Import javax.swing.\*;

Import java.awt.\*;

Import java.awt.event.\*;

Import java.io.\*;

Import java.net.URI;

Import java.util.\*;

Import org.jsoup.Jsoup;

Import org.jsoup.nodes.Document;

Import org.jsoup.select.Elements;

Public class SearchEngineApp extends JFrame {

Private final JTextField searchField;

Private final JTextArea resultArea;

Private final Map<String, Map<String, Integer>> index; // Index to store word count in files/web pages

Private final Set<String> visitedURLs;

Public SearchEngineApp() {

// Initialize the search engine index and visited URLs set (for crawling)

Index = new HashMap<>();

visitedURLs = new HashSet<>();

// Set up the frame

setTitle(“Search Engine”);

setSize(1000, 700);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new BorderLayout());

// Set background color and font

getContentPane().setBackground(new Color(245, 245, 245));

Font customFont = new Font(“Arial”, Font.PLAIN, 14);

UIManager.put(“Label.font”, customFont);

// Create the top panel with buttons and search input

JPanel topPanel = new JPanel();

topPanel.setLayout(new FlowLayout(FlowLayout.LEFT));

topPanel.setBackground(new Color(0, 123, 255));

// Buttons and text field for search

JButton addFilesButton = new JButton(“Index Files & Web Pages”);

addFilesButton.setBackground(new Color(70, 130, 180));

addFilesButton.setForeground(Color.WHITE);

addFilesButton.addActionListener(e -> indexFilesAndWebPages());

JLabel searchLabel = new JLabel(“Search:”);

searchLabel.setForeground(Color.WHITE);

searchField = new JTextField(25);

searchField.setFont(new Font(“Arial”, Font.PLAIN, 14));

JButton searchButton = new JButton(“Search”);

searchButton.setBackground(new Color(255, 140, 0));

searchButton.setForeground(Color.WHITE);

searchButton.addActionListener(e -> performSearch());

// Customize top panel layout

topPanel.add(addFilesButton);

topPanel.add(searchLabel);

topPanel.add(searchField);

topPanel.add(searchButton);

// Create the result area

resultArea = new JTextArea();

resultArea.setEditable(false);

resultArea.setLineWrap(true);

resultArea.setBackground(new Color(255, 255, 255));

resultArea.setFont(new Font(“Monospaced”, Font.PLAIN, 14));

resultArea.setForeground(new Color(50, 50, 50));

JScrollPane resultScrollPane = new JScrollPane(resultArea);

// Add components to the frame

Add(topPanel, BorderLayout.NORTH);

Add(resultScrollPane, BorderLayout.CENTER);

// Add a status bar at the bottom

JLabel statusBar = new JLabel(“Welcome to the Search Engine!”, JLabel.CENTER);

statusBar.setFont(new Font(“Arial”, Font.ITALIC, 12));

statusBar.setBackground(new Color(0, 123, 255));

statusBar.setForeground(Color.WHITE);

statusBar.setOpaque(true);

add(statusBar, BorderLayout.SOUTH);

// Make the frame visible

setVisible(true);

}

Private void indexFilesAndWebPages() {

// Choose files to index

JFileChooser fileChooser = new JFileChooser();

fileChooser.setMultiSelectionEnabled(true);

int returnValue = fileChooser.showOpenDialog(this);

if (returnValue == JFileChooser.APPROVE\_OPTION) {

File[] selectedFiles = fileChooser.getSelectedFiles();

For (File file : selectedFiles) {

Try {

indexFile(file);

} catch (IOException e) {

showError(“Error indexing file: “ + file.getName());

}

}

}

// Option to enter a URL to index a webpage

String url = JOptionPane.showInputDialog(this, “Enter URL to index:”);

If (url != null && !url.isEmpty()) {

If (!url.startsWith([http://](NULL)) && !url.startsWith(<https://)>) {

url = [http://](NULL) + url; // Add the HTTP protocol if missing

}

Try {

crawlAndIndexWebPage(url);

} catch (IOException e) {

showError(“Error indexing web page: “ + url);

}

}

JOptionPane.showMessageDialog(this, “Indexing completed!”);

}

Private void crawlAndIndexWebPage(String url) throws IOException {

// Crawl the given URL and index the page along with all linked pages

If (visitedURLs.contains(url)) return;

visitedURLs.add(url);

Document doc = Jsoup.connect(url).userAgent(“Mozilla/5.0”).timeout(5000).get();

Elements body = doc.body().select(“p”); // Select all paragraph texts

StringBuilder content = new StringBuilder();

Body.forEach(element -> content.append(element.text()).append(“ “));

indexDocument(url, content.toString());

// Find and index linked pages

Elements links = doc.select(“a[href]”); // Select all links in the web page

For (org.jsoup.nodes.Element link : links) {

String linkURL = link.attr(“href”);

// Resolve relative URLs to absolute URLs

If (!linkURL.startsWith(“http”)) {

linkURL = doc.baseUri() + linkURL; // Resolve to absolute URL

}

If (!visitedURLs.contains(linkURL)) {

crawlAndIndexWebPage(linkURL); // Crawl and index linked pages

}

}

}

Private void indexFile(File file) throws IOException {

If (file == null || !file.exists()) {

Throw new FileNotFoundException(“File not found: “ + file);

}

Try (BufferedReader br = new BufferedReader(new FileReader(file))) {

StringBuilder content = new StringBuilder();

String line;

While ((line = br.readLine()) != null) {

Content.append(line).append(“ “);

}

indexDocument(file.getName(), content.toString());

resultArea.setText(“File indexed: “ + file.getName());

}

}

Private void indexDocument(String docId, String content) {

String[] words = content.split(<\\W+>);

For (String word : words) {

Word = word.toLowerCase();

Index.putIfAbsent(word, new HashMap<>());

Map<String, Integer> postings = index.get(word);

Postings.put(docId, postings.getOrDefault(docId, 0) + 1);

}

}

Private void performSearch() {

String query = searchField.getText().toLowerCase().trim();

If (query.isEmpty()) {

resultArea.setText(“Please enter a search term.”);

return;

}

Map<String, Integer> results = index.getOrDefault(query, Collections.emptyMap());

If (results.isEmpty()) {

resultArea.setText(“No results found for: “ + query);

return;

}

// Sort and display results based on frequency

ArrayList<Map.Entry<String, Integer>> sortedResults = new ArrayList<>(results.entrySet());

sortedResults.sort((e1, e2) -> e2.getValue().compareTo(e1.getValue()));

StringBuilder sb = new StringBuilder();

Sb.append(“Results for ‘”).append(query).append(“’:\n\n”);

For (Map.Entry<String, Integer> entry : sortedResults) {

Sb.append(“[CLICK TO OPEN] “).append(entry.getKey()).append(“ (“).append(entry.getValue()).append(“ occurrences)\n”);

}

resultArea.setText(sb.toString());

addClickListenersToResults(sortedResults);

}

Private void addClickListenersToResults(ArrayList<Map.Entry<String, Integer>> sortedResults) {

// Make results clickable by adding mouse listeners

resultArea.addMouseListener(new MouseAdapter() {

@Override

Public void mousePressed(MouseEvent e) {

String resultText = resultArea.getText();

Point clickPoint = e.getPoint();

Int position = resultArea.viewToModel(clickPoint);

Try {

String[] lines = resultText.split(“\n”);

For (String line : lines) {

If (line.contains(“[CLICK TO OPEN]”)) {

String filename = line.substring(15, line.indexOf(“ (“));

File file = new File(filename);

If (file.exists()) {

Desktop.getDesktop().open(file); // Open the file with the default editor

} else if (filename.startsWith(“http”)) {

Desktop.getDesktop().browse(new URI(filename)); // Open URL in browser

}

}

}

} catch (IOException | URISyntaxException ex) {

showError(“Error opening the file or URL.”);

}

}

});

}

Private void showError(String message) {

JOptionPane.showMessageDialog(this, message, “Error”, JOptionPane.ERROR\_MESSAGE);

}

Public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

New SearchEngineApp(); // Run the app

});

}

}