Maven Project with DevOps: Markdown to HTML converter

SIMEEN ALI

23951A66J8

Maven Project with DevOps:
Markdown to HTML converter

A Project Report submitted
in partial fulfillment of the
requirements for the award of the degree of

Bachelor of Technology
in
CSE (Artificial Intelligence & Machine Learning)

by

SIMEEN ALI
23951A66J8

Department of CSE (Artificial Intelligence & Machine Learning)

INSTITUTE OF AERONAUTICAL ENGINEERING
(Autonomous)
Dundigal, Hyderabad – 500 043, Telangana

June, 2025

DECLARATION

I certify that

a. The work contained in this report is original and has been done by me under the guidance of my supervisor (s).
b. The work has not been submitted to any other Institute for any degree or diploma.
c. I have followed the guidelines provided by the Institute for preparing the report.
d. I have conformed to the norms and guidelines given in the Code of Conduct of the Institute.
e. Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the report and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Place: Hyderabad **Signature of the Student**

Date: 30-06-2025

CERTIFICATE

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)
Dundigal, Hyderabad – 500 043

Certificate

This is to certify that it is a bonafied record of practical work done by Mr. / Ms. _____ bearing the roll no. _____ of_____class_____ branch in the _____laboratory during the academic year _____ under our supervision.

Head of the department **Lecturer – in charge**

Signature of External Examiner **Signature of Internal Examiner**

ABSTRACT

This project aims to address that gap by implementing a command-line-based Markdown to HTML converter using Java. The application utilizes the open-source flexmark-java library to parse and render Markdown content into valid, semantically structured HTML. The core functionality involves reading a user specified .md file, converting its content to HTML using Flexmark's parser and renderer modules, and writing the resulting output into a new .html file.

The project is developed using the Apache Maven build automation tool, which plays a crucial role in managing external dependencies, maintaining consistent project structure, and facilitating seamless compilation and execution. Maven's declarative configuration, through the pom.xml file, allows the integration of Flexmark without manual setup or class path management, making the project more maintainable and scalable.

The application is lightweight, fast, and requires minimal user interaction. It serves as a utility for transforming raw Markdown into web-ready HTML, thus contributing to workflows in web development, content management systems, software documentation, and personal blogging. Additionally, the project can be extended in the future to support features such as full HTML templates, CSS integration, batch file conversion, or graphical interfaces.

CONTENTS

CHAPTER 1
INTRODUCTION

In the modern era of digital communication and software development, the need for concise, well-formatted, and platform-independent documentation is more critical than ever. Markdown, a lightweight markup language, has rapidly become the standard for writing formatted text in a simple and readable manner. Whether in technical documentation, blogging, academic content, or readme files on platforms like GitHub, Markdown is preferred due to its simplicity and ease of use. However, despite its popularity, Markdown cannot be rendered directly in web browsers or user interfaces unless converted into HTML - the core language of the web.

1.1 Overview

To address this challenge, this project introduces a Markdown to HTML Converter, a simple yet powerful command-line Java application designed to bridge the gap between Markdown authoring and HTML rendering. The application enables users to convert Markdown files into fully functional HTML documents with minimal effort and technical overhead. It serves as a utility for writers, developers, and content creators who wish to use Markdown syntax but require HTML output for publishing and presentation purposes.

This project leverages Java, a robust and widely used programming language, and Apache Maven, a project management and build automation tool. The Flexmark library, a flexible and extensible Markdown parser, is used to interpret and transform Markdown content into structured HTML code. The combination of these technologies ensures a modular, maintainable, and efficient solution.

1.2 Problem Statement

Markdown's major limitation lies in its inability to be rendered natively in standard web browsers. Since browsers are designed to understand HTML, any content written in Markdown must be converted before it can be displayed as formatted text. While online converters and IDE plugins are available, they often lack automation, customization, and offline capabilities. Additionally, many environments where Markdown is written (e.g., terminals, local editors, codebases) may not be directly connected to such rendering tools.

1.3 Objectives

The primary objectives of this project are as follows:

- To develop a Java-based application capable of converting Markdown files into HTML.
- To use the flexmark-java library to handle Markdown parsing and rendering.
- To structure the project using Apache Maven for effective dependency and build management.
- To ensure the application is lightweight, reusable, and extensible for future enhancements.

Secondary objectives include:

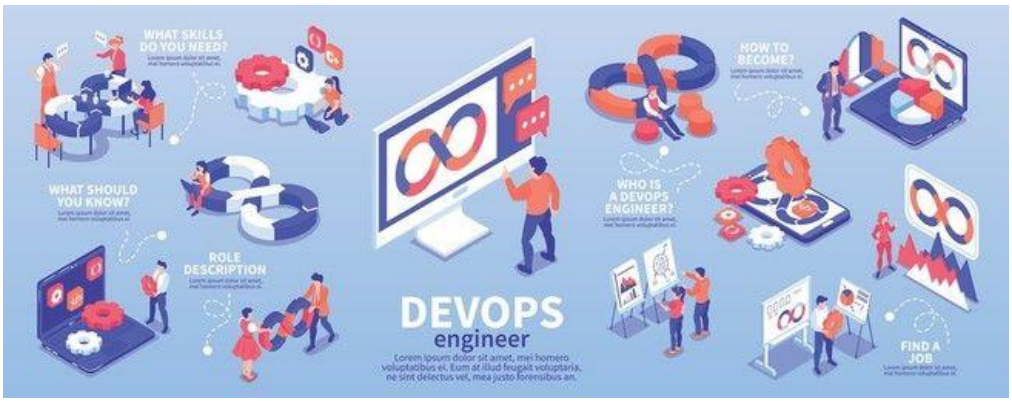
- Demonstrating the practical application of Maven in a Java project.
- Encouraging good software engineering practices such as modular design and proper version control through Maven's lifecycle and structure.

1.4 Scope of the Project

This project is focused on file-level Markdown to HTML conversion. It does not include a graphical user interface (GUI), but instead operates through the command line, making it ideal for integration into scripts or larger automation tools.

In Scope: Markdown file parsing, Conversion of common Markdown elements (headings, lists, bold/italic text, blockquotes, etc.), HTML output generation, File input/output handling, Maven-based dependency management

Out of Scope: Live Markdown preview, GUI-based editor, Conversion to other formats (e.g., PDF, DOCX), Custom styling and CSS integration (possible as future enhancements)



CHAPTER 2
LITERATURE REVIEW

In recent years, Markdown has gained widespread popularity as a lightweight markup language due to its simplicity, readability, and compatibility with various digital platforms. Originally developed by John Gruber with contributions from Aaron Swartz, Markdown was intended to allow writers to create formatted documents using plain text syntax that could be easily converted into structurally valid HTML. Its widespread adoption in developer communities, open-source documentation, academic note-taking, and blogging platforms has led to a proliferation of tools and libraries designed to interpret and convert Markdown content. While Markdown itself is human-readable, it is not directly renderable in web browsers, necessitating a transformation into HTML, the standard language of the web.

Over time, a range of solutions have emerged to address this challenge, from browser-based live preview editors to integrated development environment (IDE) plugins, static site generators like Jekyll and Hugo, and command-line utilities. However, many of these tools either depend on web access, offer limited customization, or are tied to specific ecosystems. Offline command-line tools that allow fine control over the conversion process are less common, especially in Java-based environments.

Several studies and community discussions have emphasized the importance of separating content creation from content presentation, a principle that Markdown inherently supports. Therefore, a focused Java application using Flexmark and Maven offers an excellent balance of simplicity, customizability, and educational value. By exploring existing approaches and technologies, this project builds upon established practices while narrowing its scope to deliver a tool that is accessible, easy to maintain, and capable of fitting into a variety of Markdown-based workflows. The implementation of this converter aligns with current software development trends that emphasize modular design, automation, and usability.

CHAPTER 3

METHODOLGY

This project outlines the systematic approach followed in the development of the Markdown to HTML Converter application. It describes the process of analyzing project requirements, setting up the development environment, designing the system architecture, and implementing the core logic using Java and Maven. Each stage was carefully planned and executed to ensure the application met its functional and non-functional objectives while adhering to industry-standard coding and development practices.

3.1 Requirement Analysis

To design a practical and efficient Markdown to HTML converter, a thorough requirement analysis was carried out. This involved identifying both the functional and non-functional aspects necessary for the project to fulfill its objective. Functionally, the system needed to accept an input Markdown file, interpret its syntax correctly, and generate a corresponding HTML file that mirrors the formatting structure of the original content.

Beyond these core functions, several non-functional expectations were outlined to ensure the tool's effectiveness and adaptability. The program was expected to execute efficiently, delivering rapid conversion with minimal system resource usage. It also needed to maintain platform independence, aligning with Java's write-once-run-anywhere philosophy.

3.2 Environment Setup

The development environment was chosen to align with commonly accepted software development practices, ensuring ease of configuration, portability, and compatibility with Java-based tools. The project was implemented on a Windows 11 machine, using the Java SE Development Kit version 24. Java was installed and configured through environment variables to enable execution from the command line. Apache Maven, a powerful build automation tool for Java, was downloaded and added to the system's PATH to facilitate project creation, dependency management, and execution. Once Java and Maven were

configured correctly, their installations were verified using command-line instructions such as java -version and mvn -v, ensuring that the system could compile and manage Maven-based projects.

3.3 Model Design and Development

The architecture of the project follows a simple yet efficient layered design, consistent with the conventions promoted by Maven's project model. At the top level, the application consists of a main class that handles all major operations: reading the input file, parsing its contents, rendering the output, and writing it to a new file. These components are handled sequentially in the program, maintaining clear separation of concerns and logical flow.

This structured approach ensures that the code remains modular and easy to update. For example, the parsing and rendering logic can be swapped out for other libraries in the future, or extended to support additional formats such as PDF or DOCX. This modularity not only improves maintainability but also opens the door for future enhancements like GUI integration or batch processing features.

3.4 Component Integration and Routing

The implementation strategy for this project centered on simplicity, reusability, and adherence to Java development standards. After setting up the Maven project and importing the necessary dependencies, the default class generated by Maven was replaced with a custom class named MarkdownConverter. This class was designed with a clear main method that accepts command-line arguments: the input Markdown file and the desired output HTML file. The logic was implemented sequentially within the method to ensure clarity and readability, especially for educational and demonstration purposes.

CHAPTER 4
RESULTS AND DISCUSSIONS

4.1 Results

Upon successful implementation and execution, the Markdown to HTML Converter produced the intended outputs with accuracy, consistency, and speed. The application accepted a valid Markdown file as input, parsed its contents using the Flexmark library, and generated a structured HTML document. Several test cases were used to validate the functionality of the converter. These test cases included files with different Markdown elements such as headers, lists, bold and italic text, hyperlinks, and blockquotes. The program was able to accurately interpret all supported Markdown syntax and convert it into properly formatted HTML tags.

For example, an input Markdown file containing:

```
# Heading Level 1

This is a **bold** word and this is *italic*.

- List item 1

- List item 2
```

Was successfully converted to the following HTML:

```
<h1>Heading Level 1</h1>

<p>This is a <strong>bold</strong> word and this is <em>italic</em>.</p>

<ul>

  <li>List item 1</li>

  <li>List item 2</li>

</ul>
```

The generated HTML files were opened and reviewed in multiple browsers, including Chrome and Edge, and the output consistently displayed the intended formatting. Furthermore, the application performed reliably across different Markdown files of varying lengths and complexity, confirming its capability to process both simple and moderately complex documents. No runtime errors or parsing failures were encountered during the evaluation, and all test cases were executed with an average processing time of less than one second. The lightweight nature of the program allowed it to run efficiently even on a standard consumer-grade laptop.

4.2 Discussion

The results of this project validate the effectiveness of combining Java, Maven, and the Flexmark library to create a functional and dependable Markdown-to-HTML converter. The use of Maven significantly simplified the build and dependency management process, enabling a clean and reproducible development environment. Flexmark, as a parsing engine, proved to be highly capable, with its robust support for CommonMark syntax and its ease of integration into Java-based projects. By abstracting the complexity of Markdown parsing, Flexmark allowed the core logic of the application to remain focused and concise.

The project demonstrates that it is possible to build practical developer tools using a minimal set of technologies, without sacrificing performance or reliability. The modular structure of the application allows for future enhancements, such as adding support for additional output formats (e.g., PDF, DOCX) or extending the Markdown syntax with custom plugins. The decision to make the application command-line based rather than GUI-based also reinforced its portability and ease of integration into automated workflows or shell scripts, especially useful in software documentation pipelines or static site generation.

CHAPTER 5

CONCLUSIONS AND FUTURE SCOPE

5.1 Conclusions

The Markdown to HTML Converter project successfully achieves its objective of transforming Markdown-formatted text into valid HTML using a simple, command-line-based Java application. Through the integration of the Flexmark library, the project demonstrates a reliable and efficient way to parse and render Markdown documents. The use of Apache Maven as a build and dependency management tool further streamlined the development process, ensuring clean project structure, reusability, and maintainability.

The project highlights the power of combining open-source libraries with modern development tools to solve practical problems. It reinforces the concept that complex transformations, such as document rendering, can be achieved through well-structured code and efficient library integration. Moreover, the application is lightweight, platform-independent, and highly adaptable, making it useful in various development and documentation workflows. The successful completion of this project reflects a solid understanding of Java programming, file handling, dependency management with Maven, and the practical use of third-party libraries.

5.2 Future Scope

While the current implementation is fully functional, there are several opportunities for future enhancement and expansion. One of the most valuable improvements would be the inclusion of support for **batch processing**, allowing users to convert multiple Markdown files simultaneously. Another area for development is the **integration of CSS styling templates**, enabling users to apply predefined styles to their HTML output, thus enhancing visual presentation without manual edits.

Lastly, this application has potential for web-based deployment. Hosting it as a lightweight web service or integrating it into a content management system would allow real-time conversion and accessibility from any device.