Báo cáo Project 1 - aaaaaaaaaaaa

Project I (Trường Đại học Bách khoa Hà Nội)

ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO BÀI TẬP LỚN
Project I

Đề tài: Xây dựng chương trình cho phép định dạng văn bản

Giáo viên hướng dẫn: Nguyễn Thị Thu Hương

Mã lớp học: 154018

Sinh viên thực hiện: Nguyễn Ngọc Sơn

MSSV : 20225390

Lớp : IT2-01

Hà Nội, tháng 1 năm 2025

MỤC LỤC

Lời nói đầu

Trong thời đại công nghệ phát triển mạnh mẽ, việc xử lý và định dạng văn bản đã trở thành một phần không thể thiếu trong công việc và học tập. Một văn bản được trình bày rõ ràng, thẩm mỹ không chỉ giúp người đọc dễ dàng nắm bắt thông tin mà còn thể hiện sự chuyên nghiệp của người soạn thảo.

Xuất phát từ nhu cầu đó, tôi đã thực hiện chương trình "Định dạng Văn Bản" với mong muốn cung cấp một công cụ hỗ trợ hiệu quả cho việc định dạng tài liệu. Chương trình cho phép người dùng thực hiện các thao tác định dạng cơ bản như căn chỉnh lề, thay đổi phông chữ, cỡ chữ, và các chức năng nâng cao như chèn hình ảnh, bảng biểu, hoặc tự động hóa quy trình xử lý văn bản.

Với mục tiêu đơn giản hóa và tối ưu hóa trải nghiệm của người dùng, chương trình được thiết kế để dễ sử dụng nhưng vẫn đảm bảo hiệu quả và tính ứng dụng thực tế cao. Tôi hy vọng rằng sản phẩm này không chỉ đáp ứng tốt các nhu cầu về định dạng văn bản mà còn mang lại giá trị thiết thực trong công việc và cuộc sống hàng ngày của người dùng.

Trân trọng,

Nguyễn Ngọc Sơn

I. Khảo sát bài toán

1. Mô tả yêu cầu bài toán định dạng văn bản

Một tệp văn bản đầu vào có thể chứa các dòng có độ dài khác nhau, có thể rất dài hoặc rất ngắn, gây khó khăn cho việc đọc và xử lý. Yêu cầu chính của bài toán gồm:

- Định dạng lại văn bản
- Trích xuất các từ khóa từ từng dòng trong tệp văn bản đã được định dạng.
- Lập bảng chỉ dẫn cho các từ khóa, trong đó mỗi từ khóa sẽ kèm theo số thứ tự của các dòng nơi từ khóa đó xuất hiện.

Mục tiêu

- Cải thiện khả năng đọc và xử lý văn bản, đặc biệt là trong các tài liệu dài.
- Phân tích từ khóa trong các tệp văn bản, hỗ trợ tìm kiếm nhanh hoặc xây dựng các công cụ phân tích nội dung.

2. Yêu cầu bài toán

- Định dạng lại văn bản
  - Không có dòng nào trong tệp đầu ra dài hơn một độ dài ký tự cho trước
  - Đặt càng nhiều từ lên một dòng càng tốt mà không phá vỡ từ.
  - Không cho phép dấu chấm, dấu phẩy, hoặc các dấu câu khác xuất hiện ở đầu dòng mới (nếu dòng trước kết thúc bằng dấu câu, dấu đó phải ở cuối dòng).
- Trích xuất từ khoá
  - Trích xuất các từ khóa từ từng dòng trong tệp văn bản đã được định dạng.
  - Từ khóa được định nghĩa là các từ có nghĩa trong văn bản và không nằm trong danh sách **stop words** (các từ không mang ý nghĩa quan trọng như "và", "là", "the", "a"...).
  - Lập bảng chỉ dẫn cho các từ khóa, trong đó mỗi từ khóa sẽ kèm theo số thứ tự của các dòng nơi từ khóa đó xuất hiện.

3. Công nghệ sử dụng

Để thực hiện bài toán định dạng văn bản và trích xuất từ khóa, chúng ta sử dụng các công nghệ sau:

- Ngôn ngữ lập trình: Java
- Thư viện và công cụ : Java Standard Library
- File I/O (java.io, java.nio): Đọc và ghi tệp văn bản.

- String and Regex (java.lang.String, java.util.regex): Xử lý chuỗi, loại bỏ dấu câu, và phân tích văn bản.
- HashMap và HashSet để lưu bảng từ khóa và danh sách stop words.
- ArrayList để quản lý các dòng văn bản

II. Thiết kế chương trình

1.Xử lý đọc và lưu file

1. Hàm readFile(File file)

- **Chức năng:**
  - Đọc toàn bộ nội dung của một tệp văn bản được chỉ định và trả về dưới dạng một chuỗi (String).
  - Nội dung trong tệp được đọc dòng theo dòng và nối lại thành một chuỗi duy nhất.
- **Ứng dụng:**
  - Sử dụng để tải nội dung từ tệp đầu vào (ví dụ: input.txt) vào bộ nhớ, giúp xử lý hoặc phân tích văn bản.
  - Cách hoạt động:
- **Mở tệp bằng BufferedReader để đọc dữ liệu hiệu quả.**
  - Đọc từng dòng và nối với ký tự xuống dòng (\n) để giữ nguyên định dạng gốc của tệp.

```
public static String readFile(File file) throws IOException {  1 usage  sown1812
    try (BufferedReader reader = new BufferedReader(new FileReader(file))) {
        StringBuilder content = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            content.append(line).append("\n");
        }
        return content.toString(); // trả về nội dung của tệp dưới dạng chuỗi
    }
}
```

Code hàm readFile

2. Hàm saveFile(File file, String content)

- **Chức năng:**
  - Ghi nội dung chuỗi (content) vào một tệp văn bản được chỉ định.
  - Nội dung được mã hóa bằng UTF-8, đảm bảo hỗ trợ tốt cho tiếng Việt và các ký tự đặc biệt.
- **Ứng dụng:**
  - Lưu kết quả sau khi xử lý văn bản (ví dụ: tệp văn bản đã định dạng hoặc bảng chỉ dẫn từ khóa) vào tệp đầu ra (ví dụ: formatted_output.txt).

- **Cách hoạt động:**
  - Sử dụng BufferedWriter và OutputStreamWriter để ghi dữ liệu một

cách hiệu quả.

- Ghi toàn bộ nội dung của chuỗi vào tệp, thay thế dữ liệu cũ trong tệp nếu tồn tại.

```
public static void saveFile(File file, String content) throws IOException {  1 usage  sown1812
    try (BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(file), StandardCharsets.UTF_8))) {
        writer.write(content); // ghi nội dung chuỗi vào tệp
    }
}
```

Code hàm saveFile

3. Hàm showFileChooser(Stage stage, boolean isSaveDialog)

- **Chức năng:**
  - Hiển thị hộp thoại để người dùng chọn tệp hoặc lưu tệp, tùy thuộc vào tham số isSaveDialog:
  - Nếu isSaveDialog = true: Hiển thị hộp thoại lưu tệp (Save File Dialog).
  - Nếu isSaveDialog = false: Hiển thị hộp thoại mở tệp (Open File Dialog).
- **Ứng dụng:**
  - Tạo giao diện cho người dùng để dễ dàng mở hoặc lưu tệp khi làm việc với ứng dụng JavaFX.
- **Cách hoạt động:**
  - Sử dụng FileChooser của JavaFX để hiển thị hộp thoại.
  - Bộ lọc tệp (ExtensionFilter) được thêm vào để người dùng chỉ chọn tệp văn bản (*.txt) hoặc hiển thị tất cả các loại tệp (*.*).

```
public static File showFileChooser(Stage stage, boolean isSaveDialog) {  1 usage
    FileChooser fileChooser = new FileChooser();
    fileChooser.getExtensionFilters().addAll( // thêm bộ lọc
            new FileChooser.ExtensionFilter( s: "Text Files", ...strings: "*.txt"),
            new FileChooser.ExtensionFilter( s: "All Files", ...strings: "*.*")
    );
    return isSaveDialog ?
            fileChooser.showSaveDialog(stage)
          : fileChooser.showOpenDialog(stage);
}
```

Code hàm showFileChooser

2.Định dạng văn bản

1.Lấy danh sách các từ Stop words

- **Chức năng:**
    - Tải stop words từ các tệp (trong trường hợp này là stopwords_english.txt và stopwords_vietnamese.txt).
    - Lưu stop words vào một tập hợp Set<String> để dễ dàng tra cứu và lọc trong quá trình xử lý văn bản. mới.
- **Ứng dụng:**
    - Cho phép linh hoạt thay đổi danh sách stop words bằng cách cập nhật các tệp mà không cần chỉnh sửa mã nguồn.
- **Cách hoạt động:**
    - Tạo một biến tĩnh (static) STOP_WORDS để lưu trữ các từ stop words, và giá trị của nó được gán thông qua phương thức loadStopWords().
    - Khởi tạo Set<String> stopWords: Tạo một HashSet để lưu trữ các từ không quan trọng. HashSet đảm bảo không có từ trùng lặp và có hiệu suất cao khi tra cứu.
    - Khai báo một mảng stopWordsFiles chứa tên của các tệp stop words cần tải (stopwords_english.txt và stopwords_vietnamese.txt).
    - Đọc nội dung tệp: Mỗi tệp được mở thông qua getResourceAsStream(fileName), và dữ liệu từ tệp được đọc bằng một BufferedReader.

```
private static final Set<String> STOP_WORDS = loadStopWords();  1 usage

private static Set<String> loadStopWords() {  1 usage  new *
    Set<String> stopWords = new HashSet<>();
    // Danh sách tệp stop words
    String[] stopWordsFiles = {"/stopwords_english.txt"
                             , "/stopwords_vietnamese.txt"};

    for (String fileName : stopWordsFiles) {
        try (InputStream inputStream = TextFormatter.class.getResourceAsStream(fileName);
             BufferedReader reader = new BufferedReader(new InputStreamReader(Objects.requireNonNull(inputStream)))) {
            String line;
            while ((line = reader.readLine()) != null) {
                stopWords.add(line.trim().toLowerCase());
            }
        } catch (IOException | NullPointerException e) {
            throw new RuntimeException("Failed to load stop words from file: " + fileName, e);
        }
    }

    return stopWords;
}
```

Code Stop Words

2.Hàm formatText()

- **Chức năng:**
    - Định dạng văn bản sao cho mỗi dòng không dài quá độ dài tối đa maxLineLength.

    - Đảm bảo các từ không bị ngắt giữa chừng, và dấu câu không nằm ở đầu dòng mới.

- **Ứng dụng:**
    - Định dạng văn bản gọn gàng, dễ đọc mà không phá vỡ từ hay dấu câu.**a**
- **Cách hoạt động:**
    - Khởi tạo danh sách để lưu các dòng đã được format
    - Duyệt qua từng đoạn văn bản trong danh sách originalLines**.**
    - Tách đoạn văn thành các từ.
    - Duyệt qua các từ trong đoạn văn:
        - Đối với mỗi từ trong mảng words, kiểm tra xem độ dài của từ đó có vượt quá maxLineLength hay không.
        - Nếu độ dài của từ lớn hơn maxLineLength:
            - Nếu currentLine (dòng hiện tại) không rỗng, dòng hiện tại sẽ được thêm vào formattedLines và dòng mới sẽ bắt đầu.
            - Sau đó, từ dài này sẽ được thêm trực tiếp vào formattedLines mà không cần tách ra.
            - Quay lại vòng lặp.
    - Kiểm tra độ dài dòng hiện tại:
        - Nếu độ dài của currentLine cộng với độ dài của từ tiếp theo cộng thêm một khoảng trắng sẽ vượt quá maxLineLength, dòng hiện tại sẽ được thêm vào formattedLines và một dòng mới sẽ bắt đầu.
    - Thêm từ vào dòng hiện tại:
        - Nếu currentLine không rỗng, một khoảng trắng sẽ được thêm vào trước khi thêm từ vào currentLine.
    - Cuối mỗi đoạn văn:
        - Nếu có bất kỳ nội dung nào còn lại trong currentLine (tức là dòng chưa được thêm vào danh sách formattedLines), dòng này sẽ được thêm vào danh sách.
    - Kết quả trả về là danh sách các dòng văn bản đã được định dạng**.**

```
public List<String> formatText(List<String> originalLines, int maxLineLength) {  1 usage  sown18
    List<String> formattedLines = new ArrayList<>(); // tạo danh sách lưu văn bản đã format
    for (String paragraph : originalLines) {
        if (paragraph.trim().isEmpty()) continue;

        String[] words = paragraph.split( regex: "\\s+");
        StringBuilder currentLine = new StringBuilder(); // xây dựng từng dòng
        for (String word : words) {
            if (word.length() > maxLineLength) {
                if (!currentLine.isEmpty()) {
                    formattedLines.add(currentLine.toString().trim());
                    currentLine = new StringBuilder();
                }
                formattedLines.add(word);
                continue;
            }
            if (currentLine.length() + word.length() + 1 > maxLineLength) {
                formattedLines.add(currentLine.toString().trim()); // thêm vào formatted line
                currentLine = new StringBuilder();
            }
            if (!currentLine.isEmpty()) {
                currentLine.append(" ");
            }
            currentLine.append(word);
        }

        if (!currentLine.isEmpty()) {
            formattedLines.add(currentLine.toString().trim());
        }
    }
    return formattedLines;
}
```

Code hàm formatText

3.Hàm extractKeywords()

- **Chức năng:**
  - Trích xuất từ khóa có ý nghĩa từ các dòng văn bản đã được định dạng.
  - Loại bỏ các từ thuộc danh sách STOP_WORDS.
- **Cách hoạt động:** Hàm extractKeywords() sử dụng bảng băm (HashMap) để lưu trữ và tra cứu nhanh các từ khóa (keywords) và các dòng mà chúng xuất hiện. Việc sử dụng bảng băm giúp tối ưu hóa quá trình tìm kiếm và quản lý các từ khóa trong văn bản, với độ phức

tạp thấp và hiệu suất cao.

- Khởi tạo một Map để lưu từ khóa và chỉ số dòng:
    - Một Map với khóa là từ khóa (String) và giá trị là một Set<Integer> chứa các số dòng mà từ khóa đó xuất hiện.
- Duyệt qua từng dòng văn bản:
    - Duyệt qua từng dòng văn bản trong formattedLines bằng cách sử dụng vòng lặp for với biến lineNum để giữ chỉ số dòng.
- Duyệt qua các từ trong dòng:
    - Với mỗi từ (word) trong mảng words, hàm sẽ thực hiện các bước sau:
        - Làm sạch từ: Gọi hàm cleanWord(word) để làm sạch từ (có thể loại bỏ các ký tự đặc biệt, chữ hoa thành chữ thường, v.v. tùy vào cách triển khai hàm cleanWord).
        - Kiểm tra từ khóa là từ dừng (stop word): Nếu từ đã làm sạch (cleanWord) có trong tập các từ dừng (STOP_WORDS), thì từ đó sẽ bị bỏ qua bằng cách dùng continue để tiếp tục với từ tiếp theo.
- Thêm từ khóa vào Map:
    - Sử dụng computeIfAbsent để thêm từ vào keywordIndex nếu từ đó chưa có trong bản đồ.
        - Nếu từ chưa tồn tại trong keywordIndex, một TreeSet<Integer> mới sẽ được tạo ra cho từ đó. TreeSet đảm bảo rằng các số dòng sẽ được sắp xếp theo thứ tự tăng dần.
        - Sau đó, chỉ số dòng hiện tại (lineNum + 1 vì các dòng được đánh số bắt đầu từ 1) sẽ được thêm vào TreeSet của từ đó.
    - Trả về kết quả:
        - Sau khi duyệt qua tất cả các dòng và từ, hàm sẽ trả về keywordIndex, một map chứa các từ khóa và các dòng mà từ đó xuất hiện.
    -

Code hàm extractKeyWords()

4. Hàm createKeywordEntries()

- **Chức năng:**
    - Chuyển bảng băm keywordIndex thành danh sách các đối tượng KeywordEntry
    - Mỗi đối tượng KeywordEntry chứa:
    - Từ khoá.
    - Các dòng mà từ khoá xuất hiện
    - Số lần xuất hiện
- **Ứng dụng:**
    - Tạo danh sách hiển thị từ khóa dưới dạng có cấu trúc, dễ sử dụng trong giao diện

JavaFX.

- **Cách hoạt động:**
    - Duyệt qua từng mục trong keywordIndex.
    - Chuyển đổi các chỉ số dòng thành chuỗi
    - Đếm số lần xuất hiện của từ khóa:
        - entry.getValue().size() trả về số lượng dòng mà từ khóa xuất hiện, tức là số lần từ khóa xuất hiện trong văn bản.
    - Thêm một KeywordEntry vào ObservableList:
        - Tạo một đối tượng KeywordEntry mới với ba tham số:
            - entry.getKey(): Từ khóa (chuỗi).
            - lines: Chuỗi các dòng mà từ khóa xuất hiện.
            - occurrences: Số lần xuất hiện của từ khóa.
    - Trả về ObservableList

```
public ObservableList<KeywordEntry> createKeywordEntries(Map<String, Set<Integer>> keywordIndex) {
    ObservableList<KeywordEntry> keywordEntries = FXCollections.observableArrayList();
    for (Map.Entry<String, Set<Integer>> entry : keywordIndex.entrySet()) {
        String lines = entry.getValue().stream() Stream<Integer>
                .map(String::valueOf) Stream<String>
                .collect(Collectors.joining( delimiter: ", "));
        // Đếm số lần xuất hiện
        int occurrences = entry.getValue().size();
        keywordEntries.add(new KeywordEntry(entry.getKey(), lines, occurrences));
    }
    return keywordEntries;
}
```

Code hàm KeywordEntries

5. Hàm cleanWord()

- **Chức năng:**
    - Làm sạch từ:
    - Chuyển thành chữ thường
    - Loại bỏ các ký tự không phải chữ cái hoặc số.
- **Ứng dụng**:
    - Chuẩn hoá từ trước khi kiểm tra hoặc xử lý.
- **Cách hoạt động:**
    - Chuyển từ thành chữ thường: dung hàm toLowerCase():
    - Loại bỏ các ký tự không phải chữ cái hoặc số:
    - replaceAll("[^\\p{L}\\p{N}]", "")

- Sử dụng biểu thức chính quy (regex) để thay thế các ký tự không cần thiết bằng chuỗi rỗng (""), nghĩa là loại bỏ chúng.
- [^...]: Ký hiệu phủ định, chọn các ký tự không nằm trong tập hợp.
- \p{L}: Đại diện cho tất cả các ký tự chữ cái (cả chữ cái tiếng Anh và ký tự Unicode khác như tiếng Việt).
- \p{N}: Đại diện cho tất cả các ký tự số (từ 0-9).

```
private String cleanWord(String word) {  1 usage  sown1812
    return word.toLowerCase().replaceAll( regex: "[^\\p{L}\\p{N}]",  replacement: "");
}
```

Code hàm cleanWord

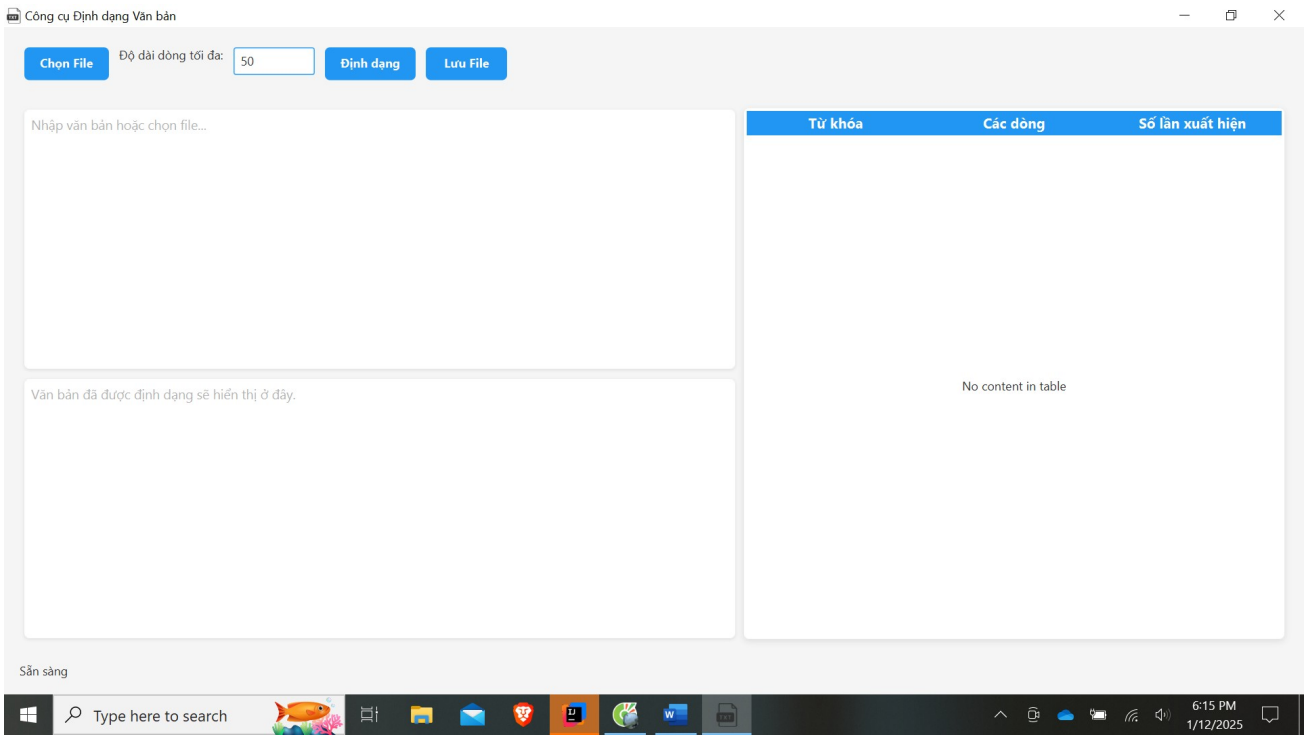
III. Xây dựng giao diện cho chương trình

3.1. Thư viện và công cụ sử dụng

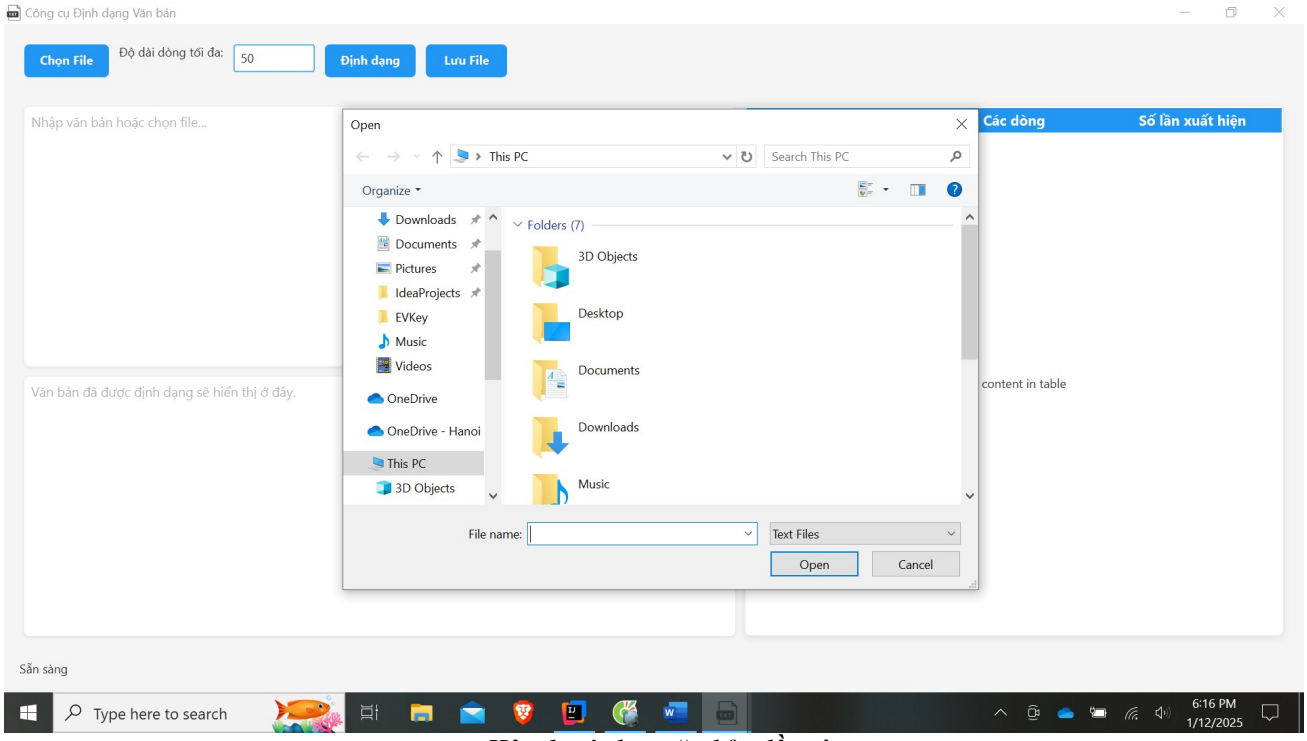
Danh sách thư viện và công cụ sử dụng :

- IDE lập trình: IntelliJ IDEA 2024.3.1.1
- Môi trường và ngôn ngữ lập trình : JDK 23 và JavaFx

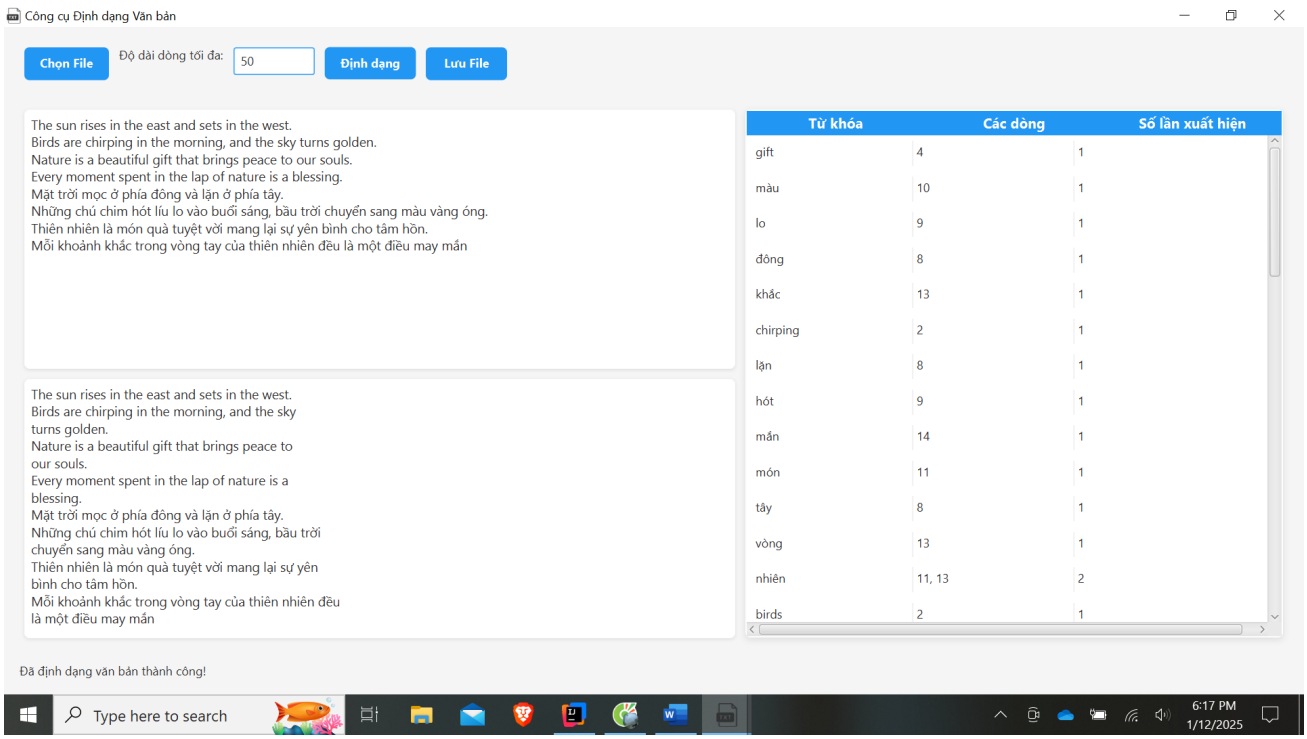
3.2. Giao diện minh hoạ các chức năng của chương trình



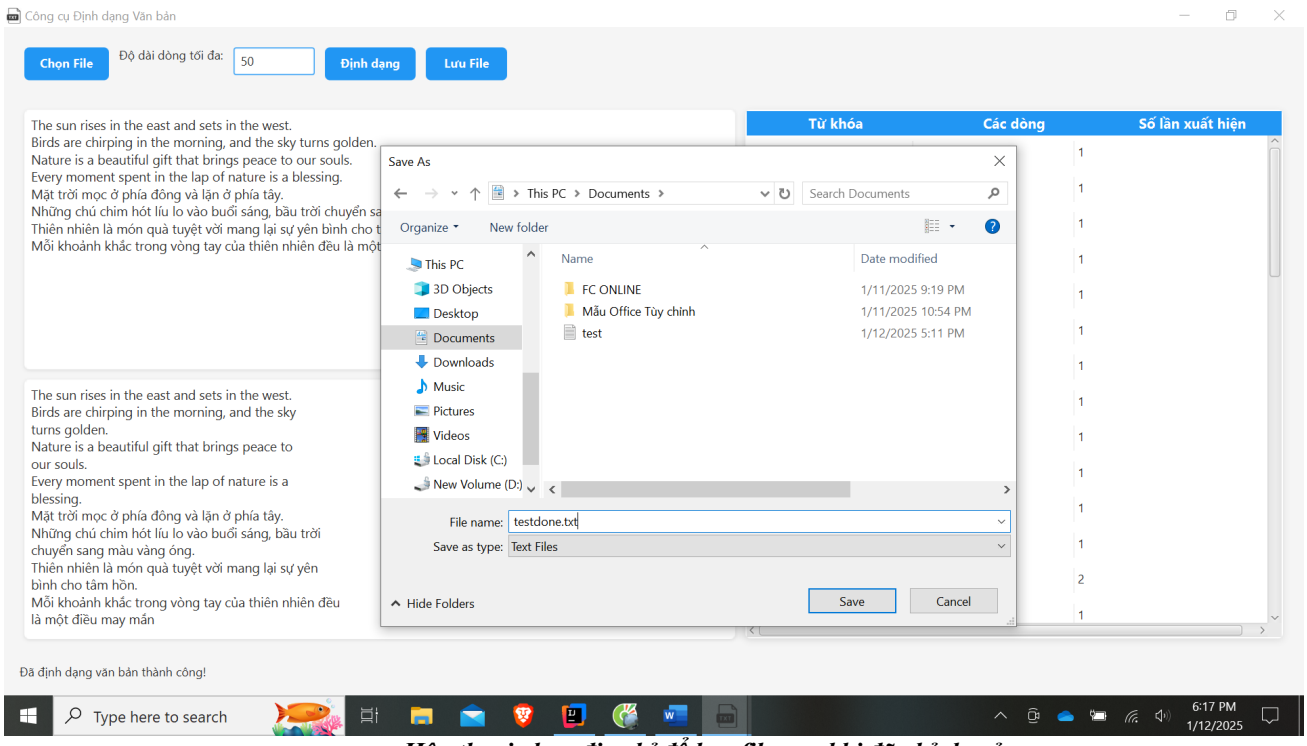
Giao diện sử dụng chương trình



Hộp thoại chọn văn bản đầu vào



Văn bản sau khi đã được chỉnh sửa



Hộp thoại chọn địa chỉ để lưu file sau khi đã chỉnh sửa

IV. Đánh giá và hướng phát triển

4.1. Ưu điểm

- **Tính năng đầy đủ và thiết thực**
    - Cho phép nhập văn bản trực tiếp hoặc từ file
    - Có thể định dạng văn bản theo độ dài dòng tùy chỉnh
    - Phân tích và thống kê từ khóa với số lần xuất hiện và vị trí
    - Hỗ trợ lưu kết quả ra file
- **Xử lý lỗi tốt:**
    - Có hệ thống thông báo lỗi đầy đủ thông qua AlertUtils
    - Kiểm tra các trường hợp đầu vào không hợp lệ
    - Xử lý ngoại lệ khi đọc/ghi file
- **Giao diện người dùng thân thiện:**
    - Layout rõ ràng, dễ sử dụng
    - Có thông báo trạng thái thông qua statusLabel
- **Khả năng mở rộng tốt:**
    - Dễ dàng thêm tính năng mới nhờ cấu trúc module hóa
    - Hỗ trợ nhiều ngôn ngữ với stopwords

4.2. Nhược điểm

- **Hiệu năng có thể cải thiện:**
    - Việc tách từ và xử lý văn bản được thực hiện tuần tự
    - Có thể tối ưu hóa bằng cách sử dụng parallel streams cho các file lớn
    - Việc load stopwords có thể được cache tốt hơn
- **Giới hạn về tính năng:**
    - Chưa có tùy chọn cho các định dạng văn bản khác nhau (như căn lề, font chữ)
    - Chưa có tính năng tìm kiếm trong văn bản
    - Chưa hỗ trợ undo/redo cho các thao tác định dạng
- **Giao diện người dùng chưa được tối ưu:**
    - Chưa có dark mode hoặc các theme tùy chọn

- Chưa hỗ trợ responsive design cho các kích thước màn hình khác nhau
- Thiếu shortcuts cho các thao tác thường xuyên

4.3. Hướng phát triển

- **: Cải thiện Core Features**
  - Thêm tính năng undo/redo cho các thao tác định dạng
  - Tích hợp spell check và grammar check
  - Bổ sung tính năng tìm kiếm và thay thế văn bản
  - Thêm các tùy chọn định dạng văn bản nâng cao (căn lề, font chữ, khoảng cách dòng)
  - Cho phép người dùng tùy chỉnh và quản lý danh sách stopwords
- **Nâng cao hiệu năng:**
  - Triển khai parallel processing cho xử lý file lớn
  - Tối ưu hóa thuật toán phân tích từ khóa
  - Thêm caching cho stopwords và các tài nguyên thường xuyên sử dụng
  - Sử dụng lazy loading cho các components không cần thiết ngay lập tức
- **Cải thiện UX/UI:**
  - Thêm phím tắt cho các thao tác phổ biến
  - Tích hợp drag-and-drop cho file
- **Mở rộng tích hợp:**
  - Hỗ trợ nhiều định dạng file (PDF, DOC, DOCX)
  - Tích hợp với lưu trữ đám mây (Google Drive, Dropbox)
  - Thêm tính năng xuất sang các định dạng khác nhau

PHỤ LỤC

[1] Source code: https://github.com/sown1812/Project-I