AWS Cloud9 for Cloud-Based Integrated Development Environment (IDE)

*(13 size) A Project Based Learning Report Submitted in partial fulfilment of the requirements for the award of the degree*

*of*

**Bachelor of Technology**

**in the Department of Computer Science & Engineering**

**Cloud Based AI/ML Speciality (22SDCS07A)**

Submitted by

**2210030252: RAKESH PENUGONDA**

Under the guidance of

**Ms. P. Sree Lakshmi**



Department of Computer Science and Engineering

Koneru Lakshmaiah Education Foundation, Aziz Nagar

Aziz Nagar – 500075

March - 2025.

**Introduction**

This project is an advanced web-based compiler that integrates AWS Cloud9 as a cloud-based development environment. It provides users with a seamless coding experience by leveraging modern web technologies and cloud infrastructure. The primary goal of this system is to allow users to write, edit, and execute code in a highly efficient and collaborative manner. The frontend of the system is built using React with Vite, ensuring a fast and responsive user interface.

It incorporates the Monaco Editor, which offers syntax highlighting, code suggestions, and real-time error detection. Users can write code in multiple programming languages, primarily JavaScript and Python, and submit it for execution through the backend. The backend, developed with Node.js and Express, handles code execution, user authentication, and session management. To facilitate secure authentication, JSON Web Tokens (JWT) are used, while user data and session logs are stored in either MongoDB or PostgreSQL.

The backend is responsible for executing JavaScript code using child processes and running Python scripts through the Python-Shell package. AWS services such as Amazon S3 are used for storing user code, and DynamoDB keeps track of user sessions and active instances. Real-time collaboration is made possible through WebSockets, allowing multiple users to interact with the same codebase simultaneously. This compiler system is designed for scalability, efficiency, and security, making it an ideal solution for developers looking for a cloud-based coding platform.

**Methodology**

The project follows a structured approach to ensure efficient execution and seamless user interaction. The frontend is designed to be lightweight, interactive, and user-friendly, using React with Vite for optimized performance.

The Monaco Editor serves as the primary interface where users can write and edit code. Axios is used to handle API requests, ensuring smooth communication between the frontend and backend. Authentication is managed using cookies, which store the JWT token to maintain user sessions securely. Upon logging in, users can access their saved code from Amazon S3 and continue working without interruptions.

The backend is implemented using Node.js and Express to handle API requests, process user code, and maintain session data. The code execution mechanism utilizes a combination of child processes and Python-Shell to ensure safe and isolated execution environments. WebSockets are integrated to enable real-time collaboration, allowing users to share and edit code simultaneously. To optimize performance and reduce costs, AWS Lambda is used to terminate inactive AWS Cloud9 instances, ensuring that resources are only used when necessary.

Database management is handled by MongoDB or PostgreSQL, depending on the deployment requirements. The server securely interacts with the database to store user credentials, session data, and metadata related to the code files. Security is a major consideration, with CORS policies enforced to prevent unauthorized access and bcrypt.js used for hashing user passwords. The entire system is built to be modular, ensuring that components can be independently updated or replaced without affecting overall functionality.

**Implementation**

The frontend is implemented with React.js, providing a dynamic and interactive UI. The Monaco Editor is embedded to create a coding environment similar to VS Code. Tailwind CSS is used for styling, ensuring responsiveness and a clean design. The frontend communicates with the backend through Axios, handling authentication, code execution, and data retrieval.

On the backend, Express.js manages API routes, handling requests related to user authentication, compilation, and real-time collaboration. The authentication system utilizes JWT tokens, which are generated at login and stored securely using js-cookie. MongoDB stores user credentials and history, while PostgreSQL manages structured data related to user activities.

The real-time collaboration feature is powered by Socket.IO, enabling multiple users to write and edit code simultaneously. Each coding session is assigned a unique identifier, allowing users to join shared sessions securely. The server listens for code updates and broadcasts them to all connected clients, ensuring synchronization.

For code execution, the backend processes code in a controlled environment. Python code is executed using Python-Shell, while other languages run in separate processes. Execution timeouts and input validation mechanisms are in place to prevent malicious code execution and server overloading.

AWS Cloud9 is configured as the cloud-based development environment. Setting it up involves logging into the AWS Management Console, selecting Cloud9, and creating an environment with an EC2 instance. The development tools are pre-installed, and any additional dependencies can be managed using npm, pip, or apt-get. The terminal allows users to test and debug code efficiently.

Security is a major aspect of the implementation. IAM roles and permissions ensure that only authorized users can access Cloud9. AWS CloudTrail is enabled to monitor activities, and Multi-Factor Authentication (MFA) is enforced to enhance security. Public access is restricted through IAM policies, and automatic session timeouts help prevent unauthorized usage.

To validate the implementation, developers test the setup by writing and running sample programs in Cloud9. The IAM access control is verified by testing user roles, and version control is checked through AWS CodeCommit. CloudWatch logs are reviewed to identify errors or performance issues.

By combining React.js, Node.js, Express.js, MongoDB, PostgreSQL, AWS Cloud9, and Socket.IO, this project delivers a powerful cloud-based compiler that enables developers to write, compile, and collaborate on code efficiently and securely.