. |
| 3. House Points Tracker | Automatically adds event points to house totals and updates a live leaderboard. |

**Boundaries**

Students **analyse** any limitations or boundaries in which this new system will need to operate. Boundaries can include but are not limited to: hardware, operating systems, security concerns etc*.*

The system must be designed to run on school-approved laptops and desktops, likely in a browser or standalone Python program. It cannot rely on internet access and should store data locally or on the school’s shared drive. Login and security should be minimal but functional — basic staff authentication. Only teachers can edit results. It must be simple enough for non-technical staff to use during high-pressure events.

### 1.2. Tools to develop ideas and generate solutions

**Identification of appropriate software development tools**Students **identify** appropriate tools for each of the given software development situations. They then **explain** how each tool is applicable or not to the current project.

|  |  |  |
| --- | --- | --- |
| Situation | Tool applicability | Reason |
| Brainstorming, mind-mapping and storyboards | Useful | Used to outline how the system will work visually, and what features are most important. |
| Data dictionaries | Useful | Helps define how variables like student name, house, and score will be stored and validated. |
| Algorithm design | Useful | Core logic for processing scores, ranking students, and updating house points. |
| Code generation | Useful | Python will be used to build the program and it will help with certain foundational aspects. |
| Testing and debugging | Useful | Required to ensure the system functions during the live carnival — bugs must be fixed quickly. |
| Installation | Limited use | If it's browser-based or local-only, setup must be fast and reliable. |
| Maintenance | Useful | Future carnivals may reuse or expand the system — modular code allows for updates. |

**Implementation method**  
Students **explain** the applicability of the implementation method for the current project. These are normally: direct, phased, parallel and pilot.

The Implementation method used will be the Pilot Implementation method, as we can test it in a junior carnival (say Year 7 to 9) and observe the performance of the program and gain feedback in live time.

## 2. Research and planning

### 2.1. Project management

**Software development approach**Students **explain** the software development approach most applicable for this current project. These are normally: Waterfall, Agile and WAgile.

**Scheduling and task allocation**Students **develop** a Gantt Chart that details the tasks required to be completed, person or people assigned to each task, timeline that does not exceed the project due date, resources required. In addition, students **identify** any collaborative tools used. For example Repl.it, GitHub and so on.

### 2.2. Quality assurance

**Quality criteria**Students **explain** quality criteria based upon the needs from Section 1.1. These quality criteria should contain qualities, characteristics or components that need to be included or visible – based on Section 1.1. – by the end of the current project.

|  |  |
| --- | --- |
| Quality criteria | Explanation |
|  |  |
|  |  |
|  |  |

**Compliance and legislative requirements**Students **explain** compliance and legislative requirements their projects need to meet and how they plan to mitigate them where possible. For example, projects that deal with sensitive personal data being publicly available may fall under the Australian [NSW Privacy and Personal Information Act (1998)](https://legislation.nsw.gov.au/view/whole/html/inforce/current/act-1998-133#statusinformation) and/or [Federal Privacy Act (1988)](https://www.legislation.gov.au/Series/C2004A03712). Alternatively, international standards on information security management such as [ISO/IEC 27001](https://www.iso.org/standard/27001) may also be applicable.

|  |  |
| --- | --- |
| Compliance or legislative issue | Methods for mitigation |
|  |  |
|  |  |
|  |  |

### 2.3. Systems modelling

Students are to **develop** the given tables and diagrams. Students should consult the [Software Engineering Course Specifications](https://library.curriculum.nsw.edu.au/341419dc-8ec2-0289-7225-6db7f2d751ef/94e1eb0a-0df7-4dbe-9b72-5d5e0d17143a/software-engineering-11-12-higher-school-certificate-course-specifications.PDF) guide should they require further detail, exemplars or information. Each subsection below should be completed with Section 1.1. in mind.

**Data dictionaries and data types**Students take the needs identified in Section 1.1. of this Systems Report. For each need, students **identify** the variables required, data types, format for display, and so on.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Need |  |  |  |  |  |  |  |
| 1. | | | | | | | |
| Variable | **Data type** | **Format for display** | **Size in bytes** | **Size for display** | **Description** | **Example** | **Validation** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

| Need |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2. | | | | | | | |
| Variable | **Data type** | **Format for display** | **Size in bytes** | **Size for display** | **Description** | **Example** | **Validation** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

| Need |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 3. | | | | | | | |
| Variable | **Data type** | **Format for display** | **Size in bytes** | **Size for display** | **Description** | **Example** | **Validation** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**Data flow diagrams**

Students **develop** data flow diagrams (DFDs) at Level 0 and Level 1. These diagrams should explicitly include the variables from the data dictionaries previously identified as well as the needs identified in Section 1.1.

*Level 0*

*Level 1*

**Structure charts**Students **develop** structure charts demonstrating how the procedures, modules or components of the final solution are interconnected.

**Class diagrams**Students **develop** class diagrams demonstrating how each class is related to the other.

**Storyboards**Students **develop** storyboards, visually representing the software solutions they will build.

**Decision trees**Students **develop** decision trees to visually outline the logic flow and chain of decisions or selections the final solution will need.

**Algorithm design**Students **develop** algorithms using methods such as pseudocode or flowcharts to solve the problem and meet the needs from Section 1.1. These algorithms should explicitly include the variables from the data dictionaries created in the previous section.

## 3. Producing and implementing

**Solution to software problem**Students are to **include** screen shots of their final developed solution here. Each screenshot should include a caption that **explains** how it links to the:

* Needs identified in Section 1.1.
* Components of Section 2.3. such as the storyboards, data dictionaries and so on.

**Version control**Students **describe** what version control system or protocol was implemented.

## 4. Testing and evaluating

### 4.1. Evaluation of code

**Methodology to test and evaluate code**Students **explain** the methodologies used to test and evaluate code. Methodologies include:

* Unit, subsystem and system testing
* Black, white and grey box testing
* Quality assurance.

**Code optimisation**Students **explain** the methodologies used to optimise code so that it runs faster and more efficiently. Methodologies include:

* Dead code elimination
* Code movement
* Strength reduction
* Common sub-expression elimination
* Compile time evaluation – constant folding and constant propagation.

### 4.2. Evaluation of solution

**Analysis of feedback**Students **analyse** feedback given to them on the new system they have just created. This feedback can be in the form of an interview, survey, focus group, observation or any other applicable method. Students should also include overall positive, negative or neutral sentiments towards the new system in their response.

**Testing methods**Students **identify** the method or methods of testing used in this current project. For each they use, students are to **explain** how and why it was used.

|  |  |  |
| --- | --- | --- |
| Method | Applicability | Reasoning |
| Functional testing |  |  |
| Acceptance testing |  |  |
| Live data |  |  |
| Simulated data |  |  |
| Beta testing |  |  |
| Volume testing |  |  |

**Test data tables**Students **identify** variables which were used for either path and/or boundary testing. Students **develop** these test data tables based on their algorithms versus their real code. Students then **state** the reason for including said variables.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Variable | Maximum | Minimum | Default Value | Expected Output | Actual Output | Reason for Inclusion |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Boundary testing

Path testing

**Analysis of solution against quality success criteria**Students are to take each quality success criteria from Section 2.2 and place it here. For each quality criteria, **analyse** the components of the solution that met or did not meet each quality criteria. Give reasons why each success criteria were or were not met.

|  |  |  |
| --- | --- | --- |
| Quality criteria | Met? | Analysis |
|  |  |  |
|  |  |  |
|  |  |  |

## Progress Diary

| **Date** | **Information** |
| --- | --- |
| **12/05/2025** | Decided to officially switch project focus from |
| **14/05/2025** | Wrote a new problem statement. I saw the key systems which were needed and researched the |
| **16/05/2025** | Started writing the Python code for this program, Set up basic event creation and registration functions. Encountered some basic coding issues with various things including syntax and logic errors which were fixed. |
| **17/05/2025** | Started to attempt to make the House point calculation and sure the code at least runs. (This is still a work in progress) |
| **19/05/2025** | Began re-doing the documentation from scratch, with the Section 1.1 and 1.2 completed. |
| **20/05/2025** | Added show event data function, continued updating the documentation with relevant information |
| **21/05/2025** | Started completing Section 2.1 of the template documentation. Had an attempt to list system limitations based on what I’ve coded so far. Finalised incomplete code version for upload to the GitHub repository tomorrow. |
| **22/05/2025** | Today we started to organize our current files and finally upload them to the GitHub Repository |