**Software Engineering Year 12**

**Programming Project**

**Project Documentation: Ryan Hanna**

Table of Contents

[Defining and understanding 3](#_Toc191399874)

[Task Definition 3](#_Toc191399875)

[Functional and non-functional specifications 4](#_Toc191399876)

[Functional Specifications 4](#_Toc191399877)

[Non-functional Specifications 4](#_Toc191399878)

[Data Types & Structures 5](#_Toc191399879)

[Planning and designing 6](#_Toc191399880)

[Storyboard 6](#_Toc191399881)

[Data Flow Diagram (Level 0 Context Diagram) 7](#_Toc191399882)

[Data Flow Diagram (Level 1) 8](#_Toc191399883)

[Structure Chart 9](#_Toc191399884)

[Algorithms 10](#_Toc191399885)

[Mainline Algorithm 10](#_Toc191399886)

[GANTT Chart 11](#_Toc191399887)

[Implementation 12](#_Toc191399888)

[Development log 12](#_Toc191399889)

[Testing 13](#_Toc191399890)

[Test Table 13](#_Toc191399891)

[Project showcase 14](#_Toc191399892)

[Repository 14](#_Toc191399893)

[Evaluation 15](#_Toc191399894)

[Project Reflection 15](#_Toc191399895)

# Defining and understanding

## Task Definition

*Communicate the core issue your software aims to address, specifying the intended users of the software to ensure the solution's relevance.*

*Define the project's scope (what it will and won’t do), confirming that it is realistic and achievable within the given constraints.*

*Identify any limitations they may encounter during development, such as time constraints, resource availability, or existing technical skills.*

*Provide a brief overview of the intended solution should be provided, outlining the primary features or functionalities that will tackle the identified problem.*

*250-word limit*

The issue I aim to address is that teaching theory of software engineering can be disengaging for younger students such as people from years 7-12, the idea for this project is to design a fun and engaging game to help teach the process of design and planning through interactive and fun gameplay elements.

Some limitations will be creating a AAA game and covering all parts of software engineering.

My intention will be to create some sort of game with a factory/tycoon style gameplay aimed at teaching fundamental design processes such as DFD diagrams, Classes, Structure Charts, etc…

Some primary features of the project will include:

* Gameplay system where the player will create a system by connecting and creating processes to complete a goal
* A zoomed-out view of the created system will be displayed as a DFD
* Functions such as a system design page where the player can layout a structure for their factory in a structure chart layout

Functional and non-functional specifications

### Functional Specifications

|  |  |
| --- | --- |
| Requirement | Description |
| Functions which replicate the process of designing and planning software projects | Functions such as the main factory layout when zoomed out should be visualised as a DFD, or when creating new factories and layouts they should be done as a structure chart like method. |
| Extension Game | Should be a small game able to be accessed as a browser extension, this means its quickly accessible and if Uploaded to the Chrome Extensions store can be easily installed and played. Also allows for offline gameplay with possible syncing capabilities for saving games when reconnecting |
| **tool for teaching Software Engineering topics**  **(DFD, IPO)** | Using functions within the game to teach certain topics within software engineering to make learning an interactive and fun experience which in turn is better for memory. |
| **Game Objectives:** | **Take Orders from clients and deliver the correct Files and packages back.** |
| **Difficulty System:** | * **More customers to serve at a time** * **Larger orders** |
| **Needed functions for game:** | * **Orders from client** * **Package generator/button** * **Files generator/button** * **Storage system for packages and Files** * **Transport system for items** |

### Non-functional Specifications

|  |  |
| --- | --- |
| Requirement | Description |
| Fun and Engaging | The design and gameplay should be fun and engaging for the user so that it doesn't feel like learning. |
| Simplistic yet pleasing Style | The game should be a simplistic and easy to learn game and the style should be pleasing to look at |
| Add a vignette to game screen | Creates a fun playful feeling scene rather than a flat harsh background |
| NATS IDEAS:   * Falling boxes cursor physics (bounce of cursor or avoid it while falling) * News Paper with factory status/leaderboard * Fun animations for***everything*** | Animation ideas:  . Pop-in/out when placing or deleting tiles  . Smooth rotation and flipping animation when transforming tiles  . Title screen animation (falling/rotating packages)  . Loading animation (Box opening and unfolding)  . Screen load animation when first loading tiles |
| Intuitive gameplay | The buttons and elements within the game should be easy to use and understand, game theme should be obvious |

* Functional specifications are key features of what the software does, while non-functional specifications describe how the software performs its functions. For example, a simple reminders app may have the following requirements:
  1. Non-functional: app should have an intuitive interface that users can navigate without a tutorial.
  2. Functional: app should enable users to enter a reminder into a list, including the time of reminding.
* Two tables, at least 5 functional requirements, 5 non-functional requirements

## Data Types & Structures

Data Dictionary

# Planning and designing

Storyboard

* A storyboard shows the various interfaces (screens) as well as the links between them.

A diagram of a few options

Description automatically generated with medium confidence

* For a Command Line Interface (CLI) application that lacks a GUI, a storyboard can be presented as a series of steps outlining each stage of interaction the user has with the application. For example, a CLI based simple reminder application could be outlined as follows:

Welcome Screen:

Display a welcome message.

Present the user with the main command options: “[A] Add Reminder, [V] View Reminders, [Q] Quit.”

Add Task Interaction:

If the user enters “A”, prompt for the task description: "Please enter the task description."

After the user inputs the description, show a confirmation message: "Task added successfully."

[and so on.]

* Use a digital drawing tool such as draw.io

Data Flow Diagram (Level 0 Context Diagram)

* Level 0 data flow diagrams, also known as Context Diagrams, provide a system-wide overview and do not include data stores or internal processes, as depicted in Data Flow Diagrams (see below).
* Below is a Level 0 data flow diagram example for a voting system:

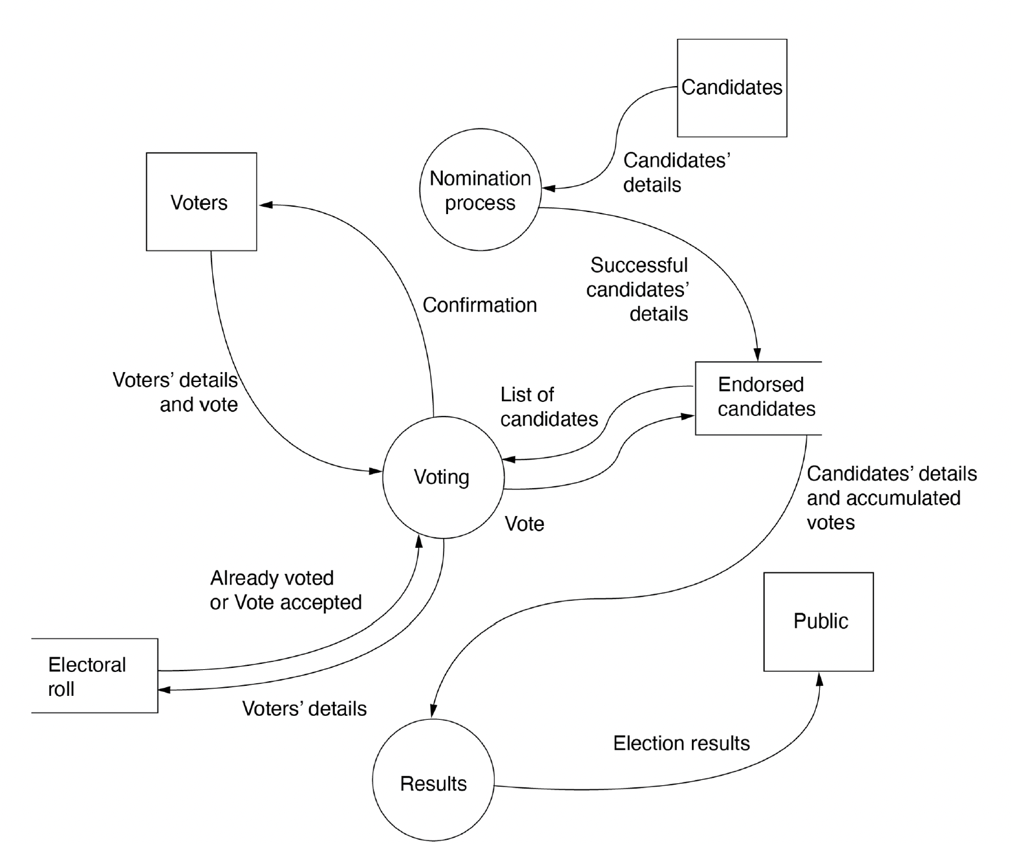
A diagram of voting

Description automatically generated

* When creating the diagram, use a straightforward system name and focus on identifying the system's inputs and outputs, along with their providers and recipients.
* Use a digital drawing tool such as draw.io

Data Flow Diagram (Level 1)

* A Data Flow Diagram (DFD) is a detailed breakdown the system identified in your Context Diagram (Level 0 DFD). It represents the flow of data within that system, including the individual processes, data stores, data flows, and external entities involved.
* The symbols you must use are outlined here:   
    
  A white paper with black text

  Description automatically generated
* Here is an example DFD for the voting system example given above: 
* Use a digital drawing tool such as draw.io

Structure Chart

* A structure chart is suited to students that are taking a functional approach to the development of their solution. It serves as a visual representation of the program's modular structure and the hierarchy of functions within the program. The symbols you must use are outlined here:

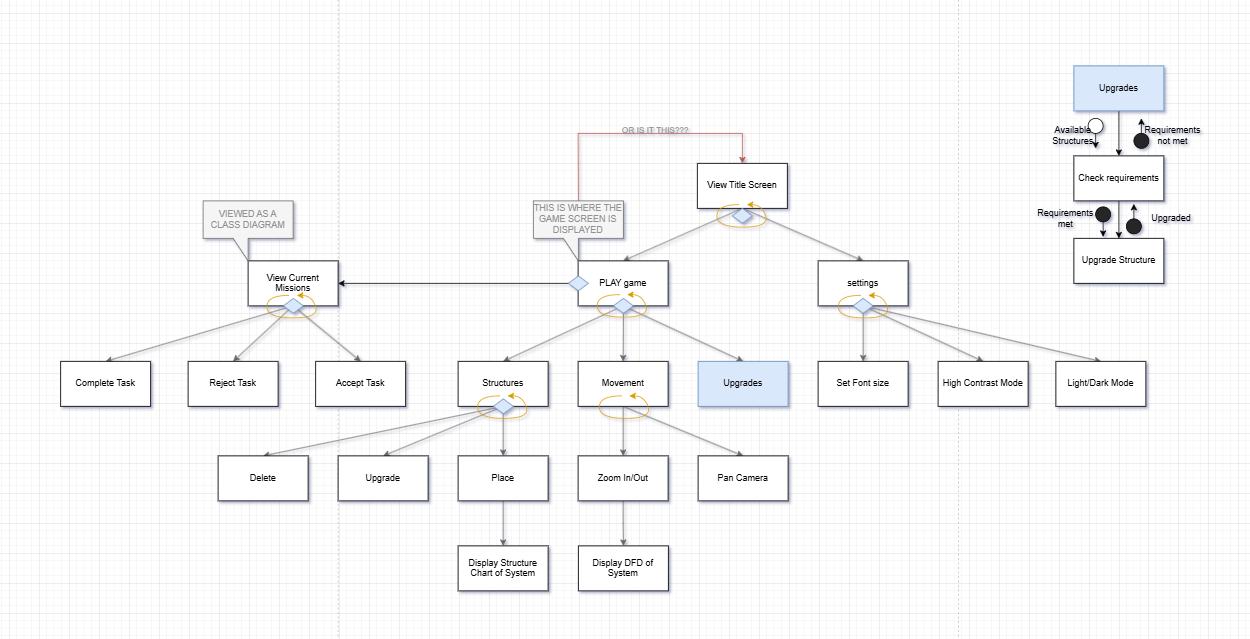
A white background with black text

AI-generated content may be incorrect.

* An example structure chart for a school library system:

A diagram of a library system

AI-generated content may be incorrect.

* Use a digital drawing tool such as draw.io
* 

Algorithms

Mainline Algorithm

GANTT Chart

* A GANTT chart should be generated at the project's outset, outlining estimated durations for key tasks related to software development. Regular review and adjustments should be made to the chart as the project progresses to reflect actual progress made.
* Students should submit a GANTT chart mapped out at the beginning of the project, and a second GANTT chart to illustrate the actual development timings experienced.
* A screenshot of a computer

  AI-generated content may be incorrect.
* Students can create GANTT charts using a web application that follows a format similar to the example provided, which can be accessed at: <https://www.onlinegantt.com/#/gantt>

# Implementation

Development log

* The purpose of the development log is to capture a snapshot of your project's evolution. It should highlight the obstacles you've encountered, the approaches you've taken to overcome them, the progress you've made, and any significant breakthroughs. Below is the recommended format for your log entries.
* Your project documentation should include at least weekly detailed log entries over the life of the project. These entries should give us a clear view of your development process. A well-crafted log is genuine, includes excerpts of code, screenshots, and conveys the story of your project's development throughout the term.
* Remember, the log should be kept up-to-date as your project unfolds. Logs created after the fact will affect your marks. Note that the log is meant to reflect the development phase, so initial brainstorming or project planning need not be documented here.
* A log entry should follow this detail:

|  |  |
| --- | --- |
| Development Log Entry 1 | |
| Date | 26/2/2025 |
| Week Number | Week 4 |
| Summary of Work Done | Began Planning and working on setting up file directory and first prototype |
| Challenges and Solutions | Describe specific problems encountered.  First issue was figuring out how to set up a chrome extension  Then I encountered a few issues with coding in mainly JavaScript and CSS.  Detail how the problems were addressed or solved.  To solve these issues i used ChatGPT to show me how to set up a chrome extension and then practice a bunch of JavaScript and CSS using w3schools a lot to learn a bunch of new syntax. |
| Milestones Achieved | Got the chrome extension working and I recently finished the title screen of the game along with the logo and some sprite designs. |

|  |  |
| --- | --- |
| Development Log Entry 2 | |
| Date | 06/3/2025 |
| Week Number | Week 6 |
| Summary of Work Done | Did a bunch of work on setting up a game screen and created some more sprites, I worked on a tiling system and have been trying to get placing and deleting structures working. |
| Challenges and Solutions | Describe specific problems encountered.  Alot! I have had a lot of issues with getting the game screen to work as i need a tile system and first tried making a grid then snapping structures to the grid but that would be way to complicated and a messy way. Finally, I encountered a TON of bugs with trying to add animation to the tiles when rotating, flipping, and placing.  Detail how the problems were addressed or solved.  To fix the game screen I learnt how to make a tiling system using tables and cells to create a grid and then edit their style with structures, next I had to find a way to send variables to my CSS file as I needed to update the transform functions dynamically when rotating or flipping tiles. This took me a few days to get working to a good level, but I'm still stuck with some glitching that i will leave to fix later as it's not really impacting the gameplay or my progress. |
| Milestones Achieved | Game screen works with a functioning tiling system.  structures can be placed, rotated, flipped, and deleted. |

NEXT WEEK!

|  |  |
| --- | --- |
| Development Log Entry 3 | |
| Date | 13/3/2025 |
| Week Number | Week 7 |
| Summary of Work Done | Did a bunch of work on setting up a game screen and created some more sprites, I worked on a tiling system and have been trying to get placing and deleting structures working. |
| Challenges and Solutions | Describe specific problems encountered.  Alot! I have had a lot of issues with getting the game screen to work as i need a tile system and first tried making a grid then snapping structures to the grid but that would be way to complicated and a messy way. Finally, I encountered a TON of bugs with trying to add animation to the tiles when rotating, flipping, and placing.  Detail how the problems were addressed or solved.  To fix the game screen i learnt how to make a tiling system using tables and cells to create a grid and then edit their style with structures, next I had to find a way to send variables to my CSS file as I needed to update the transform functions dynamically when rotating or flipping tiles. This took me a few days to get working to a good level, but I'm still stuck with some glitching that i will leave to fix later as it's not really impacting the gameplay or my progress. |
| Milestones Achieved | Game screen works with a functioning tiling system.  structures can be placed, rotated, flipped, and deleted. |

# Testing

Test Table

* Students should complete their test table with a robust suite of tests [minimum 10 tests] that address a range of test types, including boundary values, path coverage, and exception handling. The most effective testing will lead to bug fixes, which have been briefly documented within the test table.
* Here is an example section of test table for a software implementation of the game of Hangman.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test ID | Category | Test Case Description | Input to Provide | Expected Output | Actual Output | Pass/Fail | Action Taken |
| Test 1 | Path Coverage | Verify attempts increment on multiple failures | An incorrect letter six times | Attempts counter reaches 6 and game ends | “You Lost”  “Play Again (Y/N): ” | Pass | N/A |
| Test 2 | Boundary Value | Check behavior on last attempt | Correct letter after 5 incorrect guesses | Game indicates a win condition | “You Lost”  “Play Again (Y/N): ” | Pass | N/A |
| Test 3 | Path Coverage | Validate win condition with minimum guesses | Correct letters of the word in order | Game should indicate win before max attempts | “You Lost”  “Play Again (Y/N): ” | Pass | N/A |
| Test 4 | Faulty Data | Input non-alphabetic characters as guess | '1', '@', '-' | Game should prompt for correct input format | Game rejected invalid characters and prompted for letters | Fail | Updated get\_guess() function to validate user input more effectively using isalnum(), which returns False if a string does not contain alphanumeric characters. |
| Test 5 | Abnormal Data | Enter an already guessed letter | Correct letter guessed twice | Game notifies letter was already guessed | “You already guessed that letter!  Guess a letter:” | Pass |  |

# Project showcase

Repository

A well-structured repository is essential for project management and presentation. Your README.md file serves as a guide and introduction to your project and should include:

|  |  |
| --- | --- |
| Heading | Detail |
| Description | A fun game where you take orders for files and get them packed into packages and delivered by truck back to the clients. The game is intended for teaching students and anyone interested in software engineering development topics and tools such as DFD, server, packages, files, storage, and so on... |
| Installation Instructions | This is a chrome extension game intended to be installed of the chrome extension store under the name: “PACK!” by ryan hanna |
| How to Use | KeyBinds:  Place Structure: “1”, then, “M1/Click”  Delete Structures: “e”, then, “click/M1”  Rotate Tiles: “r”  Flip Tiles: “f/u”  Change Tile: “t”  Open Settings: “NA” |
| License Information | Under the google chrome extensions licence and created by Ryan Hanna the projects Assets and designs are all made by Ryan Hanna and should not be taken or misused. |
| Visuals |  |
| Acknowledgements | This Project has been made on VSCode and I have used ChatGPT, w3schools, stacks, and other websites to gain knowledge on coding syntax and how key functions such as a tiling system work. |
| Author Details | This project was made completely by me, Ryan Hanna including all pixel art sprites and code.  Contacts:  Online Name: Rewy or Rewy10  Email: [ry007h@gmail.com](mailto:ry007h@gmail.com)  Phone: 0492803235 |
| Clean Directory Structure | Organize your files in a logical manner, with folders for different types of files if necessary (e.g., separate folders for source code, assets, and documentation). |
| Additional Details | If your project has more complex setup procedures, dependencies, or configurations, document these thoroughly. |

**Review** [**What is README.md File? - GeeksforGeeks**](https://www.geeksforgeeks.org/what-is-readme-md-file/)

# Evaluation

Project Reflection

For students aspiring to attain the highest grade in their project reflection, it is crucial to deliver a sincere and analytical evaluation of their project. It is recommended that they include the following aspects:

* A brief re-introduction of the project, including its aim and scope. Mention the main objectives you intended to achieve with the project.
* Identify the parts of the project that were successful. This could include completed objectives, or any positive feedback received.
* Reflect on any difficulties you faced during the project. This could relate to technical issues, conceptual misunderstandings, or limitations in resources. Discuss how these challenges impacted the project and what was learned from addressing them.
* Outline specific areas where the project could be improved. This might include features that were not implemented, aspects of the project that could be refined, or knowledge gaps that were identified. Mention how these improvements could enhance the project’s outcomes in the future.
* Include a brief reflection, regarding the project management process. This reflection should highlight lessons learned in managing the project's timeline and tasks effectively.
* Conclude with a critical evaluation of the overall development journey. Reflect on how the experience has contributed to your learning and development as a software engineer. Summarize the key takeaways and how they will influence your future work.

Your project reflection should not exceed 500 words.