CODETANTRA

AN INDUSTRIAL REPORT ON

"Online Code Editor"

Submitted in Partial Fulfillment for the award of the degree of

Bachelor of Technology

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

Submitted by

Student Name : Poojitha Varikallu

Department of Artificial Intelligence and Data Science

PACE Institute of Technology and Sciences
(Autonomous)

Approved by AICTE and Govt of Andhra Pradesh Accredited by NAAC (A grade), Recognized under 2(f) & 12(B) of UGC Permanently Affiliated to JNTUK, Kakinada. A.P., An ISO 9001:2015 and ISO 50001:2018 Certified Institution
NH-16, Near Valluramma Temple, ONGOLE - 523 272, A.P.

CODETANTRA

PACE INSTITUTE OF TECHNOLOGY AND SCIENCE

DEPARTMENT OF AIDS

CERTIFICATE

This is to certify that this internship report on **"Online Code Editor"** is the Bonafide work of **"Poojitha Varikallu - 22KQ1A5435"** who has carried out the work under my supervision and submitted in partial fulfillment for the award of **CodeTantra Web Development Internship** during the year 2025(May-26)-2025(June-24).

Signature of External Trainer

Signature Internal Trainer

Prof. & HoD

ACKNOWLEDGEMET

I would like to express my sincere gratitude to all those who supported and guided me throughout my internship journey titled **"Online Code Editor."**

First and foremost, I extend my heartfelt thanks to **Dr. G. V. K. Murthy**, Principal of **PACE Institute of Technology and Science**, for providing me with an encouraging environment and the opportunity to undertake this internship.

I am deeply grateful to **Dr. G. Ganesh Naidu**, Head of the AIDS, for paving the path and supporting me with all the necessary resources and guidance required to complete this internship.

My sincere thanks to **Ms. V. Mounika**, Internship In-charge, for her valuable suggestions, constant encouragement, and insightful feedback that played a significant role in the successful completion of this internship.

I would like to place on record my sincere appreciation and gratitude to technical trainer **P. Mani Sampath** from **CodeTantra**, the organization where I had the opportunity to pursue my internship. I am thankful for the exposure, technical learning, and real-time insights into **Web Development** that I gained during this period.

Finally, I am immensely thankful to my friends and family for their unwavering moral support throughout this journey.

.

Poojitha Varikallu
22KQ1A5435

ABSTRACT

The **Online Code Editor** is a web-based application designed to provide a comprehensive platform for coding in various programming languages. This project aims to create an interactive environment where users can write, edit, and execute code seamlessly. The Online Code Editor supports multiple programming languages, including **HTML, CSS, JavaScript, Python, C, C++, and Java**, making it versatile for different coding needs.

The application integrates with the **Judge0 API**, which allows users to run their code and receive immediate feedback on the output. This feature is particularly beneficial for learners and developers who wish to practice coding and debug their programs in real-time. The user interface is designed to be **intuitive and user-friendly**, ensuring that even beginners can navigate the platform with ease.

In addition to its core functionalities, the Online Code Editor emphasizes **security and performance**. User inputs are sanitized to prevent code injection attacks, and the execution environment is isolated to ensure that malicious code does not affect the server or other users. The project not only serves as a practical tool for coding but also as a learning platform for users to improve their programming skills.

Overall, the Online Code Editor represents a significant step towards enhancing the coding experience for users, providing them with the tools they need to learn, practice, and excel in programming.

INDEX

CHAPTER 1

INTRODUCTION

1.1 Overview

The Online Code Editor is a web-based platform that allows users to write, edit, and execute code in various programming languages. The primary goal of this project is to create an accessible and efficient coding environment that caters to both novice and experienced programmers. With the increasing demand for coding skills in today's job market, providing a platform that simplifies the coding process is essential.

The Online Code Editor is designed to be **user-friendly**, enabling users to focus on their coding tasks without being overwhelmed by complex features. The application supports multiple programming languages, allowing users to choose their preferred language for coding. This flexibility is crucial for learners who may be exploring different programming languages or for developers who work on diverse projects.

1.2 Features

The Online Code Editor comes equipped with several key features that enhance the user experience:

- **Multi-language Support**: The editor supports a variety of programming languages, including HTML, CSS, JavaScript, Python, C, C++, and Java. This allows users to work in their preferred language and switch between languages as needed.
- **Real-time Code Execution**: Users can execute their code and see the output instantly. This feature is particularly beneficial for debugging and learning, as it provides immediate feedback on code performance.
- **User-friendly Interface**: The interface is designed to be clean and intuitive, making it easy for users to navigate and utilize the features. The layout includes a language selection dropdown, a code editor area, and an output display section.
- **Code Highlighting and Formatting**: The editor provides syntax highlighting and formatting options to improve code readability. This feature helps users identify errors and understand the structure of their code more easily.

- **Error Detection and Debugging Tools**: The application includes features to help users identify and fix errors in their code. This is especially important for beginners who may struggle with debugging.
- **Collaboration Features (Future Work)**: Plans for future enhancements include adding collaborative coding features, allowing multiple users to work on the same project simultaneously.

1.3 Advantages

The Online Code Editor offers several advantages that make it a valuable tool for users:

- **Accessibility**: The application can be accessed from any device with internet connectivity, allowing users to code from anywhere. This is particularly useful for students and professionals who may need to work on projects outside of a traditional classroom or office setting.
- **Encourages Learning**: The real-time execution feature encourages users to experiment with their code and learn from their mistakes. This hands-on approach to learning is effective in building coding skills.
- **Instant Feedback**: Users receive immediate feedback on their code execution, which helps them understand the impact of their changes and improve their coding abilities.
- **Enhanced Learning Experience**: The combination of practical application and instant results enhances the overall learning experience for users, making coding more engaging and less intimidating.

1.4 Scope

The scope of the Online Code Editor is broad, with potential for expansion in several areas:

- **Additional Language Support**: Future versions of the application could include

- support for more programming languages, catering to a wider audience.
- **Integration with Version Control Systems**: Implementing features that allow users to save their projects and collaborate with others through version control systems like Git could enhance the platform's functionality.
- **User Accounts and Project Storage**: Adding user accounts would enable users to save their projects and access them from any device. This feature would also facilitate collaboration among users.

Advanced Features: Future enhancements may include AI-driven code suggestions, advanced debugging tools, and integration with third-party libraries and frameworks.

1.5 Future Work

The future work for the Online Code Editor includes several exciting enhancements:

- **Cloud Storage for User Projects**: Implementing cloud storage would allow users to save their projects securely and access them from any device. This feature would enhance the user experience and encourage users to return to the platform.
- **User Authentication**: Adding user authentication would enable users to create accounts, save their work, and collaborate with others. This feature would also enhance security by ensuring that only authorized users can access certain functionalities.
- **Community Forum**: Creating a community forum where users can share their code, ask questions, and collaborate on projects would foster a sense of community and support among users.
- **Integration with Learning Resources**: Partnering with educational platforms to provide users with access to tutorials, coding challenges, and other learning resources could further enhance the platform's value.

CHAPTER 2

TOOLS REQUIRED

2.1 Tools and Softwares used:

The development of the Online Code Editor involved several tools and software:

- **HTML, CSS, JavaScript**: These core web technologies were used for front-end development, creating the user interface and interactive elements of the application. HTML was used for structuring the content, CSS for styling, and JavaScript for adding interactivity.
- **Node.js**: This JavaScript runtime was used for back-end services, handling requests from the client and integrating with external APIs. Node.js allows for efficient handling of asynchronous operations, making it suitable for real-time applications.
- **Judge0 API**: This API was utilized for code execution, allowing users to run their code and receive output in real-time. The Judge0 API supports multiple programming languages and provides a secure environment for code execution.
- **CodeMirror or Ace Editor**: These libraries were considered for implementing the code editing functionalities, providing features like syntax highlighting, auto-completion, and error detection.

Version Control System (Git): Git was used for version control during the development process, allowing the team to track changes, collaborate effectively, and manage the codebase.

CHAPTER 3

PROJECT ARCHITECTURE

3.1 Architecture

The architecture of the Online Code Editor consists of a **client-server model**, which is essential for web applications. The architecture can be broken down into the following components:

- **Client Side**: The client side includes the user interface where users can write and execute code. It consists of various components such as the language selector, code editor, and output display area. The client-side code is primarily written in HTML, CSS, and JavaScript, ensuring a responsive and interactive user experience.
- **Server Side**: The server side handles requests from the client, processes the code execution through the Judge0 API, and returns the results to the client. The server is built using Node.js, which allows for efficient handling of multiple requests and real-time communication.
- **API Integration**: The application integrates with the Judge0 API to execute code securely. When a user submits their code for execution, the server sends a request to the Judge0 API, which compiles and runs the code in a secure environment. The results, including any output or error messages, are then returned to the user.
- **Data Flow**: The data flow in the application follows a request-response model. When a user writes code and clicks the "Run" button, the client sends the code to the server, which processes the request and communicates with the Judge0 API. The API executes the code and sends the output back to the server, which then relays the results to the client for display.

CONCLUSION

The **Online Code Editor** project successfully demonstrates the ability to create a web-based coding platform that is both functional and user-friendly. Throughout the development process, the team encountered various challenges, including ensuring the security of the code execution environment and optimizing the user interface for a seamless experience. However, these challenges were addressed through careful planning and implementation.

The project not only serves as a practical tool for coding but also as a **learning platform** for users to improve their programming skills. The integration of real-time code execution and user-friendly features enhances the overall learning experience, making coding more accessible to a broader audience.

In conclusion, the Online Code Editor represents a significant step towards enhancing the coding experience for users. The project has the potential for further development and expansion, with plans for additional features and improvements in the future. The skills and knowledge gained during this internship will be invaluable as I continue my journey in the field of web development.

Future Scope

The Online Code Editor project has great potential for future enhancements and scalability. As technology and user expectations evolve, the platform can be extended in the following ways:

- AI-Powered Code Assistance: Implementing AI-based features such as code suggestions, auto-completion, and bug detection to enhance coding efficiency and learning.
- Mobile Optimization: Developing a mobile-friendly version or app to allow users to code on the go using smartphones or tablets.
- Offline Mode: Enabling limited offline coding functionality with automatic syncing once the internet is available.
- Integrated Debugger: Adding an in-browser step-by-step debugger to trace code execution and better understand logic errors.
- Gamified Learning Modules: Incorporating interactive challenges and coding games to motivate users and make learning more engaging.
- Voice-Based Coding: Exploring speech-to-code functionality for accessibility and productivity enhancement.
- Live Pair Programming: Allowing users to connect in real-time for collaborative programming and mentorship sessions.
- Customization Options: Letting users customize themes, layouts, and keyboard shortcuts to personalize their experience.

REFERENCES

- Judge0 API Documentation
- CodeMirror Documentation
- Node.js Documentation
- W3Schools - HTML, CSS, JavaScript Tutorials
- Mozilla Developer Network (MDN) Web Doc

CODE

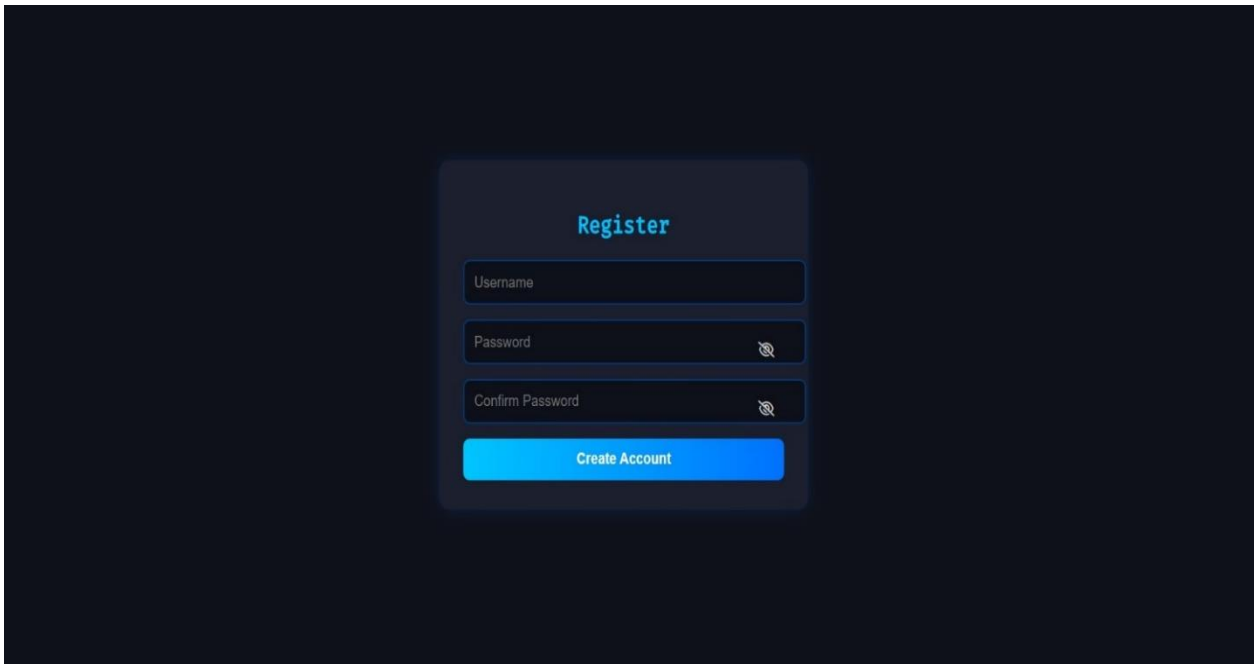
register.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
 <title>Register - Universal Code Editor</title>
 <link href="https://fonts.googleapis.com/css2?family=Fira+Code&display=swap"
rel="stylesheet">
 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.5.0/css/all.min.css">
 <style>
  body {
   font-family: 'Fira Code', monospace;
   background: #0f111a;
   color: #e0e8f0;
   display: flex;
   justify-content: center;
   align-items: center;
   height: 100vh;
   margin: 0;
  }
  .register-container {
   background: #1c1f2e;
   padding: 30px;
   border-radius: 12px;
   box-shadow: 0 0 15px rgba(0, 114, 255, 0.2);
   max-width: 400px;
   width: 100%;
  }
  .register-container h2 {
   color: #00c6ff;
   text-align: center;
   margin-bottom: 20px;
  }
  .register-container input {
   width: 100%;
   padding: 12px;
   margin-bottom: 15px;
   border: 2px solid #0072ff66;
   border-radius: 8px;
   background: #0f111a;
   color: #e0e8f0;
   font-size: 1rem;
```

```
  }
  .register-container button {
    width: 100%;

    padding: 12px;
    background: linear-gradient(90deg, #00c6ff, #0072ff);
    color: white;
    border: none;
    border-radius: 8px;
    font-weight: bold;
    font-size: 1rem;
    cursor: pointer;
  }
  .password-wrapper {
    position: relative;
  }
  .toggle-password {
    position: absolute;
    right: 12px;
    top: 50%;
    transform: translateY(-50%);
    cursor: pointer;
    color: #ccc;
  }
 </style>
</head>
<body>
 <div class="register-container">
  <h2>Register</h2>
  <input type="text" id="reg-username" placeholder="Username" required />
  <div class="password-wrapper">
   <input type="password" id="reg-password" placeholder="Password" required />
   <span class="toggle-password" onclick="toggleVisibility('reg-password', this)">
    <i class="fa-regular fa-eye-slash"></i>
   </span>
  </div>
  <div class="password-wrapper">
   <input type="password" id="reg-confirm" placeholder="Confirm Password" required />
   <span class="toggle-password" onclick="toggleVisibility('reg-confirm', this)">
    <i class="fa-regular fa-eye-slash"></i>
   </span>
  </div>
  <button onclick="register()">Create Account</button>
 </div>
 <script>
  function toggleVisibility(id, icon) {
   const field = document.getElementById(id);
   const isPassword = field.type === "password";
```

```
    field.type = isPassword ? "text" : "password";

    const iTag = icon.querySelector("i");
    iTag.classList.toggle("fa-eye");
    iTag.classList.toggle("fa-eye-slash");
   }
   function validatePassword(password) {
    const minLength = 6;
    const hasNumber = /[0-9]/.test(password);
    const hasSpecialChar = /[!@#$%^&*(),.?":{}|<>]/.test(password);
    return password.length >= minLength && hasNumber && hasSpecialChar;
   }
   function register() {
    const username = document.getElementById("reg-username").value.trim();
    const password = document.getElementById("reg-password").value.trim();
    const confirm = document.getElementById("reg-confirm").value.trim();
    if (!username || !password || !confirm) {
     alert("Please fill in all fields.");
     return;
    }
    if (password !== confirm) {
     alert("Passwords do not match.");
     return;
    }
    if (!validatePassword(password)) {
     alert("Password must be at least 6 characters long, include a number and a special
character.");
     return;
    }
    localStorage.setItem("registeredUser", username);
    localStorage.setItem("registeredPass", password);
    alert("Account created successfully! Please sign in.");
    window.location.href = "login.html";
   }
  </script>
</body>
</html>
```

Output



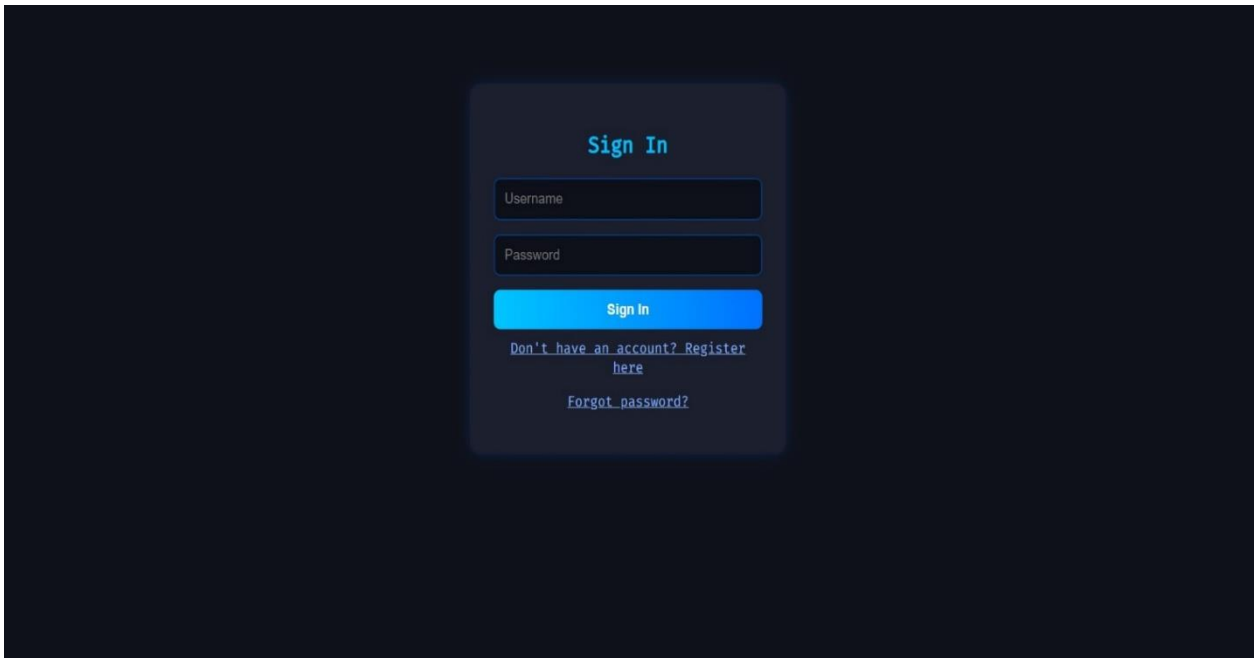
login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
 <title>Sign In - Universal Code Editor</title>
 <link href="https://fonts.googleapis.com/css2?family=Fira+Code&display=swap"
rel="stylesheet">
 <link rel="stylesheet" href="styles.css" />
 <style>
  .login-container {
   max-width: 400px;
   margin: 100px auto;
   background: #1c1f2e;
   padding: 30px;
   border-radius: 12px;
   box-shadow: 0 0 15px rgba(0, 114, 255, 0.2);
  }
  .login-container h2 {
   color: #00c6ff;
   text-align: center;
   margin-bottom: 20px;
  }
  .login-container input {
   width: 100%;
   padding: 12px;
   margin-bottom: 15px;
```

```
    border: 2px solid #0072ff66;
    border-radius: 8px;
    background: #0f111a;
    color: #e0e8f0;
    font-size: 1rem;
   }
   .login-container input:focus {
    border: 1px solid #00c6ff;
    box-shadow: 0 0 10px #00c6ff88;
    outline: none;
   }
   .login-container button {
    width: 100%;
    padding: 12px;
    background: linear-gradient(90deg, #00c6ff, #0072ff);
    color: white;
    border: none;
    border-radius: 8px;
    font-weight: bold;
    font-size: 1rem;

    cursor: pointer;
    transition: background 0.3s ease-in-out;
   }
   .login-container button:hover {
    background: linear-gradient(90deg, #0072ff, #00c6ff);
   }
  </style>
</head>
<body>
  <div class="login-container">
   <h2>Sign In</h2>
   <input type="text" id="username" placeholder="Username" required />
   <input type="password" id="password" placeholder="Password" required />
   <button onclick="signIn()">Sign In</button>
   <p style="text-align: center; margin-top: 10px;">
  <a href="register.html" style="color:#8ab4f8;">Don't have an account? Register here</a>
</p>
<p style="text-align: center; margin-top: 5px;">
  <a href="forgot.html" style="color:#8ab4f8;">Forgot password?</a>
</p>
  </div>
  <script>
   function signIn() {
  const username = document.getElementById("username").value.trim();
  const password = document.getElementById("password").value.trim();
  const savedUser = localStorage.getItem("registeredUser");
  const savedPass = localStorage.getItem("registeredPass");
```

```
  if (username === savedUser && password === savedPass) {
    localStorage.setItem("user", username);
    window.location.href = "index.html";
  } else {
    alert("Invalid credentials. Please try again.");
  }
}
  </script>
</body>
</html>
```

Output



forgot.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Reset Password - Universal Code Editor</title>
  <link href="https://fonts.googleapis.com/css2?family=Fira+Code&display=swap"
rel="stylesheet">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.5.0/css/all.min.css">
  <style>
    body {
```

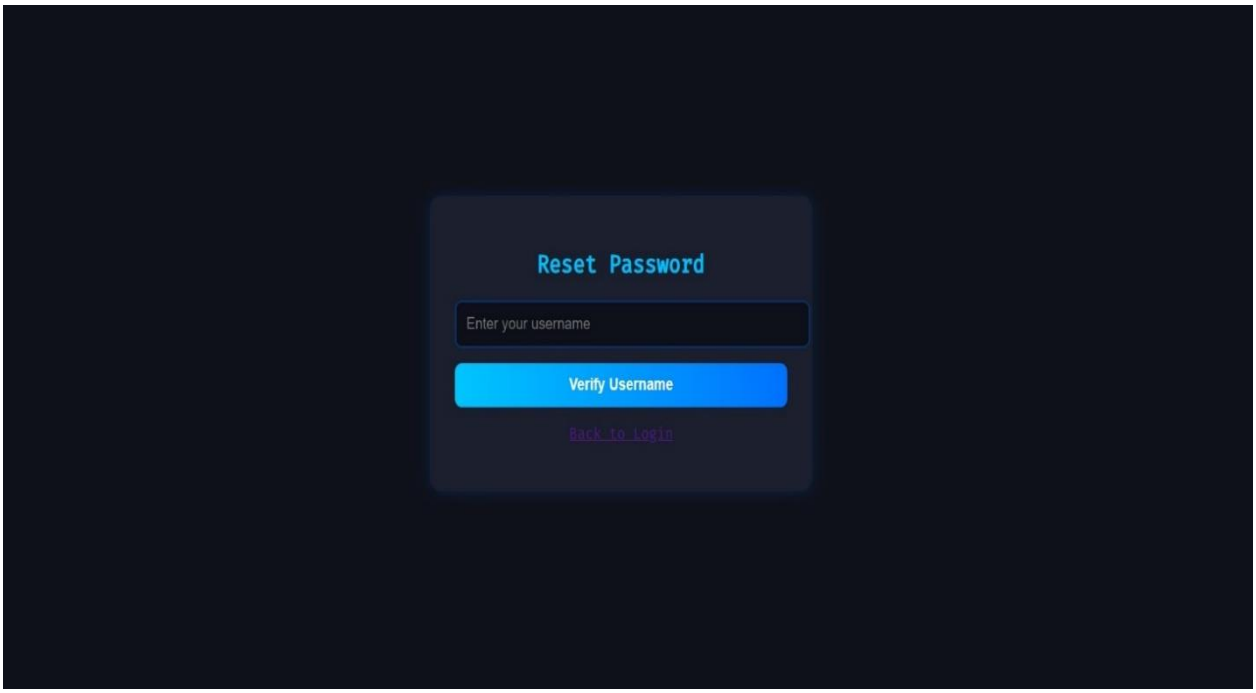
```
  font-family: 'Fira Code', monospace;
  background: #0f111a;
  color: #e0e8f0;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  margin: 0;
}
.forgot-container {
  background: #1c1f2e;
  padding: 30px;
  border-radius: 12px;
  box-shadow: 0 0 15px rgba(0, 114, 255, 0.2);

  max-width: 400px;
  width: 100%;
}
.forgot-container h2 {
  color: #00c6ff;
  text-align: center;
  margin-bottom: 20px;
}
.forgot-container input {
  width: 100%;
  padding: 12px;
  margin-bottom: 15px;
  border: 2px solid #0072ff66;
  border-radius: 8px;
  background: #0f111a;
  color: #e0e8f0;
  font-size: 1rem;
}
.forgot-container button {
  width: 100%;
  padding: 12px;
  background: linear-gradient(90deg, #00c6ff, #0072ff);
  color: white;
  border: none;
  border-radius: 8px;
  font-weight: bold;
  font-size: 1rem;
  cursor: pointer;
}
.back-link {
  text-align: center;
  margin-top: 15px;
}
```

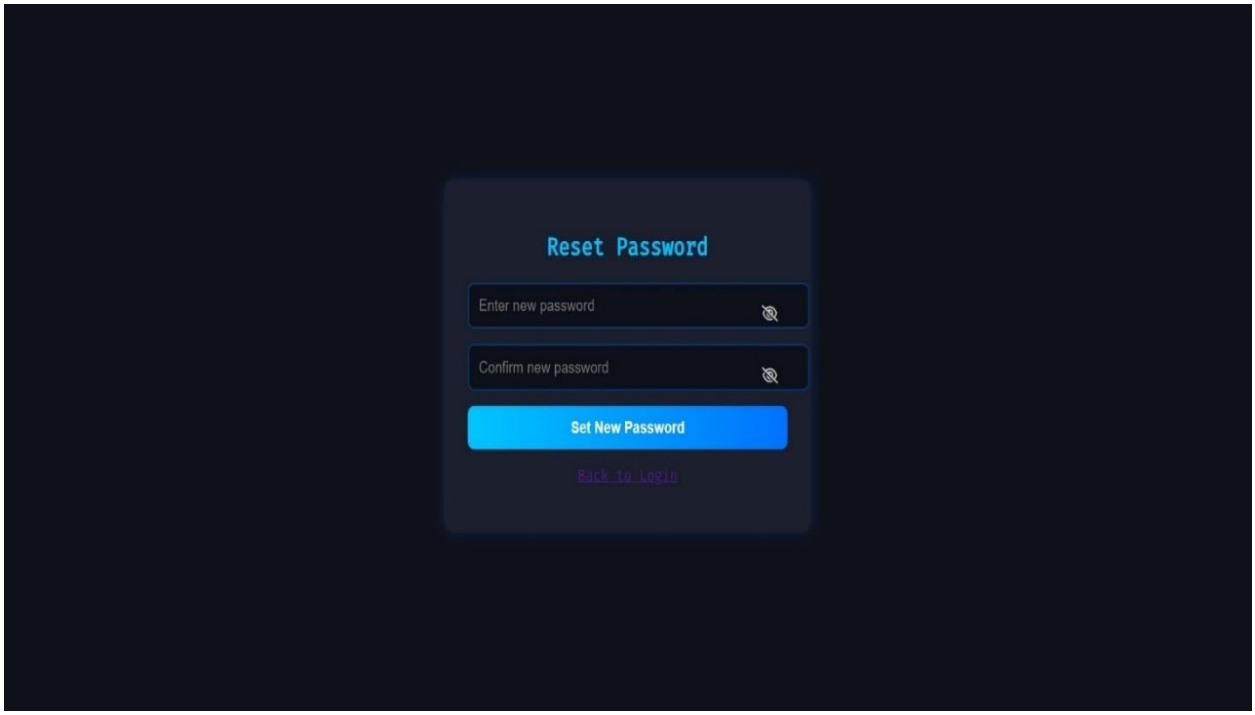
```
   .password-wrapper {
    position: relative;
   }
   .toggle-password {
    position: absolute;
    right: 12px;
    top: 50%;
    transform: translateY(-50%);
    cursor: pointer;
    color: #ccc;
   }
   #new-password-section {
    display: none;
   }
  </style>
</head>
<body>
  <div class="forgot-container">
   <h2>Reset Password</h2>
   <div id="verify-username-section">
    <input type="text" id="forgot-username" placeholder="Enter your username" required
/>
    <button onclick="verifyUsername()">Verify Username</button>
   </div>
   <div id="new-password-section">
    <div class="password-wrapper">
     <input type="password" id="new-password" placeholder="Enter new password"
required />
     <span class="toggle-password" onclick="toggleVisibility('new-password', this)">
      <i class="fa-regular fa-eye-slash"></i>
     </span>
    </div>
    <div class="password-wrapper">
     <input type="password" id="confirm-password" placeholder="Confirm new
password" required />
     <span class="toggle-password" onclick="toggleVisibility('confirm-password', this)">
      <i class="fa-regular fa-eye-slash"></i>
     </span>
    </div>
    <button onclick="setNewPassword()">Set New Password</button>
   </div>
   <p class="back-link">
    <a href="login.html">Back to Login</a>
   </p>
  </div>
  <script>
   function toggleVisibility(id, iconSpan) {
    const input = document.getElementById(id);
```

```
    const icon = iconSpan.querySelector('i');
    if (input.type === 'password') {
      input.type = 'text';
      icon.classList.remove('fa-eye-slash');
      icon.classList.add('fa-eye');
    } else {
      input.type = 'password';
      icon.classList.remove('fa-eye');
      icon.classList.add('fa-eye-slash');
    }
  }
  function verifyUsername() {
    const username = document.getElementById("forgot-username").value.trim();
    const savedUser = localStorage.getItem("registeredUser");
    if (username === savedUser) {
      document.getElementById("verify-username-section").style.display = "none";
      document.getElementById("new-password-section").style.display = "block";
    } else {
      alert("Username not found.");
    }
  }
  function validatePassword(password) {
    const minLength = 6;
    const hasNumber = /[0-9]/.test(password);
    const hasSpecialChar = /[!@#$%^&*(),.?":{}|<>]/.test(password);
    return password.length >= minLength && hasNumber && hasSpecialChar;
  }
  function setNewPassword() {
    const newPassword = document.getElementById("new-password").value.trim();
    const confirmPassword = document.getElementById("confirm-password").value.trim();
    if (!newPassword || !confirmPassword) {
      alert("Please fill in both fields.");
      return;
    }
    if (newPassword !== confirmPassword) {
      alert("Passwords do not match.");
      return;
    }
    if (!validatePassword(newPassword)) {
      alert("Password must be at least 6 characters, include a number and a special
character.");
      return;
    }
    localStorage.setItem("registeredPass", newPassword);
    alert("Password updated successfully! Please sign in.");
    window.location.href = "login.html";
  }
 </script>
```

</body>
</html>

Output





styles.css

```
* {
  box-sizing: border-box;
}
body {
  margin: 0;
  padding: 0;
  font-family: 'Fira Code', monospace;
  background: #0f111a;
  color: #e0e8f0;
  display: flex;
  flex-direction: column;
  min-height: 100vh;
}
header {
  background: linear-gradient(90deg, #00c6ff, #0072ff);
  color: white;
  padding: 1px;
  text-align: center;
  font-size: 1.2rem;
  font-weight: bolder;
  text-shadow: 0 0 10px #ffffffaa;
}
.controls {
  display: flex;
  flex-wrap: wrap;
  align-items: center;

  padding: 20px;
  background: #1a1d2b;
  border-bottom: 2px solid #0072ff66;
  gap: 20px;
}
.controls label {
  font-weight: bold;
  font-size: 1rem;
  color: #8ab4f8;
}
select, .run-btn {
  padding: 10px 16px;
  font-size: 1rem;
  border-radius: 8px;
  border: none;
  outline: none;
  transition: all 0.3s ease-in-out;
}
```

```
select {
  background: #222738;
  color: #8ab4f8;
  border: 1px solid #0072ff88;
  box-shadow: 0 0 6px #0072ff33;
}
select:hover {
  border-color: #00c6ff;
  box-shadow: 0 0 10px #00c6ff88;
}
.run-btn {
  background: linear-gradient(90deg, #00c6ff, #0072ff);
  color: white;
  font-weight: bold;
  cursor: pointer;
  box-shadow: 0 0 15px #00c6ff88;
}
.run-btn:hover {
  background: linear-gradient(90deg, #0072ff, #00c6ff);
}
.editor {
  display: grid;
  grid-template-columns: 1fr 1fr;
  gap: 20px;
  padding: 20px;
  flex-grow: 1;
}
.code-block {
  background: #1c1f2e;
  padding: 16px;
  border-radius: 12px;
  box-shadow: 0 0 15px rgba(0, 114, 255, 0.15);
  display: flex;
flex-direction: column;
}
.code-block label {
  margin-bottom: 10px;
  color: #00c6ff;
  font-weight: bold;
  font-size: 1.1rem;
}
textarea {
  background: #0f111a;
  border: 2px solid #0072ff66;
  border-radius: 8px;
  padding: 12px;
  font-size: 0.95rem;
  color: #e0e8f0;
```

```
  resize: vertical;
  min-height: 300px;
  transition: border 0.3s, box-shadow 0.3s;
  width: 100%;
}
textarea:focus {
  border: 1px solid #00c6ff;
  box-shadow: 0 0 10px #00c6ff88;
  outline: none;
}
iframe#output-frame,
pre#output {
  width: 100%;
  height: 100%;
  border-radius: 10px;
  background: #0f111a;
  border: 2px solid #0072ff66;
  color: #8affc1;
  padding: 12px;
  overflow: auto;
  box-shadow: inset 0 0 10px #0072ff33;
}
pre#output {
  white-space: pre-wrap;
  word-wrap: break-word;
}
.output-block {
  background: #1c1f2e;
  padding: 16px;
  border-radius: 12px;
  display: flex;
  flex-direction: column;
}
#java-hint {
  margin-top: 10px;
  background: #fff6e640;
  color: #ffd700;
  padding: 10px;
  border-left: 4px solid #ffd700;
  font-style: italic;
  border-radius: 6px;
}
@media (max-width: 800px) {
  .editor {
    grid-template-columns: 1fr;
  }
}
```

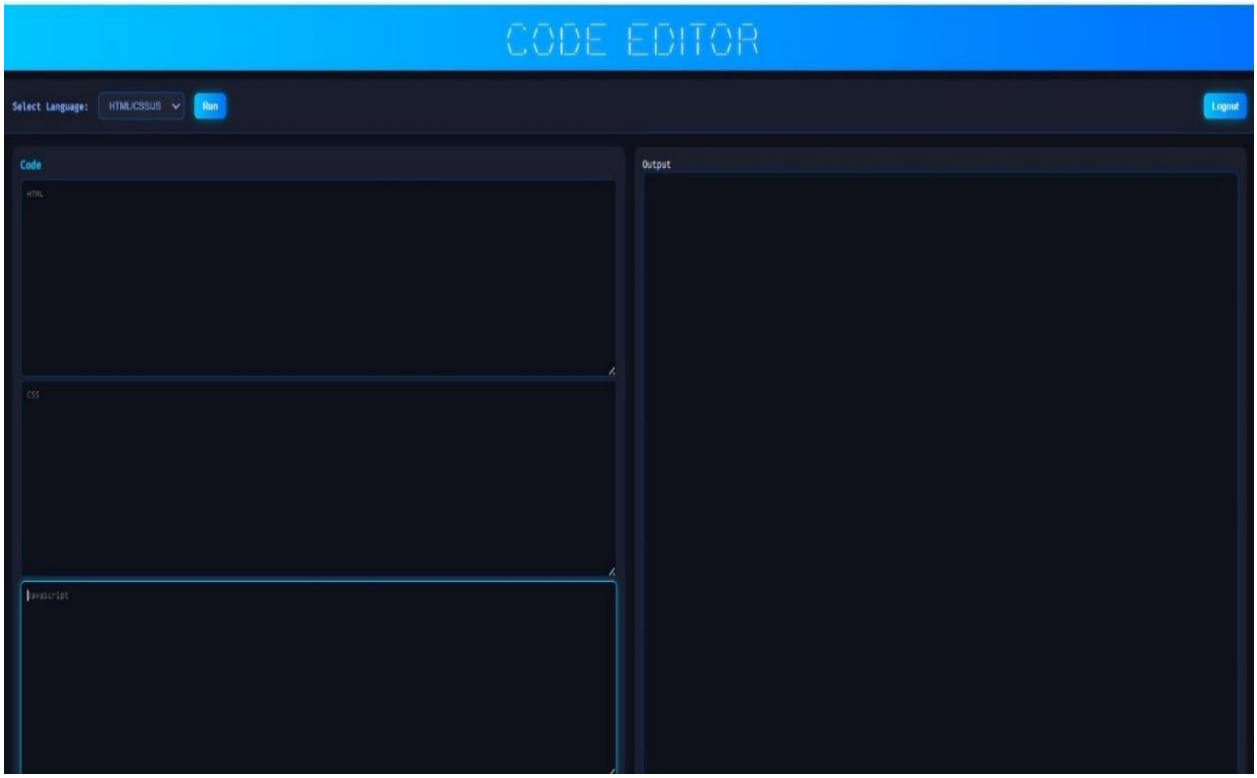
index.html

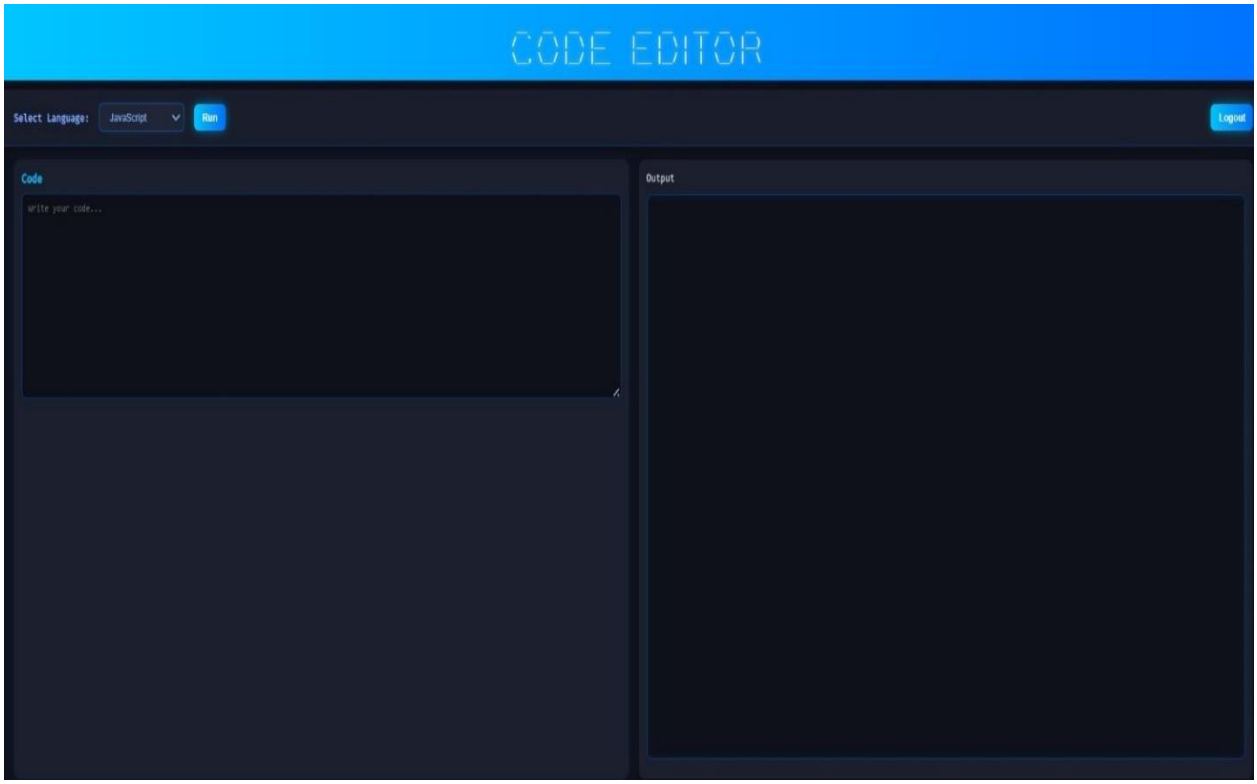
```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
 <title>Universal Code Editor</title>
 <link href="https://fonts.googleapis.com/css2?family=Fira+Code&display=swap"
rel="stylesheet">
 <link rel="stylesheet" href="styles.css">
</head>
<body>
 <pre><header>
  __   __   __   ___   ___  __    ___ __   __
 / ` /  \ |  \ |__    |__  |  \ | |  /  \ |__)
 \__, \__/ |__/ |___   |___ |__/ |  | \__/ |  \
 </header></pre>
 <div class="controls">
  <label for="language">Select Language:</label>
  <select id="language" onchange="toggleEditor()">
   <option value="html">HTML/CSS/JS</option>
   <option value="javascript">JavaScript</option>
   <option value="python">Python</option>
   <option value="c">C</option>
   <option value="cpp">C++</option>
   <option value="java">Java</option>
  </select>
  <button class="run-btn" onclick="runCode()">Run</button>
  <button style="margin-left:auto;" class="run-btn" onclick="logout()">Logout</button>
 </div>
 <div class="editor">
  <div class="code-block">
   <label id="editor-label">Code</label>
   <div id="web-editors" style="display: none;">
    <textarea id="html-code" placeholder="HTML"></textarea><br />
    <textarea id="css-code" placeholder="CSS"></textarea><br />

    <textarea id="js-code" placeholder="JavaScript"></textarea>
   </div>
   <div id="code-editor-block">
    <textarea id="code-editor" placeholder="Write your code..."></textarea>
    <div id="java-hint" style="display: none;"> ⚠️ Java Hint: Class will be renamed to
`Main` if not already.</div>
   </div>
  </div>
  <div class="output-block">
```

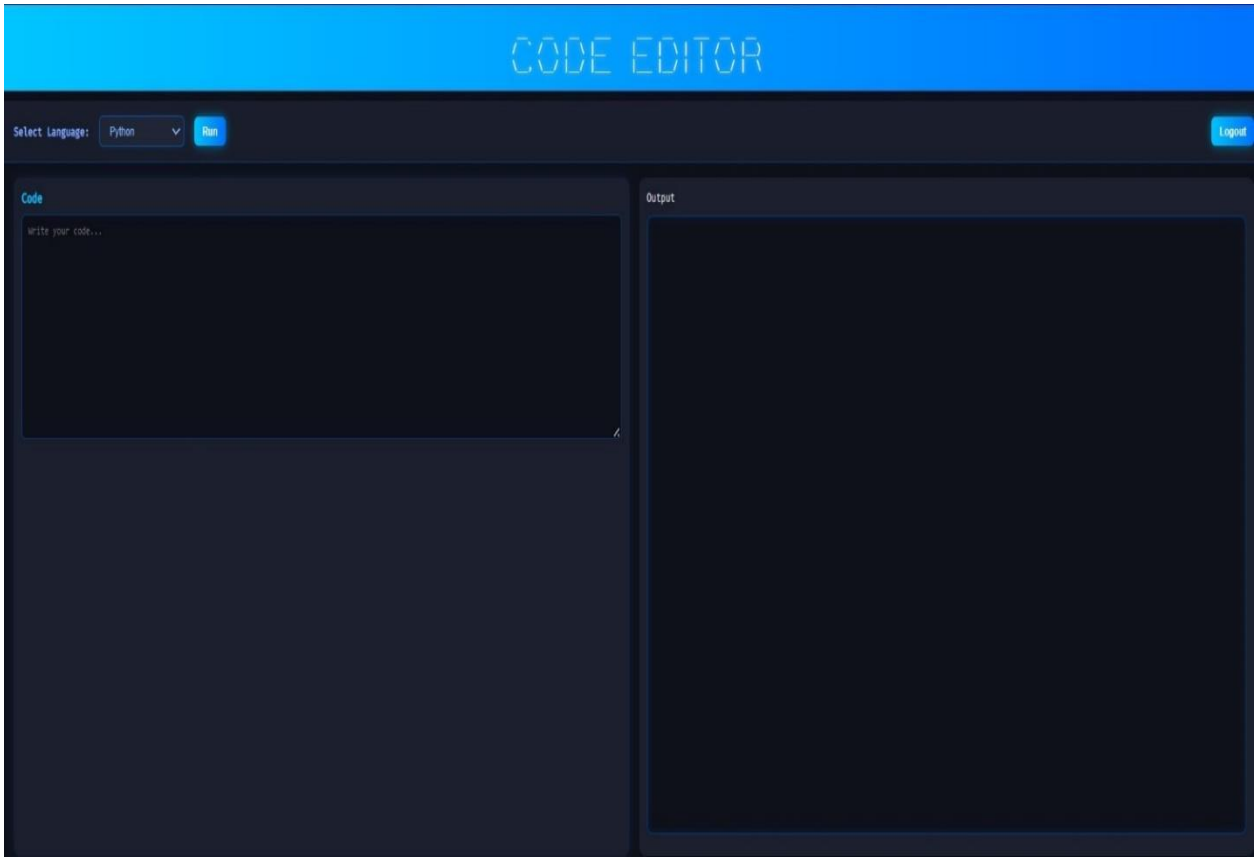
```
    <label>Output</label>
    <iframe id="output-frame" style="display: none;"></iframe>
    <pre id="output"></pre>
   </div>
  </div>
  <script>
   // Redirect to login if not authenticated
if (!localStorage.getItem("user")) {
  window.location.href = "login.html";
}
const langMap = {
    "c": 50,
    "cpp": 54,
    "java": 62,
    "python": 71,
    "javascript": 63
   };
   function toggleEditor() {
    const lang = document.getElementById("language").value;
    const webEditors = document.getElementById("web-editors");
    const codeEditorBlock = document.getElementById("code-editor-block");
    const outputFrame = document.getElementById("output-frame");
    const outputPre = document.getElementById("output");
    const javaHint = document.getElementById("java-hint");
    if (lang === "html") {
     webEditors.style.display = "block";
     codeEditorBlock.style.display = "none";
     outputFrame.style.display = "block";
     outputPre.style.display = "none";
     javaHint.style.display = "none";
    } else {
     webEditors.style.display = "none";
     codeEditorBlock.style.display = "block";
     outputFrame.style.display = "none";
     outputPre.style.display = "block";
     javaHint.style.display = (lang === "java") ? "block" : "none";
    }
   }
   function runCode() {
    const lang = document.getElementById("language").value;
    const output = document.getElementById("output");
    const iframe = document.getElementById("output-frame");
    if (lang === "html") {
     const html = document.getElementById("html-code").value;
     const css = document.getElementById("css-code").value;
     const js = document.getElementById("js-code").value;
     const content = `
      <html>
```

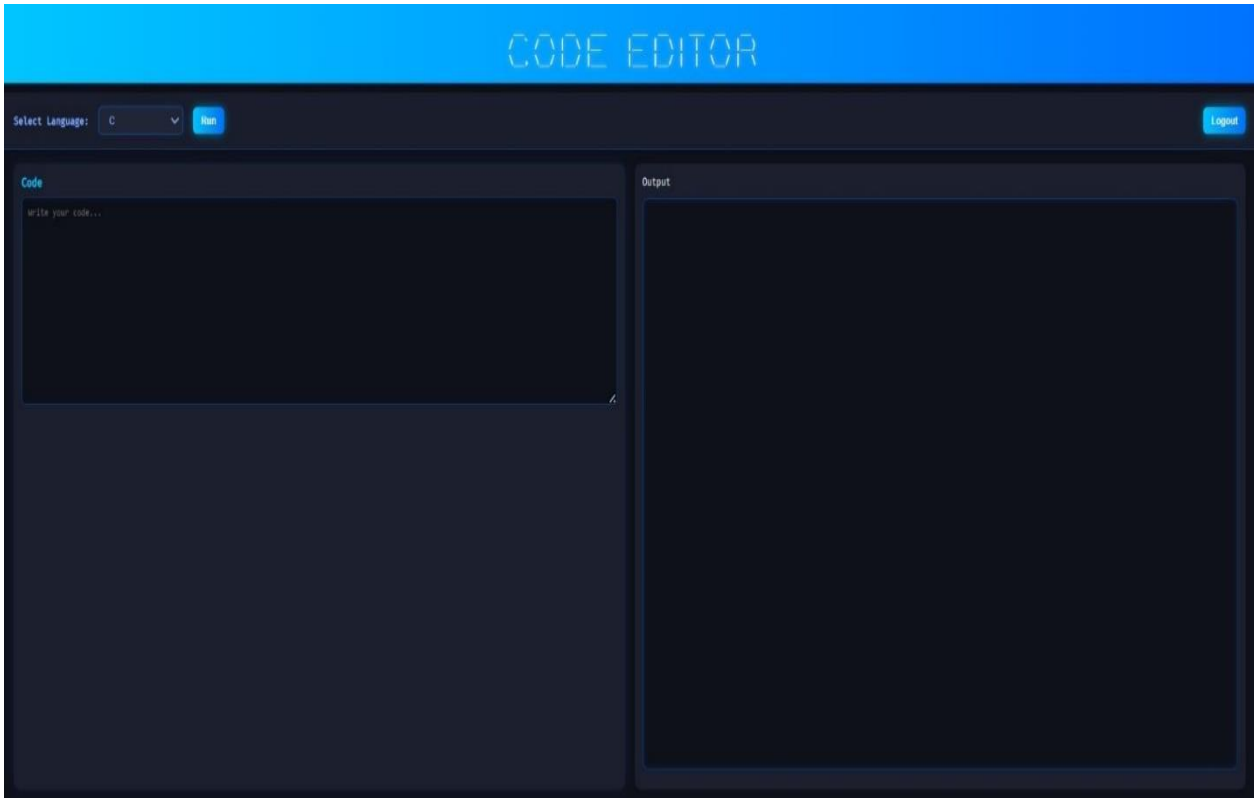
```
          <head><style>${css}</style></head>
          <body>${html}<script>${js}<\/script></body>
        </html>`;
       iframe.srcdoc = content;
      } else {
       let code = document.getElementById("code-editor").value;
       if (lang === "java") {
        const match = code.match(/class\s+(\w+)/);
        if (match && match[1] !== "Main") {
         const className = match[1];
         const regex = new RegExp(`\\b${className}\\b`, "g");
         code = code.replace(regex, "Main");
        }
       }
       output.textContent = "Running...";
       fetch("https://judge0-
ce.p.rapidapi.com/submissions?base64_encoded=false&wait=true", {
        method: "POST",
        headers: {
         "content-type": "application/json",
         "X-RapidAPI-Key": "0d404c0408msh10c6f3dd5852773p179ee6jsn3f8491618dc4",
         "X-RapidAPI-Host": "judge0-ce.p.rapidapi.com"
        },
        body: JSON.stringify({
         source_code: code,
         language_id: langMap[lang]
        })
       })
       .then(res => res.json())
       .then(data => {
        output.textContent = data.stdout || data.stderr || data.compile_output || "No output.";
       })
       .catch(err => {
        output.textContent = "Error: " + err.message;
       });
      }
     }
     function logout() {
  localStorage.removeItem("user");
  window.location.href = "login.html";
}
     window.onload = toggleEditor;
  </script>
</body>
</html>
```

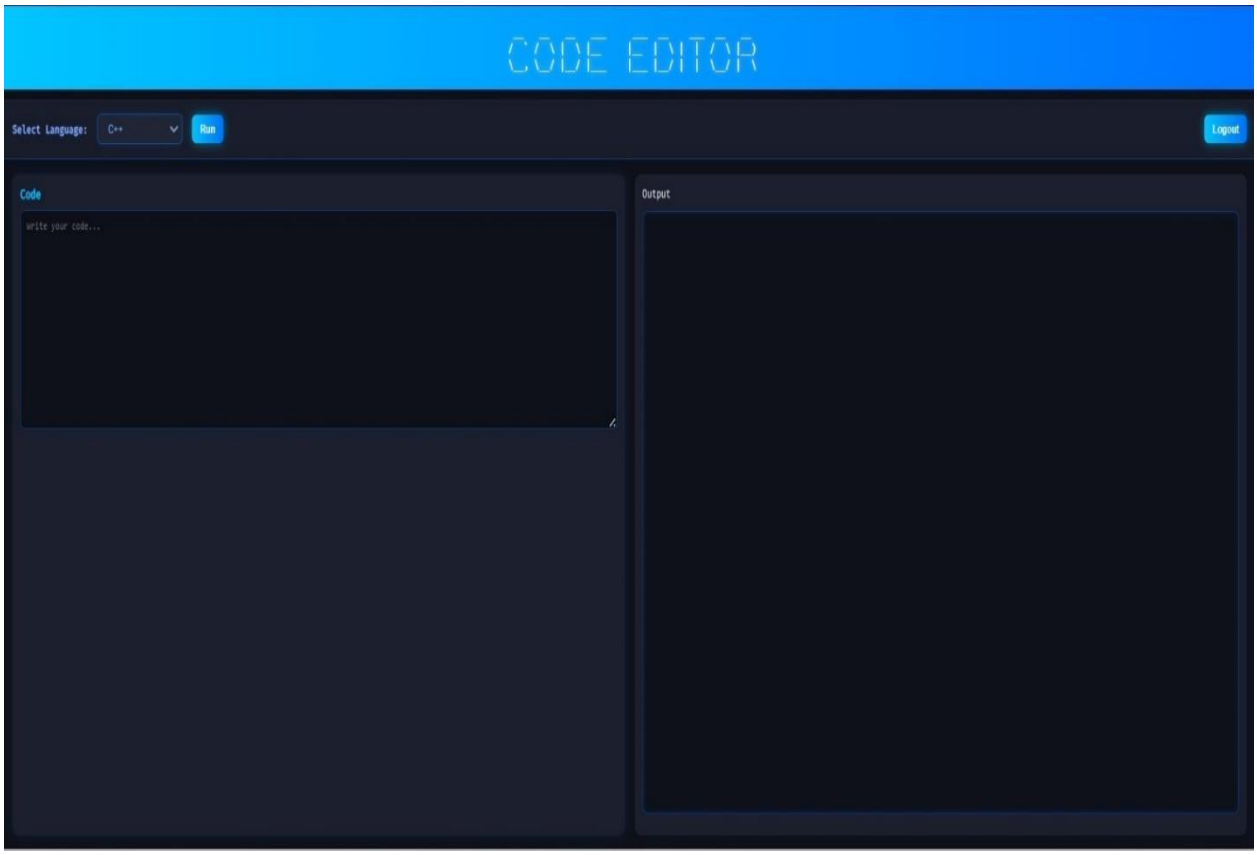
Output

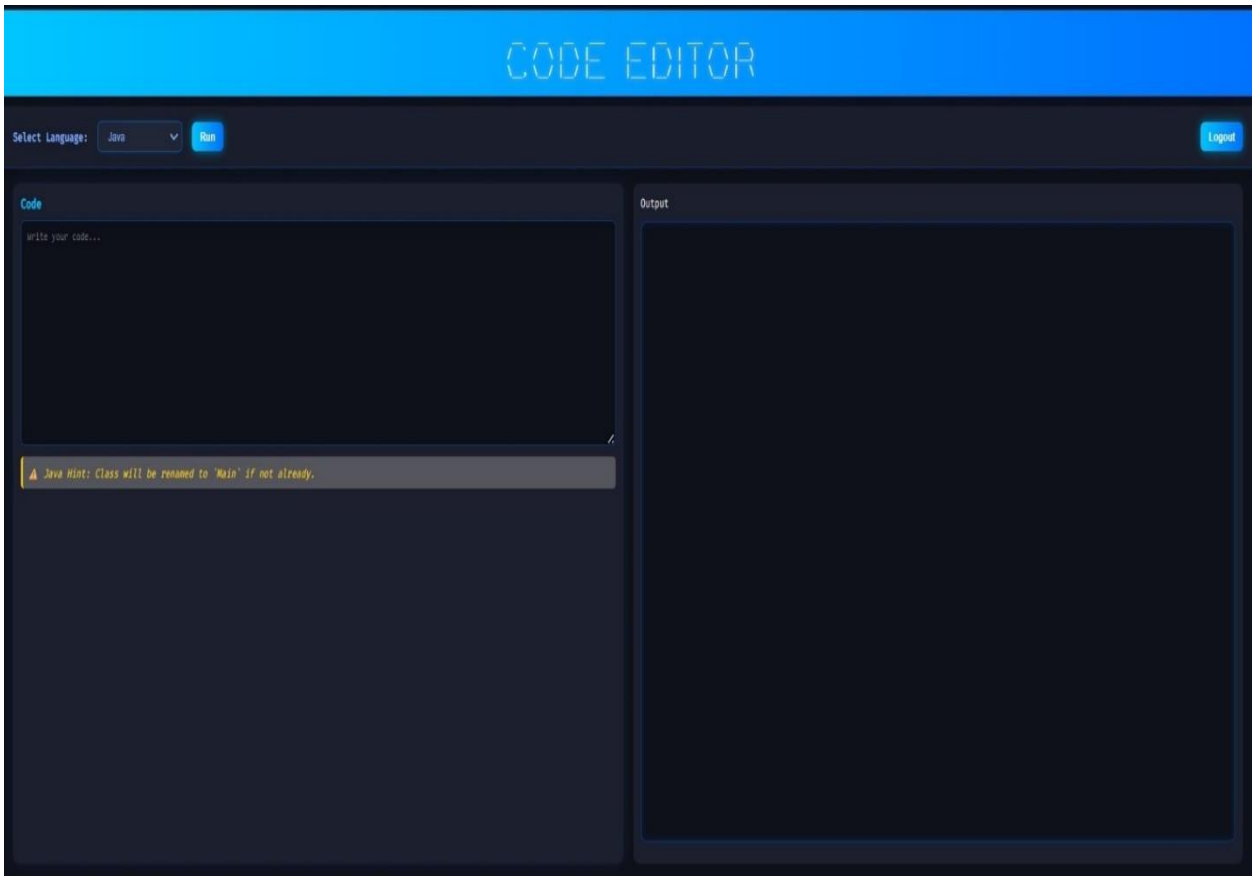

CODE EDITOR
Select Language:
HTML/CSS/JS
Run
Logout
Code
HTML
CSS
JavaScript
Output


CODE EDITOR
Select Language:
JavaScript
Run
Logout
Code
write your code...
Output


CODE EDITOR
Select Language:
Python
Run
Logout
Code
Write your code...
Output


CODE EDITOR
Select Language:
C
Run
Logout
Code
Write your code...
Output


CODE EDITOR
Select Language:
C++
Run
Logout
Code
Write your code...
Output


CODE EDITOR
Select Language:
Java
Run
Logout
Code
Write your code...
Java Hint: Class will be renamed to 'Main' if not already.
Output