CHAPTER 1

INTRODUCTION

In an increasingly digital world, the importance of robust and secure passwords cannot be overstated. With the proliferation of online services, applications, and sensitive information stored in the digital realm, a reliable password generator is an essential tool for users to safeguard their data.

This project, titled "Password Generator," aims to provide users with a user-friendly and versatile tool to generate strong and unique passwords tailored to their specific needs. By leveraging modern web technologies, such as HTML, CSS, and JavaScript, this web-based application offers a seamless experience for generating secure passwords with various customizable options.

The core functionalities of this Password Generator include:

1. **Password Length Customization:** Users can specify the desired length of their password, ensuring it meets the requirements of different online platforms.
2. **Character Set Customization:** Users have the flexibility to include or exclude uppercase letters, lowercase letters, numbers, and symbols in their generated passwords, allowing them to adhere to specific password policies.
3. **Clipboard Integration:** For user convenience, a built-in clipboard feature enables one-click copying of generated passwords for immediate use.

CHAPTER 2

IMPLEMENTATION

HTML Code

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="style.css" />
    <title>Password Generator</title>
  </head>
  <body>
    <div class="container">
      <h2>Password Generator</h2>
      <div class="result-container">
        <span id="result"></span>
        <button class="btn" id="clipboard">
          <i class="far fa-clipboard"></i>
        </button>
      </div>
      <div class="settings">
        <div class="setting">
          <label>Password Length</label>
          <input type="number" id="length" min="4" max="20" value="20">
        </div>
        <div class="setting">
          <label>Include uppercase letters</label>
          <input type="checkbox" id="uppercase" checked>
        </div>
        <div class="setting">
          <label>Include lowercase letters</label>
          <input type="checkbox" id="lowercase" checked>
        </div>
        <div class="setting">
          <label>Include numbers</label>
          <input type="checkbox" id="numbers" checked>
        </div>
        <div class="setting">
          <label>Include symbols</label>
          <input type="checkbox" id="symbols" checked>
        </div>
      </div>
```

```
      <button class="btn btn-large" id="generate">
        Generate Password
      </button>
    </div>
    <script src="script.js"></script>
  </body>
</html>
```

This HTML code defines a basic webpage structure for a password generator tool. It incorporates metadata, external CSS stylesheets, and JavaScript functionality. The webpage comprises a container division containing a title, a prominent "Password Generator" heading, a result display area, settings for password customization (including length and character types), and a "Generate Password" button. The user can tailor the password by adjusting checkboxes and an input field. When the "Generate Password" button is activated, JavaScript, presumably embedded in an external script.js file, generates a password based on the user's preferences. The generated password is then presented in the result container. Additionally, users can easily copy the generated password to their clipboard using the provided clipboard button. In essence, this HTML code establishes a user-friendly interface for generating secure passwords with customizable options.

The settings section, which includes checkboxes for character type inclusion and an input field for password length, offers users flexibility in tailoring the generated passwords to meet their specific security requirements. The "Generate Password" button acts as the trigger for the password generation process, employing JavaScript to create a random password based on the user's chosen parameters.

CSS Code

```css
@import url('https://fonts.googleapis.com/css?family=Muli&display=swap');

* {
  box-sizing: border-box;
}

body {
  background-color: #3b3b98;
  color: #fff;
  font-family: 'Muli', sans-serif;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  height: 100vh;
  overflow: hidden;
  padding: 10px;
  margin: 0;
}

h2 {
  margin: 10px 0 20px;
  text-align: center;
}

.container {
  background-color: #fe3017;
  box-shadow: 0px 2px 10px rgba(255, 255, 255, 0.2);
  padding: 20px;
  width: 350px;
  max-width: 100%;
}

.result-container {
  background-color: rgba(0, 0, 0, 0.4);
  display: flex;
  justify-content: flex-start;
  align-items: center;
  position: relative;
  font-size: 18px;
  letter-spacing: 1px;
  padding: 12px 10px;
  height: 50px;
  width: 100%;
}
```

```
.result-container #result {
  word-wrap: break-word;
  max-width: calc(100% - 40px);
  overflow-y: scroll;
  height: 100%;
}

#result::-webkit-scrollbar {
  width: 1rem;
}

.result-container .btn {
  position: absolute;
  top: 5px;
  right: 5px;
  width: 40px;
  height: 40px;
  font-size: 20px;
}

.btn {
  border: none;
  background-color: #3b3b98;
  color: #fff;
  font-size: 16px;
  padding: 8px 12px;
  cursor: pointer;
}

.btn-large {
  display: block;
  width: 100%;
}

.setting {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin: 15px 0;
}
```

The CSS code begins with an **@import** statement, which imports the "Muli" font from Google Fonts and makes it available for use throughout the webpage. This is a common practice to ensure consistent typography across the site.

Moving on to the global styling, the ***** selector selects all elements on the page. The **box-sizing: border-box;** property is set, which is a popular CSS reset technique. It ensures that an element's total width includes its padding and border, making it easier to control layout and alignment consistently across different elements.

The styling for the **body** element is defined next. The background color is set to **#3b3b98**, which is a deep blue color, and the text color is set to **#fff**, making the text white. The chosen font for the entire webpage is "Muli," a sans-serif font, which is loaded from Google Fonts using the **@import** statement at the beginning. The **display: flex;** property is applied, making the body a flex container. This is a common technique for creating responsive layouts. By setting **flex-direction: column;**, the content within the body will be stacked vertically. **align-items: center;** and **justify-content: center;** center the content both vertically and horizontally within the body. The **height: 100vh;** rule ensures that the body takes up the full viewport height, and **overflow: hidden;** hides any content that overflows the viewport. Finally, **padding: 10px;** and **margin: 0;** set the padding and margin of the body element.

The CSS code also defines styles for specific elements within the webpage, such as headings, containers, result displays, buttons, and settings. For example, the h2 element is centered with a top margin, the .container has a background color and box shadow to create a card-like appearance, and the **.btn** class sets the style for buttons, including their background color, text color, and cursor behavior. These rules help create a visually appealing and user-friendly interface for the password generator web page. Overall, the CSS code complements the HTML structure to achieve the desired styling and layout for the webpage.

Javascript Code

```
const resultEl = document.getElementById('result')
const lengthEl = document.getElementById('length')
const uppercaseEl = document.getElementById('uppercase')
const lowercaseEl = document.getElementById('lowercase')
const numbersEl = document.getElementById('numbers')
const symbolsEl = document.getElementById('symbols')
const generateEl = document.getElementById('generate')
const clipboardEl = document.getElementById('clipboard')

const randomFunc = {
    lower: getRandomLower,
    upper: getRandomUpper,
    number: getRandomNumber,
    symbol: getRandomSymbol
}

clipboardEl.addEventListener('click', () => {
    const password = resultEl.innerText;
  if (!password) {
    return;
  }
  navigator.clipboard.writeText(password);
    alert('Password copied to clipboard!')
})

generateEl.addEventListener('click', () => {
    const length = +lengthEl.value
    const hasLower = lowercaseEl.checked
    const hasUpper = uppercaseEl.checked
    const hasNumber = numbersEl.checked
    const hasSymbol = symbolsEl.checked

    resultEl.innerText = generatePassword(hasLower, hasUpper, hasNumber,
hasSymbol, length)
})

function generatePassword(lower, upper, number, symbol, length) {
    let generatedPassword = ''
    const typesCount = lower + upper + number + symbol
    const typesArr = [{lower}, {upper}, {number}, {symbol}].filter(item =>
Object.values(item)[0])

    if(typesCount === 0) {
        return ''
    }
```

```
    for(let i = 0; i < length; i += typesCount) {
        typesArr.forEach(type => {
            const funcName = Object.keys(type)[0]
            generatedPassword += randomFunc[funcName]()
        })
    }

    const finalPassword = generatedPassword.slice(0, length)

    return finalPassword
}

function getRandomLower() {
    return String.fromCharCode(Math.floor(Math.random() * 26) + 97)
}

function getRandomUpper() {
    return String.fromCharCode(Math.floor(Math.random() * 26) + 65)
}

function getRandomNumber() {
    return String.fromCharCode(Math.floor(Math.random() * 10) + 48)
}

function getRandomSymbol() {
    const symbols = '!@#$%^&*(){}[]=<>/,.'
    return symbols[Math.floor(Math.random() * symbols.length)]
}
```

It begins by selecting various HTML elements using their id attributes, enabling interaction with these elements. Notably, it captures user input, including the desired password length and options for including lowercase letters, uppercase letters, numbers, and symbols.

The code includes a set of random character generation functions, organized within the randomFunc object. These functions generate random characters for lowercase and uppercase letters, numbers, and symbols, contributing to the password generation process.

The clipboardEl element is equipped with an event listener, allowing users to copy the generated password to their clipboard with a simple click. This feature

enhances user convenience and security.

The core of the script lies in the generatePassword function, which constructs a password based on the user's selected criteria. It calculates the number of character types to include and builds the password accordingly, ensuring that the generated password adheres to the specified length and character types.

CHAPTER 3

OUTPUT