SOFTWARE ENGINEERING LAB

PROJECT REPORT

On

HTML Document Editor

Submitted By:

Ranjit Singh (120CS0224)

Megha Sinha (120CS0226)

Suraj Kumar Swain (120CS0229)

Submitted To:

Dr. Puneet Kumar Jain

Department of Computer Science and Engineering

Department of Computer Science and Engineering

NATIONAL INSTITUTE OF TECHNOLOGY

ROURKELA

APRIL, 2023

ABSTRACT

The HTML Document Editor is a software application that provides a range of features and tools to help users create, edit, and format web pages using HTML code. HTML is the standard markup language used to create web pages, and the editor simplifies the process of writing code by providing an intuitive interface that is easy to navigate.

One of the key features of the HTML Document Editor is real-time previewing. This means that as the user writes HTML code, they can see the changes reflected in real-time in a preview window. This feature helps users to visualize their web pages as they create them and make changes quickly and efficiently.

Usability, flexibility, and responsiveness are important design principles of the HTML Document Editor. The user interface is designed to be user-friendly and easy to navigate, regardless of the user's skill level. The editor is flexible, allowing users to customize the interface and configure settings to their preferences. Additionally, the editor is responsive, meaning that it can be used on a range of devices and screen sizes without compromising functionality.

Overall, the HTML Document Editor is a powerful tool for anyone who wants to create professional-looking web pages using HTML code. Its range of features and tools streamline the coding process, enhance the coding experience, and help users produce error-free code, ultimately leading to better, more polished web pages.

TABLE OF CONTENTS

List of Figures

1. LIST OF ABBREVIATIONS

HTML	Hyper Text Markup
Language	
CSS	Cascading Style Sheets
JS	JavaScript
DOM	Document Object Model
WYSIWYG	What You See Is What You Get
FTP	File Transfer Protocol
IDE	Integrated Development Environment
API	Application Programming Interface
URL	Uniform Resource Locator
HTTP	Hypertext Transfer Protocol
XML	Extensible Markup Language
UTF-8	Unicode Transformation Format 8-bit
SQL	Structured Query Language.

2. INTRODUCTION

An HTML document editor is a software application used to create, edit, and manage HTML documents. HTML (Hypertext Markup Language) is the standard markup language used to create web pages and web applications. HTML documents are the foundation of the World Wide Web, and they are used to structure content, define layouts, and add multimedia elements to web pages.
The purpose of an HTML document editor is to provide a user-friendly interface that allows users to create and edit HTML documents efficiently. The editor should have a wide range of features that make it easy to create and format text, add images and multimedia elements, and organize the layout of the web page.
In summary, an HTML document editor is an essential tool for web developers and designers as it enables them to create and modify HTML documents easily. It provides a user-friendly interface and a wide range of features that make HTML document creation and editing efficient and straightforward.

2.1. Background

HTML editors emerged as a response to the growing need for a more efficient way of creating and editing HTML documents. In the early days of the World Wide Web, web pages were created by writing HTML code using a basic text editor. This approach was tedious and error-prone, especially for complex web pages.
As the web grew in popularity, the demand for tools that could simplify the process of creating and editing web pages increased.
The first HTML editors were simple text editors with basic HTML support, such as Notepad and vi. These editors allowed users to write HTML code, but they lacked features like syntax highlighting and auto-completion.

Later editors were developed having a more user-friendly interface and a wide range of features such as syntax highlighting, error checking, and auto-completion.

2.2. Motivation

The motivation for creating an HTML editor is to simplify the process of creating and editing HTML documents. HTML is the standard markup language used to create web pages, and it can be challenging to write and maintain error-free HTML code using a basic text editor. An

HTML editor provides a user-friendly interface that simplifies the process of creating and editing HTML documents, making it easier for non-technical users to create web pages.
HTML editors provide a range of features that help users to write error-free HTML code, such as syntax highlighting, auto-completion, and error checking. These features reduce the learning curve for beginners and make it easier for experienced web developers to create and edit HTML documents efficiently.
Another motivation for HTML editors is to provide a WYSIWYG editing interface that allows users to create web pages without writing HTML code. This is particularly useful for non-technical users who may not be familiar with HTML code but still need to create web pages.

2.3. Objectives

The main objectives of HTML editors are:

1. Simplify HTML document creation and editing: HTML editors provide a user-friendly interface that simplifies the process of creating and editing HTML documents, reducing the learning curve for beginners and making it easier for experienced web developers to create and edit HTML documents efficiently.
2. Ensure error-free HTML code: HTML editors provide a range of features that help users to write error-free HTML code, such as syntax highlighting, auto-completion, and error checking. These features ensure that HTML code is correct and free of errors, making it easier to maintain and update web pages.
3. Provide a WYSIWYG editing interface: HTML editors provide a WYSIWYG editing interface that allows users to create web pages without writing HTML code. This is particularly useful for non-technical users who may not be familiar with HTML code but still need to create web pages.

2.4. Problem statement

The problem statement is to develop a HTML Document Editor software ,which should help create and manipulate HTML files using a GUI.

2.5. Scope of Project

The scope of an HTML document editor includes the following:

Creating and editing HTML documents: The HTML editor should provide a user-friendly interface for creating and editing HTML documents, with features such as syntax highlighting, auto-completion, and error checking.

Code reusability: The editor should allow users to save commonly used HTML snippets as templates, which can be reused in different web pages, saving time and effort and ensuring consistency across web pages.

WYSIWYG editing interface: The editor should provide a WYSIWYG editing interface that allows users to create web pages without writing HTML code.

Collaboration: Online HTML editors should enable collaboration among team members working on a web project, allowing multiple users to work on the same HTML document simultaneously.

Advanced HTML features: The editor should support advanced HTML features such as cascading style sheets (CSS), JavaScript, and multimedia elements, making it easier to create visually appealing and interactive web pages.

Error-free HTML code: The editor should ensure that HTML code is correct and free of errors, making it easier to maintain and update web pages.

Productivity: The editor should increase productivity by providing features that simplify HTML document creation and editing, such as code snippets, auto-complete, and error checking.

In summary, the scope of an HTML document editor includes creating and editing HTML documents, enabling code reusability, providing a WYSIWYG editing interface, facilitating collaboration, supporting advanced HTML features, ensuring error-free HTML code, and increasing productivity.

3. REQUIREMENT ANALYSIS

Requirement analysis for an HTML document editor involves identifying the functional and non-functional requirements that the editor should meet. The following are some of the requirements that an HTML document editor should meet:

Functional Requirements:

Text editing features: The editor should provide text editing features such as cut, copy, paste, undo, and redo.

Syntax highlighting: The editor should highlight HTML tags and attributes to improve readability and make it easier to identify errors.

Auto-completion: The editor should provide auto-completion for commonly used HTML tags and attributes, saving time and effort when typing.

Code snippets: The editor should allow users to save commonly used HTML snippets as templates, which can be reused in different web pages.

Error checking: The editor should check for errors in the HTML code and provide suggestions for correcting them.

WYSIWYG editing interface: The editor should provide a WYSIWYG editing interface that allows users to create web pages without writing HTML code.

Collaboration: The editor should allow multiple users to work on the same HTML document simultaneously and enable version control to track changes made by different users.

Advanced HTML features: The editor should support advanced HTML features such as cascading style sheets (CSS), JavaScript, and multimedia elements.

Non-functional Requirements:

User interface: The editor should have an intuitive and user-friendly interface that is easy to navigate.

Performance: The editor should be fast and responsive, with minimal lag when opening, saving, and editing HTML documents.

Compatibility: The editor should be compatible with different operating systems and web browsers.

Security: The editor should be secure and protect users' data from unauthorized access.

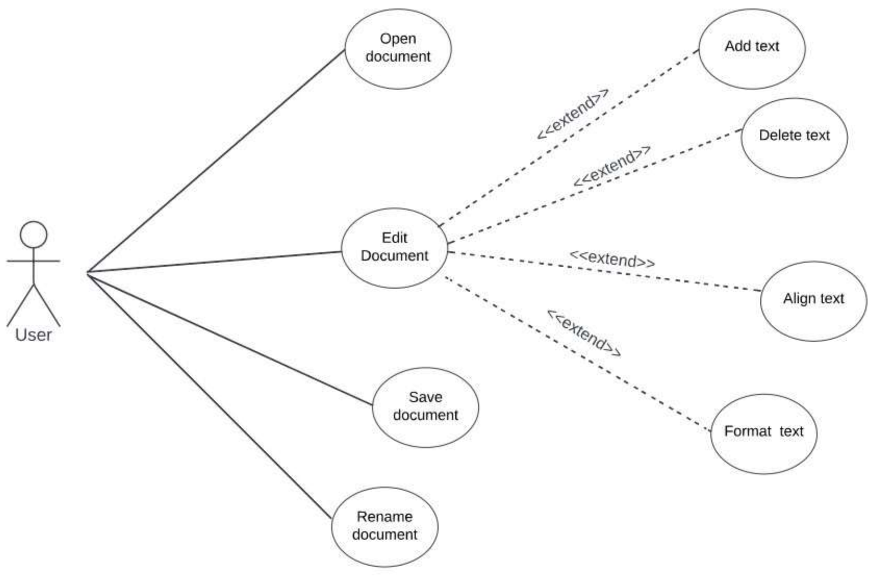
Accessibility: The editor should be accessible to users with disabilities and meet accessibility standards.

Documentation: The editor should provide clear and comprehensive documentation to help users get started and troubleshoot issues.

In summary, an HTML document editor should meet functional requirements such as text editing features, syntax highlighting, auto-completion, code snippets, error checking, WYSIWYG editing interface, collaboration, and support for advanced HTML features, as well as non-functional requirements such as user interface, performance, compatibility, security, accessibility, and documentation.

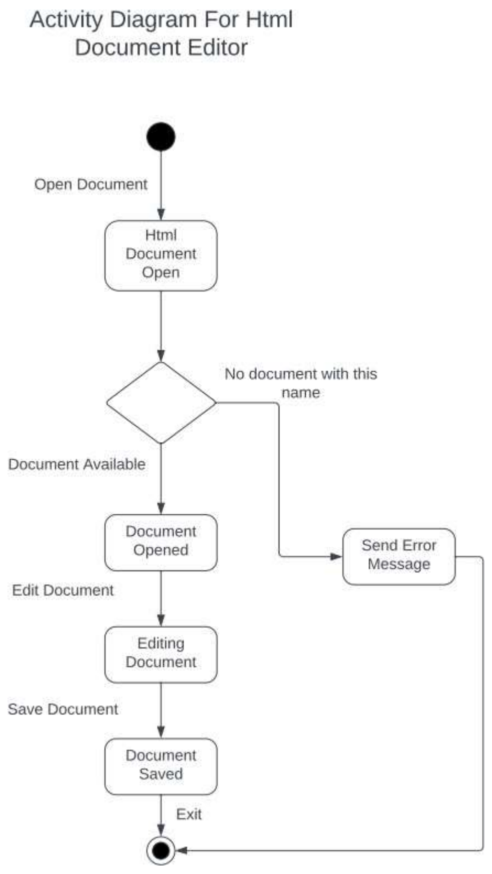
4. UML DIAGRAMS

4.1. Use Case Diagram

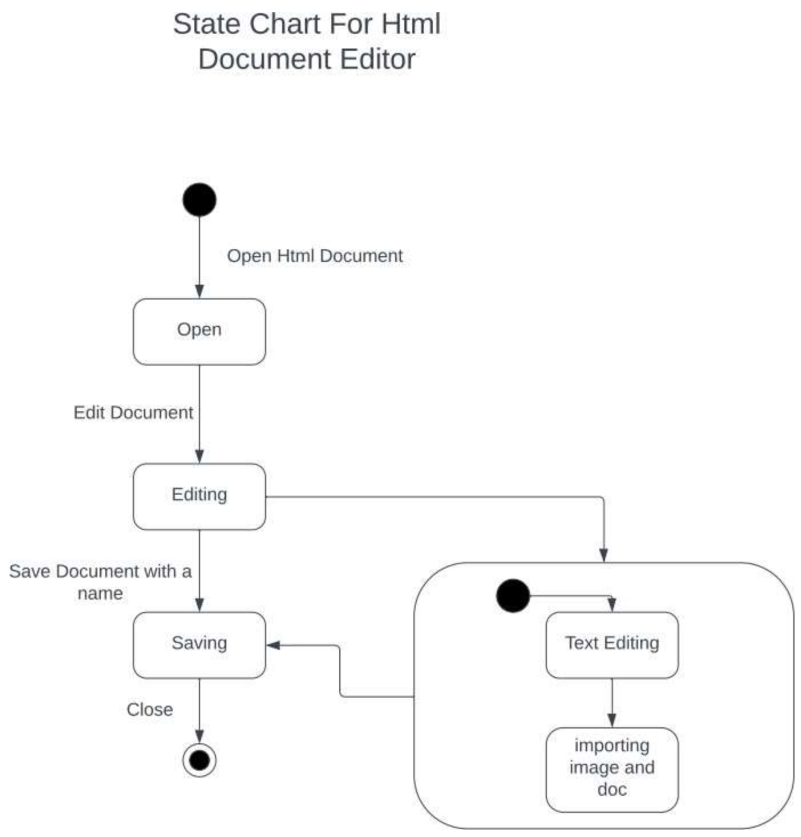


USE CASE DIAGRAM

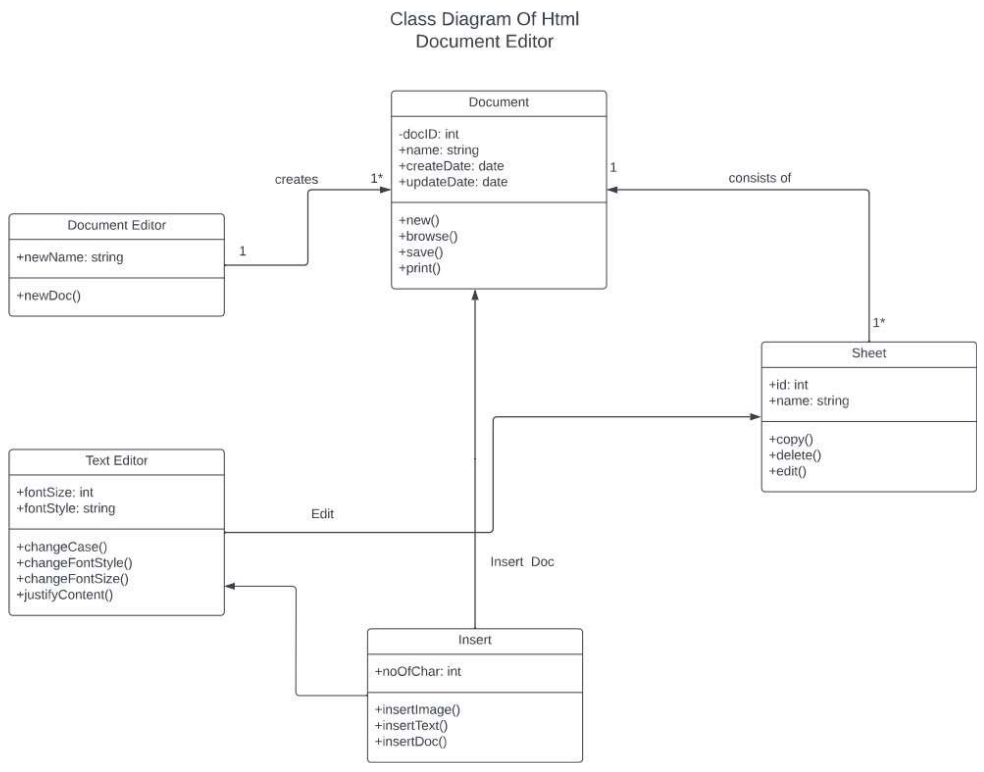
4.2. Activity Diagram



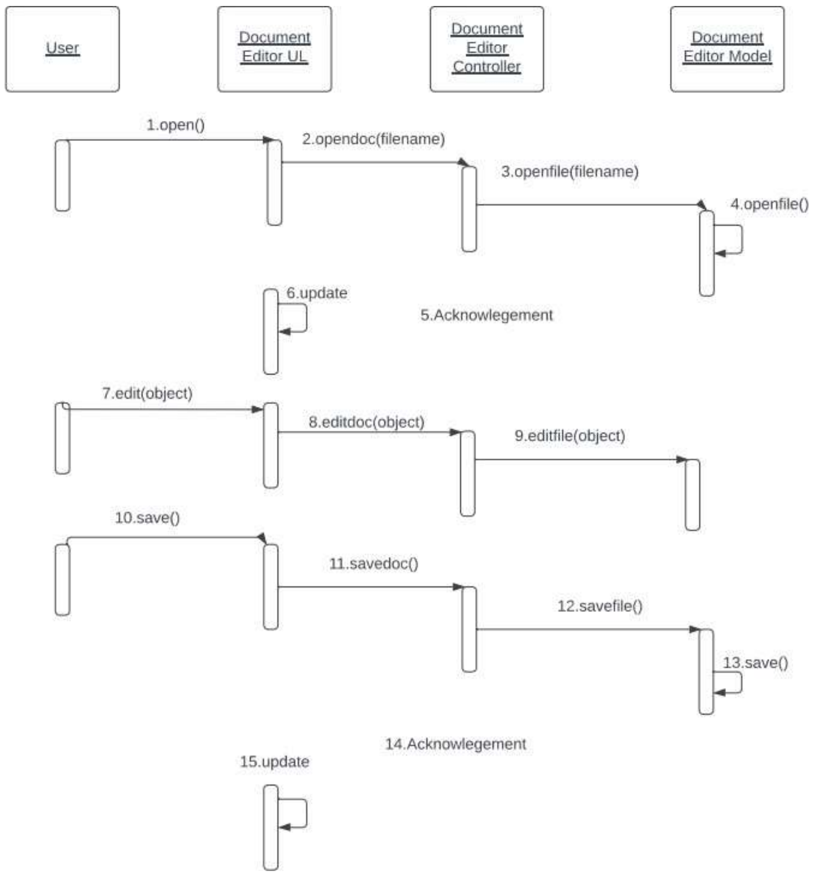
4.3. State Diagram



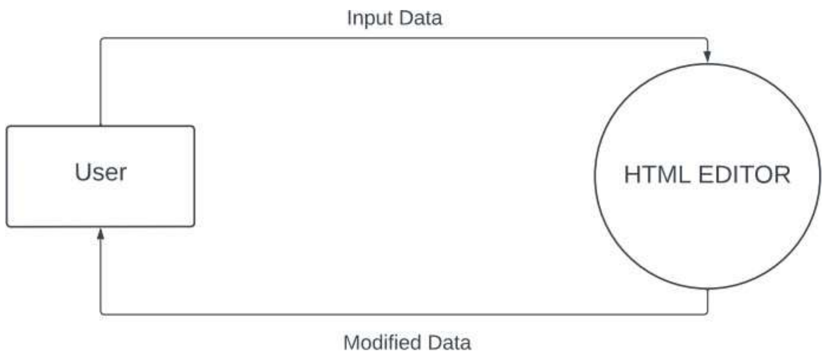
4.4. Class Diagram



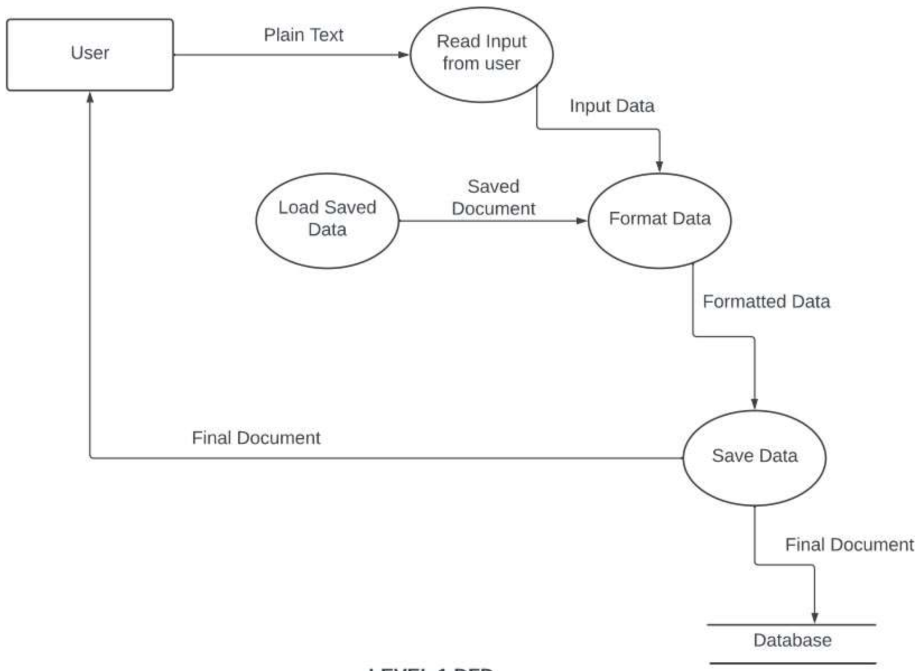
4.5. Sequence Diagram



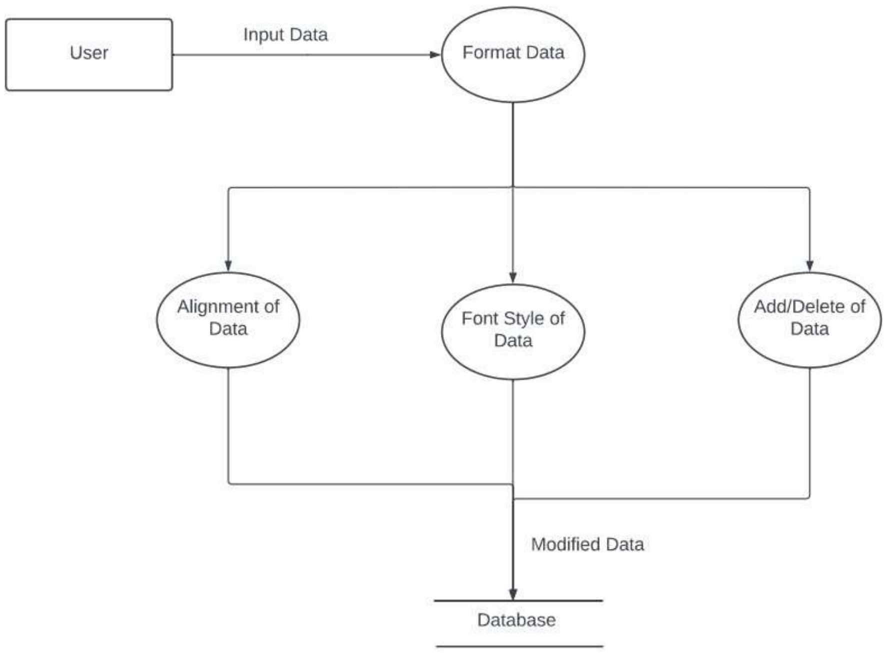
4.6. Data Flow Diagram



0 LEVEL DFD



LEVEL 1 DFD



Formatting of Document

LEVEL 2 DFD

5. IMPLEMENTATION DETAILS

Working Code:

HTML:

```html
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Document Editor</title>
    <link rel="stylesheet" type="text/css" href="style.css" />
  </head>
  <body>
    <div>
      <h2>HTML DOCUMENT EDITOR</h2>
    </div>
    <div class="toolbar">
      <select id="font-family">
        <option value="Arial">Arial</option>
        <option value="Verdana">Verdana</option>
        <option value="Tahoma">Tahoma</option>
        <option value="Helvetica">Helvetica</option>
        <option value="Times New Roman">Times New Roman</option>
      </select>
      <select id="font-size">
        <option value="12">12</option>
        <option value="14">14</option>
        <option value="16">16</option>
        <option value="18">18</option>
        <option value="20">20</option>
        <option value="22">22</option>
        <option value="24">24</option>
        <option value="26">26</option>
        <option value="28">28</option>
        <option value="30">30</option>
      </select>
      <button id="left-justify">Left</button>
      <button id="right-justify">Right</button>
      <button id="save-file">Save</button>
      <button id="open-file">Open</button>
      <button id="format-button">Format Text</button>
      <button id="stats">Document Statistics</button>
      <select id="format-select">
        <option value="bold">Bold</option>
        <option value="italic">Italic</option>
        <option value="sup">Superscript</option>
        <option value="sub">Subscript</option>
      </select>
      <input type="file" id="image-input" />
    </div>

    <div contenteditable="true" id="document">
      <h3>Clear it and enter your text.</h3>
    </div>
    <input type="file" id="file-input" style="display: none" />
    <script src="script.js"></script>
  </body>
</html>
```

CSS:

```css
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  background-image: linear-gradient(to right, #0f0c29, #302b63, #24243e);
}

.toolbar {
  background-image: linear-gradient(to right, #0f0c29, #302b63, #24243e);
  padding: 3px;
  font-size: 25px;
}
.toolbar select {
  font-size: 20px;
  font-weight: 80px;
  margin: 10px auto;
  width: fit-content;
  border: 1px solid black;
  padding: 10px 10px;
  border-radius: 6px;
  text-decoration: none;
  color: #000;
  transition: background 0.5s;
  cursor: pointer;
}

#document {
  border-radius: 5px;
  color: aliceblue;
}

.toolbar input {
  cursor: pointer;
}
.toolbar button {
  font-size: 20px;
  font-weight: 80px;
  margin: 10px auto;
  width: fit-content;
  border: 1px solid black;
  padding: 10px 10px;
  border-radius: 6px;
  text-decoration: none;
  color: #000;
  transition: background 0.5s;
  cursor: pointer;
}

.toolbar button:hover {
  background: #404040;
}
.toolbar select :hover {
  background: #404040;
}

#document {
  border: 1px solid #ccc;
  margin: 7px;
  padding: 7px;
  background-color: rgb(69, 60, 86);
}
h2 {
  font-weight: 12px;
  font-family: Arial, Helvetica, sans-serif;
  text-align: center;
  font-size: 40px;
  padding: 4px;
  margin: 0;
  color: #ccc;
  background-image: linear-gradient(to right, #0f0c29, #302b63, #24243e);
}
```

JS:

```js
const fontFamilySelect = document.getElementById("font-family");
const fontSizeSelect = document.getElementById("font-size");
const documentEditor = document.getElementById("document");

const leftJustifyButton = document.getElementById("left-justify");
const rightJustifyButton = document.getElementById("right-justify");
const saveFileButton = document.getElementById("save-file");
const openFileButton = document.getElementById("open-file");
const fileInput = document.getElementById("file-input");
const statsButton = document.getElementById("stats");
const formatButton = document.getElementById("format-button");
```

```javascript
const formatSelect = document.getElementById("format-select");
const imageInput = document.getElementById("image-input");

fontFamilySelect.addEventListener("change", (event) => {
  documentEditor.style.fontFamily = event.target.value;
});

fontSizeSelect.addEventListener("change", (event) => {
  documentEditor.style.fontSize = `${event.target.value}px`;
});

leftJustifyButton.addEventListener("click", () => {
  documentEditor.style.textAlign = "left";
});

rightJustifyButton.addEventListener("click", () => {
  documentEditor.style.textAlign = "right";
});

openFileButton.addEventListener("click", () => {
  fileInput.click();
});

fileInput.addEventListener("change", (event) => {
  const file = event.target.files[0];
  const reader = new FileReader();
  reader.readAsText(file);
  reader.onload = () => {
    documentEditor.innerHTML = reader.result;
  };
});

saveFileButton.addEventListener("click", () => {
  const content = documentEditor.innerHTML;
  const fileName = prompt("Enter a file name:", "document.html");
  if (fileName !== null) {
    const file = new Blob([content], { type: "text/plain" });
    const a = document.createElement("a");
    a.href = URL.createObjectURL(file);
    a.download = fileName;
    a.click();
    URL.revokeObjectURL(a.href);
  }
});
statsButton.addEventListener("click", () => {
  const content = documentEditor.innerText;
  const charCount = content.length;
  const wordCount = content.split(/\s+/).length;
  const lineCount = content.split("\n").length;
  const paraCount = content.split("\n\n").length;
  alert(
    `Characters: ${charCount}\nWords: ${wordCount}\nLines:
${lineCount}\nParagraphs: ${paraCount}`
  );
});

formatButton.addEventListener("click", () => {
  const content = documentEditor.innerText;
  const maxWidth = prompt(
    "Enter the maximum number of characters per line:",
    "132"
  );
  if (maxWidth !== null) {
    const formattedContent = content.replace(
      new RegExp(`(.{1,${maxWidth}})(\\s+|$)`, "g"),
      "$1\n"
    );
    documentEditor.innerText = formattedContent;
  }
});

formatSelect.addEventListener("change", (event) => {
  const format = event.target.value;
  switch (format) {
    case "bold":
      document.execCommand("bold");
      break;
    case "italic":
      document.execCommand("italic");
      break;
    case "sup":
      document.execCommand("superscript");
      break;
    case "sub":
      document.execCommand("subscript");
      break;
    default:
      break;
  }
});

imageInput.addEventListener("change", (event) => {
  const file = event.target.files[0];
  const reader = new FileReader();
  reader.readAsDataURL(file);
  reader.onload = () => {
    const img = document.createElement("img");
    img.src = reader.result;
    documentEditor.appendChild(img);
  };
});
```

6. CONCLUSION

In conclusion, the HTML document editor project aims to provide a user-friendly interface for creating and editing HTML documents. The need for such an editor arises from the popularity and widespread use of HTML in web development, and the complex nature of HTML code that can make it challenging for non-technical users to create and edit HTML documents.

The objective of the project is to create an editor that is easy to use, provides useful features for HTML development, and is accessible to users with different levels of technical expertise. The implementation details may include features such as file management, drag-and-drop editing, code completion, code validation, support for multiple programming languages, custom plugins and extensions, and project management features.

Overall, an HTML document editor can greatly simplify the process of creating and editing HTML documents, and can be a valuable tool for web developers and non-technical users alike. By providing a user-friendly interface and useful features, the editor can help users create high-quality HTML documents more efficiently and with greater ease.

A. APPENDIX

User manual: A detailed guide on how to use the HTML document editor, including step-by-step instructions on how to create, edit, and save HTML documents.

Sample code: Sample HTML code snippets that demonstrate how to use various features of the editor, such as inserting images, creating tables, and adding links.

Design documents: If the HTML document editor was part of a larger software development project, design documents such as software requirements, architecture diagrams, or use cases could be included in the appendix.

Bibliography: A list of sources consulted during the project, such as books, academic papers, or online tutorials.

Technical specifications: Detailed technical specifications for the HTML document editor, such as system requirements or performance benchmarks.

Sample output: Examples of HTML documents created using the editor, to showcase the editor's capabilities and potential uses.

Acknowledgments: A section to thank anyone who contributed to the project, such as mentors, colleagues, or project sponsors.

References

References for an HTML document editor are:

1. "HTML and CSS: Design and Build Websites" by Jon Duckett. This book provides a beginner-friendly introduction to HTML and CSS, with clear explanations and examples.
2. "Pro HTML5 and CSS3 Design Patterns" by Michael Bowers, Dionysios Synodinos, and Victor Sumner. This book covers advanced topics in HTML and CSS, with a focus on best practices and design patterns.
3. "Bootstrap" (https://getbootstrap.com/). Bootstrap is a popular front-end framework for building responsive web pages and applications. The official documentation provides a comprehensive guide to using Bootstrap.
4. "MDN Web Docs" (https://developer.mozilla.org/en-US/docs/Web). MDN Web Docs is a comprehensive resource for web developers, with articles and tutorials on HTML, CSS, JavaScript, and other web technologies.
5. "W3Schools" (https://www.w3schools.com/). W3Schools provides a variety of online tutorials and reference materials for web development, including HTML, CSS, JavaScript, and many other topics.
6. "Sublime Text Documentation" (https://www.sublimetext.com/docs/). Sublime Text is a popular code editor for web development. The official documentation provides information on using the editor, including tips and tricks for working with HTML and CSS.
7. "Atom Flight Manual" (https://flight-manual.atom.io/). Atom is another popular code editor for web development. The Atom Flight Manual provides a comprehensive guide to using the editor, including information on working with HTML and CSS.
8. "Visual Studio Code Documentation" (https://code.visualstudio.com/docs). Visual Studio Code is a free, open-source code editor for web development. The official documentation provides information on using the editor, including tips and tricks for working with HTML and CSS.
9. "Emmet Documentation" (https://emmet.io/docs/). Emmet is a plugin for code editors that makes it easier to write HTML and CSS code. The documentation provides a guide to using Emmet, including shortcuts and syntax.
10. "Stack Overflow" (https://stackoverflow.com/). Stack Overflow is a popular Q&A site for programmers, with a vast archive of questions and answers on HTML, CSS, JavaScript, and other web development topics.