DLithe™
Be Agile..Be in Demand

SKILL DEVELOPMENT

Project Report On

String Reverser

DLithe Consultancy Services Pvt. Ltd.

DLithe™
Be Agile..Be in Demand

Project Report Assessment

Student Name: Seema Udapudi

Reg. no: 2JR23CS087

Assignment: Java

Organization: DLithe Consultancy Services Pvt. Ltd.

Supervisor's Name: Archana SM

Submitted to

Signature of Training Supervisor

Date: 19-05-2025

Signature of Students

Date:19-05-2025

DLithe™
Be Agile..Be in Demand

Table of Contents

INTRODUCTION

A **String Reverser** is a basic yet important utility in computer programming that reverses the order of characters in a given string. For example, the string "OpenAI" would be reversed to "IAnepO". This simple task helps programmers understand how to work with strings, manipulate character arrays, and use control structures such as loops or recursion. Despite being a common beginner-level exercise, it provides valuable insight into how strings are stored and processed in memory.

The logic behind reversing a string involves traversing it from the end to the beginning and building a new string by appending each character in reverse order. There are various ways to implement this, including using for or while loops, built-in string functions, or recursive methods. Each approach can demonstrate different programming concepts, such as iteration, stack memory in recursion, or the efficiency of using immutable versus mutable data structures.

Beyond its educational use, string reversal has practical applications. It is often used in algorithms involving **palindrome checking**, **data encoding/decoding**, and **text processing**. In more complex systems, reversing strings or sequences can play a part in undo functionalities, cryptographic transformations, and specialized search algorithms. Therefore, understanding how to reverse a string is not only essential for learning programming but also useful in solving real-world problems efficiently

DLithe™
Be Agile..Be in Demand

USE CASE

Functional Requirements:

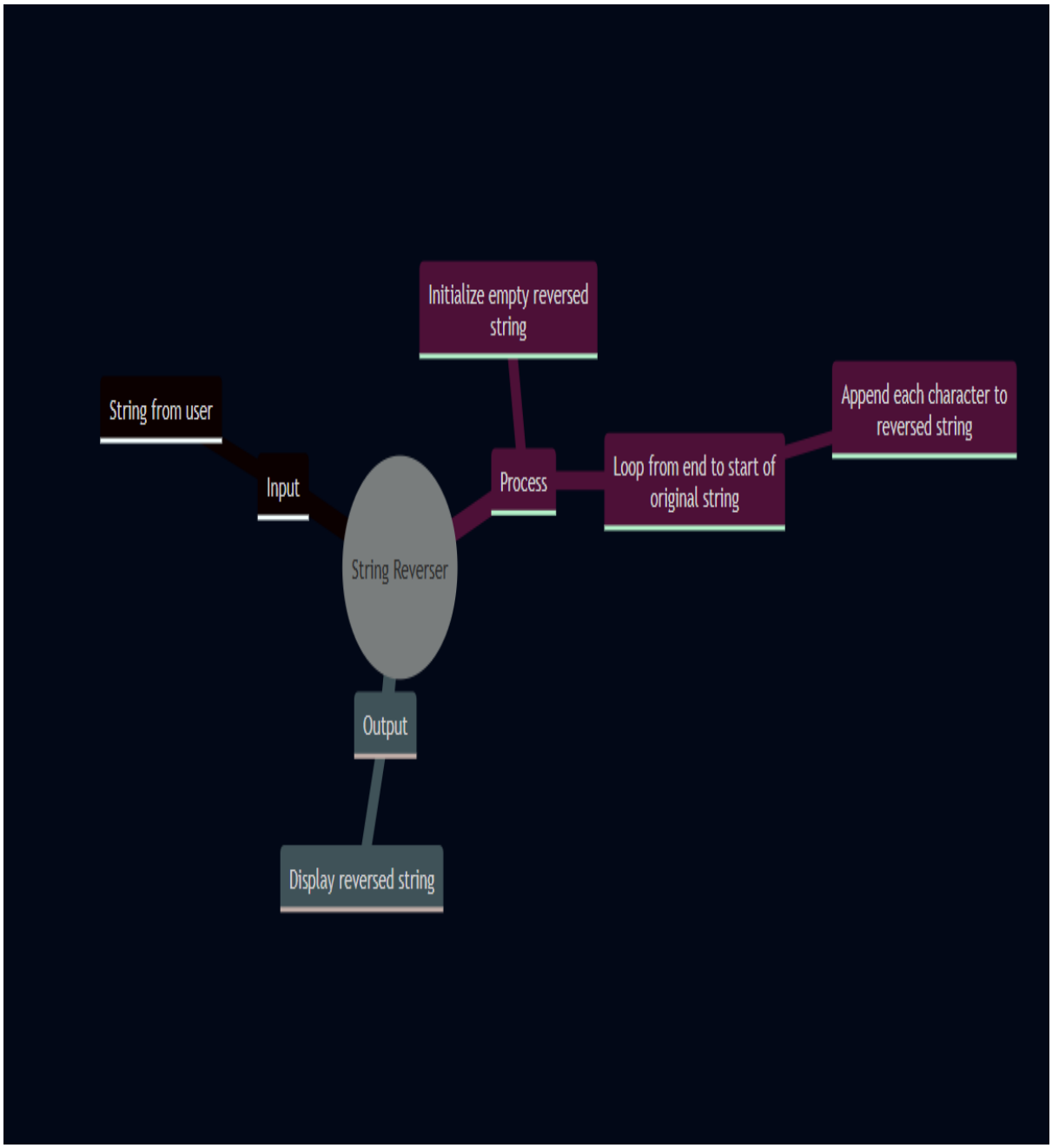
1. **FR1 – Input Handling**:The system shall accept a string input from the user (via console, GUI, or API, depending on the interface).
2. **FR2 – Reverse Logic**:The system shall reverse the characters in the input string using a loop, built-in method, or recursion.
3. **FR3 – Output Display**:The system shall display or return the reversed string as output.
4. **FR4 – Edge Case Handling**:The system shall correctly handle edge cases such as:
    - Empty string input
    - Single-character strings
    - Strings with special characters, spaces, or numbers

Non-Functional Requirements

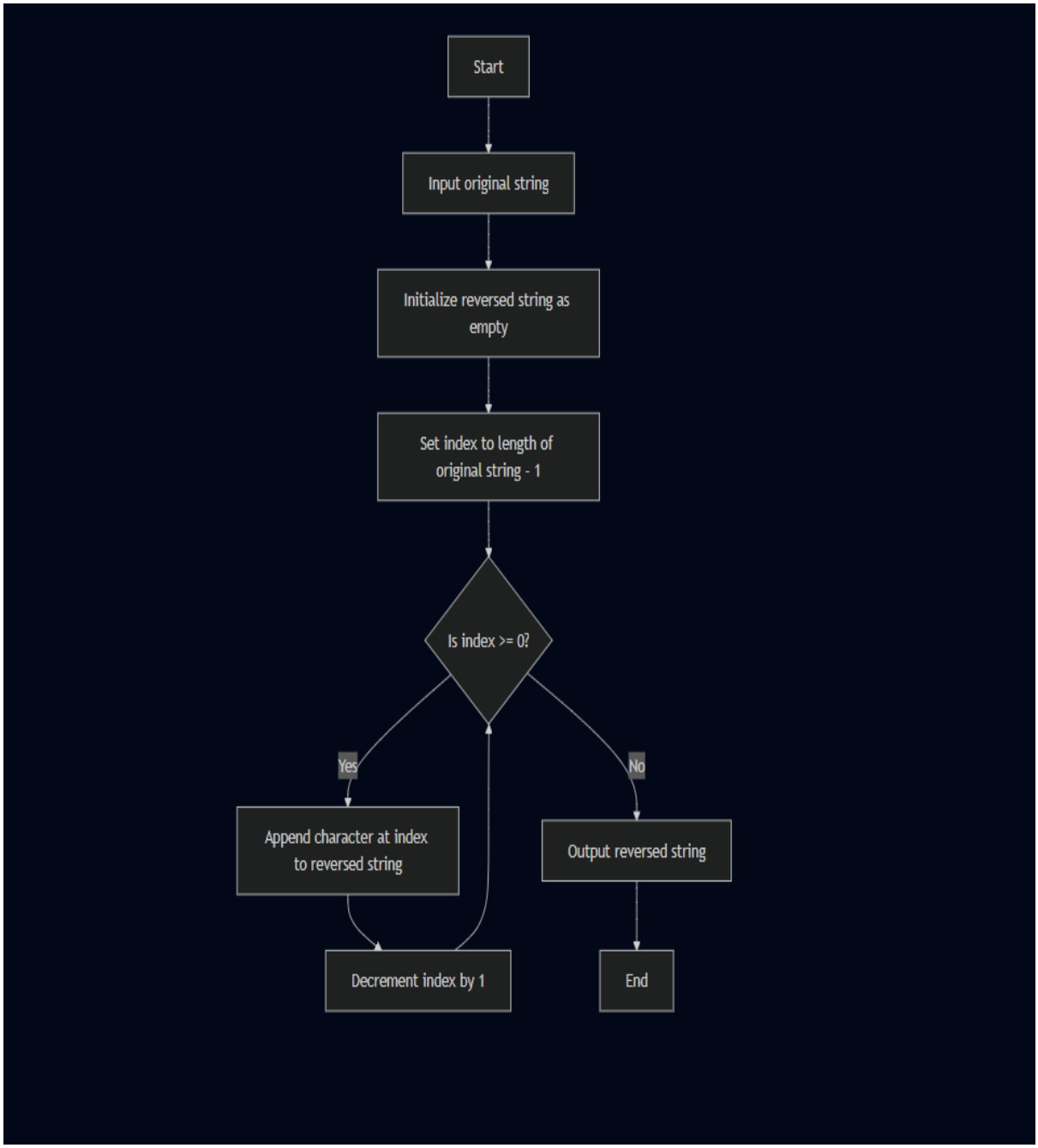
1. **NFR1 – Performance**:The system shall reverse strings of up to 10,000 characters with minimal latency.
2. **NFR2 – Usability**:The interface (if applicable) shall be intuitive, allowing the user to enter and receive strings easily.
3. **NFR3 – Reliability:**The system shall provide correct and consistent output across multiple runs.
4. **NFR4 – Language Support (Optional)**:The system may support Unicode and multi-language input (e.g., Arabic, Chinese) depending on implementation.

DLithe™
Be Agile..Be in Demand

MIND MAP



FLOWCHART


Start
Input original string
Initialize reversed string as empty
Set index to length of original string - 1
Is index >= 0?
Yes
No
Append character at index to reversed string
Output reversed string
Decrement index by 1
End

DLithe™
Be Agile..Be in Demand

PROJECT IMPLEMENTATION IMAGES

```java
import java.util.Scanner;

public class StringReverser {

    // Method to reverse a given string
    public static String reverseString(String input) {
        StringBuilder reversed = new StringBuilder();

        // Loop through the string from the end to the beginning
        for (int i = input.length() - 1; i >= 0; i--) {
            reversed.append(input.charAt(i));
        }

        return reversed.toString();
    }

    Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Prompt user for input
        System.out.print(s:"Enter a string to reverse: ");
        String original = scanner.nextLine();

        // Reverse the string using the method
        String reversed = reverseString(original);

        // Display the reversed string
        System.out.println("Reversed string: " + reversed);

        scanner.close();
    }
}

```

OUTPUT

```
Reversed string: olleh
PS D:\Desktop\StringReverser.java> ^C
PS D:\Desktop\StringReverser.java>
PS D:\Desktop\StringReverser.java> d:; cd 'd:\Desktop\StringReverser.java'; & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.7.6-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Asus\AppData\Roaming\Code\User\workspaceStorage\4dc8753ab0f99ae4c215eb
eb9eb52634\redhat.java\jdt_ws\StringReverser.java_da8dda23\bin' 'StringReverser'
Enter a string to reverse: seema
Reversed string: amees
PS D:\Desktop\StringReverser.java> ^C
PS D:\Desktop\StringReverser.java>
PS D:\Desktop\StringReverser.java> d:; cd 'd:\Desktop\StringReverser.java'; & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.7.6-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Asus\AppData\Roaming\Code\User\workspaceStorage\4dc8753ab0f99ae4c215eb
eb9eb52634\redhat.java\jdt_ws\StringReverser.java_da8dda23\bin' 'StringReverser'
Enter a string to reverse: shital
Reversed string: latihs
PS D:\Desktop\StringReverser.java>
```

TECHNOLOGY USED

Technology	Purpose
Java	Programming language

TRAINING EXPERIENCE

Working on this project gave me real-time exposure to practical Java development. I became confident with writing modular code using OOP, handling user inputs, validations, and file operations in Java.

I understood how small applications can incorporate real-world components like authentication systems, which are foundational in enterprise-level applications.

KEY LEARNINGS

- Implemented a secure, menu-driven console app
- Learned Java file handling (read/write)
- Built reusable classes using OOP
- Applied mathematical logic in code
- Improved error handling and validation techniques

CONCLUSION

The String Reverser program is a fundamental yet powerful example of string manipulation in programming. It not only helps beginners understand the core concepts of loops, indexing, and character handling, but also serves as a foundation for more complex string-processing tasks such as palindrome checking and text analysis.

Through a straightforward implementation in Java, this program demonstrates how a simple algorithm can be both effective and efficient. By reinforcing problem-solving skills and logical thinking, the string reversal exercise proves to be a valuable learning tool in the early stages of software development.

As strings are a key data type in virtually all programming languages, mastering basic operations like reversing a string prepares developers for more advanced applications and real-world challenges in software engineering.