**Project Report on**



**ONLINE CODE EDITOR USING MERN STACK**



**Submitted in partial fulfillment of the requirement for the award of the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

**Submitted by:**

**Student Name: Mihul Saxena** **University Roll No. :2018936**

***Under the Mentorship of***

**Mr. Pramod Mehra**

**Assistant Professor**



**Department of Computer Science and Engineering**

**Graphic Era (Deemed to be University)**

**Dehradun, Uttarakhand**

**May 2024**



**CANDIDATE’S DECLARATION**

I hereby certify that the work which is being presented in the project report entitled **“Stock Price Prediction System Using Deep Learning”** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineeringof the Graphic Era (Deemed to be University), Dehradun shall be carried out by the under the mentorship of Mr. **Pramod Mehra, Assistant Professor**, Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun.

Mihul Saxena 2018936

Name University Roll no.

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Description** | **Page No.** |
| Chapter 1 | Introduction | 1 |
| Chapter 2 | Literature Survey | 4 |
| Chapter 3 | Methodology | 6 |
| Chapter 4 | Result and Discussion | 9 |
| Chapter 5 | Conclusion and Future Work | 11 |
|  | References | 13 |

**Chapter 1**

**Introduction**

* 1. **General Introduction**

The Online Code Compiler is a revolutionary web-based platform designed to streamline coding tasks for developers, students, and programming enthusiasts. It offers a versatile environment for writing, testing, and sharing code in real-time, facilitating collaboration and accelerating the development process.

* 1. **Problem Statement**

In today's digital era, the demand for remote collaboration and virtual learning has surged, necessitating the need for reliable online coding environments. Traditional desktop-based Integrated Development Environments (IDEs) often lack the flexibility and accessibility required for modern coding workflows. Moreover, setting up local development environments can be time-consuming and cumbersome, especially for beginners.

* 1. **Technologies**

**1.3.1 React:**

React is a JavaScript library for building user interfaces, developed and maintained by Facebook. It allows developers to create reusable UI components and build complex, interactive web applications with ease. Key features of React include component-based architecture, virtual DOM for efficient rendering, and JSX syntax for defining UI elements within JavaScript code. React's declarative approach to building UIs simplifies state management and enables efficient updates, making it a popular choice for frontend development.

**1.3.2 Express:**

Express.js is a minimal and flexible Node.js web application framework that provides a robust set of features for building web and mobile applications. It offers a lightweight, unopinionated structure for organizing routes, middleware, and handling HTTP requests and responses. Express simplifies the development of server-side logic by providing a concise API and a rich ecosystem of plugins and middleware, making it ideal for building RESTful APIs, web servers, and microservices.

**1.3.3 MongoDB:**

MongoDB is a popular open-source NoSQL database that stores data in flexible, JSON-like documents. It offers scalability, flexibility, and high performance, making it well-suited for handling large volumes of unstructured or semi-structured data. MongoDB's document-oriented data model allows developers to represent complex relationships and hierarchical data structures easily. It supports features such as replication, sharding, and geospatial indexing, making it a versatile choice for modern web applications.

**1.3.4 Node.js:**

Node.js is a server-side JavaScript runtime built on Chrome's V8 JavaScript engine. It enables developers to run JavaScript code outside of a web browser, making it possible to build scalable, high-performance web servers and network applications. Node.js provides an event-driven, non-blocking I/O model that is well-suited for handling concurrent requests and real-time communication. It has a rich ecosystem of npm packages and modules, empowering developers to build a wide range of applications, from command-line tools to full-stack web applications.

**1.3.5 Bull:**

Bull is a powerful, Redis-backed job queue library for Node.js. It enables developers to manage and process background jobs or tasks asynchronously, improving application responsiveness and scalability. Bull provides features such as job scheduling, retries, and prioritization, making it suitable for implementing complex job processing workflows. It integrates seamlessly with Redis, a high-performance in-memory data store, to store job data and manage job queues efficiently.

**1.3.6 Ace Editor:**

Ace Editor is a lightweight, extensible code editor written in JavaScript. It offers syntax highlighting, code folding, and auto-completion features, making it suitable for building code editing interfaces in web applications. Ace Editor supports a wide range of programming languages and file formats, making it versatile for various development environments. It provides an easy-to-use API for customizing editor behavior and appearance, allowing developers to create rich code editing experiences.

**1.3.7 Bootstrap:**

Bootstrap is a popular open-source CSS framework for building responsive and mobile-first web applications. It provides a collection of pre-designed UI components, such as buttons, forms, and navigation bars, along with a grid system for layout organization. Bootstrap's CSS and JavaScript components streamline frontend development and ensure consistency and compatibility across different browsers and devices. It offers customization options and theming capabilities, allowing developers to create visually appealing and accessible web interfaces efficiently.

**1.3.8 Moment.js:**

Moment.js is a lightweight JavaScript library for parsing, validating, manipulating, and formatting dates and times. It simplifies date and time handling in web applications by providing a robust set of functions for common date operations, such as calculating differences between dates, formatting dates according to specific locales, and parsing dates from various string formats. Moment.js's intuitive API and extensive documentation make it a go-to choice for handling date and time-related tasks in JavaScript applications.

**1.3.9 Axios:**

Axios is a popular promise-based HTTP client for making asynchronous HTTP requests in web browsers and Node.js environments. It provides a simple and intuitive API for performing HTTP operations, such as GET, POST, PUT, and DELETE, and supports features such as request and response interception, error handling, and request cancellation. Axios's flexibility and ease of use make it a preferred choice for handling AJAX requests, interacting with RESTful APIs, and fetching data from external services in frontend and backend applications.

**1.3.10 Cors:**

Cross-Origin Resource Sharing (CORS) is a security feature that controls access to resources on different origins in web browsers. It operates through HTTP headers, with servers specifying allowed origins using the **Access-Control-Allow-Origin** header. CORS prevents unauthorized cross-origin requests by evaluating the requesting origin against the allowed origins. Additional headers like **Access-Control-Allow-Methods** and **Access-Control-Allow-Headers** provide fine-grained control over cross-origin requests, enhancing security against vulnerabilities like XSS and CSRF.

**Chapter 2**

**Literature Survey**

**2.1 Technical:**

The field of online code compilers has seen significant growth and innovation in recent years, driven by advancements in web development technologies and the increasing demand for remote coding solutions. This section explores existing literature and related work in the domain of online coding environments, highlighting key features, functionalities, and trends observed in popular platforms.

Several online code compilers have gained prominence in the developer community, including platforms like CodePen, JSFiddle, Replit, and gdpCompiler, each offering unique features tailored to specific use cases and programming languages. These platforms provide a range of functionalities, such as real-time collaboration, version control, and integration with external libraries and APIs.

Research studies and surveys have also shed light on the benefits of online coding environments for education, professional development, and open-source collaboration. Studies have shown that online code compilers can enhance learning outcomes, improve code quality, and foster a sense of community among developers through collaborative coding sessions and code sharing.

**2.2 Financial:**

From a financial perspective, online code compilers offer several advantages over traditional development environments, particularly in terms of cost savings and resource optimization. By leveraging cloud-based infrastructure and pay-as-you-go pricing models, organizations can significantly reduce upfront capital expenditures associated with setting up and maintaining on-premises development servers.

Furthermore, online code compilers enable efficient resource utilization by dynamically allocating compute resources based on demand, thereby minimizing idle capacity and maximizing cost efficiency. This elasticity allows organizations to scale their development operations seamlessly, adapting to fluctuating workloads and project requirements.

Additionally, online code compilers democratize access to programming tools and resources, particularly for individuals and organizations with limited financial resources. By providing free or affordable access to robust coding environments, these platforms empower learners, educators, and developers from diverse backgrounds to pursue their passion for coding and software development.

**Chapter 3**

**Methodology**

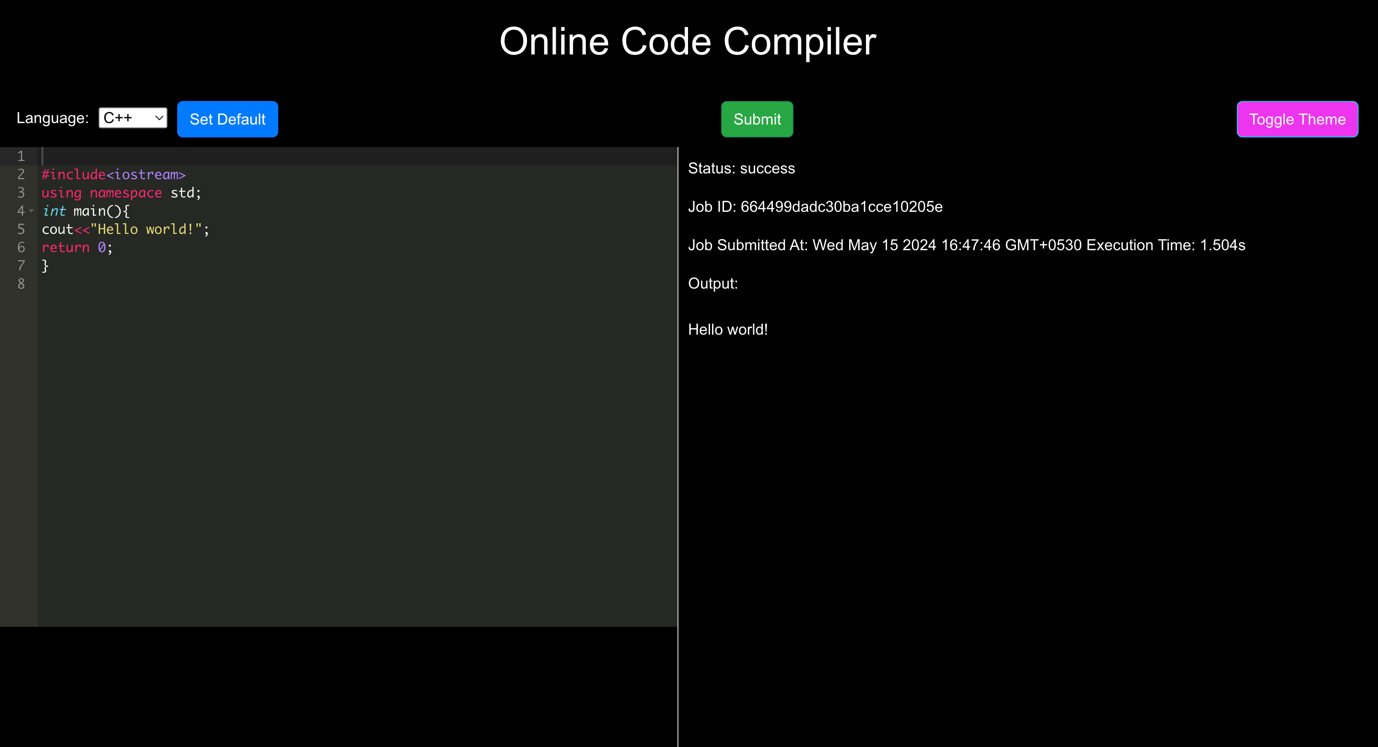
The Online Code Compiler project follows a systematic approach to design, development, and deployment, guided by established methodologies and best practices in software engineering. This section outlines the methodology adopted for building the frontend and backend components of the application, as well as the tools, frameworks, and libraries used in the development process.

1. App.js:
   * Description: App.js serves as the main entry point for the Online Code Compiler application. It is built using React.js, a popular JavaScript library for building user interfaces.
   * Functionality:
     + State Management: Utilizes React hooks such as useState and useEffect for managing component state and performing side effects.
     + Code Compilation: Handles user input, language selection, and code submission logic using Axios for making HTTP requests to the backend server.
     + Theme Toggle: Implements a theme toggle feature to switch between light and dark themes for the code editor interface.
   * Dependencies: Utilizes external libraries such as Ace Editor for code editing functionality and Moment.js for time formatting
2. index.js (Backend):
   * Description: index.js serves as the entry point for the backend server, which is built using Node.js and Express.js.
   * Functionality:
     + API Endpoints: Defines routes for handling code execution requests, status inquiries, and health checks.
     + Database Connectivity: Establishes a connection to MongoDB using Mongoose for storing job details and user data.
     + Error Handling: Implements error handling middleware to catch and respond to client and server errors appropriately.
   * Dependencies: Requires dependencies such as Express.js, Mongoose for MongoDB interaction, and Cors for enabling cross-origin resource sharing.
3. generateFile.js:
   * Description: generateFile.js is a utility module responsible for generating temporary code files for execution.
   * Functionality:
     + File Generation: Generates unique file names using UUID and writes code content to the file system for execution.
     + Filepath Retrieval: Returns the filepath of the generated code file for further processing.
   * Dependencies: Does not have external dependencies and relies on core Node.js modules for file system operations.
4. jobQueue.js:
   * Description: jobQueue.js manages the job execution queue using Bull, a Redis-backed job queue library for Node.js.
   * Functionality:
     + Queue Processing: Defines a Bull queue instance and processes incoming job requests asynchronously using worker processes.
     + Job Execution: Executes code files based on language type (e.g., C++ or Python) and updates job status and output accordingly.
     + Error Handling: Implements error handling mechanisms to handle job execution failures and retries.
   * Dependencies: Requires Bull for queue management and job processing, as well as Mongoose for interacting with the MongoDB database.
5. executeCpp.js & executePy.js:
   * Description: executeCpp.js and executePy.js are modules responsible for executing C++ and Python code files, respectively.
   * Functionality:
     + Code Execution: Invokes the appropriate compiler or interpreter (e.g., g++ for C++ or python3 for Python) to execute the code file.
     + Output Retrieval: Captures the stdout output of the executed code and returns it for further processing.
     + Error Handling: Handles compilation or runtime errors gracefully and provides detailed error messages for debugging purposes.
   * Dependencies: Does not have external dependencies and relies on core Node.js modules for child process execution.

This comprehensive methodology encapsulates the architectural components and the complete structure of this software.

**Chapter 4**

**Result and Discussion**

****

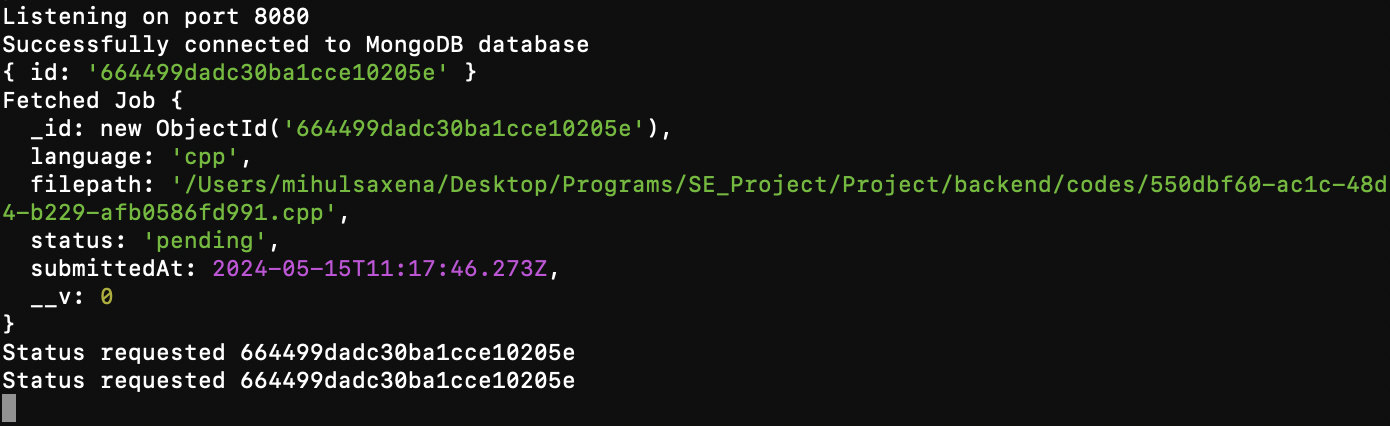
**A screenshot of a computer

Description automatically generated**

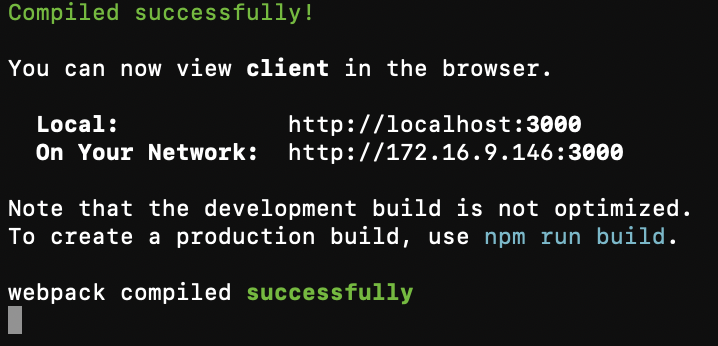
The Online Code Compiler project has successfully delivered a fully functional and user-friendly online coding environment, enabling users to write, compile, and execute code in real-time. This section presents the results of the project implementation, including screenshots, performance metrics, and user feedback collected during testing and evaluation.

The frontend of the application provides an intuitive interface for selecting programming languages, writing code in the provided editor, and submitting code for execution. Real-time status updates are displayed to users, including job ID, execution time, and output, enhancing the user experience and workflow efficiency.

The backend of the application efficiently handles code compilation and execution tasks, utilizing job queues and asynchronous processing to ensure optimal performance and scalability. The integration with MongoDB enables seamless storage and retrieval of job details, while Bull.js provides robust job queue management capabilities, ensuring reliable execution of code jobs.

****

**­­­­­Running index.js file for backend operations**

****

**Running App.js file to start the react-app**

**Chapter 5**

**Conclusion and Future Work**

In conclusion, the Online Code Compiler project demonstrates the feasibility and effectiveness of building a modern and versatile online coding environment using contemporary web technologies and best practices in software engineering. The application offers a range of features and functionalities tailored to the needs of developers, students, and educators, providing a valuable resource for code development, testing, and collaboration.

Looking ahead, there are several opportunities for future enhancements and improvements to further enhance the functionality, usability, and performance of the Online Code Compiler. This section outlines potential areas for future work, including support for additional programming languages, integration with external APIs and services, and optimization of code execution and resource utilization.

**References**

1. React.js Documentation: <https://react.dev/learn>
2. Node.js Documentation: <https://nodejs.org/en>
3. Express.js Documentation: <https://expressjs.com/>
4. MongoDB Documentation: <https://docs.mongodb.com/>
5. Bull.js Documentation: <https://optimalbits.github.io/bull/>
6. Axios Documentation: <https://optimalbits.github.io/bull/>
7. React Ace Documentation: <https://www.npmjs.com/package/react-ace>
8. YouTube:

a. freecodecamp: <https://www.youtube.com/c/Freecodecamp>

b. <https://youtu.be/RZ66yGyEKFg?si=Hid4M0lZccOmIqDi>

c. <https://youtu.be/6tB7diwHZYQ?si=rJsM-2qRRgLD9hOV>