A logo for a university

AI-generated content may be incorrect.

Student Name: Le Mac Hoang King

University of Greenwich ID Number: 001342002

FPT Student ID Number: GCD220068

Module Code: COMP-1551-M06-2024-25-371

Module Assessment Title: Application Development

Lecturer Name: Nguyen The Nghia

Submission Date: 25/04/2024

**Table of Contents**

[I. Software Requirements Specification 4](#_Toc196534516)

[A. Introduction 4](#_Toc196534517)

[1. Objective 4](#_Toc196534518)

[2. Project Scope 4](#_Toc196534519)

[B. Overall Description 4](#_Toc196534520)

[1. Product Overview 4](#_Toc196534521)

[2. Intended Users 4](#_Toc196534522)

[3. Operational Environment 4](#_Toc196534523)

[C. System Features 4](#_Toc196534524)

[1. Description 4](#_Toc196534525)

[2. Functional Requirements 5](#_Toc196534526)

[D. User Interface Requirements 5](#_Toc196534527)

[E. Platform Requirements 5](#_Toc196534528)

[F. Quality Attributes 5](#_Toc196534529)

[1. Performance 5](#_Toc196534530)

[2. Security 5](#_Toc196534531)

[3. Safety 5](#_Toc196534532)

[II. CLASS String Processing 6](#_Toc196534533)

[A. Overview 6](#_Toc196534534)

[B. Functionalities and Explanation 6](#_Toc196534535)

[1. Constructor 6](#_Toc196534536)

[2. Method Encode (Process) 6](#_Toc196534537)

[3. Method Print 7](#_Toc196534538)

[4. Method InputCode 7](#_Toc196534539)

[5. Method OutputCode 7](#_Toc196534540)

[6. Method Sort 7](#_Toc196534541)

[7. Input Validation 8](#_Toc196534542)

[8. Vowel Replacement 8](#_Toc196534543)

[9. Mirror String Processing 9](#_Toc196534544)

[C. Database Operations 9](#_Toc196534545)

[1. Save to Database 9](#_Toc196534546)

[2. Display 10](#_Toc196534547)

[3. Find 11](#_Toc196534548)

[4. Remove 12](#_Toc196534549)

[5. Edit 13](#_Toc196534550)

[6. Clear 15](#_Toc196534551)

[D. User Authentication (Login) 15](#_Toc196534552)

[E. Menu-Driven Interface 16](#_Toc196534553)

[III. User Interface Requirements 17](#_Toc196534554)

[A. Class Diagram 17](#_Toc196534555)

[B. Use Case Diagram 18](#_Toc196534556)

[IV. Upcoming Enhancements 18](#_Toc196534557)

[V. Testing 18](#_Toc196534558)

[VI. Appendix 21](#_Toc196534559)

# I. Software Requirements Specification

## Introduction

### Objective

This document specifies the requirements for a C# console application designed for educational purposes, enabling students to explore string processing and object-oriented programming (OOP) through a Caesar cipher implementation. The application facilitates learning by demonstrating input validation, string manipulation, and ASCII code handling in a controlled environment.

### 2. Project Scope

The application processes an input string (uppercase A-Z, ≤40 characters) and a shift value (-25 to 25) using a StringProcessing class, producing an encoded string, ASCII codes, and a sorted string. It includes a test to verify encoding reversibility by using the encoded string as input with the opposite shift. The scope excludes graphical interfaces, data persistence, or advanced cryptographic features, focusing on console-based interaction and core OOP principles.

## B. Overall Description

### **1. Product Overview**

The product is a C# console application centered on the StringProcessing class, which encapsulates string processing logic. It validates inputs, applies a Caesar cipher with alphabet wrap-around, generates ASCII codes for input/output strings, and sorts the input alphabetically. The class ensures data integrity through encapsulated properties and methods, providing a robust learning tool.

### 2. Intended Users

The primary users are computer science students and novice C# developers learning string manipulation and OOP. Users interact via console input/output, requiring only basic familiarity with command-line interfaces and no advanced programming skills.

### 3. Operational Environment

The application operates on any platform supporting a standard C# compiler, such as .NET Framework 4.8 or .NET 6.0, including Windows, macOS, and Linux. It relies solely on standard .NET libraries, ensuring portability and ease of deployment in educational settings.

## C. System Features

### 1. Description

The application processes strings using a Caesar cipher, transforming an input string by shifting characters based on a user-specified value. It supports validation, ASCII code output, sorting, and a reversibility test, serving as an educational tool for understanding string operations and OOP.

### 2. Functional Requirements

|  |  |
| --- | --- |
| **ID** | **Requirement** |
| FR1 | Accept an input string (A-Z, ≤40 characters) and shift value (-25 to 25). |
| FR2 | Validate inputs, prompting re-entry for invalid strings (non-A-Z, up to 40 characters) or shifts (outside -25 to 25). |
| FR3 | Encode the string using a Caesar cipher, wrapping around the alphabet (e.g., 'Z' + 1 = 'A'). |
| FR4 | Output the encoded string, ASCII codes for input/output, and sorted input string. |
| FR5 | Test reversibility by encoding the output string with the opposite shift, verifying the original string is restored. |

## D. User Interface Requirements

The application uses a console interface:

* Prompts for string and shift input with clear instructions.
* Displays error messages and re-prompts for invalid inputs.
* Output results in a structured, labeled format.

## E. Platform Requirements

* Compiler: Standard C# compiler, for example, .NET Framework 4.8 or .NET 6.0.
* Operating System: Windows, macOS, or Linux with .NET support.
* Dependences: None; uses standard .NET libraries, for example, System, System.Linq, and System.

## F. Quality Attributes

### 1. Performance

The application processes strings up to 40 characters instantly, with no delay due to efficient operations like shifting and sorting.

### 2. Security

Input validation ensures valid strings (A-Z, up to 40 characters) and shifts (-25 to 25), maintaining data integrity. No external inputs or network access minimizes risks.

### 3. Safety

The application is safe, performing in-memory operations without interacting with system resources or external devices.

# II. CLASS String Processing

## A. Overview

The String Processing System is implemented through a hierarchy of classes, with StringProcessorBase as an abstract base class and derived classes CaesarCipherProcessor, VowelReplacerProcessor, and MirrorProcessor. These classes encapsulate string manipulation logic, input validation, and database interaction, providing a robust educational tool for learning object-oriented programming (OOP) and string processing in C#.

## B. Functionalities and Explanation

### 1. Constructor

1. public StringProcessorBase(string inputString)

2. {

3. S = inputString;

4. }

5.

6. public CaesarCipherProcessor(string inputString, int shift) : base(inputString)

7. {

8. N = shift;

9. }

10.

11. public VowelReplacerProcessor(string inputString) : base(inputString)

12. {

13. N = 0;

14. }

15.

16. public MirrorProcessor(string inputString) : base(inputString)

17. {

18. N = 0;

19. }

**Description**: The StringProcessorBase constructor initializes the input string, validated via S. CaesarCipherProcessor sets the shift value, while VowelReplacerProcessor and MirrorProcessor set N to 0, as shifts are not applicable.

### 2. Method Encode (Process)

1. // CaesarCipherProcessor

2. public override string Process()

3. {

4. StringBuilder result = new StringBuilder();

5. foreach (char c in S)

6. {

7. if (char.IsLetter(c))

8. {

9. char shifted = (char)(c + N);

10. if (shifted > 'Z') shifted = (char)(shifted - 26);

11. if (shifted < 'A') shifted = (char)(shifted + 26);

12. result.Append(shifted);

13. }

14. else

15. {

16. result.Append(c);

17. }

18. }

19. string encodedString = result.ToString();

20. Output = encodedString;

21. return encodedString;

22. }

**Description**: The Process method in CaesarCipherProcessor applies a Caesar cipher, shifting letters by N with alphabet wrap-around. Non-letters are preserved, and the result is stored in Output.

### 3. Method Print

1. public virtual string Print()

2. {

3. return $"Input: {S}, Output: {Output}";

4. }

5.

6. public override string Print() // In CaesarCipherProcessor

7. {

8. return $"Input: {S}, Output: {Output}, Shift: {N}";

9. }

**Description:** The Print method formats input and output strings. CaesarCipherProcessor overrides it to include the shift value, enhancing output clarity.

### 4. Method InputCode

1. public virtual int[] InputCode()

2. {

3. if (S == null)

4. throw new InvalidOperationException("Input string (S) is null.");

5. return S.Select(c => (int)c).ToArray();

6. }

**Description**: The InputCode method converts input string characters to ASCII values, returning an array with a null check.

### 5. Method OutputCode

1. public virtual int[] OutputCode()

2. {

3. if (Output == null)

4. throw new InvalidOperationException("Output string is null.");

5. return Output.Select(c => (int)c).ToArray();

6. }

**Description**: The OutputCode method converts the encoded string’s characters to ASCII values, ensuring the output is not null.

### 6. Method Sort

1. public virtual string Sort()

2. {

3. if (S == null)

4. throw new InvalidOperationException("Input string (S) is null.");

5. char[] sortedChars = S.ToCharArray();

6. Array.Sort(sortedChars);

7. return new string(sortedChars);

8. }

**Description**: The Sort method sorts the input string’s characters alphabetically, with a null check.

### 7. Input Validation

1. public string S

2. {

3. get => \_s;

4. set

5. {

6. if (value == null)

7. throw new ArgumentNullException(nameof(value), "Input string cannot be null.");

8. if (value.Length > 40 || !System.Text.RegularExpressions.Regex.IsMatch(value, @"^[A-Z]+$"))

9. throw new ArgumentException("Input string must contain only uppercase letters (A-Z) with no lowercase, special characters, or diacritics, and be no longer than 40 characters.");

10. \_s = value;

11. }

12. }

13.

14. public virtual int N

15. {

16. get => \_n;

17. set

18. {

19. if (value < -25 || value > 25)

20. throw new ArgumentException("Shift value must be between -25 and 25.");

21. \_n = value;

22. }

23. }

24.

25. public static class InputValidator

26. {

27. public static void ValidateString(string input)

28. {

29. if (string.IsNullOrEmpty(input))

30. throw new ArgumentException("Input string cannot be null or empty.");

31. if (input.Length > 40)

32. throw new ArgumentException("Input string must be no longer than 40 characters.");

33. if (!System.Text.RegularExpressions.Regex.IsMatch(input, @"^[A-Z]+$"))

34. throw new ArgumentException("Input string must contain only uppercase letters (A-Z) with no lowercase, special characters, or diacritics.");

35. }

36.

37. public static void ValidateShift(int shift)

38. {

39. if (shift < -25 || shift > 25)

40. throw new ArgumentException("Shift value must be between -25 and 25.");

41. }

42. }

**Description**: The S property validates input strings as uppercase A-Z, up to 40 characters, using regex. The N property restricts shift values to -25 to 25 for CaesarCipherProcessor. The InputValidator class enforces these rules in the Program class, ensuring robust input handling.

### 8. Vowel Replacement

1. public override string Process() // In VowelReplacerProcessor

2. {

3. StringBuilder result = new StringBuilder();

4. foreach (char c in S)

5. {

6. if ("AEIOU".Contains(c))

7. result.Append('\*');

8. else

9. result.Append(c);

10. }

11. string encodedString = result.ToString();

12. Output = encodedString;

13. return encodedString;

14. }

**Description**: The VowelReplacerProcessor replaces vowels (A, E, I, O, U) with asterisks (\*), demonstrating conditional character substitution.

### 9. Mirror String Processing

1. public override string Process() // In MirrorProcessor

2. {

3. char[] reversedChars = S.ToCharArray();

4. Array.Reverse(reversedChars);

5. string reversedString = new string(reversedChars);

6. string encodedString = S + reversedString;

7. Output = encodedString;

8. return encodedString;

9. }

**Description**: The MirrorProcessor concatenates the input string with its reverse, teaching array manipulation.

## C. Database Operations

### 1. Save to Database

1. public static void SaveToDatabase(string inputString, string encodedString, string inputCode = null, string outputCode = null, string algorithm = null, int? shift = null, bool checkForDuplicate = false)

2. {

3. if (checkForDuplicate && IsStringExist(inputString))

4. {

5. throw new ArgumentException("Error: This input string already exists in the database. Please use a different string.");

6. }

7. string connectionString = "Server=localhost;Database=stringprocessingdb;User Id=root;Password=;Charset=utf8mb4;";

8. string query = "UPDATE History SET EncodedString = @EncodedString, InputCode = @InputCode, OutputCode = @OutputCode, Algorithm = @Algorithm, Shift = @Shift WHERE InputString = @InputString";

9. using (MySqlConnection conn = new MySqlConnection(connectionString))

10. {

11. MySqlCommand cmd = new MySqlCommand(query, conn);

12. cmd.Parameters.AddWithValue("@InputString", inputString);

13. cmd.Parameters.AddWithValue("@EncodedString", encodedString);

14. cmd.Parameters.AddWithValue("@InputCode", (object)inputCode ?? DBNull.Value);

15. cmd.Parameters.AddWithValue("@OutputCode", (object)outputCode ?? DBNull.Value);

16. cmd.Parameters.AddWithValue("@Algorithm", (object)algorithm ?? DBNull.Value);

17. cmd.Parameters.AddWithValue("@Shift", (object)shift ?? DBNull.Value);

18. try

19. {

20. conn.Open();

21. int rowsAffected = cmd.ExecuteNonQuery();

22. if (rowsAffected == 0)

23. {

24. string insertQuery = "INSERT INTO History (UserId, InputString, EncodedString, InputCode, OutputCode, Algorithm, Shift) VALUES (1, @InputString, @EncodedString, @InputCode, @OutputCode, @Algorithm, @Shift)";

25. MySqlCommand insertCmd = new MySqlCommand(insertQuery, conn);

26. insertCmd.Parameters.AddWithValue("@InputString", inputString);

27. insertCmd.Parameters.AddWithValue("@EncodedString", encodedString);

28. insertCmd.Parameters.AddWithValue("@InputCode", (object)inputCode ?? DBNull.Value);

29. insertCmd.Parameters.AddWithValue("@OutputCode", (object)outputCode ?? DBNull.Value);

30. insertCmd.Parameters.AddWithValue("@Algorithm", (object)algorithm ?? DBNull.Value);

31. insertCmd.Parameters.AddWithValue("@Shift", (object)shift ?? DBNull.Value);

32. insertCmd.ExecuteNonQuery();

33. Console.WriteLine("Data inserted successfully (new record).");

34. }

35. else

36. {

37. Console.WriteLine("Data updated successfully.");

38. }

39. }

40. catch (Exception ex)

41. {

42. Console.WriteLine($"Error saving to database: {ex.Message}");

43. }

44. }

45. }

**Description:** The SaveToDatabase method stores or updates string processing results, checking for duplicates if specified. It attempts to update an existing record or inserts a new one, storing input string, encoded string, ASCII codes, algorithm, and shift.

### 2. Display

1. static void Display()

2. {

3. string connectionString = "Server=localhost;Database=stringprocessingdb;User Id=root;Password=;Charset=utf8mb4;";

4. string query = "SELECT InputString, EncodedString, InputCode, OutputCode, Algorithm, Shift FROM History";

5. using (MySqlConnection conn = new MySqlConnection(connectionString))

6. {

7. try

8. {

9. MySqlCommand cmd = new MySqlCommand(query, conn);

10. conn.Open();

11. MySqlDataReader reader = cmd.ExecuteReader();

12. if (reader.HasRows)

13. {

14. while (reader.Read())

15. {

16. Console.WriteLine($"- Original Code (InputString): {reader["InputString"]}");

17. Console.WriteLine($"1. Encoded String: {reader["EncodedString"]}");

18. Console.WriteLine($"2. Algorithm: {(reader.IsDBNull(reader.GetOrdinal("Algorithm")) ? "N/A" : reader["Algorithm"])}");

19. if (!reader.IsDBNull(reader.GetOrdinal("Shift")))

20. {

21. Console.WriteLine($"3. Shift: {reader["Shift"]}");

22. }

23. if (!reader.IsDBNull(reader.GetOrdinal("InputCode")))

24. {

25. Console.WriteLine($"4. InputCode: {reader["InputCode"]}");

26. }

27. if (!reader.IsDBNull(reader.GetOrdinal("OutputCode")))

28. {

29. Console.WriteLine($"5. OutputCode: {reader["OutputCode"]}");

30. }

31. Console.WriteLine();

32. }

33. }

34. else

35. {

36. Console.WriteLine("No records found.");

37. }

38. }

39. catch (Exception ex)

40. {

41. Console.WriteLine($"Error displaying records: {ex.Message}");

42. }

43. }

44. }

**Description:** The Display method retrieves and shows all records from the History table, including input string, encoded string, algorithm, shift, and ASCII codes. It handles empty databases by informing the user.

### 3. Find

1. static void Find()

2. {

3. Console.Write("Enter the original string to search for: ");

4. string searchOriginalString = Console.ReadLine();

5. string connectionString = "Server=localhost;Database=stringprocessingdb;User Id=root;Password=;Charset=utf8mb4;";

6. string query = "SELECT InputString, EncodedString, InputCode, OutputCode, Algorithm, Shift FROM History WHERE InputString = @InputString";

7. using (MySqlConnection conn = new MySqlConnection(connectionString))

8. {

9. try

10. {

11. MySqlCommand cmd = new MySqlCommand(query, conn);

12. cmd.Parameters.AddWithValue("@InputString", searchOriginalString);

13. conn.Open();

14. MySqlDataReader reader = cmd.ExecuteReader();

15. if (reader.Read())

16. {

17. Console.WriteLine($"- Original Code (InputString): {reader["InputString"]}");

18. Console.WriteLine($"- Encoded String: {reader["EncodedString"]}");

19. Console.WriteLine($"- Algorithm: {(reader.IsDBNull(reader.GetOrdinal("Algorithm")) ? " WATCHN/A" : reader["Algorithm"])}");

20. if (!reader.IsDBNull(reader.GetOrdinal("Shift")))

21. {

22. Console.WriteLine($"- Shift: {reader["Shift"]}");

23. }

24. if (!reader.IsDBNull(reader.GetOrdinal("InputCode")))

25. {

26. Console.WriteLine($"- InputCode: {reader["InputCode"]}");

27. }

28. if (!reader.IsDBNull(reader.GetOrdinal("OutputCode")))

29. {

30. Console.WriteLine($"- OutputCode: {reader["OutputCode"]}");

31. }

32. }

33. else

34. {

35. Console.WriteLine("Original string not found.");

36. }

37. }

38. catch (Exception ex)

39. {

40. Console.WriteLine($"Error finding record: {ex.Message}");

41. }

42. }

43. }

**Description:** The Find method searches for an input string in the History table, displaying its details or indicating if not found.

### 4. Remove

1. static void Remove()

2. {

3. Console.Write("Enter the original string to remove: ");

4. string removeOriginalString = Console.ReadLine();

5. string connectionString = "Server=localhost;Database=stringprocessingdb;User Id=root;Password=;Charset=utf8mb4;";

6. string selectQuery = "SELECT EncodedString, Algorithm, Shift FROM History WHERE InputString = @InputString";

7. string deleteQuery = "DELETE FROM History WHERE InputString = @InputString";

8. using (MySqlConnection conn = new MySqlConnection(connectionString))

9. {

10. try

11. {

12. conn.Open();

13. MySqlCommand selectCmd = new MySqlCommand(selectQuery, conn);

14. selectCmd.Parameters.AddWithValue("@InputString", removeOriginalString);

15. MySqlDataReader reader = selectCmd.ExecuteReader();

16. if (reader.Read())

17. {

18. string encodedString = reader["EncodedString"].ToString();

19. string algorithm = reader.IsDBNull(reader.GetOrdinal("Algorithm")) ? "N/A" : reader["Algorithm"].ToString();

20. string shift = reader.IsDBNull(reader.GetOrdinal("Shift")) ? "N/A" : reader["Shift"].ToString();

21. reader.Close();

22. Console.WriteLine($"Found record: InputString={removeOriginalString}, EncodedString={encodedString}, Algorithm={algorithm}, Shift={shift}");

23. Console.Write("Confirm removal (y/n)? ");

24. if (Console.ReadLine().ToLower() != "y")

25. {

26. Console.WriteLine("Removal cancelled.");

27. return;

28. }

29. MySqlCommand deleteCmd = new MySqlCommand(deleteQuery, conn);

30. deleteCmd.Parameters.AddWithValue("@InputString", removeOriginalString);

31. int rowsAffected = deleteCmd.ExecuteNonQuery();

32. if (rowsAffected > 0)

33. {

34. Console.WriteLine("Original string removed.");

35. }

36. }

37. else

38. {

39. Console.WriteLine("Original string not found.");

40. }

41. }

42. catch (Exception ex)

43. {

44. Console.WriteLine($"Error removing record: {ex.Message}");

45. }

46. }

47. }

**Description:** The Remove method deletes a record after user confirmation, showing its details first. It informs if the string is not found.

### 5. Edit

1. static void Edit()

2. {

3. Console.Write("Enter the original string to edit: ");

4. string editOriginalString = Console.ReadLine();

5.

6. string connectionString = "Server=localhost;Database=stringprocessingdb;User Id=root;Password=;Charset=utf8mb4;";

7. string query = "SELECT InputString, EncodedString, Algorithm, Shift FROM History WHERE InputString = @InputString";

8.

9. using (MySqlConnection conn = new MySqlConnection(connectionString))

10. {

11. try

12. {

13. MySqlCommand cmd = new MySqlCommand(query, conn);

14. cmd.Parameters.AddWithValue("@InputString", editOriginalString);

15.

16. conn.Open();

17. MySqlDataReader reader = cmd.ExecuteReader();

18.

19. if (reader.Read())

20. {

21. Console.WriteLine($"Current record: InputString={reader["InputString"]}, EncodedString={reader["EncodedString"]}, Algorithm={(reader.IsDBNull(reader.GetOrdinal("Algorithm")) ? "N/A" : reader["Algorithm"])}, Shift={(reader.IsDBNull(reader.GetOrdinal("Shift")) ? "N/A" : reader["Shift"])}");

22. reader.Close();

23.

24. bool isValidInput = false;

25. string newInputString = null;

26.

27. while (!isValidInput)

28. {

29. Console.Write("Enter new string of capital letters (max 40 characters): ");

30. newInputString = Console.ReadLine();

31. InputValidator.ValidateString(newInputString);

32.

33. string checkDuplicateQuery = "SELECT COUNT(\*) FROM History WHERE InputString = @NewInputString AND InputString != @EditOriginalString";

34. MySqlCommand checkCmd = new MySqlCommand(checkDuplicateQuery, conn);

35. checkCmd.Parameters.AddWithValue("@NewInputString", newInputString);

36. checkCmd.Parameters.AddWithValue("@EditOriginalString", editOriginalString);

37.

38. int count = Convert.ToInt32(checkCmd.ExecuteScalar());

39. if (count > 0)

40. {

41. Console.WriteLine($"Error: The string '{newInputString}' already exists in the database. Please enter a different string.");

42. }

43. else

44. {

45. isValidInput = true;

46. }

47. }

48.

49. Console.WriteLine("Select algorithm:");

50. Console.WriteLine("1. Caesar Cipher");

51. Console.WriteLine("2. Vowel/Consonant Replacement");

52. Console.WriteLine("3. Mirror String");

53. Console.Write("Choose an option (1-3): ");

54. if (!int.TryParse(Console.ReadLine(), out int algoChoice) || algoChoice < 1 || algoChoice > 3)

55. {

56. throw new ArgumentException("Invalid algorithm choice.");

57. }

58.

59. string algorithm;

60. int? shift = null;

61. StringProcessorBase processor;

62.

63. switch (algoChoice)

64. {

65. case 1:

66. Console.Write("Enter shift value between -25 to 25: ");

67. if (!int.TryParse(Console.ReadLine(), out int newShiftValue))

68. {

69. throw new ArgumentException("Invalid shift value.");

70. }

71. InputValidator.ValidateShift(newShiftValue);

72. algorithm = "Caesar";

73. shift = newShiftValue;

74. processor = new CaesarCipherProcessor(newInputString, newShiftValue);

75. break;

76. case 2:

77. algorithm = "Vowel";

78. processor = new VowelReplacerProcessor(newInputString);

79. break;

80. case 3:

81. algorithm = "Mirror";

82. processor = new MirrorProcessor(newInputString);

83. break;

84. default:

85. throw new ArgumentException("Invalid algorithm choice.");

86. }

87.

88. string encodedString = processor.Process();

89. if (encodedString == null)

90. {

91. throw new InvalidOperationException("Encoded string cannot be null.");

92. }

93.

94. string inputCode = string.Join(", ", processor.InputCode());

95. string outputCode = string.Join(", ", processor.OutputCode());

96.

97. RemoveOldRecord(editOriginalString);

98.

99. SaveToDatabase(newInputString, encodedString, inputCode, outputCode, algorithm, shift, false);

100.

101. Console.WriteLine($"- Updated Encoded String: {encodedString}");

102. Console.WriteLine($"- {processor.Print()}");

103. Console.WriteLine($"- Sorted Input: {processor.Sort()}");

104. Console.WriteLine($"- Algorithm: {algorithm}");

105. if (shift.HasValue)

106. {

107. Console.WriteLine($"- Shift: {shift.Value}");

108. }

109. Console.WriteLine($"- InputCode (ASCII values): {inputCode}");

110. Console.WriteLine($"- OutputCode (ASCII values): {outputCode}");

111. }

112. else

113. {

114. Console.WriteLine("Original string not found.");

115. }

116. }

117. catch (ArgumentException ex)

118. {

119. Console.WriteLine($"Error: {ex.Message}");

120. }

121. catch (InvalidOperationException ex)

122. {

123. Console.WriteLine($"Error: {ex.Message}");

124. }

125. catch (Exception ex)

126. {

127. Console.WriteLine($"Error editing record: {ex.Message}");

128. }

129. }

130. }

131.

**Description:** The Edit method updates a record with a new string and algorithm, removing the old record before saving the new details.

### 6. Clear

1. static void Clear()

2. {

3. string connectionString = "Server=localhost;Database=stringprocessingdb;User Id=root;Password=;Charset=utf8mb4;";

4. string query = "DELETE FROM History";

5. using (MySqlConnection conn = new MySqlConnection(connectionString))

6. {

7. try

8. {

9. MySqlCommand cmd = new MySqlCommand(query, conn);

10. conn.Open();

11. cmd.ExecuteNonQuery();

12. Console.WriteLine("All entries have been cleared.");

13. }

14. catch (Exception ex)

15. {

16. Console.WriteLine($"Error clearing records: {ex.Message}");

17. }

18. }

19. }

20.

**Description**: The Clear method deletes all records from the History table, resetting the database.

## D. User Authentication (Login)

1. static bool Login()

2. {

3. Console.Write("Enter username: ");

4. string username = Console.ReadLine();

5. Console.Write("Enter password: ");

6. string password = Console.ReadLine();

7. string connectionString = "Server=localhost;Database=stringprocessingdb;User Id=root;Password=;Charset=utf8mb4;";

8. string query = "SELECT COUNT(\*) FROM Users WHERE Username = @Username AND Password = @Password";

9. using (MySqlConnection conn = new MySqlConnection(connectionString))

10. {

11. MySqlCommand cmd = new MySqlCommand(query, conn);

12. cmd.Parameters.AddWithValue("@Username", username);

13. cmd.Parameters.AddWithValue("@Password", password);

14. try

15. {

16. conn.Open();

17. int result = Convert.ToInt32(cmd.ExecuteScalar());

18. if (result > 0)

19. {

20. Console.WriteLine("Login successful!");

21. return true;

22. }

23. else

24. {

25. Console.WriteLine("Invalid username or password.");

26. return false;

27. }

28. }

29. catch (Exception ex)

30. {

31. Console.WriteLine("Error: " + ex.Message);

32. return false;

33. }

34. }

35. }

**Description**: The Login method authenticates users against a MySQL database, granting access with valid credentials.

## E. Menu-Driven Interface

1. static void Main(string[] args)

2. {

3. bool isLoggedIn = false;

4. while (!isLoggedIn)

5. {

6. isLoggedIn = Login();

7. if (!isLoggedIn)

8. {

9. Console.WriteLine("Login failed. Please try again.");

10. }

11. }

12. bool running = true;

13. while (running)

14. {

15. Console.WriteLine("\n<> Menu String Processing System <>");

16. Console.WriteLine("0. Process Default Input (HELLO WORLD)");

17. Console.WriteLine("1. New Input (Caesar Cipher)");

18. Console.WriteLine("2. New Input (Vowel/Consonant Replacement)");

19. Console.WriteLine("3. New Input (Mirror String)");

20. Console.WriteLine("4. Display String");

21. Console.WriteLine("5. Find String");

22. Console.WriteLine("6. Remove String");

23. Console.WriteLine("7. Edit String");

24. Console.WriteLine("8. Clear All");

25. Console.WriteLine("9. Exit");

26. // ... (handles user input)

27. }

28. }

**Description**: The Main method provides a menu-driven interface for processing inputs, managing records, or exiting.

# III. User Interface Requirements

## A. Class Diagram

A screenshot of a computer

AI-generated content may be incorrect.

This Class Diagram depicts the String Processing System in C#. The abstract StringProcessorBase class, with attributes (\_s, \_n, \_output) and methods (InputCode(), Process()), is inherited by CaesarCipherProcessor, VowelReplacerProcessor, and MirrorProcessor for specific string operations. InputValidator offers static validation methods, while Program manages user interactions and database tasks via "uses" associations.

## Use Case Diagram

A black grid with white lines and dots

AI-generated content may be incorrect.

This Use Case Diagram shows interactions in the String Processing System. The User performs actions like Login, string processing (Apply Caesar Cipher, Replace Vowels, Mirror String), and database operations (Display Strings, Edit String). The Database supports data-related use cases (Save to Database, Clear All). Included use cases (Validate Input, Save to Database) ensure validation and storage, with use cases grouped for clarity.

# IV. Upcoming Enhancements

The following improvements are proposed to improve the String Processing System’s functionality, security, and usability:

* **Secure Password Storage**: Replace plain-text passwords with salted hashing (e.g., bcrypt) to enhance user data security.
* **Graphical Interface**: Transition from a console to a user-friendly GUI using Windows Forms or WPF.
* **Automated Testing & CI**: Add unit tests for encoding, sorting, and validation, integrated into a continuous integration pipeline for quality assurance.
* **External Configuration**: Move hardcoded settings (e.g., connection strings, validation rules) to external files (JSON/XML) or environment variables for easier maintenance.

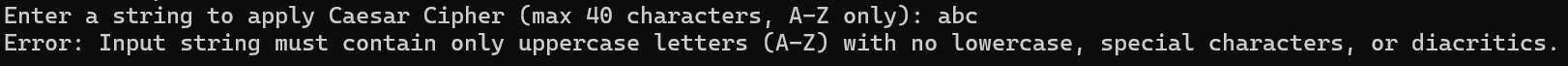
These improvements will enhance the system’s security, usability, and maintainability.

# V. Testing

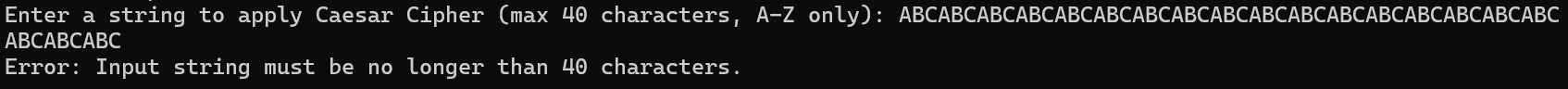
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Case ID | Function | Test Description | Input | Output | Verification | Result |
| 1 | Input String Validation | Test invalid string (lowercase) | "abc" | Error: "Input string must contain only uppercase letters (A-Z)" | [Evidence](#_Case_1:_Input) |  |
| 2 | Input String Validation | Test string exceeding 40 characters | 41-character string (e.g., "A...A") | Error: "Input string must be no longer than 40 characters." | [Evidence](#_Case_2:_Input) |  |
| 3 | Shift Value Validation | Test invalid shift value (outside -25 to 25) | Shift = 26 | Error: "Shift value must be between -25 and 25." | [Evidence](#_Case_3:_Shift) |  |
| 4 | Caesar Cipher Encoding | Encode string with shift | "EXAMPLE", Shift = 3 | Encoded: "HAPSOH" | [Evidence](#_Case_4:_Caesar) |  |
| 5 | Caesar Cipher Encoding | Test edge case: shift wraps around (Z+1=A) | "XYZ", Shift = 3 | Encoded: "ABC" | [Evidence](#_Case_5:_Caesar) |  |
| 6 | Vowel Replacement | Replace vowels with asterisks | "HELLO" | Encoded: "H*LL*" | [Evidence](#_Case_6:_Vowel) |  |
| 7 | Vowel Replacement | Test string with no vowels | "BCD" | Encoded: "BCD" (unchanged) | [Evidence](#_Case_7:_Vowel) |  |
| 8 | Mirror String Processing | Create mirror string | "EXAMPLE" | Encoded: "EXAMPLEELPMAXE" | [Evidence](#_Case_8:_Mirror) |  |
| 9 | ASCII Code Conversion (Input) | Convert input string to ASCII codes | "EXAMPLE" | [69, 88, 65, 77, 80, 76, 69] | [Evidence](#_Case_9:_ASCII) |  |
| 10 | ASCII Code Conversion (Output) | Convert encoded string to ASCII codes (Caesar Cipher) | "EXAMPLE", Shift = 3 | 72, 65, 80, 83, 79, 72] (for "HAPSOH") | [Evidence](#_Case_10:_ASCII) |  |
| 11 | Display Database Records | Display all records in database | Option 4 | List of all records | [Evidence](#_Case_11:_Display) |  |
| 12 | Display Database Records | Test empty database | Option 4. No records in database | "No records found." | [Evidence](#_Case_12:_Display) |  |
| 13 | Find Record | Search for existing string | "EXAMPLE" | Displays record details (Encoded: "HAPSOH", Algorithm: "Caesar") | [Evidence](#_Case_13:_Find) |  |
| 14 | Find Record | Search for non-existent string | "NOTFOUND" | "Original string not found." | [Evidence](#_Case_14:_Find) |  |
| 15 | Remove Record | Remove existing record with confirmation | "EXAMPLE", Confirm = "y" | "Original string removed." | [Evidence](#_Case_15:_Remove) |  |
| 16 | Remove Record | Cancel removal | "EXAMPLE", Confirm = "n" | "Removal cancelled." | [Evidence](#_Case_16:_Remove) |  |
| 17 | Remove Record | Remove non-existent record | "NOTFOUND" | "Original string not found." | [Evidence](#_Case_17:_Remove) |  |
| 18 | Edit Record | Edit existing record with new string (Caesar Cipher) | Old: "EXAMPLE", New: "DEMO", Shift = 5 | Updated Encoded: "IJRT", Algorithm: "Caesar", Shift: 5 | [Evidence](#_Case_18:_Edit) |  |
| 19 | Edit Record | Edit with Vowel Replacement | Old: "EXAMPLE", New: "HELLO", Algo = Vowel | Updated Encoded: "H*LL*" | [Evidence](#_Case_19:_Edit) |  |
| 20 | Edit Record | Edit with invalid algorithm choice | Algo choice = 4 | Error: "Invalid algorithm choice."  Error: "Invalid algorithm choice." | [Evidence](#_Case_20:_Edit) |  |
| 21 | Clear Database | Clear all records | Option 8 | "All entries have been cleared." | [Evidence](#_Case_21:_Clear) |  |
| 22 | Login | Successful login | Username: "King", Password: "king123" | "Login successful!" | [Evidence](#_Case_22:_Login) |  |
| 23 | Login | Failed login (wrong credentials) | Username: "King", Password: "wrong" | "Invalid username or password." | [Evidence](#_Case_23:_Login) |  |
| 24 | Default Input Processing | Process default input (HELLO WORLD) | Option 0 | Caesar: "KHOORZRUOG", Vowel: "H*LL*W\*RLD", Mirror: "HELLOWORLDDLROWOLLEH" | [Evidence](#_Case_24:_Default) |  |
| 25 | Menu Navigation | Test invalid menu option | Option = 10 | "Invalid option. Please try again." | [Evidence](#_Case_25:_Menu) |  |
| 26 | Menu Navigation | Test non-numeric menu input | Option = "abc" | "Invalid input. Please enter a number between 0 and 9." | [Evidence](#_Case_26:_Menu) |  |
| 27 | Edit Record Duplicate | Check For Duplicate | Old: "AZAZ", New: "KINGLE" | The string already exists in the database. | [Evidence](#_Case_27:_Edit) |  |
| 28 | Caesar Cipher Encoding Duplicate | Check For Duplicate | “KINGLE”, Shift = 1 | The string already exists in the database. | [Evidence](#_Case_28:) |  |

# VI. Appendix

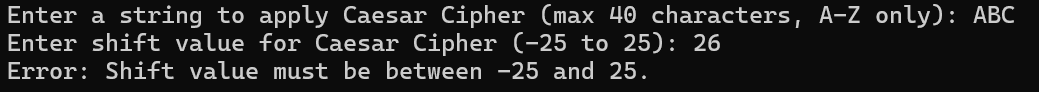
Case 1: Input String Validation



Case 2: Input String Validation



Case 3: Shift Value Validation



Case 4: Caesar Cipher Encoding.

A black screen with white text

AI-generated content may be incorrect.

Case 5: Caesar Cipher Encoding

A black screen with white text

AI-generated content may be incorrect.

Case 6: Vowel Replacement

A black screen with white text

AI-generated content may be incorrect.

Case 7: Vowel Replacement

A black background with white text

AI-generated content may be incorrect.

Case 8: Mirror String Processing

A black screen with white text

AI-generated content may be incorrect.

Case 9: ASCII Code Conversion (Input)

A black screen with white text

AI-generated content may be incorrect.

Case 10: ASCII Code Conversion (Output)

A black screen with white text

AI-generated content may be incorrect.

Case 11: Display Database Records

A screenshot of a computer

AI-generated content may be incorrect.

Case 12: Display Database Records

White text on a black background

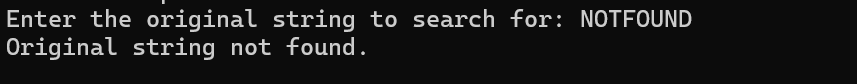
AI-generated content may be incorrect.

Case 13: Find Record

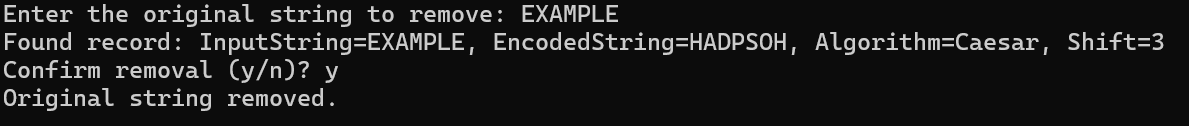
A black screen with white text

AI-generated content may be incorrect.

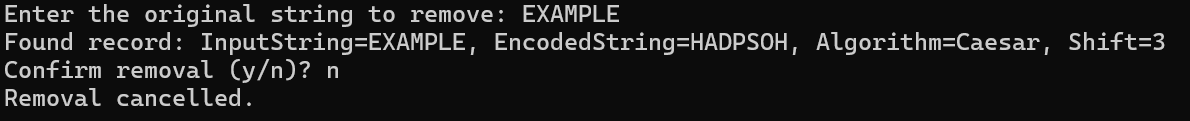
Case 14: Find Record



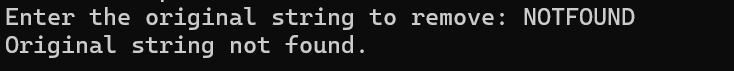
Case 15: Remove Record



Case 16: Remove Record



Case 17: Remove Record



Case 18: Edit Record

A computer screen with white text

AI-generated content may be incorrect.

Case 19: Edit Record

A screenshot of a computer program

AI-generated content may be incorrect.

Case 20: Edit Record

A screen shot of a computer

AI-generated content may be incorrect.

Case 21: Clear Database

A black background with white text

AI-generated content may be incorrect.

Case 22: Login

A black background with white text

AI-generated content may be incorrect.

Case 23: Login

A black background with white text

AI-generated content may be incorrect.

Case 24: Default Input Processing

A screen shot of a computer program

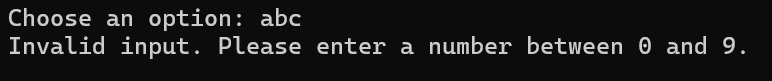
AI-generated content may be incorrect.

Case 25: Menu Navigation

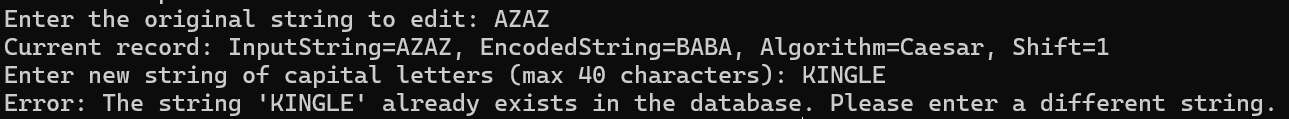
A black background with white text

AI-generated content may be incorrect.

Case 26: Menu Navigation



Case 27: Edit Record Duplicate



Case 28: Caesar Cipher Encoding Duplicate

