**Text Processing Tool Requirements Document**

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to define the functional and non-functional requirements for the development of the Text Processing Tool. This tool aims to provide users with advanced capabilities for text manipulation using regular expressions (regex) and efficient data management using Java collections.

**1.2 Scope**

The Text Processing Tool will include modules for regex operations (search, match, replace) and data management (CRUD operations on ArrayList, Set, and Map). It will feature a JavaFX-based user interface for seamless interaction.

**2. Functional Requirements**

**2.1 Regular Expressions Module**

**Search Operation**

Allow users to search for occurrences of regex patterns in input text.

Display all matches found, including their positions in the text.

**Match Operation**

Verify if the entire input text matches a specified regex pattern.

Provide clear feedback on whether a match is found or not.

**Replace Operation**

Enable users to replace all occurrences of a regex pattern with a specified replacement text.

Display the modified text after replacement.

**2.2 Data Management Module**

**Collection Types**

Support three types of Java collections: ArrayList, Set, and Map.

Allow users to select and manage entries within these collections based on their requirements.

**CRUD Operations**

Create: Add new entries to the selected collection type.

Read: Display all entries currently stored in the collection.

Update: Modify existing entries, ensuring consistency and validation checks.

Delete: Remove entries from the collection based on user selection.

**2.3 User Interface (JavaFX)**

**Input Components**

Text areas and input fields for entering text data, regex patterns, and collection entries.

**Output Components**

Display areas to show results of regex operations (matches, replacements) and collection content.

Enable seamless transition between inputting data, defining regex operations, and managing collections.

Implement event handling to respond to user actions promptly (e.g., button clicks, text input changes).

**3. Non-Functional Requirements**

**3.1 Performance**

Responsiveness: The tool should respond to user actions within acceptable time limits, even with large datasets.

Efficiency: Regex operations and collection manipulations should be optimized for performance.

**3.2 Usability**

Intuitiveness: The user interface should be easy to navigate and understand for users with basic technical knowledge.

Accessibility: Ensure accessibility features for users with disabilities where feasible.

**3.3 Reliability**

Error Handling: Implement robust error handling mechanisms to prevent crashes and provide informative error messages to users.

Data Integrity: Ensure that data in collections remains consistent and valid throughout all operations.

**4. Use Case Diagram**

Use Case Descriptions:

Actors:User

Use Cases:

Manage Collections:

**Add Entry:** Allows the user to add entries to a selected collection (List, Set, Map).

**Edit Entry:** Enables the user to modify existing entries in the collections.

**Delete Entry:** Allows the user to remove entries from the collections.

**Search Entry:** Facilitates searching for specific entries within the collections.

**Find Item:** Helps locate items within the collections based on specified criteria.

**Clear Collection:** Clears all entries from the selected collection.

**Text Processing:**

**Search Text:** Enables the user to search for occurrences of a pattern in the input text.

**Match Text:** Checks if the entire input text matches a specified pattern.

**Replace Text:** Allows the user to replace occurrences of a pattern in the input text.