

# SEARCH ENGINE



## A PROJECT REPORT

*Submitted by*

**THANUJA V (2303811710422168)**

*in partial fulfillment of requirements for the award of the course*

**CGB1201 - JAVA PROGRAMMING**

*In*

**COMPUTER SCIENCE AND ENGINEERING**

**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY**

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

**SAMAYAPURAM – 621 112**

**NOVEMBER- 2024**

**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY  
(AUTONOMOUS)**

**SAMAYAPURAM – 621 112**

**BONAFIDE CERTIFICATE**

Certified that this project report on “**SEARCH ENGINE**” is the bonafide work of **THANUJA V (2303811710422168)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING  
Dr.A.DELPHIN CAROLINA RANI, M.E., Ph.D.,  
HEAD OF THE DEPARTMENT  
PROFESSOR

**SIGNATURE**

Dr.A.Delphin Carolina Rani, M.E., Ph.D.,

**HEAD OF THE DEPARTMENT**

**PROFESSOR**

Department of CSE

K.Ramakrishnan College of Technology  
(Autonomous)

Samayapuram–621112.

CGB1201-JAVA PROGRAMMING  
Mr. A. MALARMANNAN A, M.E.,  
ASSISTANT PROFESSOR

**SIGNATURE**

Mr. A. Malarmannan, M.E.,

**SUPERVISOR**

**ASSISTANT PROFESSOR**

Department of CSE

K.Ramakrishnan College of Technology  
(Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on .....06/12/2024.....

CGB1201-JAVA PROGRAMMING  
Mr. A. MALARMANNAN A, M.E.,  
INTERNAL EXAMINER  
ASSISTANT PROFESSOR

**INTERNAL EXAMINER**

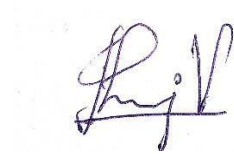
CGB1201-JAVA PROGRAMMING  
Mr. R. KARTHICK, M.E.,  
EXTERNAL EXAMINER  
ASSISTANT PROFESSOR  
8138-SCE, TRICHY.

**EXTERNAL EXAMINER**

## DECLARATION

I declare that the project report on “**SEARCH ENGINE**” is the result of original work done by us and best of our knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201 - JAVA PROGRAMMING**.

Signature



---

THANUJA V

Place: Samayapuram

Date: 06/12/2024

## ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution “**K.Ramakrishnan College of Technology (Autonomous)**”, for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN, B.E.**, for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.**, Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E.,Ph.D.**, Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **MR. A. MALARMANNAN, M.E.**, Department of **COMPUTER SCIENCE AND ENGINEERING**, for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

## **VISION OF THE INSTITUTION**

To serve the society by offering top-notch technical education on par with global standards

## **MISSION OF THE INSTITUTION**

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

## **VISION OF DEPARTMENT**

To be a center of eminence in creating competent software professionals with research and innovative skills.

## **MISSION OF DEPARTMENT**

**M1: Industry Specific:** To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

**M2: Research:** To prepare students for research-oriented activities.

**M3: Society:** To empower students with the required skills to solve complex technological problems of society.

## **PROGRAM EDUCATIONAL OBJECTIVES**

### **1. PEO1: Domain Knowledge**

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

### **2. PEO2: Employability Skills and Research**

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

### **3. PEO3: Ethics and Values**

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

### **PROGRAM SPECIFIC OUTCOMES (PSOs)**

#### **PSO 1: Domain Knowledge**

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

#### **PSO 2: Quality Software**

To apply software engineering principles and practices for developing quality software for scientific and business applications.

#### **PSO 3: Innovation Ideas**

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

### **PROGRAM OUTCOMES (POs)**

Engineering students will be able to:

- 1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- 2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
- 3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
- 4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

- 5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations
- 6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
- 7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
- 8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- 9. Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- 10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- 11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- 12. Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

## **ABSTRACT**

This project presents the development of a Search Engine implemented in Java using the Swing library for the graphical user interface (GUI). The primary aim of the project is to provide a simple and interactive platform for managing and searching web pages based on their titles and contents. Users can add, update, delete, and view web pages, as well as perform keyword searches across stored pages. The search functionality supports both case-sensitive and case-insensitive queries, with results ranked by relevance based on the frequency of keyword occurrences within the content. The project uses Java's collections framework, particularly the HashMap class, for efficient data storage and retrieval. The application demonstrates basic search engine capabilities, offering a foundation for potential future enhancements such as advanced search algorithms and data persistence.



### ABSTRACT WITH POs AND PSOs MAPPING

#### CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
<p>This project presents the development of a Search Engine implemented in Java using the Swing library for the graphical user interface (GUI). The primary aim of the project is to provide a simple and interactive platform for managing and searching web pages based on their titles and contents. Users can add, update, delete, and view web pages, as well as perform keyword searches across stored pages. The search functionality supports both case-sensitive and case-insensitive queries, with results ranked by relevance based on the frequency of keyword occurrences within the content. The project uses Java's collections framework, particularly the HashMap class, for efficient data storage and retrieval. The application demonstrates basic search engine capabilities, offering a foundation for potential future enhancements such as advanced search algorithms and data persistence.</p>	<p><b>PO1 -3</b> <b>PO2 -3</b> <b>PO3 -3</b> <b>PO4 -3</b> <b>PO5 -3</b> <b>PO6 -3</b> <b>PO7 -3</b> <b>PO8 -3</b> <b>PO9 -3</b> <b>PO10 -3</b> <b>PO11 -3</b> <b>PO12 -3</b></p>	<p><b>PSO1 -3</b> <b>PSO2 -3</b> <b>PSO3 -3</b></p>

Note: 1- Low, 2-Medium, 3- High

## TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	<b>ABSTRACT</b>	viii
<b>1</b>	<b>INTRODUCTION</b>	1
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming concepts	1
<b>2</b>	<b>PROJECT METHODOLOGY</b>	2
	2.1 Proposed Work	2
	2.2 Block Diagram	2
<b>3</b>	<b>MODULE DESCRIPTION</b>	3
	3.1 Main Module	3
	3.2 GUI Module	3
	3.3 Add, Delete, Update, View Pages Module	3
	3.4 Search Logic Module	3
	3.5 Utility Functions	3
<b>4</b>	<b>CONCLUSION &amp; FUTURE SCOPE</b>	4
	4.1 Conclusion	4
	4.2 Future Scope	4
	<b>APPENDIX A (SOURCE CODE)</b>	5
	<b>APPENDIX B (SCREENSHOTS)</b>	13
	<b>REFERENCES</b>	15

# CHAPTER 1

## INTRODUCTION

### 1.1 Objective

The objective of this project is to develop a simple search engine using Java, where users can manage web pages by adding, updating, deleting, and viewing them. It allows users to perform case-sensitive and case-insensitive keyword searches, with results ranked by relevance based on keyword frequency. The project utilizes Java's HashMap for efficient data storage and retrieval, and provides an intuitive GUI using Java Swing.

### 1.2 Overview

The code implements a basic search engine application with a graphical user interface (GUI) built using Java Swing. It manages a collection of web pages stored in a HashMap, where each entry consists of a page title and its content. Users can perform operations like adding new pages, updating or deleting existing ones, viewing all pages, and searching for keywords within the content.

### 1.3 Java Programming Concepts

The Java programming concepts used in the above code include:

- **Object-Oriented Programming (OOP):** Encapsulation through methods and data handling (HashMap for storing web pages).
- **Control Structures:** Conditional statements (if-else) for input validation and decision-making.
- **Exception Handling:** Handling user input validation and displaying error messages using dialog boxes.
- **Multithreading (Event Dispatch Thread):** GUI creation is invoked using SwingUtilities. The `invokeLater()`, ensuring thread safety in the Swing environment.

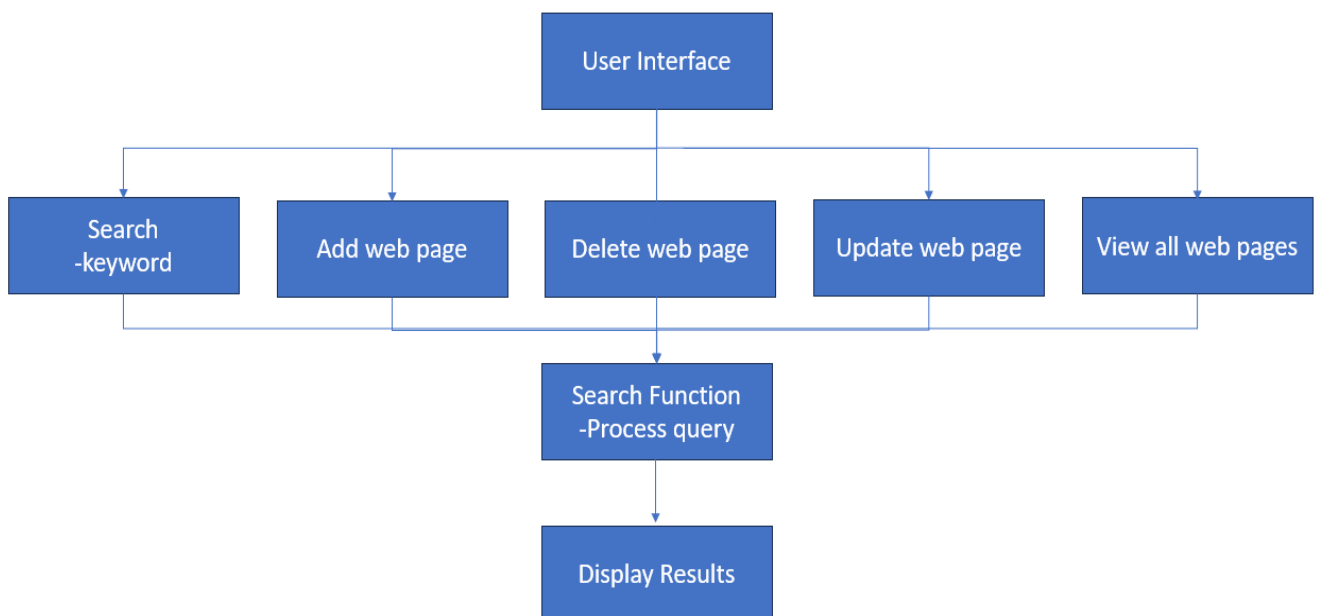
## CHAPTER 2

### PROJECT METHODOLOGY

#### 2.1 Proposed Work

The proposed work focuses on developing a simple yet functional search engine application with a user-friendly graphical interface for managing and searching web pages. The system will allow users to add new pages, update existing ones, delete unwanted entries, and view all stored pages in a structured format. The core functionality includes keyword-based search with options for case-sensitive or case-insensitive queries, displaying results ranked by relevance based on keyword frequency. The application will utilize Java Swing for building an intuitive interface and a HashMap for efficient data storage and retrieval.

#### 2.2 Block Diagram



## **CHAPTER 3**

### **MODULE DESCRIPTION**

#### **3.1 Main Module**

The main class is responsible for initializing the graphical user interface (GUI) and handling interactions with the user.

#### **3.2 Search Module**

The search module handles keyword-based searches across stored web pages. It prompts the user for a query and whether the search should be case-sensitive.

#### **3.3 Add, Delete, Update, View Pages Module**

The `showAddDialog()` adds the page to the `webPages` map if the title is unique, `showDeleteDialog()` removes the page from the `webPages` map if it exists, `showUpdateDialog()` allows the user to modify the content of the page and updates the map and `showViewDialog()` displays a list of all stored web pages (titles and their content) to the user.

#### **3.4 Search Logic Module**

The `search()` splits the query into individual keywords and checks the relevance by counting the occurrences of these keywords in the page content. Allows for case-sensitive or case-insensitive search based on the user's preference.

#### **3.5 Utility Functions**

This function splits the content into words and matches them with the keyword, ensuring accurate keyword relevance calculations for search queries.

## **CHAPTER 4**

### **CONCLUSION & FUTURE SCOPE**

#### **4.1 CONCLUSION**

In conclusion, this Java search engine offers a user-friendly interface for managing and searching web pages. It allows users to add, delete, update, view pages, and search for content using keyword queries with case-sensitive or case-insensitive options. Results are ranked based on keyword relevance, determined by the frequency of occurrences. The application demonstrates core Java concepts such as event handling, collections, and string manipulation. Future improvements could include features like keyword stemming, advanced search filters, and persistent storage for better scalability and real-world use.

#### **4.2 FUTURE SCOPE**

The future scope of this Java search engine includes several potential enhancements. One key improvement could be integrating more advanced search algorithms, such as stemming and fuzzy matching, to handle variations in keywords and improve search accuracy. Additionally, the system could benefit from implementing a more efficient data storage solution, such as a database, to support larger datasets and enable persistent storage of web pages. Incorporating features like full-text search, filters for sorting results, and ranking algorithms could further refine the search experience. Lastly, expanding the application to support multimedia content or integrating machine learning for smarter search results would make it more robust and applicable in real-world scenarios.

## **APPENDIX A**

### **(SOURCE CODE)**

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.*;
import java.util.stream.Collectors;

public class Main {

    private static final Map<String, String> webPages = new HashMap<>();

    public static void main(String[] args) {
        SwingUtilities.invokeLater(Main::createAndShowGUI);
    }

    // GUI setup
    private static void createAndShowGUI() {
        JFrame frame = new JFrame("Advanced Search Engine");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 400);

        // Main panel
        JPanel panel = new JPanel();
        panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));

        // Buttons
```

```
JButton searchButton = new JButton("Search for a keyword");
JButton addButton = new JButton("Add a new web page");
JButton deleteButton = new JButton("Delete a web page");
JButton updateButton = new JButton("Update a web page");
JButton viewButton = new JButton("View all web pages");
JButton exitButton = new JButton("Exit");
```

```
panel.add(searchButton);
panel.add(addButton);
panel.add(deleteButton);
panel.add(updateButton);
panel.add(viewButton);
panel.add(exitButton);
```

```
frame.add(panel);
frame.setVisible(true);
```

```
// Action Listeners for Buttons
```

```
searchButton.addActionListener(e -> showSearchDialog(frame));
addButton.addActionListener(e -> showAddDialog(frame));
deleteButton.addActionListener(e -> showDeleteDialog(frame));
updateButton.addActionListener(e -> showUpdateDialog(frame));
viewButton.addActionListener(e -> showViewDialog(frame));
exitButton.addActionListener(e -> System.exit(0));
```

```
}
```

```
// Show search dialog
```

```
private static void showSearchDialog(JFrame frame) {
```

```
    String query = JOptionPane.showInputDialog(frame, "Enter search query:");
```



```

        if (query == null || query.trim().isEmpty()) {
            JOptionPane.showMessageDialog(frame, "Query cannot be empty!",
"Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        int caseSensitiveOption = JOptionPane.showConfirmDialog(frame, "Case
sensitive search?", "Search Options", JOptionPane.YES_NO_OPTION);
        boolean caseSensitive = caseSensitiveOption == JOptionPane.YES_OPTION;

        Map<String, Integer> results = search(query, caseSensitive);
        StringBuilder resultMessage = new StringBuilder();
        if (results.isEmpty()) {
            resultMessage.append("No results found.");
        } else {
            resultMessage.append("Results found:\n");
            results.entrySet().stream()
                .sorted(Map.Entry.<String, Integer>comparingByValue().reversed())
                .forEach(entry -> resultMessage.append(entry.getKey()).append("
(Relevance: ").append(entry.getValue()).append(")\n"));
        }

        JOptionPane.showMessageDialog(frame, resultMessage.toString(), "Search
Results", JOptionPane.INFORMATION_MESSAGE);
    }

    // Show add dialog
    private static void showAddDialog(JFrame frame) {
        String title = JOptionPane.showInputDialog(frame, "Enter page title:");
    }

```

```

        if (title == null || title.trim().isEmpty()) {
            JOptionPane.showMessageDialog(frame, "Page title cannot be empty!",
"Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        String content = JOptionPane.showInputDialog(frame, "Enter page content:");
        if (content == null || content.trim().isEmpty()) {
            JOptionPane.showMessageDialog(frame, "Page content cannot be empty!",
"Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        if (webPages.containsKey(title)) {
            JOptionPane.showMessageDialog(frame, "Page with this title already
exists!", "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            webPages.put(title, content);
            JOptionPane.showMessageDialog(frame, "Page added successfully!",
"Success", JOptionPane.INFORMATION_MESSAGE);
        }
    }

    // Show delete dialog
    private static void showDeleteDialog(JFrame frame) {
        String title = JOptionPane.showInputDialog(frame, "Enter page title to
delete:");

        if (title == null || title.trim().isEmpty()) {
            JOptionPane.showMessageDialog(frame, "Page title cannot be empty!",

```

```

"Error", JOptionPane.ERROR_MESSAGE);
    return;
}

if (webPages.remove(title) != null) {
    JOptionPane.showMessageDialog(frame, "Page deleted successfully!",
"Success", JOptionPane.INFORMATION_MESSAGE);
} else {
    JOptionPane.showMessageDialog(frame, "Page not found!", "Error",
JOptionPane.ERROR_MESSAGE);
}
}

// Show update dialog
private static void showUpdateDialog(JFrame frame) {
    String title = JOptionPane.showInputDialog(frame, "Enter page title to
update:");
    if (title == null || title.trim().isEmpty()) {
        JOptionPane.showMessageDialog(frame, "Page title cannot be empty!",
"Error", JOptionPane.ERROR_MESSAGE);
        return;
    }

    if (!webPages.containsKey(title)) {
        JOptionPane.showMessageDialog(frame, "Page not found!", "Error",
JOptionPane.ERROR_MESSAGE);
        return;
    }
}

```

```

        String newContent = JOptionPane.showInputDialog(frame, "Enter new page
content:");

        if (newContent == null || newContent.trim().isEmpty()) {
            JOptionPane.showMessageDialog(frame, "Page content cannot be empty!",
"Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        webPages.put(title, newContent);
        JOptionPane.showMessageDialog(frame, "Page updated successfully!",
"Success", JOptionPane.INFORMATION_MESSAGE);
    }

    // Show view dialog
    private static void showViewDialog(JFrame frame) {
        if (webPages.isEmpty()) {
            JOptionPane.showMessageDialog(frame, "No web pages available!", "Info",
JOptionPane.INFORMATION_MESSAGE);
            return;
        }

        StringBuilder message = new StringBuilder("Available Web Pages:\n");
        for (Map.Entry<String, String> entry : webPages.entrySet()) {
            message.append("- ").append(entry.getKey()).append(":
").append(entry.getValue()).append("\n");
        }

        JOptionPane.showMessageDialog(frame, message.toString(), "Web Pages",
JOptionPane.INFORMATION_MESSAGE);
    }

```

```
}
```

```
// Search method
```

```
private static Map<String, Integer> search(String query, boolean caseSensitive) {
```

```
    Map<String, Integer> results = new HashMap<>();
```

```
    String[] keywords = query.trim().split("\\s+");
```

```
    for (Map.Entry<String, String> entry : webPages.entrySet()) {
```

```
        String content = entry.getValue();
```

```
        if (!caseSensitive) {
```

```
            content = content.toLowerCase();
```

```
        }
```

```
        int relevance = 0;
```

```
        for (String keyword : keywords) {
```

```
            if (!caseSensitive) {
```

```
                keyword = keyword.toLowerCase();
```

```
            }
```

```
            relevance += countWordOccurrences(content, keyword);
```

```
        }
```

```
        if (relevance > 0) {
```

```
            results.put(entry.getKey(), relevance);
```

```
        }
```

```
    }
```

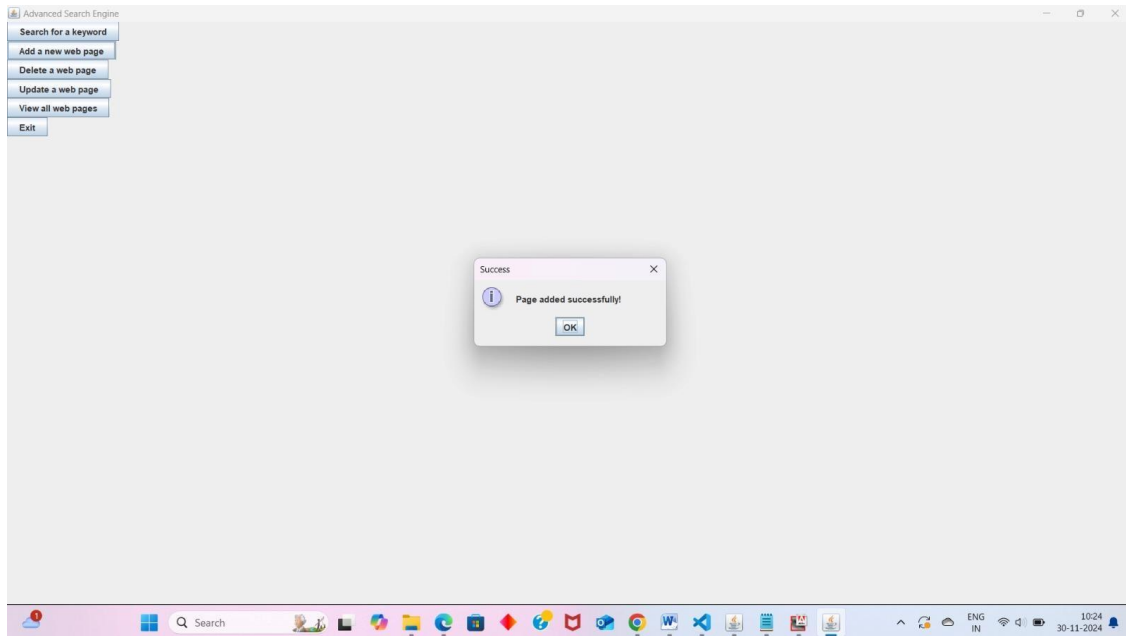
```
    return results;
```

```
}
```

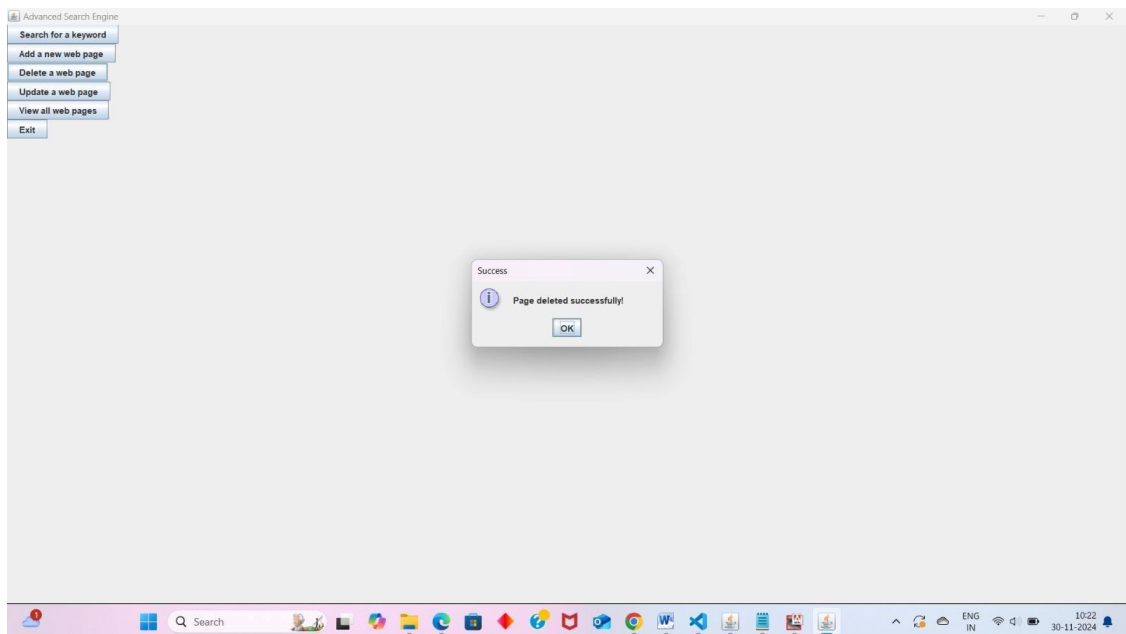
```
// Count word occurrences
private static int countWordOccurrences(String content, String keyword) {
    String[] words = content.split("\\W+");
    int count = 0;
    for (String word : words) {
        if (word.equals(keyword)) {
            count++;
        }
    }
    return count;
}
}
```

## APPENDIX B (SCREENSHOTS)

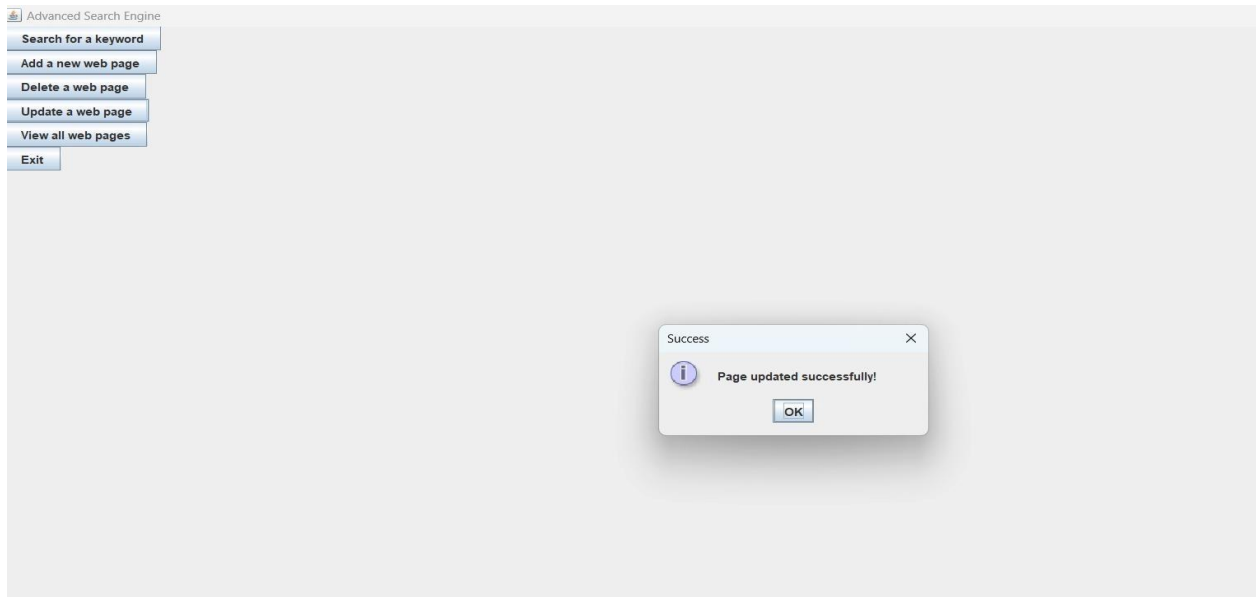
### Adding a New Web Page:



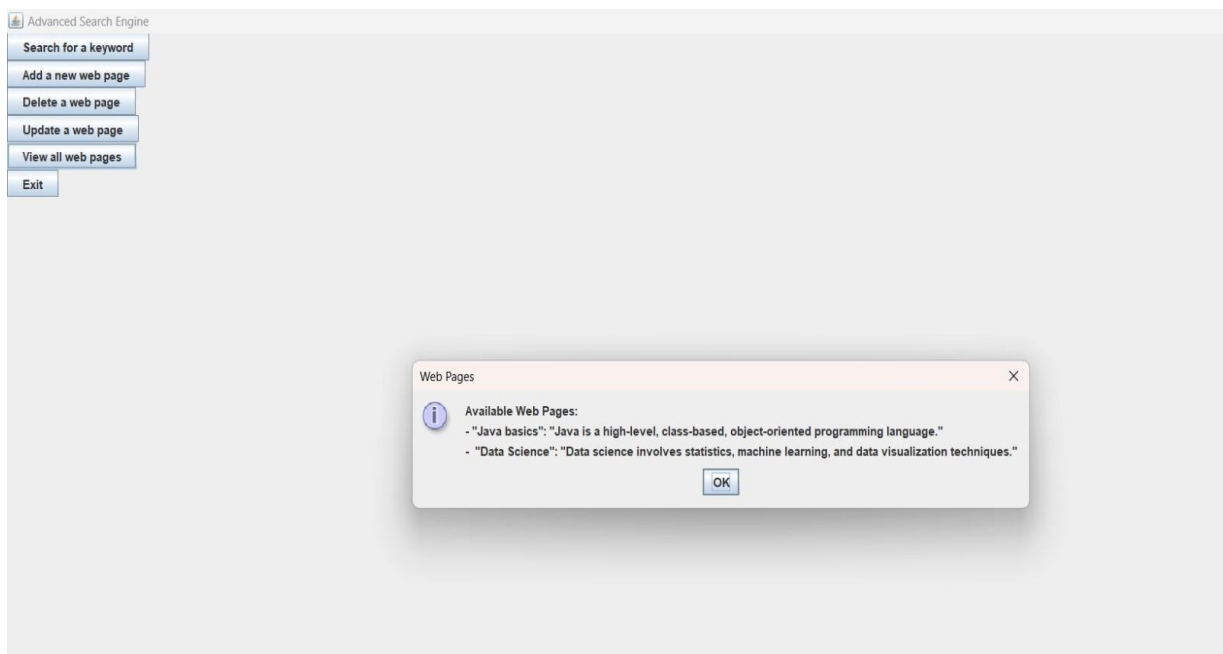
### Deleting a Web Page:



## Updating a Web Page:



## Viewing All Web Pages:





## REFERENCES

1. Robinson, M., Vorobiev, P. (2002) Swing: A GUI Toolkit for Java, Prentice Hall, Upper Saddle River, NJ, pp. 1-350.
2. Croft, B., Metzler, D., Strohman, T. (2009) Search Engines: Information Retrieval in Practice, Pearson, Boston, MA, pp. 1-384.
3. Eckstein, R., Loy, M., Wood, D. (2004) Java Swing, 2nd ed., O'Reilly Media, Sebastopol, CA, pp. 1-408.