External Documentation

Created by

**Leanne J Dong**

For

**TAFE Queensland Creative Art Studio**

**PROJECT REFERENCE:**

**DATE:**

Contents

[1 Input process output (IPO) chart 2](#_Toc83252066)

[2 Task object event (TOE) chart 2](#_Toc83252067)

[3 Stylised data flow diagram (DFD) 3](#_Toc83252068)

[4 Selected pseudo code algorithms 3](#_Toc83252069)

[4.1 Algorithm name 3](#_Toc83252070)

[5 Structured charts 3](#_Toc83252071)

[5.1 Managing the GUI elements 3](#_Toc83252072)

[5.2 Managing the user actions 4](#_Toc83252073)

[6 Class diagram/s 4](#_Toc83252074)

[7. Debugging and testing 4](#_Toc83252075)

[7.1 Testing Plan 4](#_Toc83252076)

[7.2 Test cases 4](#_Toc83252077)

[7.3 Evidence of debugging 5](#_Toc83252078)

[8 System generated program documentation 5](#_Toc83252079)

[9 User Manual 5](#_Toc83252080)

[10 Evidence of Correspondence 5](#_Toc83252081)

# Identify Entities

Entities list:

Features:

* Title : string/varchar
* Image URL : string
* Web link : string
* Primary material: main materials required for implementing suggestion, string
* Construction hints: string

With above features, the re-purposing suggestion entity can be represented as a class in Java:

public class RepurposingSuggestion {

private String title;

private String imageLocation;

private String webLink;

private String primaryMaterials;

private String constructionProcess;

// Constructors, getters, and setters

}

# 1 Input process output (IPO) chart

Provide an introduction.

Describe how the business processes information. Write the descriptions in plain English – not coding required at this stage.

|  |  |  |  |
| --- | --- | --- | --- |
| INPUT | PROCESS | OUTPUT | STORAGE |
| Save button   * Idea Title * Image file * Web link * Primary material * Construction hints | Save all the entries in RAM Memory into the specified output file. | Export of all records (entries). | All entries saved in a new copy of the data file. |
| Delete button | Remove selected entry from the list | Update list of suggestions | Array of entries remove from memory |
| New button | Clear input | Input field | When the new button is clicked, it triggers the process of clearing input fields in the UI. However, this action does not result in any specific output, and no data is stored or updated in the storage. |
| Find button | Search by suggestions matching the specified Idea Title | Display of matching suggestions or appropriate message |  |
| Filter by Text Field | Read text keyword  Find all entries  Matching keyboard  Update Table | Update Table/list to show new entry |  |
| Find Text Field | Find all entries matching Idea Title | NA |  |
| Filter By Button | Filter suggestions based on user-provided criteria (title, material) | Display filtered list of suggestions | No data storage |
| Sort by Idea Title Button | Sort suggestions alphabetically by idea title | Display sorted list of suggestions | No data storage or updates |
| Binary Search By Idea Title Button | Perform binary search on the sorted list of suggestions by idea title | Display result of binary search (found or not found) | Storage: No data storage or updates |

# 2 Task object event (TOE) chart

Provide an introduction.

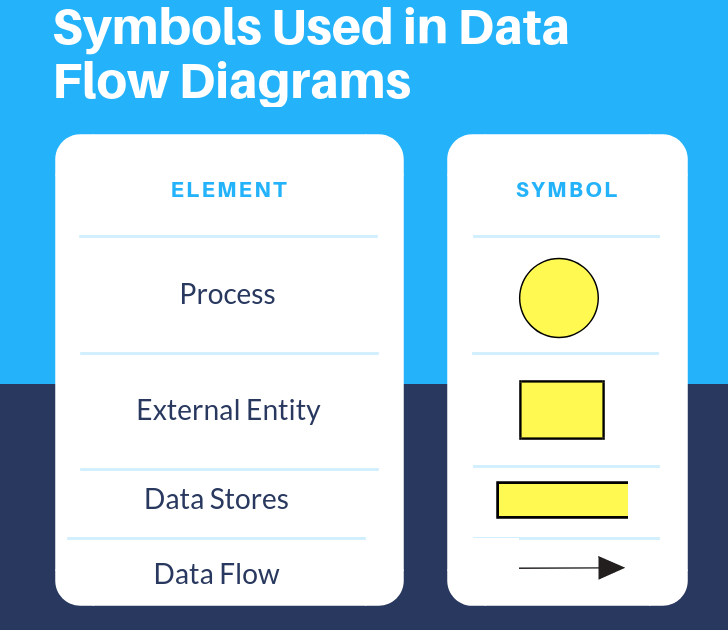
Identify the tasks. For each task identify the objects that the task requires and the event/s.

|  |  |  |
| --- | --- | --- |
| TASK | OBJECT | EVENT |
| Save all entries to a file | Save button | Click |
| Clear screen ready for new entry | New button | Click |
| Load Data On Startup | Window/Form | On Opening |
| Save New Entry | Save Button | Click |
| Delete Entry | Delete Button | Click |
| Save Edited Edit | Save Button | Click |
| Filter By | Filter By Button | Click |
| Sort by Idea Title | Sort by Idea Title Button | Click |
| Binary Search by Idea Title | Binary Search by Idea Title Button | Click |
| Find | Find Text Field | On text changed |

# 3 Stylised data flow diagram (DFD)

Prepare a simple data flow diagram (DFD) to map the flow of information for any process or system.

There are a number of symbols and notations used to create DFD select the one that is appropriate for the project and do not mix the notation of different DFD approaches.

Source: <https://www.intellspot.com/data-flow-diagram-examples>

1. Define the process

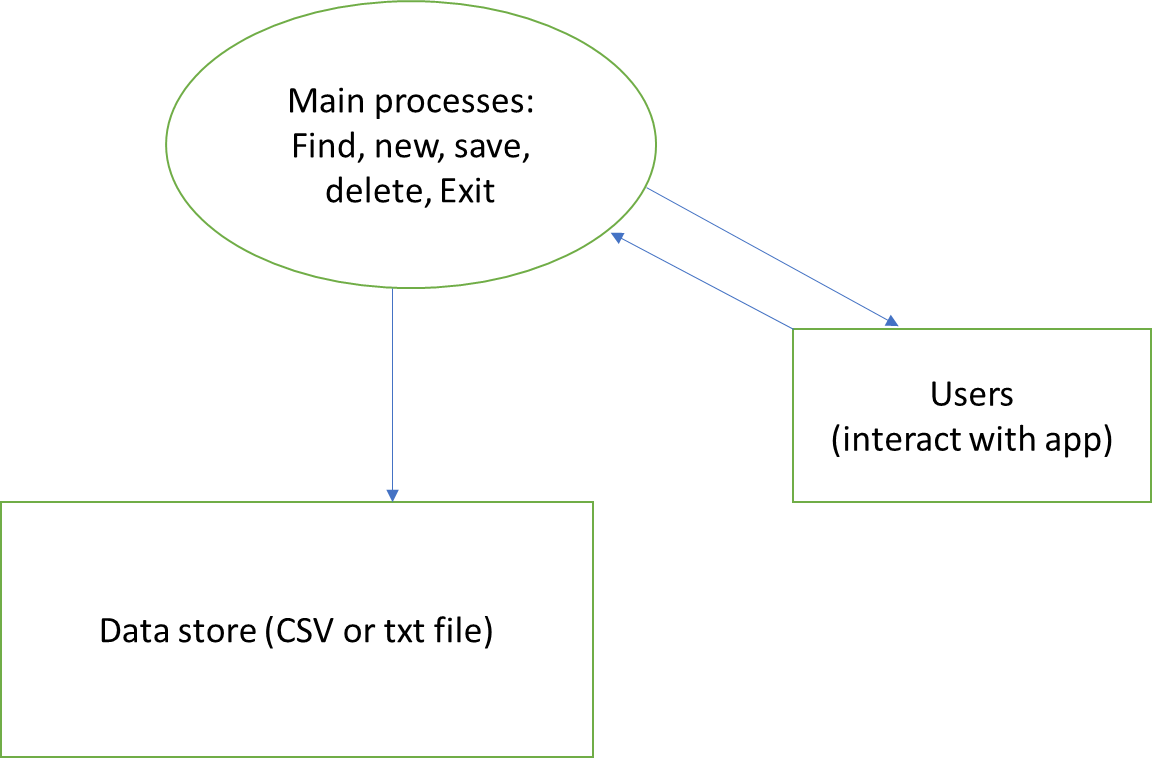
* Main process: Represents the core process of Re-purposing Suggestions Management, including finding, creating new, **loading, saving**, deleting and exiting suggestions.

1. Create a list of all external entities (people and systems)
   * User
   * Load recycle txt file
2. Create a list of a data stores

* Re-purposing suggestion data store (txt file)

1. Create a list of the data flows:

* Input Data Flow (from User to Main Process)
* Output Data Flow (from Main Process to User)
* File Data Flow (between Main Process and Local Recycler Data File)



# 4 Selected pseudo code algorithms

## 4.1 Algorithm name

Create sub-headings for each algorithm – 4.2, 4.3 etc.

**Sample algorithm**

FUNCTION WriteToFile(RePurposingSuggestionsData: Array of RepurposingSuggestion)

TRY

OPEN fileName for writing

FOR EACH suggestion IN RePurposingSuggestionsData

WRITE suggestion details to fileName

END FOR

CLOSE fileName

CATCH Exception AS ex

PRINT "Error: " + ex.message

END TRY

END FUNCTION

This code defines a function called WriteToFile under FileManager.java that takes an array of RepurposingSuggestion objects as input. The function attempts to open a file named fileName for writing. It then iterates over each suggestion in the input array and writes the details of each suggestion to the file. After writing all suggestions, the file is closed. If an exception occurs during the file writing process, the function catches the exception and prints an error message that

FUNCTION ReadFromFile() RETURNS FileData

DECLARE data AS FileData

DECLARE input AS BufferedReader

data.fileData = new Array[100] of RepurposingSuggestion

data.count = 0

TRY

OPEN fileName for reading

WHILE (line = READ next line from fileName) IS NOT NULL

PARSE line into suggestion details

CREATE new RepurposingSuggestion from parsed details

ADD new suggestion to data.fileData

INCREMENT data.count

END WHILE

CLOSE fileName

CATCH Exception AS ex

PRINT "Error: " + ex.message

SET data.fileData to NULL

SET data.count to 0

END TRY

RETURN data

END FUNCTION

This code defines a function called ReadFromFile that reads data from a file and returns it as FileData. It initializes a FileData object, opens a file for reading, reads each line from the file, parses the line into suggestion details, creates a new RepurposingSuggestion object from the parsed details, adds the new suggestion to the fileData array, and increments the count of suggestions. If an exception occurs during the process, it catches the exception, prints an error message, sets the fileData to NULL, and sets the count to 0. Finally, it returns the data object containing the fileData and count.

**FUNCTION handleNewEntry()**

**IF numberOfEntries EQUALS 100 THEN**

**DISPLAY "The Re-Purposing Suggestion List is Full. Cannot add any more Entries"**

**RETURN**

**END IF**

**CLEAR txtTitle, txtImageLocation, txtWebLink, txtPrimaryMaterials, txtConstructionHints**

**SET isNewEntry to TRUE**

**END FUNCTION**

This code defines a function called handleNewEntry.

* It first checks if the variable numberOfEntries is equal to 100. If it is, it displays a message saying that the Re-Purposing Suggestion List is full and cannot add any more entries, and then exits the function.
* If the numberOfEntries is less than 100, it clears the text fields txtTitle, txtImageLocation, txtWebLink, txtPrimaryMaterials, and txtConstructionHints.
* Finally, it sets the variable isNewEntry to TRUE.

The delete button (logic and algorithm)

In the main class, we will find the declaration of the delete button

`JButton btnDelete;`

Then in the constructor of the MainForm class, we initialized the delete button. It typically include setting text, size position and adding an ActionListner to handle button clicks.

```

btnDelete = UIBuilderLibrary.BuildJButtonInlineBelow(120, 25, "DELETE", 5, this, layout, btnSave);

add(btnDelete);

```

Under the actionPerformed method, there are various actions for various buttons. Delete is one of them. Here is the logic, which typically involving an entry from the data structure (such as an array or list) and updating the UI accordingly.

if (e.getSource() == btnDelete) {

int result = JOptionPane.showConfirmDialog(this,

"Are you sure you want to delete this entry?","Delete Confirmation", JOptionPane.YES\_NO\_OPTION, JOptionPane.ERROR\_MESSAGE);

if (result != JOptionPane.YES\_OPTION) {

return;

}

for(int i = currentEntry; i < numberOfEntries; i++) {

if(i == numberOfEntries - 1){

RepurposingSuggestionArray[i] = null;

break;

}

RepurposingSuggestionArray[i] = RepurposingSuggestionArray[i + 1];

}

numberOfEntries--;

if(currentEntry == numberOfEntries){

currentEntry--;

}

DisplayCurrentEntry();

fileManager.WriteToFile(RepurposingSuggestionArray);

}

The delete button logic involve interacting with `FileManager` class to update the data stored in the file system.

The delete algorithm does

* + - 1. When the delete is clicked, the app should prompt the user to confirm the deletion of the selection entry
      2. If the user confirms the deletion, the application should remove the selected entry from the list of suggestions.
      3. The application should update the UI to reflect the removal of the entry
      4. If necessary, our app should also update the data stored in the file system to persist the changes.

**Pseudocode**

// When the delete button is clicked

onButtonClick(deleteButton):

// Prompt the user to confirm deletion

confirmation = showConfirmationDialog("Are you sure you want to delete this entry?")

// If the user confirms deletion

if confirmation is true:

// Remove the selected entry from the list

removeSelectedEntryFromList()

// Update the UI to reflect the removal

updateUI()

// Optionally, update data in the file system

updateDataInFileSystem()

// Function to remove the selected entry from the list

removeSelectedEntryFromList():

// Determine the index of the selected entry

selectedIndex = getSelectedIndex()

// Remove the entry at the selected index from the list

removeEntryAtIndex(selectedIndex)

// Function to update the UI after deletion

updateUI():

// Refresh the display to reflect the changes in the list

refreshDisplay()

// Function to update data in the file system

updateDataInFileSystem():

// Open the data file for writing

openDataFileForWriting()

// Write the updated list of suggestions to the file

writeListToFile()

// Close the data file

closeDataFile()

# 5 Structured charts

Structured chart display the hierarchical structure of modules and their connections to other modules. Structured charts represent the software architecture of the system to a manageable low level.

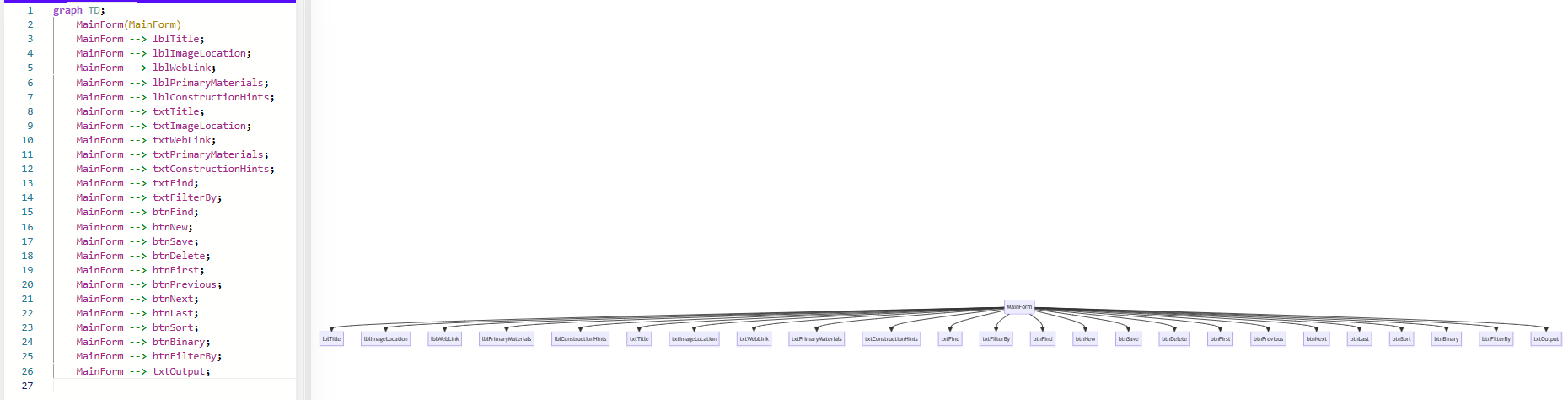
Make sure that you clearly identify the modules, sub-modules, libraries, data flow, control flow and loops or repetitive calls.

## 5.1 Managing the GUI elements

Create a structured chart for the GUI elements

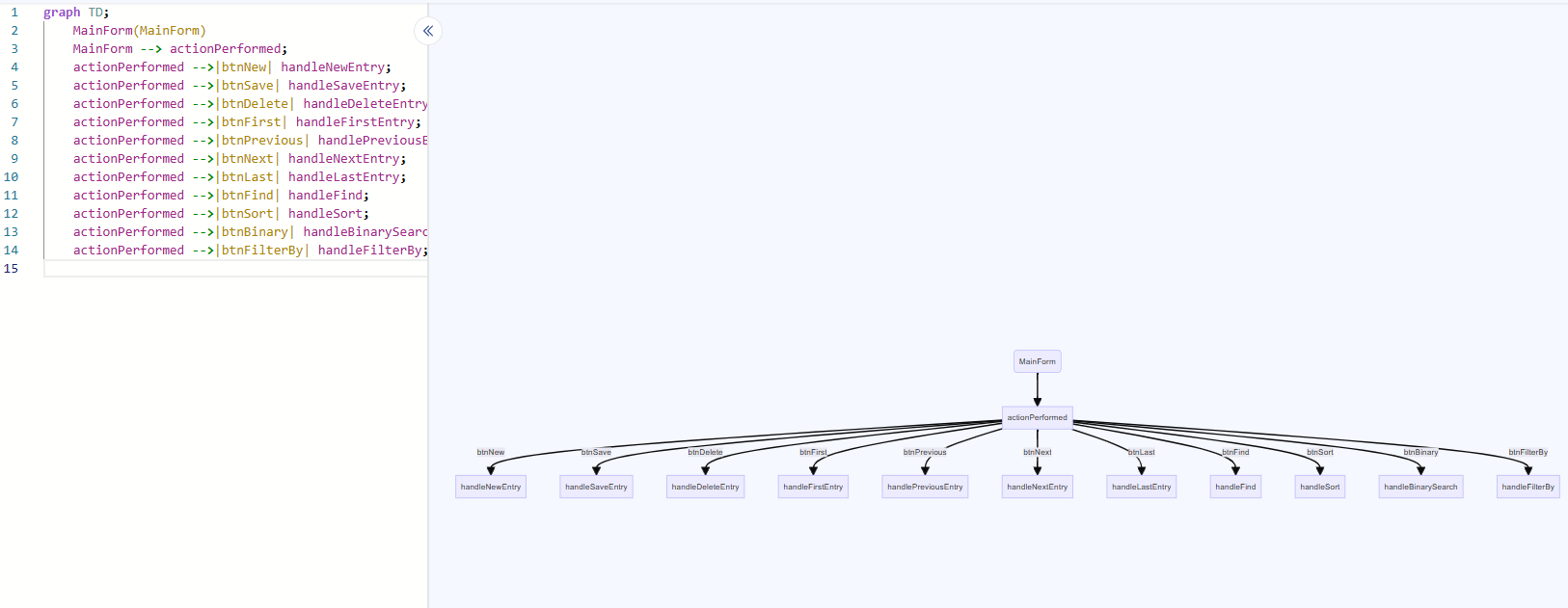
[GitHub - leannejdong/RepurposeR](https://github.com/leannejdong/RepurposeR?tab=readme-ov-file#managing-gui-elements)

* **Identify main GUI components**: Determine the main components of GUI, such as windows, panels, buttons, text fields, and text areas.
* **Identify user actions**: Identify the actions that users can perform in GUI, such as clicking buttons, entering text, selecting items from lists, etc.
* **Map out the hierarchy**: Start with the main GUI component (e.g., MainForm) and its immediate child components. Then, expand the chart to include the child components of each component.
* **Connect user actions**: Connect the user actions to the GUI components that trigger them. For example, if a user action is clicking a button, draw an arrow from the button component to the action it triggers.
* **Add details**: Add any additional details or annotations that clarify the flow of control or interaction between components.
* **Refine and organize**: Review the structure chart and make any necessary refinements to ensure clarity and organization. Group related components together and arrange them in a logical manner.
* **Finalize the chart**: Once you're satisfied with the structure and organization of the chart, finalize it by adding any necessary formatting or styling.



## 5.2 Managing the user actions

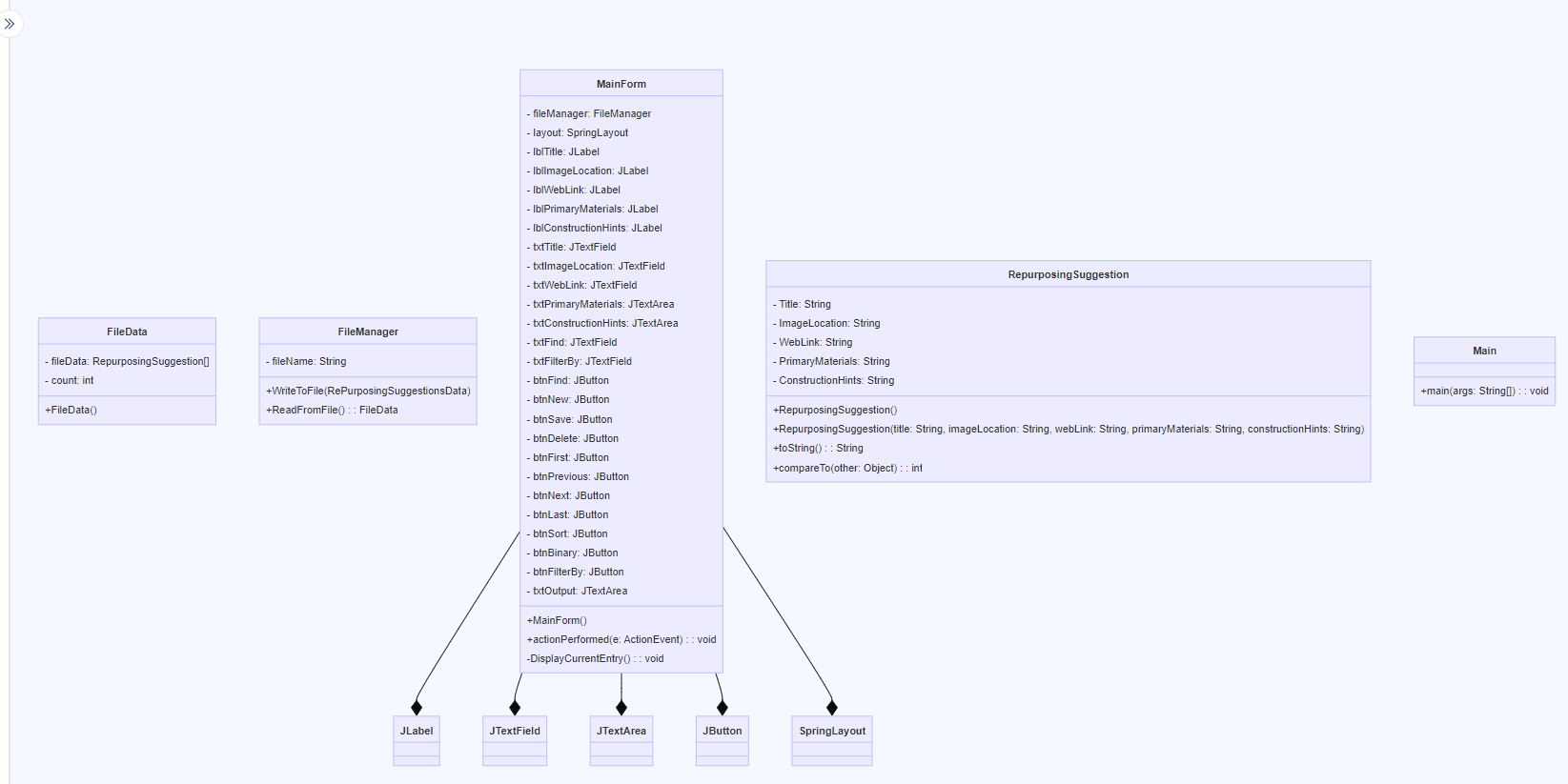
Create a structured chart for the user actions



# 6 Class diagram/s

Class diagrams are the building block of object-oriented modelling.

Use a UML diagramming tool such as Visual Paradigm or similar to create the diagrams.



# 7. Debugging and testing

## 7.1 Testing Plan

The approach to testing that will be used is as follows:

1. Review the project requirements
2. Prepare a solution design
3. Develop the required application.
4. **Unit testing** – test each component (method / class) as it is being developed
5. **Integrated Testing** - Prepare a set of data (test cases) to check how the application will behave when appropriate data is read in and entered via the screen.
6. Prepare several sets of data (test cases) to check how the application will behave when data that will generate exceptions is read in from the data file.
7. Enter several sets of data (test cases) to check how the application will behave when data that will generate exceptions is entered via the screen.
8. **System Testing** – Deploy the application to a computer system/network/web host site similar to that of the client’s and retest selected test cases.
9. **Acceptance Testing** - Deploy the application to the client’s computer system/network/web host site and look to the client to retest using sample live data.
10. Archive the test plan, test cases and results to an appropriate backup location.

Complete the section below.

**Test Objectives:**

* Ensure that the application meets all specified requirements outlined in the project requirements document.
* Validate the functionality and usability of the application under various scenarios and input conditions.
* Identify and address any defects or inconsistencies in the application's behaviours.
* Verify that the application performs reliably and efficiently in different environments.

**Test Strategy**: Unit testing, integrated testing, exception handling, system testing, acceptance testing and documentation and archiving

**Resource Requirements:**

* Personal: team of testers
* Computers: Cross platform development on any OS
* Software: JUnit5 in IntelliJ

**Roles and Responsibilities:**

In a large organisation, there are

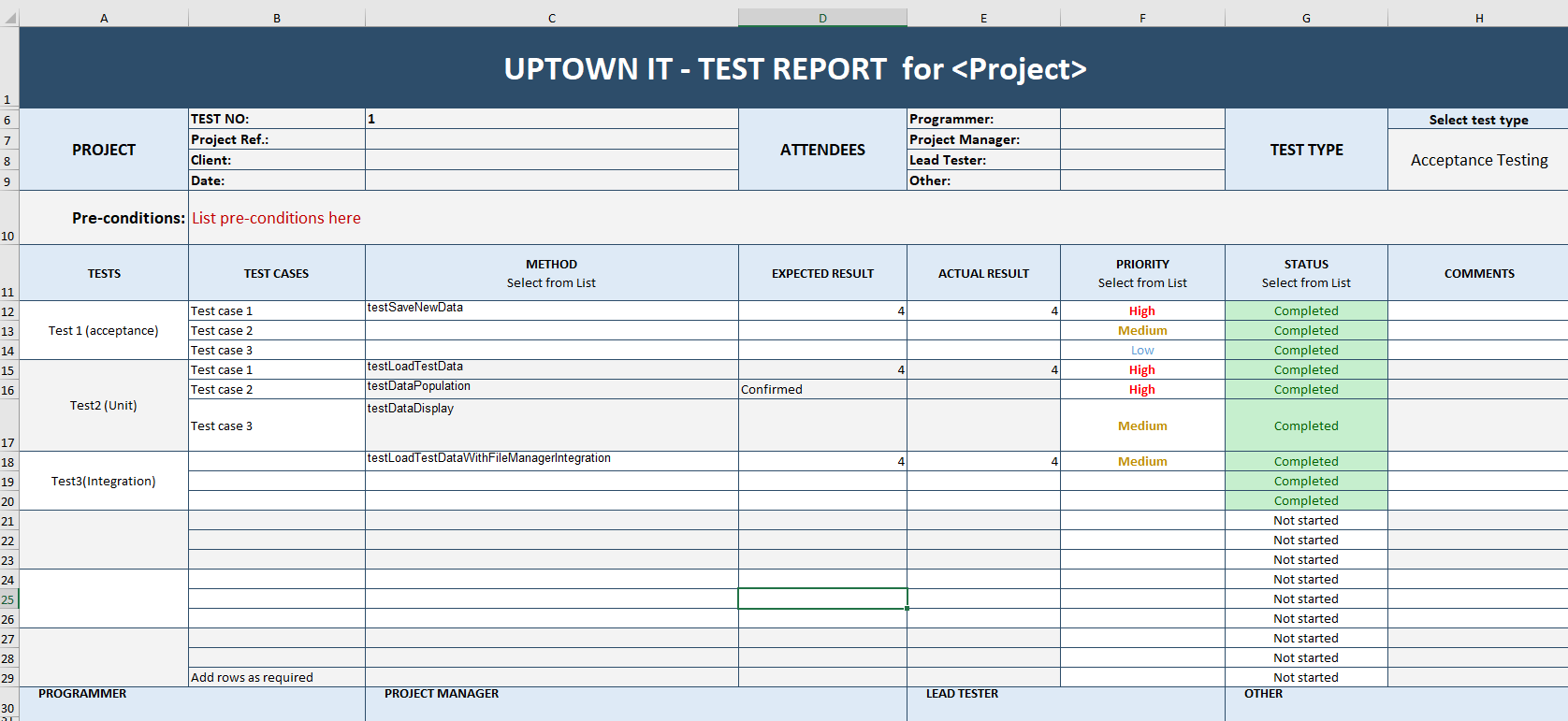
* Test Lead: Responsible for overseeing the testing process, creating the test plan, and assigning tasks to testing team members.
* Testing Team Members: Responsible for executing test cases, documenting test results, and reporting any issues found during testing.
* Development Team: Collaborates with the testing team to address and resolve reported bugs and issues.

In a small place, it is just myself for all above.

**Bug Reporting:**

* Bugs and issues found during testing will be documented using bug tracking software.
* Each bug report will include details such as the steps to reproduce the issue, expected behavior, actual behavior, severity level, and any relevant screenshots or logs.
* Bug reports will be regularly reviewed by the development team, and fixes will be prioritized and implemented accordingly.
* Regular status updates and bug resolution progress will be communicated to the project stakeholders, including the project manager and development team.

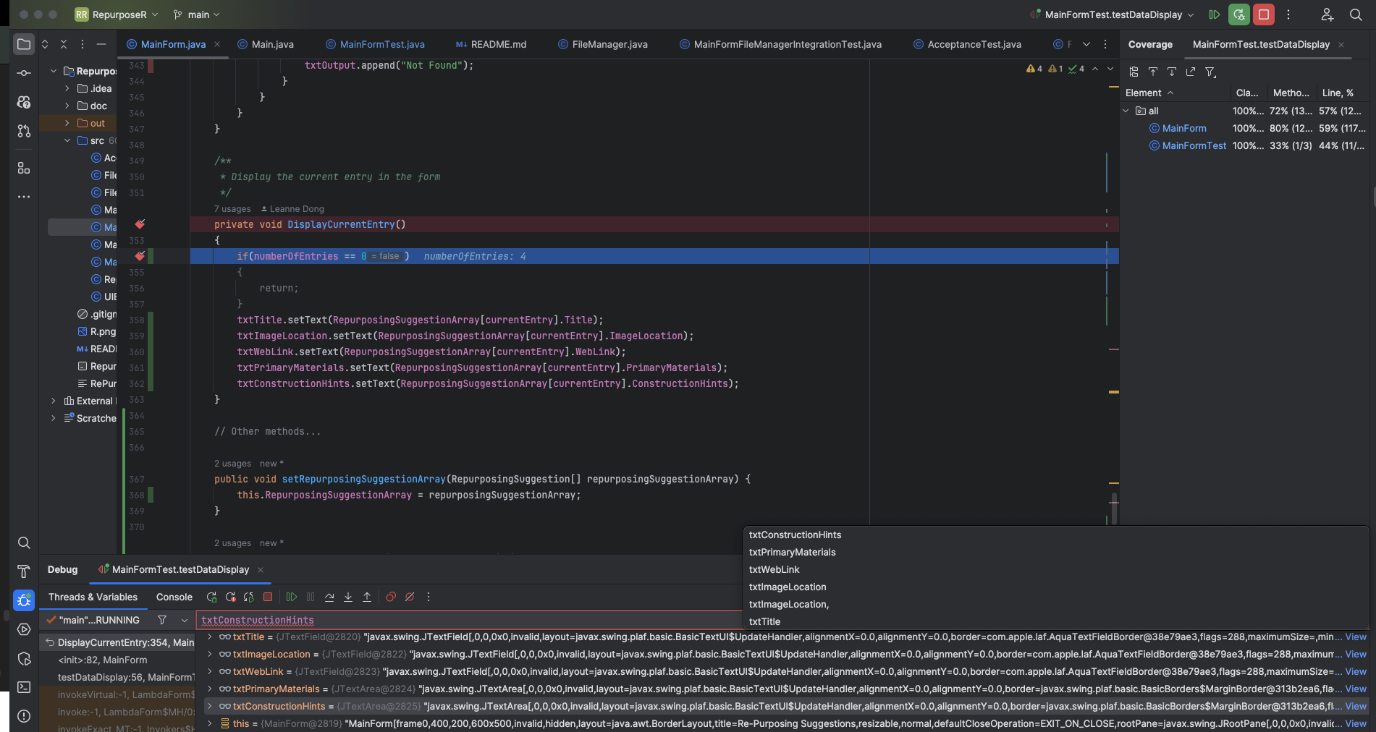
## 7.2 Test cases

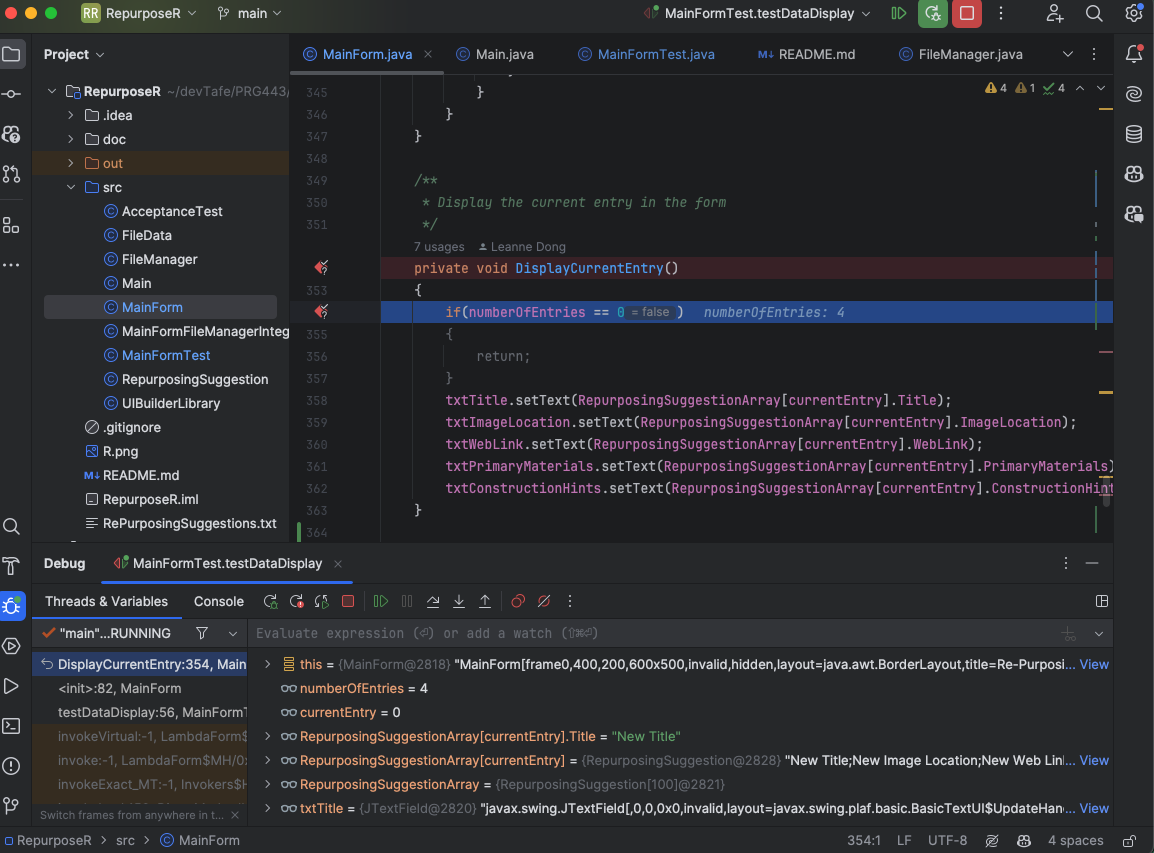


## 7.3 Evidence of debugging

Provide screenshots of different instances of debugging. Make sure that you include examples of:

1. a breakpoint
2. a set of associated watches
3. an instance of tracing through several lines of code
4. I placed a breakpoint at the `DisplayCurrentEntry()` method to verify whether the text fields are being populated correctly with the data from the `RepurposingSuggestionArray` to verify the text fields are being populated correctly with the data from the `RepurposingSuggestionArray`.
5. Click the debug button
6. Inspect variables.
7. Stepping through the code
8. Watch variables. I have setted up watches to monitor the values of specific variables such as txtTitle, txtImageLocation, txtWebLink, txtPrimaryMaterials, and txtConstructionHints to see how their values change during execution.





# 8 System generated program documentation

Prepare automated program documentation using a facility provided within your IDE, such as Javadoc, or using a third-party product such as ESS Model.

Ensure this automated documentation is provided within your submission.

# 9 User Manual

# Re-purposing Suggestions Application User Manual

## Table of Contents

1. [Introduction](bookmark://_1._Introduction)
2. [Getting Started](bookmark://_2._Getting_Started)
3. [Main Interface Overview](bookmark://_3._Main_Interface)
4. [Managing Suggestions](bookmark://_4._Managing_Suggestions)
5. [Searching and Filtering](bookmark://_5._Searching_and)
6. [Sorting and Binary Search](bookmark://_6._Sorting_and)
7. [Exiting the Application](bookmark://_7._Exiting_the)

## 1. Introduction

Welcome to the Re-purposing Suggestions application! This application allows you to manage, search, and filter various re-purposing ideas for different materials.

## 2. Getting Started

To start the application:

* Open the application executable or run the **Main** class in your IDE.

## 3. Main Interface Overview

The main interface of the application consists of the following components:

* Input fields for adding or editing suggestions.
* Navigation buttons for browsing through suggestions.
* Action buttons for saving, deleting, and searching suggestions.
* Output area for displaying suggestions and search results.

## 4. Managing Suggestions

### Adding a New Suggestion

* Click the "New" button to create a new suggestion entry.
* Enter the details for the new suggestion in the provided input fields.
* Click the "Save" button to save the new suggestion.

### Editing an Existing Suggestion

* Use the navigation buttons to find the suggestion you want to edit.
* Modify the details of the suggestion in the input fields.
* Click the "Save" button to save the changes.

### Deleting a Suggestion

* Use the navigation buttons to find the suggestion you want to delete.
* Click the "Delete" button to remove the selected suggestion.

## 5. Searching and Filtering

### Searching by Title

* Enter the title of the suggestion you want to find in the "Find" input field.
* Click the "FIND" button to search for suggestions with the specified title.

### Filtering by Construction Hints

* Enter a keyword or phrase in the "Filter by" input field.
* Click the "Filter by" button to display suggestions containing the specified keyword or phrase in their construction hints.

## 6. Sorting and Binary Search

### Sorting by Idea Title

* Click the "Sort by Idea Title" button to arrange suggestions alphabetically by their titles.

### Binary Search by Idea Title

* Enter the title of the suggestion you want to find in the "Find" input field.
* Click the "Binary Search by Idea Title" button to quickly locate the suggestion using binary search.

## 7. Exiting the Application

* To exit the application, simply close the application window or click the "Exit" button.

# 10 Evidence of Correspondence

Maintain appropriate ongoing communication with your manager.

This can be done with a series of applicable emails.

1. Email your manager before the commencement of this project to secure the project specification and the organisation’s programming and documentation standards.

Dear Troy,

I hope this mail find you well. I am writing to confirm the requirement for the Repurposing Suggestion Software project as outlined in our recent discussion. I want to ensure I understand the project scope before proceeding with development.

Based on the information provided, it seems that my client, uptown IT Solutions Department, is a department within a larger organization that specializes in software development and solutions. They may offer custom software development services or have in-house teams dedicated to creating software applications to address specific needs within their organization or for their clients.

The Repurposing Suggestions Software project is aimed at developing a Java application that assists in managing and utilizing waste items more efficiently. The application will serve as a database for storing and retrieving repurposing suggestions for various waste materials.

Here's a breakdown of the key components and functionalities of the application I am meant to deliver:

1. Database for Repurposing Suggestions:

* The application will provide a platform for users to input, store, and manage repurposing suggestions for waste items.
* Each suggestion will include details such as the idea title, image file location, web link (if applicable), primary materials required, and construction process or hints.

1. User Interface:

* The application will feature a user-friendly interface with navigation buttons (First, Last, Next, Previous), search functionality (Find), and data management options (New, Save, Delete).
* Users will be able to view a list of repurposing suggestions and navigate through them efficiently.

1. Data Management:

* Repurposing suggestion details will be saved and loaded from a comma-delimited text file named "RePurposingSuggestions.txt".
* The application will handle data storage and retrieval seamlessly, ensuring that users can access the information they need effectively.

1. Search and Filtering:

* Users will be able to search for existing repurposing suggestions by idea title and view suggestions categorized by primary material.
* Additionally, the application will implement a binary search functionality to locate records by idea title within a sorted list, enhancing search efficiency.

Overall, the Repurposing Suggestions Software project aims to streamline the process of repurposing waste items by providing staff with a centralized platform to access and utilize repurposing ideas effectively. By delivering this application, my task is to enable UpTown IT Software Solutions Department to optimize resource utilization and contribute to sustainable practices within the organization.

Below, I have summarized the requirements as I understood.

1. Project Overview

* Develop a software solution to manage repurposing suggestions for waste items within UpTown IT Software Solution Department.

1. Functionality Requirements:

* The application should allow users to
  + Enter new repurposing suggestion details, including
    - Idea Title
    - Image File Location
    - Web link
    - A memo field for listing the primary materials required
    - A memo field to hold the construction process or hints
* Navigate through records with First, Last, Next and Previous record buttons
* Search for existing records by Idea Title using a Find button
* Utilize New, Save and Delete buttons for data management.
* View a list of repurposing suggestions by Primary material
* Implement a binary search functionality to find existing records by Idea Title within a sorted list.

1. Data Management

* The application should save and load repurposing suggestion details to and from CSV file names “RePurposingSuggestions.txt”

1. Additional Details:

* Each repurposing suggestion includes specific details such as the materials required and construction process, aiding users in efficiently repurposing waste items.
* The application aims to provide staff with a convenient tool to locate potential repurposing suggestions for selected waste items.

Please review the outlined requirements and let us know if there are any additional specifications or corrections needed. In case of any specific preferences regarding the UI or additional features, feel free to share your thoughts.

Once I receive your confirmation on these requirements, I will proceed with the development phase and keep you updated on the progress regularly.

Thank you for the guidance and support throughout this project. I look forward to deliver a high quality software solution that meets our objectives

Best regards,

Dr Leanne J Dong

[leanne@spacetimenoise.dev](mailto:leanne@spacetimenoise.dev)

1. Prepare a closing email to your manager regarding the success of your project development endeavours. You should suggest any areas of concern or risk to your manager for inclusion in the development team’s risk matrix.

Dear Troy,

I hope this email finds you well.

I am pleased to inform you that the development of the Re-purposing Suggestions application has been successfully completed. Through meticulous design, rigorous development, and comprehensive testing, we have delivered a robust and user-friendly solution that meets the requirements outlined by our stakeholders.

**Key Achievements:**

* Successfully implemented all specified functionalities, including data loading, saving, editing, and searching.
* Designed an intuitive user interface using Java Swing, providing a seamless user experience.
* Implemented various testing strategies, including unit tests, integrated tests, system tests, and acceptance testing, ensuring the reliability and functionality of the application.
* Documented the codebase comprehensively and provided thorough user documentation for easy adoption and maintenance.

**Areas of Concern and Risk Mitigation**: While the development process went smoothly, there are a few areas of concern and potential risks that I would like to highlight for inclusion in the development team's risk matrix:

* Data Integrity: Although we have implemented data validation mechanisms, there is a risk of data integrity issues if users input invalid or incorrect data.
* Performance Optimization: As the application scales with a larger dataset, there may be performance bottlenecks that could impact user experience. To address this, we should conduct performance profiling and optimization periodically to ensure optimal performance.
* Platform Compatibility: While the application has been developed and tested primarily on the IntelliJ platform, there may be compatibility issues with other development environments. We should conduct cross-platform testing to ensure compatibility across different IDEs.

**User Feedback Incorporation**: As users begin to interact with the application, we may receive feedback regarding additional features or improvements. It's essential to have a process in place for gathering and incorporating user feedback into future iterations of the application.

Overall, I am confident that the Re-purposing Suggestions application will add value to our users and contribute to our organization's goals. I would like to express my gratitude to the development team for their hard work and dedication throughout this project.

Please feel free to reach out if you have any questions or require further information.

Thank you for your support and guidance throughout the development process.

Best regards,

Leanne

# 11 Data structure and algorithms

In this report we have only used Array. Nevertheless, some general explanations and example for common data structures are below

<https://leanne.godbolt.org/z/xM9WGh6vo>