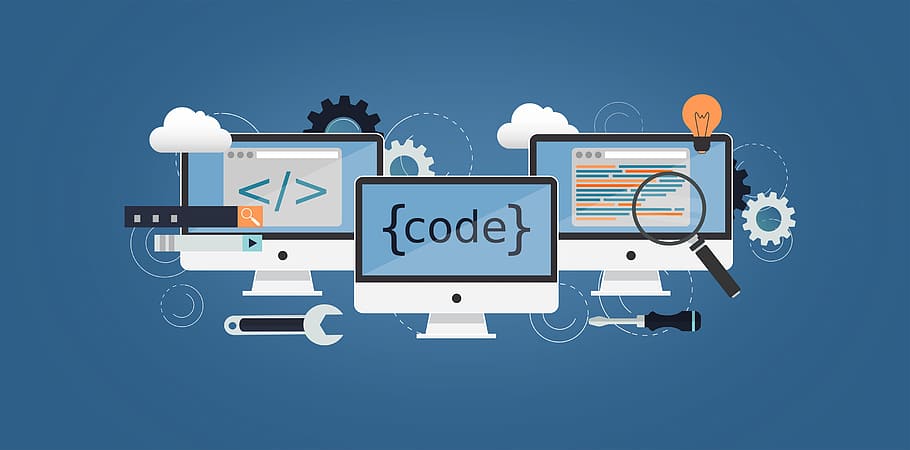
# 

# HSC Software Engineering

# Systems Report



#### Name: Hayden Mercier

#### Class: 12SWE1

Contents

[1. Identifying and defining 2](#_Toc135061669)

[1.1. Define and analyse problem requirements 2](#_Toc135061670)

[1.2. Tools to develop ideas and generate solutions 3](#_Toc135061671)

[2. Research and planning 4](#_Toc135061672)

[2.1. Project management 4](#_Toc135061673)

[2.2. Quality assurance 5](#_Toc135061674)

[2.3. Systems modelling 6](#_Toc135061675)

[3. Producing and implementing 12](#_Toc135061676)

[4. Testing and evaluating 13](#_Toc135061677)

[4.1. Evaluation of code 13](#_Toc135061678)

[4.2. Evaluation of solution 14](#_Toc135061679)

## 1. Identifying and defining

### 1.1. Define and analyse problem requirements

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

As part of the Software Engineering Major Project, I need to develop a piece of software that solves the problem identified by my clients.

My clients, a group of Y11 students, would like to be able to enjoy playing games during Y11 and Y12, which is a busy time for many students. However, the current games they play, such as Hypixel Skyblock, sink large amounts of time and can lead to an extreme loss of productivity. They want to be able to play similar games, but without the massive time loss that occurs due to the focus required. Even in the most basic skill in Hypixel Skyblock, farming, without Macros, which are not allowed on their servers, requires one hand on the mouse to move the character and farm crops in a custom farm if keys are rebound.

To solve the problem, I have determined that a semi-idle farming game that has a gameplay loop where a significant chunk of time can be spent away from the game, studying or exercising, is necessary, and that continued communication between myself and the client will ensure that the optimal solution is developed.

As part of this major project, I will also need to complete the following to ensure my solution is feasible, functional, and meets the assessment requirements:

* The game is largely glitch-free due to significant testing.
* The game can write to a file to save the game. This save file should also be able to be loaded by the game.
  + This save file should be able to resist tampering in order to ensure that my clients can fairly compete against each other.
* The game has an intuitive and aesthetically pleasing GUI, and settings can be changed during the game to improve the user experience.
* The project uses an object-oriented programming language, as specified in the assessment requirements.
* All game assets and dependencies are sourced legally and ethically, using original content, creative commons assets, or licensed material
* Fulfil the requirements of my clients, which include:
  + The game is an idle game that enables users to have fun and compete against each other while not requiring huge amounts of gameplay time.
  + The game has a core gameplay loop that is enough time for one study session to occur, which can range from 30-60 minutes.
  + The game looks aesthetically pleasing, and the assets are detailed and have been approved by the clients.
  + The game comes with instructions or a tutorial on how to play the game, to ensure that all users understand the game sufficiently.
  + The game is largely scalable to ensure that future changes can be made to the game by me or by other developers, such as my clients or my friends, if they want to continue with the project.

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

A successful project meets all needs and a maximum subset of wants.

|  |  |
| --- | --- |
| Need | Description |
| 1. Intuitive GUI that provides relevant information to the user. |  |
| 2. Ability to both save and load the game. |  |
| 3. |  |

**Boundaries**

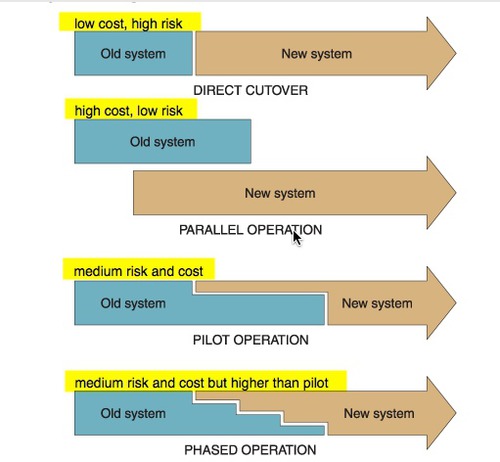
**Analyse** any limitations or boundaries in which this new system will need to operate. Boundaries can include but are not limited to: hardware, operating systems, security concerns etc*.*

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

**Identification of appropriate software development tools DON’T DO IT  
Identify** appropriate tools for each of the given software development situations. Then **explain** how each tool is applicable or not to your current project.

|  |  |  |
| --- | --- | --- |
| Situation | Tool applicability | Reason |
| Brainstorming, mind-mapping and storyboards |  |  |
| Data dictionaries |  |  |
| Algorithm design |  |  |
| Code generation |  |  |
| Testing and debugging |  |  |
| Installation |  |  |
| Maintenance |  |  |

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



Direct or pilot

Direct is like financial changes such as the inclusion of GST.

## 2. Research and planning

### 2.1. Project management

**Software development approach  
Explain** the software development approach most applicable for your project. These are normally: Waterfall, Agile and WAgile.

Waterfall is typically only used for very large projects which need to be done in a specific order, such as large construction projects.

Software solutions are usually never at this stage and hence most benefit from the agile development approach. Everytime feedback is received in the agile approach, it is gotten from the client and the public.

<https://www.reddit.com/r/agile/comments/pgi6l6/the_amount_of_hate_i_feel_for_wagile_waterfall/>

use waterfall for more structure with more strucure and documentation and whern requirements are not expected to change.

Use agile for less structure and changing requirements, adapting to new circumstances.

Use Wagile to leverage the benefits of both project management techniques.

**Scheduling and task allocation  
Develop** a Gantt Chart that details the tasks required to be completed and timeline that does not exceed the project due date. In addition, **identify** any collaborative tools used. For example Repl.it, GitHub and so on.

The use of [Gantt Project](https://www.ganttproject.biz/) is recommended.

### 2.2. Quality assurance

**Quality criteria  
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 |
| Game operates on low end systems with minimum hardware requirements. | **Detailed quality planning in product design and process development**  Detailed quality planning is required to specify the minimum specifications and minimum performance benchmark to meet.  **Ongoing audits and risk assessment**  An audit of current typical minimum specifications for machines in 2025 will be identified and used as a baseline.  **Continual measurement and analysis of quality data**  Analysis of minimum game performance benchmarks is required to accurately measure the quality of the solution.  **Extensive collaboration and communication across teams**  Extensive collaboration and communication with your client will determine approval of the game performing at the minimum performance benchmark on a minimum specification machine. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Compliance and legislative requirements  
Explain** compliance and legislative requirements your project needs 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 |
| Age | * Violence and Content warnings * No gambling cause minimum M rating |
| Personal information | * Not collecting personal information due to the lack of information on the |
| Copyright and Creative Commons |  |
| Security Standards |  |
| Software dependencies |  |
| Consumer rights and guarantees |  |

-

### 2.3. Systems modelling

**Develop** the given tables and diagrams. You 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 you require further detail, exemplars or information. Each subsection below should be completed with Section 1.1. in mind.

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

1 byte: 256

2 bytes: 65,535

3 bytes: 16,777,215

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Need |  |  |  |  |  |  |  |
| 1. | | | | | | | |
| Variable | **Data type** | **Format for display** | **Size in bytes** | **Size for display** | **Description** | **Example** | **Validation** |
|  | Float | NN.NNN | 4 | 7 (includes dot point) |  |  | >=0 |
|  | Bool | True/False | 1 Bit | 5 |  |  |  |
|  | Int | NNNNN | However many bytes are required for display. |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

| 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** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

Cap at 30 variables using a range of types

**Data flow diagrams**

**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  
Develop** a structure chart demonstrating how the modules of the final solution are interconnected. A maximum 20 modules are recommended.

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

**Storyboards  
Develop** storyboards, visually representing the software solution you will build.

**Decision trees  
Develop** a decision tree to visually outline the logic flow of at least one process and chain of decisions or selections the final solution will need.

**Algorithm design  
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  
Include** screen shots of your 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  
Describe** what version control system or protocol was implemented in your solution. If none, make a proposal.

I used the version control software Git and GitHub in order to ensure that I was never at risk of losing progress due to the Godot Engine accidentally crashing or making reverse progress when attempting to fix major issues.

## 4. Testing and evaluating

### 4.1. Evaluation of code

**Methodology to test and evaluate code  
Explain** the methodologies used to test and evaluate code in your solution. Methodologies include:

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

**Code optimisation  
Explain** the methodologies used to optimise code in your solution 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  
Analyse** feedback given to you on the new system you have just created. This feedback can be in the form of an interview, survey, focus group, observation or any other applicable method. You should also include overall positive, negative or neutral sentiments towards the new system in their response.

**Testing methods  
Identify** the method or methods of testing used in this current project. For each use, you need 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  
Identify** variables which were used for either path and/or boundary testing. **Develop** these test data tables based on your algorithms versus their real code. Then **state** the reason for including said variables.

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

**Boundary testing**

**Analysis of solution against quality success criteria**Copy each quality success criteria from Section 2.2 and paste below. 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 |
|  |  |  |
|  |  |  |
|  |  |  |

## References

NSW Legislation (2023) [NSW Privacy and Personal Information Act (1998)](https://legislation.nsw.gov.au/view/whole/html/inforce/current/act-1998-133#statusinformation), website, **accessed 26 May 2023.**

Federal Register of Legislation (2023) [Federal Privacy Act (1988)](https://www.legislation.gov.au/Series/C2004A03712), Australian Government website, **accessed 26 May 2023.**

ISO Standards (2023) [ISO/IEC 27001](https://www.iso.org/standard/27001)**, website, accessed 26 May 2023.**

NESA (2023) [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), **NESA website, accessed 26 May 2023.**