# CLOUD BASED AI TRAINING PLATFORM

**A DESIGN PROJECT REPORT**

***submitted by***


**R. ARAVINTH KRISHNA**

**K. ARUN**

**S. BALAJI**


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

***of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER SCIENCE AND ENGINEERING**


**K RAMAKRISHNAN COLLEGE OF TECHNOLOGY**

**(An Autonomous Institution, affiliated to Anna University Chennai, Approved by AICTE, New Delhi)**

**Samayapuram – 621 112**


**JUNE 2025**

# CLOUD BASED AI TRAINING PLATFORM

**A DESIGN PROJECT REPORT**

***submitted by***


**R. ARAVINTH KRISHNA (811722104012)**

**K. ARUN (811722104016)**

**S. BALAJI (811722103302)**


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

***of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER SCIENCE AND ENGINEERING**


**K RAMAKRISHNAN COLLEGE OF TECHNOLOGY**

**(An Autonomous Institution, affiliated to Anna University Chennai, Approved by AICTE, New Delhi)**

**Samayapuram – 621 112**


**JUNE 2025**

# K RAMAKRISHNAN COLLEGE OF TECHNOLOGY

## (AUTONOMOUS)

**SAMAYAPURAM – 621 112**

## BONAFIDE CERTIFICATE

Certified that this project report titled **"CLOUD BASED AI TRAINING PLATFORM"** is Bonafide work of **R. ARAVINTH KRISHNA (811722104012), K. ARUN (811722104016), S. BALAJI (811722104302)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion or on this or any other candidate.

**SIGNATURE**

Dr. A Delphin Carolina Rani, M.E., Ph.D.,

**HEAD OF THE DEPARTMENT**

PROFESSOR

Department of CSE

K. Ramakrishnan College of Technology

(Autonomous)

Samayapuram – 621 112

**SIGNATURE**

Mr. D. P. Devan, M.E.,

**SUPERVISOR**

Assistant Professor

Department of CSE

K. Ramakrishnan College of Technology

(Autonomous)

Samayapuram – 621 112

Submitted for the viva-voice examination held on ……

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

# DECLARATION

We jointly declare that the project report on **"CLOUD BASED AI TRAINING PLATFORM"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of Bachelor of Engineering. This project report is submitted on the partial fulfillment of the requirement of the award of Degree of Bachelor of Engineering.

**Signature**

R. ARAVINTH KRISHNA

K. ARUN

S. BALAJI

Place: Samayapuram

Date:

# ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and indebtedness to our institution **"K RAMAKRISHNAN COLLEGE OF TECHNOLOGY"**, for providing us with the opportunity to do this project.

We are glad to credit and praise our honorable and respected chairman sir **Dr. K RAMAKRISHNAN, B.E.,** for having provided for the facilities during our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S KUPPUSAMY, MBA, Ph.D.,** for forwarding our project and offering adequate duration to complete it.

We would like to thank **Dr. N VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave the opportunity to frame the project with full satisfaction.

We heartily thank Dr. **A DELPHIN CAROLINA RANI, M.E., Ph.D.,** Head of the Department, **COMPUTER SCIENCE AND ENGINEERING** for providing her support to pursue this project.

We express our deep and sincere gratitude and thanks to our project guide **Mr. D. P. DEVAN, M.E.,** Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry on this project.

We give our sincere thanks to the Course Coordinator and other staff members for providing valuable information during the course. We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

# ABSTRACT

The exponential rise in demand for artificial intelligence (AI) development and training environments has highlighted the need for scalable, accessible, and cost-efficient platforms. This project, titled "Cloud Based AI Training Platform," introduces a cloud-native web-based Integrated Development Environment (IDE) that enables users to write, train, and execute AI models directly from their browsers. The system is designed to mirror the capabilities of leading platforms like Amazon SageMaker Studio by integrating a VS Code–like interface with real-time performance monitoring and AI-powered assistance.

The platform employs a serverless architecture using AWS Lambda for code execution, AWS Cognito for secure user authentication, and DynamoDB for persistent storage. The front-end is developed using Next.js 15 with TypeScript and CSS Modules, ensuring a modern and responsive user experience. A dedicated AI chatbot, powered by GROQ AI, provides contextual coding support with features like fix suggestions, optimizations, and explanations.

Additional functionalities include a user profile management system, session-based resource tracking, PDF invoice generation, and a pay-as-you-go usage calculator that offers 86,000 seconds of free compute time. The architecture supports dynamic scaling and offers the flexibility to extend runtime support to other machine learning languages such as R and Julia.

This platform aims to democratize AI model training by reducing the entry barrier for learners, developers, and researchers, offering a robust, intuitive, and efficient environment for cloud-based AI experimentation and deployment.

# TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF ABBREVIATIONS

| ABREVIATION | FULL FORM |
|---|---|
| AI | Artificial Intelligence |
| IDE | Integrated Development Environment |
| CSS | Cascading Style Sheets |
| DB | Database |
| AWS | Amazon Web Services |
| UI | User Interface |
| CPU | Central Processing Unit |
| GROQ | General-purpose Retrieval and Optimization Querying AI |
| GPT | Generative Pre-trained Transformer |
| CDN | Content Delivery Network |
| API | Application Programming Interface |

# CHAPTER 1

# INTRODUCTION

## 1.1 OVERVIEW

In recent years, the rapid growth of artificial intelligence (AI) has created an increasing demand for scalable, accessible, and efficient development environments. Traditional setups for AI model training often require powerful local machines, extensive configuration, and substantial resource investment, making them impractical for students, researchers, and small enterprises. To address these challenges, this project presents a **Cloud Based AI Training Platform** — a browser-accessible, cloud-native Integrated Development Environment (IDE) designed specifically for AI training and development.

This platform combines the flexibility of modern web frameworks with the computational power of cloud services to provide a seamless coding and training experience. Inspired by platforms like **Amazon SageMaker Studio**, the system integrates features such as real-time code execution, intelligent assistance, and usage-based billing to deliver a comprehensive and user-friendly AI development ecosystem. It supports dynamic code execution using AWS Lambda, real-time resource monitoring, secure user authentication via AWS Cognito, and persistent data storage using DynamoDB.

The platform is designed to simplify the AI model training workflow, enabling users to focus more on innovation and less on infrastructure. It promotes accessibility by eliminating the need for physical installations and expensive hardware, making AI development more inclusive and widely available.

## 1.2 PROBLEM STATEMENT

The existing ecosystem for AI model training is often limited by high hardware requirements, complex setups, and lack of scalability. Many developers and learners face barriers such as limited computing resources, insufficient technical expertise to manage environments, and high operational costs. Additionally, traditional local IDEs do not provide real-time collaboration, remote accessibility, or integrated analytics, which are essential in modern AI workflows.

This project aims to resolve these issues by offering a cloud-native platform that abstracts infrastructure complexities and provides a fully managed environment for AI development. It allows users to write, execute, and monitor their code online, with intelligent guidance and flexible compute scaling, ensuring an efficient, collaborative, and scalable solution.

## 1.3 OBJECTIVE

The main objective of the **Cloud Based AI Training Platform** is to design and implement a scalable, accessible, and intelligent cloud-native web IDE that enables users to develop, train, and execute AI models with ease. The platform aims to bridge the gap between complex AI workflows and user-friendly, low-cost solutions suitable for learners, researchers, and professionals.

- Develop a browser-based IDE for AI training.
- Enable serverless code execution using AWS Lambda.
- Integrate AI chatbot for real-time coding assistance.
- Implement secure login and basic usage tracking.
- Ensure cross-platform accessibility.

# CHAPTER 2

# LITERATURE SURVEY

## 1. "Modern Integrated Development Environment (IDEs)"

**Authors:** G. Fylaktopoulos et al. (2016)

**Modern Integrated Development Environment (IDEs)** by G. Fylaktopoulos et al. (2016) offers a thorough overview of the evolving landscape of cloud-based software development tools, focusing on integrated development environments that operate fully or partially in the cloud. The paper explores various architectural approaches for cloud IDEs, discussing how these platforms enable seamless development workflows by integrating programming, database, version control, and deployment tools into a single browser-accessible environment. It highlights the growing trend towards supporting multiple programming languages and frameworks within a single environment, enabling developers to work flexibly across technologies.

Beyond simple code editing, the survey emphasizes advanced productivity features such as intelligent code completion, debugging, and real-time collaboration, which are essential for modern distributed teams. Furthermore, the study covers complementary tools integrated into cloud IDEs, including software modeling and documentation utilities, which help maintain software quality and streamline design processes. Additionally, the paper discusses how modern cloud IDEs extend their functionality to application lifecycle management, incorporating orchestration and deployment tools that simplify the management of cloud-hosted applications. These comprehensive insights underscore the critical role of cloud-based IDEs in enabling efficient, scalable, and collaborative software development—an essential foundation for building AI training platforms that require powerful, accessible, and integrated development environments.

2. **"Comprehensive Review of Performance Optimization Strategies for Serverless Applications on AWS Lambda"**

   **Authors:** Mohamed Lemine El Bechir, Cheikh Sad Bouh, and Abobakr Shuwail (2024)

**Comprehensive Review of Performance Optimization Strategies for Serverless Applications on AWS Lambda** by Mohamed Lemine El Bechir, Cheikh Sad Bouh, and Abobakr Shuwail (2024) delves into the challenges and solutions related to optimizing serverless computing environments, with a particular focus on AWS Lambda. It synthesizes the latest research on performance bottlenecks, such as cold start latency, which can severely impact the responsiveness of serverless functions, especially in real-time or resource-intensive applications like AI model training. The review examines techniques to reduce these delays through strategies like function pre-warming and provisioned concurrency, which ensure faster invocation times.

It also discusses the importance of efficient resource management, highlighting adaptive resource allocation based on workload profiling to balance performance and cost. The paper reviews the impact of different runtime environments on function execution and advocates for selecting or customizing runtimes to minimize startup overhead. Observability and monitoring are also key themes, as they provide the necessary insights to fine-tune performance and detect anomalies in serverless applications.

The authors further explore workload-aware execution patterns, emphasizing how understanding invocation triggers and usage can optimize throughput and reduce unnecessary executions. Cost efficiency is a recurring focus, with recommendations on function consolidation and scheduling that minimize cloud expenses without sacrificing responsiveness.

### 3. A Survey on Cloud Computing Security: Issues, Threats, and Solutions

**Authors:** Subashini, S., & Kavitha, V. (2011)

This influential survey provides an exhaustive overview of security concerns, threat vectors, and proposed mitigation techniques in the realm of cloud computing. The authors classify the core issues into categories such as data security, network security, virtualization-related threats, and compliance challenges. They argue that the abstraction and multi-tenancy of cloud environments, while enhancing scalability and flexibility, also introduce new vulnerabilities—particularly in shared infrastructure and virtual machine (VM) isolation. One significant insight is the risk of side-channel attacks and VM hopping, where an attacker on the same physical host can potentially gain access to neighbouring virtual machines.

To address these threats, the paper evaluates technologies like advanced encryption for data at rest and in transit, secure virtualization layers (e.g., using hypervisors like Xen and KVM), and fine-grained access controls using public key infrastructure (PKI) and federated identity management. The survey also explores standards and best practices including Service Level Agreements (SLAs) with embedded security clauses and regular third-party audits. In the context of developing a cloud-based AI training platform, these concerns are especially relevant. AI applications often handle confidential datasets, proprietary models, and user interactions, all of which must be shielded from unauthorized access or leakage. Moreover, the paper emphasizes the need for continuous threat modelling and real-time monitoring, which are essential as AI training pipelines typically involve data ingestion, transformation, and model training at scale.

## 4. A Survey on Serverless Computing: Platforms and Applications

**Authors:** Jonas, E., Schleier-Smith, J., Sreekanti, V., et al. (2019)

This comprehensive survey addresses the rising adoption of serverless computing, specifically Function-as-a-Service (FaaS), and its implications for modern cloud applications. The authors present a detailed analysis of how serverless platforms shift operational complexity away from developers, enabling code to run in response to events without provisioning or managing servers. However, the paper also identifies several limitations—such as cold start latency (delays during the first invocation of a function), limited execution time (typically under 15 minutes), and difficulty maintaining state across function calls.

The authors explore strategies to overcome these constraints through techniques like function pre-warming, state externalization via databases or object stores, and coupling FaaS with orchestration frameworks like Apache OpenWhisk and AWS Step Functions. Specifically, it underscores how serverless is ideal for AI workflows that require high parallelism and event-driven computation, such as training model segments or running inference on incoming data streams. For developers of cloud-based AI training platforms, the findings are especially pertinent. Serverless architecture enables scalable training pipelines triggered by data upload or user interaction, with optimized resource usage and reduced costs. The decoupling of compute from infrastructure management also aligns with the needs of browser-based AI systems that must support multiple concurrent users and adapt dynamically to workload variations. Ultimately, this survey offers strategic direction for leveraging serverless paradigms in building responsive and resource-efficient AI systems. Additionally, the paper highlights emerging trends such as serverless machine learning pipelines and the integration of GPUs in FaaS environments.

## 5. Cloud-Based Machine Learning Platforms: A Review

**Authors:** Zaharia, S., Chen, A., Davidson, A., et al. (2020)

This comprehensive review surveys the capabilities and architectures of leading cloud-based machine learning platforms, including Google AI Platform, Amazon SageMaker, Microsoft Azure Machine Learning, and several open-source alternatives. The authors provide a structured evaluation of each platform based on features that support the full machine learning lifecycle—from data acquisition and preprocessing to model training, tuning, evaluation, deployment, and monitoring. Central to their analysis is the role of infrastructure abstraction: these platforms allow users to access high-performance computing environments (such as GPU/TPU clusters) without manual configuration, thus democratizing access to advanced AI capabilities.

The paper also discusses the increasing role of AutoML, which automates model selection and hyperparameter tuning, and MLOps practices that enforce version control, reproducibility, and continuous integration/continuous deployment (CI/CD). One of the most insightful discussions in the paper is the trade-off between flexibility and ease of use. The authors highlight emerging trends such as hybrid-cloud deployment models and interoperability between platforms using containers (e.g., Docker) and orchestration tools (e.g., Kubernetes). For a browser-based AI training platform, the practical implications are significant: cloud ML platforms offer not only technical infrastructure but also ecosystem support—APIs, SDKs, data labeling tools, and monitoring dashboards—that reduce development time and operational overhead.By synthesizing the strengths and weaknesses of each solution, this paper serves as a valuable guide for architects aiming to design a comprehensive, scalable, and user-friendly AI platform in the cloud.

# CHAPTER 3

# SYSTEM ANALYSIS

## 3.1 EXISTING SYSTEM

Existing AI development environments, such as traditional desktop IDEs or Jupyter Notebook-based platforms, offer powerful tools for building and training models but come with significant limitations. These systems often require high-performance local hardware, manual installation of dependencies, and extensive environment configuration. They also lack seamless integration with cloud-based compute resources and generally do not support real-time scalability or remote collaboration.

Furthermore, managing runtime environments, securing user access, and tracking resource usage can be complex and time-consuming in conventional setups. Although enterprise-grade platforms like Amazon SageMaker or Google AI Platform provide some of these features, they are often costly and not beginner-friendly, making them less accessible to students or independent developers.

### 3.1.1 Disadvantages of the Existing System

- Requires high-end hardware and manual setup
- Limited scalability and automation
- No real-time collaboration or cloud-native support
- Complex resource tracking and cost estimation
- Expensive for individual learners or small-scale usage
- Poor accessibility from low-resource environments

## 3.2 PROPOSED SYSTEM

The **Cloud Based AI Training Platform** overcomes the limitations of traditional environments by offering a cloud-native, web-based IDE with integrated support for code execution, monitoring, and intelligent assistance. It enables users to write and train AI models directly in the browser, eliminating the need for local installations or hardware dependencies.

This system uses **AWS Lambda** to execute code in a serverless manner, allowing dynamic scaling based on workload. It integrates **AWS Cognito** for secure user authentication and **DynamoDB** for storing persistent data such as chat history, user preferences, and runtime metrics. The front-end is built using **Next.js 15**, offering a responsive and modern user experience. An AI-powered chatbot, backed by **GROQ AI**, assists users with context-aware suggestions, code explanations, and optimizations.

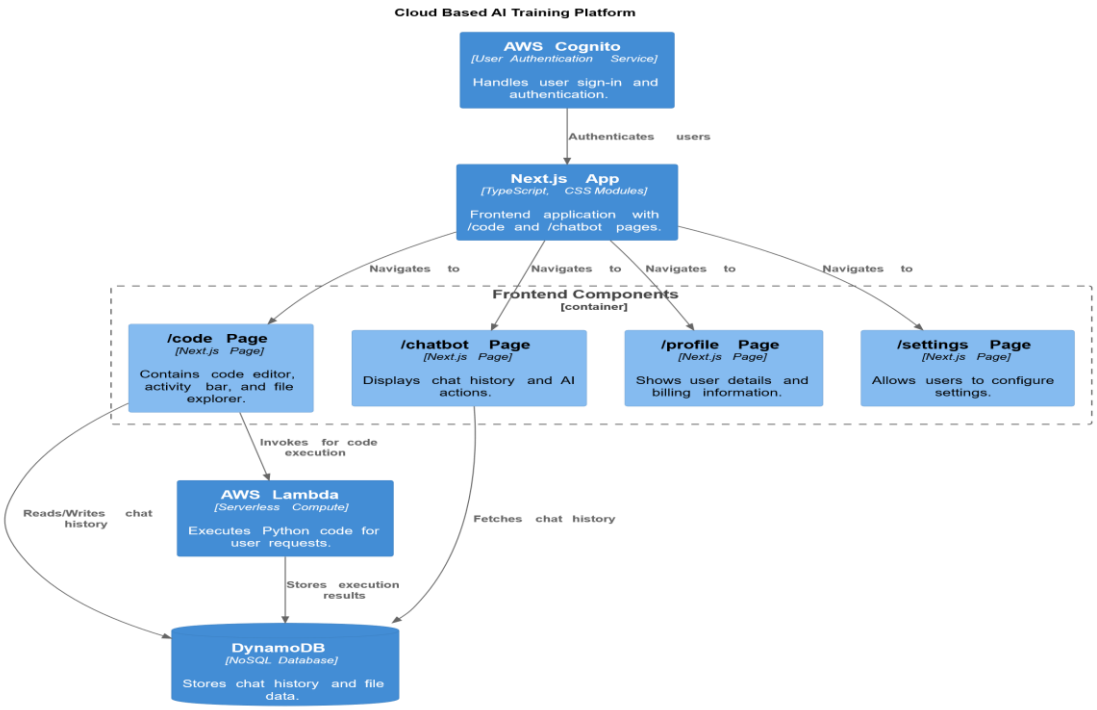


Fig 3.2.1 System Architecture

### 3.2.1 Advantages of the Proposed System

- Fully browser-based; no installation required
- Serverless compute with auto-scaling support
- AI-powered code assistance and real-time help
- Integrated metrics and monitoring dashboard
- Secure authentication and data storage
- Cost-effective usage model with pay-as-you-go billing
- Extensible for future support of R, Julia, and collaborative coding

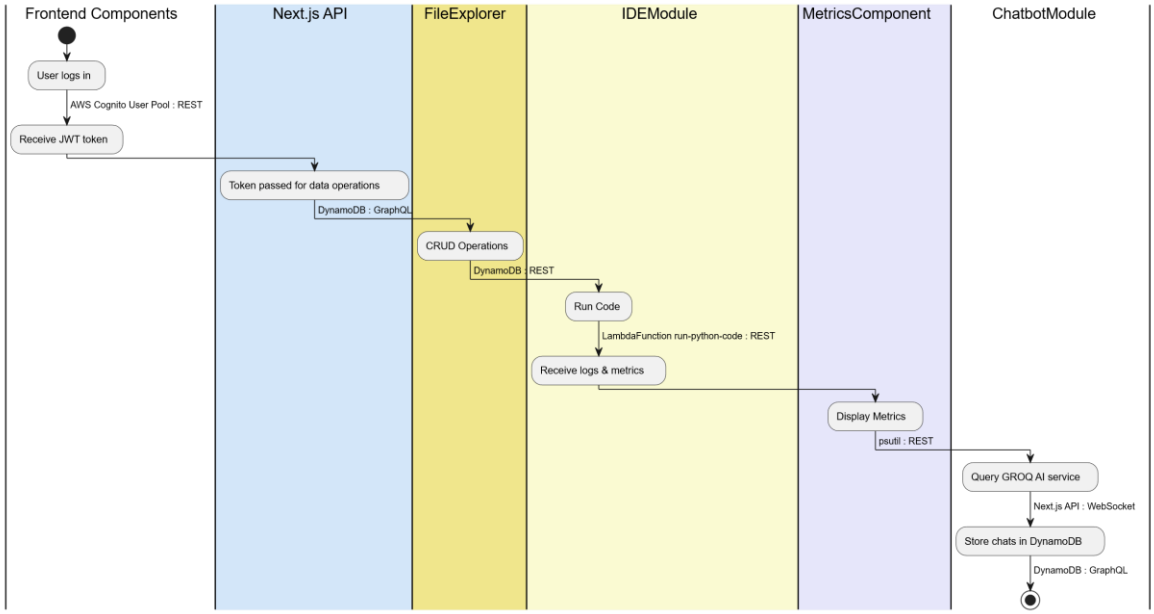


Fig 3.2.1.1 Proposed System

## 3.3 SYSTEM REQUIREMENTS

This section outlines the necessary hardware and software prerequisites required for the development, deployment, and operation of the Cloud-Based AI Training Platform. These requirements ensure that the system functions smoothly, supports user interactions, and efficiently handles AI training workloads in a cloud environment.

### 3.3.1 Hardware Requirements

The platform is designed to be accessible via standard computing devices using a web browser, minimizing the need for specialized hardware on the user side. However, the backend infrastructure hosted on the cloud requires the following minimum hardware specifications to ensure reliable performance and scalability:

- Client-Side (End User):
    - Processor: Dual-core 2.0 GHz or higher
    - RAM: 4 GB minimum
    - Storage: 500 MB free space (for browser cache and temporary data)
    - Display: 1280x720 resolution or higher
    - Internet: Stable connection with at least 5 Mbps bandwidth
- Server-Side (Cloud Environment):
    - CPU: 8-core (or vCPUs in cloud instances)
    - RAM: Minimum 32 GB (scalable as per workload)
    - Storage: SSD-based storage with a minimum of 500 GB

### 3.3.2 Software Requirements

The software stack required includes both development tools and runtime environments that support AI model development, training, and deployment in a cloud-based setup:

- **Client-Side (Browser-based IDE Access):**
    - Web Browser: Google Chrome, Firefox, or Microsoft Edge (latest version)

- Operating System: Platform-independent (Windows/Linux/macOS)

- **Server-Side (Cloud Backend):**
  - Operating System: Ubuntu 20.04 LTS or Amazon Linux 2
  - Programming Languages: Python 3.8+, JavaScript (Node.js), Shell scripting
  - AI Libraries: PyTorch, TensorFlow, Scikit-learn, NumPy, Pandas
  - Development Tools: Jupyter Notebook, VS Code Server, Docker
  - Cloud Services: AWS (S3, Lambda, EC2, SageMaker), or equivalent from GCP/Azure
  - Containerization: Docker 20.10+, Kubernetes (for orchestration)
  - Database: MongoDB / PostgreSQL (for storing user data, training logs)
  - Authentication: OAuth 2.0 / Firebase Auth (for secure login)

# CHAPTER 4

# MODULES

## 4.1 MODULE DESCRIPTION

- Runtime & Metrics Module
- Chatbot Module
- Profile & Settings Module
- Billing and Usage Module
- Code Editor Interface Module

### 4.1.1 Runtime & Metrics Module

This module handles the execution of user-submitted code using a serverless infrastructure powered by **AWS Lambda**. It invokes a helper Lambda function that securely runs Python code in an isolated environment and returns the output to the frontend. It also captures **CPU and memory usage metrics** during execution to provide feedback and performance statistics in real time. This helps users understand the efficiency of their code and manage their compute resources effectively.

### 4.1.2 Chatbot Module

The chatbot module is integrated with GROQ AI, providing a context-aware virtual assistant for users. It supports:

- Natural language queries for explaining, fixing, or optimizing code
- Persistent chat history, saved in DynamoDB for user reference

- File upload support, allowing the assistant to use custom datasets or code snippets for deeper understanding
- Quick actions like “One-click Fix,” “Explain,” and “Optimize” for real-time code improvement

### 4.1.3 Profile & Settings Module

This module manages user authentication and personalization using AWS Cognito. Features include:

- Updating personal details such as name, email, and password
- Displaying usage statistics and historical session data
- Usage calculator to track computing time (free tier up to 86,000 seconds; then $0.02/hour)
- Option to download invoices in PDF format for payment and billing transparency

### 4.1.4 Billing and Usage Module

Tracks the user's activity time and calculates usage cost based on active runtime. It helps users monitor compute usage, manage billing, and understand how much time is left in their free quota. It also generates downloadable billing invoices and will support **integration with payment gateways (Stripe or PayPal)** in future enhancements.

### 4.1.5 Code Editor Interface

Built using **Next.js 15 with TypeScript**, this module offers a browser-based, responsive **code editing environment**. It provides syntax highlighting, real-time error prompts, and an intuitive UI that closely mimics popular editors like **VS Code**. The editor interacts with the runtime module to execute code and display results instantly.

# CHAPTER 5

# TEST RESULT AND ANALYSIS

## 5.1 TESTING

Testing was conducted to ensure the stability, accuracy, and reliability of the system. All modules were tested in real cloud environments to validate performance and integration. The system underwent structured testing to ensure all components worked as intended. Testing focused on validating code execution, chatbot functionality, usage tracking, and user interaction. Both manual and automated tests were conducted in the actual cloud environment to reflect real user conditions.

## 5.2 TEST OBJECTIVES

- Verify functionality of all core modules
- Detect and fix bugs in runtime execution and UI
- Confirm secure authentication and proper resource tracking
- Validate chatbot responses and PDF generation

## 5.3 Program Testing

Testing was performed in stages:

- **Unit testing** validated individual modules like the runtime engine and chatbot.
- **Integration testing** checked the interaction between frontend, Lambda, Cognito, and DynamoDB.
- **System testing** ensured the full workflow (login → code execution → billing) functioned without issues.

## 5.4 Testing and Correctness

### 5.4.1 Unit Testing

Unit testing was conducted to validate the core components of the platform in isolation, ensuring that each feature behaves as expected when tested individually. Key areas tested include the Lambda-based code execution module, the AI-powered chatbot assistant, profile creation and update functionality, and invoice generation logic. Test cases were designed to handle a wide range of valid and invalid inputs to confirm that each unit consistently produced the correct outputs and handled edge cases gracefully.

### 5.4.2 Integration Testing

Integration testing focused on verifying that the various modules and services within the platform communicate and function together as expected. This included ensuring smooth interaction between the frontend (built with Next.js) and backend services hosted on AWS, such as Lambda functions, DynamoDB, and Cognito for authentication. A primary area of focus was the Lambda function that handles real-time Python code execution; it was tested to confirm that the results returned from the cloud were rendered accurately in the browser IDE.

### 5.4.3 Functional Testing

Functional testing simulated actual user interactions with the platform to ensure the system meets its specified functional requirements. Test scenarios included typical actions such as logging in via Cognito, writing and executing Python code, receiving AI chatbot suggestions, uploading files for context enhancement, editing user profiles, and downloading usage invoices in PDF

format. Each function was tested in sequence to replicate a real user's journey through the platform.

### 5.4.4 White Box Testing

White box testing was applied to examine the internal logic and structure of critical backend functions, particularly the Lambda helper service responsible for code execution and usage tracking. The focus was on logic branches, loops, and exception handling within the Python scripts that interpret user-submitted code and calculate usage time.

Internal logs were traced to ensure accurate measurement of active code execution time, proper sandboxing of untrusted code, and error reporting. This helped verify that the system enforces constraints on resource usage and protects against potential abuse, such as infinite loops or excessive memory allocation.

### 5.4.5 Black Box Testing

Black box testing was conducted from the end-user's perspective without any knowledge of the internal code structure. The primary objective was to verify that, given a specific input, the system consistently returns the expected output. Test cases included chatbot conversations, code execution results, and invoice downloads—all evaluated based on inputs and UI outputs alone.

The chatbot's relevance and response quality were assessed using a range of context-specific queries, while the PDF generation feature was tested for completeness and formatting. This type of testing validated that the platform's external behavior aligns with user expectations and functional requirements, regardless of the underlying implementation.

## 5.5 Analysis

Test results showed the platform to be reliable and responsive. The system successfully handled real-time execution with minimal delays. The chatbot produced relevant responses, and the usage calculator performed accurately. There were no major UI issues across browsers, and cloud integrations worked smoothly.

## 5.6 Feasibility Study

- **Technical Feasibility**: Built using robust AWS services and modern frameworks.
- **Economic Feasibility**: Cost-effective with pay-as-you-go model and free tier.
- **Operational Feasibility**: Easy to use, minimal setup, and accessible on any modern browser.

# CHAPTER 6
# RESULT AND DISCUSSION

## 6.1 RESULT

The Cloud Based AI Training Platform successfully met its intended objectives by delivering a responsive, secure, and intelligent cloud-based development environment for AI training. The serverless code execution module performed well, with minimal latency and reliable output generation through AWS Lambda. The chatbot, powered by GROQ AI, provided relevant suggestions, bug fixes, and code explanations, improving user productivity.

User authentication via AWS Cognito ensured data security, and usage tracking with invoice generation functioned accurately. Real-time performance monitoring, persistent user settings, and seamless UI interactions were validated through multiple testing rounds. The platform remained stable under different load levels and demonstrated cross-browser compatibility.

## 6.2 CONCLUSION

This project introduced a practical and scalable solution for AI development without requiring high-end local hardware. By combining modern web technologies with cloud computing services, it provides a flexible environment where users can write, execute, and monitor AI code efficiently.

Key contributions include:

- A fully cloud-native development interface
- Serverless, on-demand code execution
- AI-assisted programming support
- Usage-based cost tracking and billing

The platform reduces barriers to entry for AI learners and developers, especially those without access to expensive systems, making AI more accessible and inclusive.

## 6.3 FUTURE ENHANCEMENT

To expand the platform's functionality and reach, several enhancements are planned:

- **Payment Integration**: Add gateways like Stripe or PayPal for direct billing
- **Multi-language Support**: Extend runtime to support R, Julia, and others
- **Collaborative Coding**: Introduce real-time pair programming features
- **Advanced Analytics**: Build dashboards for usage trends and cost forecasting
- **Mobile Support**: Optimize the platform for mobile access and interaction

These improvements aim to enhance user experience, support broader use cases, and scale the platform for wider adoption.

## APPENDIX – 1

## SOURCE CODE

**package.json**

```
{
  "name": "cloud-canvas",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev --turbopack",
    "build": "next build",
    "start": "next start",
    "lint": "next lint"
  },
  "dependencies": {
    "@aws-amplify/ui-react": "^6.11.0",
    "@aws-sdk/client-dynamodb": "^3.799.0",
    "@monaco-editor/react": "^4.7.0",
    "aws-amplify": "^6.14.2",
    "chart.js": "^4.4.9",
    "groq-sdk": "^0.21.0",
    "jspdf": "^3.0.1",
    "next": "15.3.0",
    "react": "^19.0.0",
    "react-chartjs-2": "^5.3.0",
    "react-dom": "^19.0.0",
    "react-full-screen": "^1.1.1",
    "react-icons": "^5.5.0",
    "stripe": "^18.1.0"
```

```
  },
  "devDependencies": {
    "@aws-amplify/backend": "^1.14.3",
    "@aws-amplify/backend-cli": "^1.5.0",
    "@eslint/eslintrc": "^3",
    "@types/node": "^20",
    "@types/react": "^19",
    "@types/react-dom": "^19",
    "aws-cdk": "^2.1003.0",
    "aws-cdk-lib": "^2.180.0",
    "constructs": "^10.4.2",
    "esbuild": "^0.25.2",
    "eslint": "^9",
    "eslint-config-next": "15.3.0",
    "tsx": "^4.19.3",
    "typescript": "^5.8.3"
  }
}
```

**app\page.tsx**

```
import Link from "next/link";
import styles from "./page.module.css";
// If Navbar is NOT in layout.tsx, uncomment the next line
import HomeNavbar from "@/components/HomeNavbar/HomeNavbar";
import Footer from "@/components/Footer/Footer";

// Optional: For icons, install react-icons: npm install react-icons
// import { FiCpu, FiCode, FiDatabase, FiDollarSign, FiPlayCircle } from
'react-icons/fi';
```

```
export default function Home() {
  return (
    <div className={styles.pageWrapper}>
      <HomeNavbar />

      <main className={styles.mainContent}>
        {/* Hero Section */}
        <section className={styles.hero}>
          <div className={styles.heroContent}>
            <h1 className={styles.headline}>
              Unlock AI Development{" "}
              <span className={styles.highlight}>Without Barriers</span>
            </h1>
            <p className={styles.subheadline}>
              Cloud Canvas provides a ready-to-use PyTorch environment in your
              browser. Focus on learning and building, not on hardware setup.
              Pay only for what you compute.
            </p>
            <Link href="/code">
              <button className={styles.ctaButton}>Start Coding Now</button>
            </Link>
            <p className={styles.ctaSubtext}>No credit card required*</p>
            {/* (*Add asterisk if you have a free tier or trial) */}
          </div>
          {/* Optional: Add a visual element here (illustration, code snippet
animation) */}
          {/* <div className={styles.heroVisual}> ... </div> */}
        </section>
```

```
      {/* About/Problem Section */}
      <section id="about" className={styles.section}>
        <h2 className={styles.sectionTitle}>Stop Worrying About
Hardware</h2>
        <p className={styles.sectionText}>
          High-end GPUs and complex local setups shouldn&amp;apos;t block your AI
          journey. Cloud Canvas eliminates the need for expensive hardware,
          offering a seamless, cloud-based PyTorch editor accessible from any
          device. Learn, experiment, and iterate faster.
        </p>
      </section>

      {/* Features Section */}
      <section className={styles.section}>
        <h2 className={styles.sectionTitle}>Features Designed For You</h2>
        <div className={styles.featuresGrid}>
          <div className={styles.featureCard}>
            {/* <FiCode size={40} className={styles.featureIcon} /> */}
            <span className={styles.featureIcon}>📝</span>{" "}
            {/* Placeholder Icon */}
            <h3>Web-Based Editor</h3>
            <p>
              Code directly in your browser with a familiar interface. No
              installation needed.
            </p>
          </div>
          <div className={styles.featureCard}>
            {/* <FiPlayCircle size={40} className={styles.featureIcon} /> */}
```

```
  <span className={styles.featureIcon}>🚀</span>{" "}
  {/* Placeholder Icon */}
  <h3>PyTorch Ready</h3>
  <p>
    Jump straight into development with PyTorch and essential
    libraries pre-installed.
  </p>
</div>
<div className={styles.featureCard}>
  {/* <FiCpu size={40} className={styles.featureIcon} /> */}
  <span className={styles.featureIcon}>☁️</span>{" "}
  {/* Placeholder Icon */}
  <h3>Cloud Powered</h3>
  <p>
    Leverage cloud compute resources on demand. Say goodbye to local
    limitations.
  </p>
</div>
<div className={styles.featureCard}>
  {/* <FiDatabase size={40} className={styles.featureIcon} /> */}
  <span className={styles.featureIcon}>💾</span>{" "}
  {/* Placeholder Icon */}
  <h3>Persistent Storage</h3>
  <p>
    Save your code files and projects securely. Pick up right where
    you left off.
  </p>
</div>
```

```
        </div>
      </section>

      {/* How it Works Section */}
      <section className={`${styles.section} ${styles.howItWorks}`}>
        <h2 className={styles.sectionTitle}>Get Started in Minutes</h2>
        <div className={styles.stepsGrid}>
          <div className={styles.step}>
            <div className={styles.stepNumber}>1</div>
            <h3>Sign Up / Log In</h3>
            <p>Create your Cloud Canvas account quickly using AWS
Cognito.</p>
          </div>
          <div className={styles.step}>
            <div className={styles.stepNumber}>2</div>
            <h3>Create & Code</h3>
            <p>
              Open the editor, create a new file, and start writing your
              PyTorch code.
            </p>
          </div>
          <div className={styles.step}>
            <div className={styles.stepNumber}>3</div>
            <h3>Run & See Output</h3>
            <p>
              Execute your code on our cloud infrastructure and view the
              results instantly.
            </p>
          </div>
```

```
          <div className={styles.step}>
            <div className={styles.stepNumber}>4</div>
            <h3>Save Your Work</h3>
            <p>
              Your files are saved automatically to DynamoDB, accessible
              anytime.
            </p>
          </div>
        </div>
      </section>

      {/* Pricing Section */}
      <section id="pricing" className={styles.section}>
        <h2 className={styles.sectionTitle}>Simple, Transparent Pricing</h2>
        <div className={styles.pricingCard}>
          {/* <FiDollarSign size={40} className={styles.featureIcon} /> */}
          <span className={styles.featureIcon}>💰</span>{" "}
          {/* Placeholder Icon */}
          <h3>Pay Per Use</h3>
          <p>
            You only pay for the computational resources (like CPU/GPU time
            and memory) you consume while running your code. File storage is
            included*.
            {/* (*Clarify storage limits/costs if applicable) */}
          </p>
          {/* --- IMPORTANT: ADD YOUR SPECIFIC PRICING DETAILS
HERE --- */}
          <div className={styles.pricingDetails}>
            <p>
```

```
            <strong>Compute Time:</strong> $0.02 per hour
          </p>
          {/* Add more details as needed - e.g., different instance types? */}
          <p className={styles.pricingNote}>
            Clear usage tracking available in your dashboard.
          </p>
        </div>
        <Link href="/code">
          <button className={`${styles.ctaButton} ${styles.pricingCta}`}>
            Start Building
          </button>
        </Link>
      </div>
      {/* Optional: Add tiers if you have them (e.g., Free Tier, Pro Tier) */}
    </section>

    {/* Final CTA Section */}
    <section className={`${styles.section} ${styles.finalCtaSection}`}>
      <h2>Ready to Start Your AI Journey?</h2>
      <p>
        Join Cloud Canvas today and experience the easiest way to learn and
        build AI models.
      </p>
      <Link href="/code">
        <button className={styles.ctaButton}>Go to Editor</button>
      </Link>
    </section>
  </main>
  <Footer />
```

```
    </div>
  );
}
```

**app\layout.tsx**

```
"use client";

import "./globals.css";

import { Amplify } from "aws-amplify";
import outputs from "@/amplify_outputs.json";

Amplify.configure(outputs);

export default function RootLayout({
  children,
}: Readonly<{
  children: React.ReactNode;
}>) {
  return (
    <html lang="en">
      <body>{children}</body>
    </html>
  );
}
```

**amplify\auth\resource.ts**

```
import { defineAuth } from "@aws-amplify/backend";
```

```
export const auth = defineAuth({
  loginWith: {
    email: true,
  },
});
```

**amplify\data\resource.ts**

```
// amplify/data/resource.ts
import { type ClientSchema, a, defineData } from "@aws-amplify/backend";

const schema = a.schema({
  // Don't care about this model, just ignore it completely
  FileItem: a
    .model({
      name: a.string(),
      type: a.string(), // "file" or "folder"
      content: a.string(), // only applicable for files
      parentId: a.string(), // for hierarchical structuring
    })
    .authorization((allow) => [allow.owner()]),

  File: a
    .model({
      name: a.string(),
      content: a.string(), // file content
    })
    .authorization((allow) => [allow.publicApiKey()]),

  ChatEntry: a
```

```
    .model({
      name: a.string(), // a title or label, searchable in sidebar
      content: a.string(), // the chat message text
    })
    .authorization((allow) => [allow.publicApiKey()]), // or owner(), if you want
private history

  Usage: a
    .model({
      month: a.string(), // e.g., "2025-05"
      totalDuration: a.float(), // seconds used
      runs: a.integer(), // number of code runs
    })
    .authorization((allow) => [allow.publicApiKey()]),
});

export type Schema = ClientSchema<typeof schema>;

export const data = defineData({
  schema,
  authorizationModes: {
    defaultAuthorizationMode: "apiKey",},});
```

# APPENDIX – 2

# SCREENSHOTS

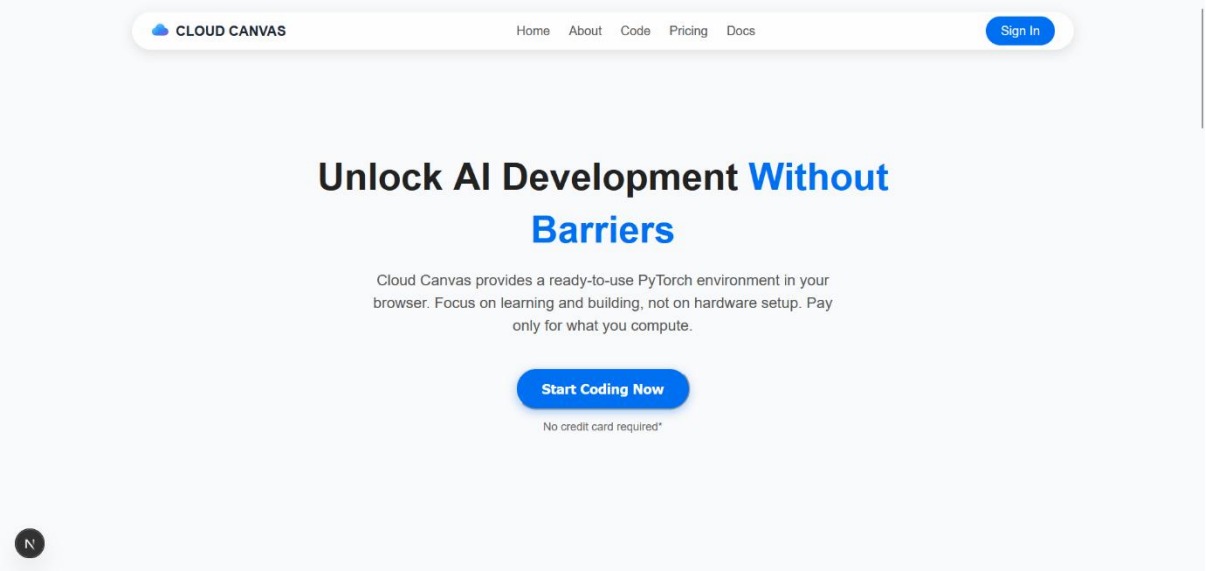


Fig A.2.1 Landing page 1

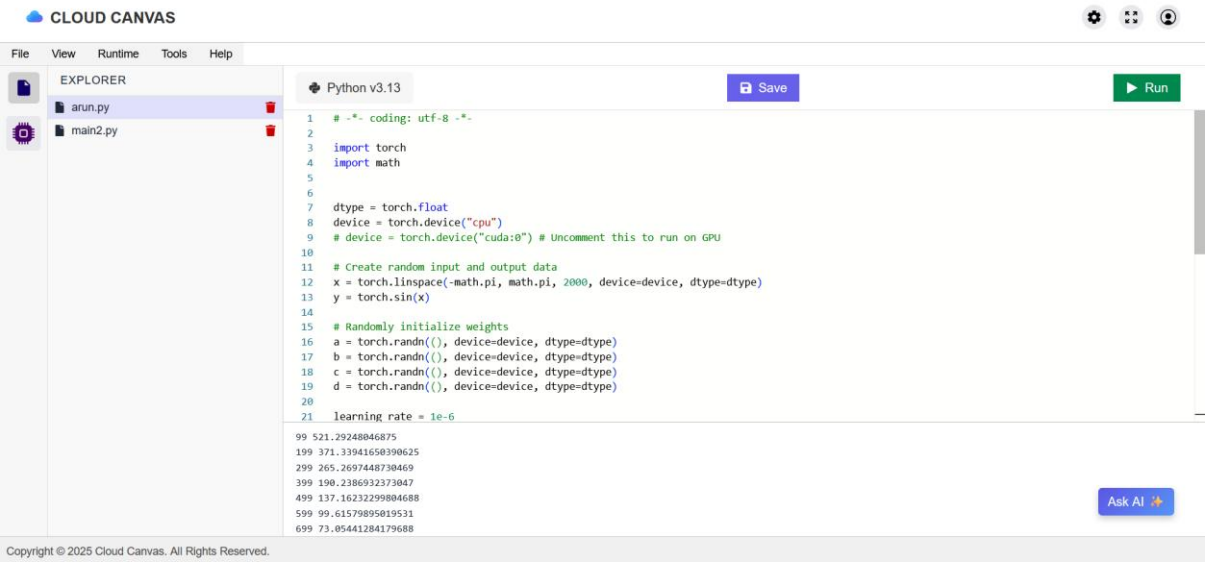


Fig A.2.2 Code Editor

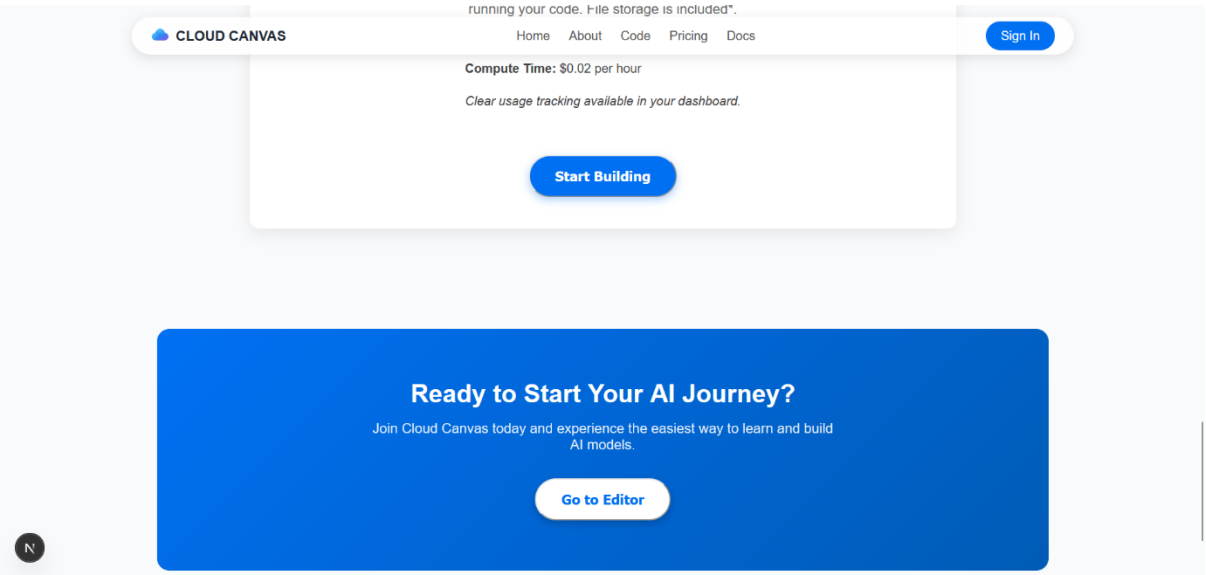


Fig A.2.3 Landing page 2

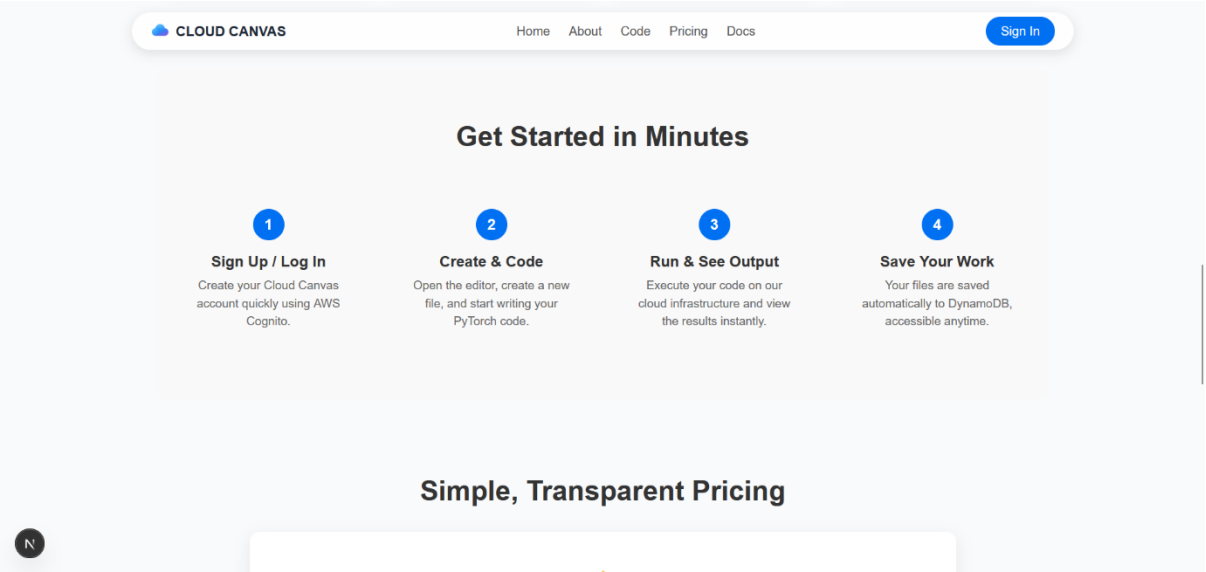


Fig A.2.4 Landing Page 3

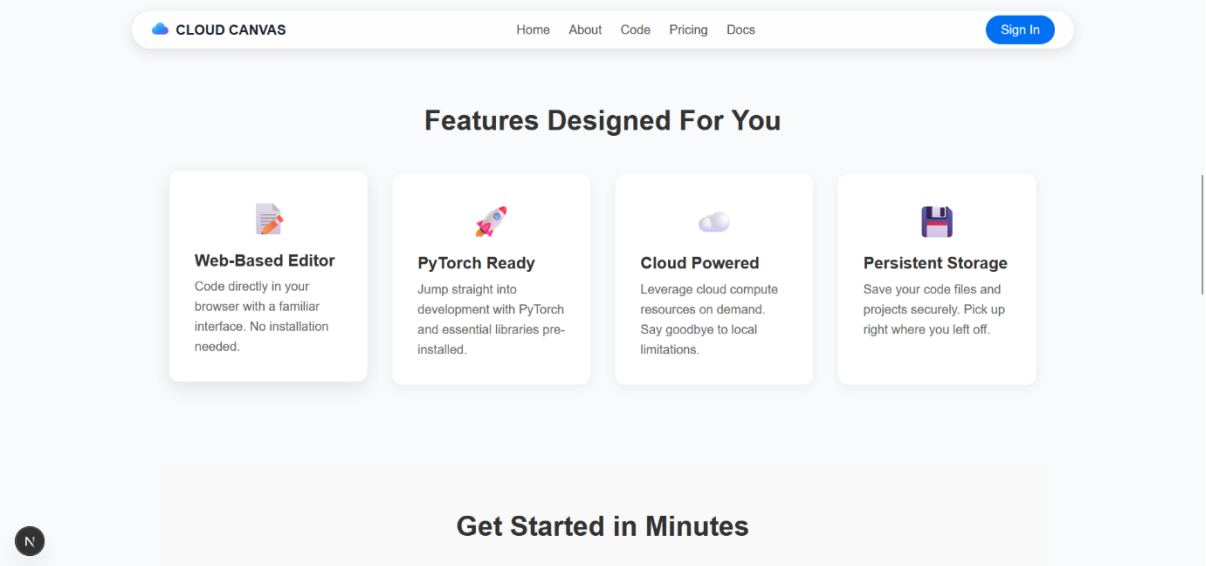


Fig A.2.5 Landing Page 4

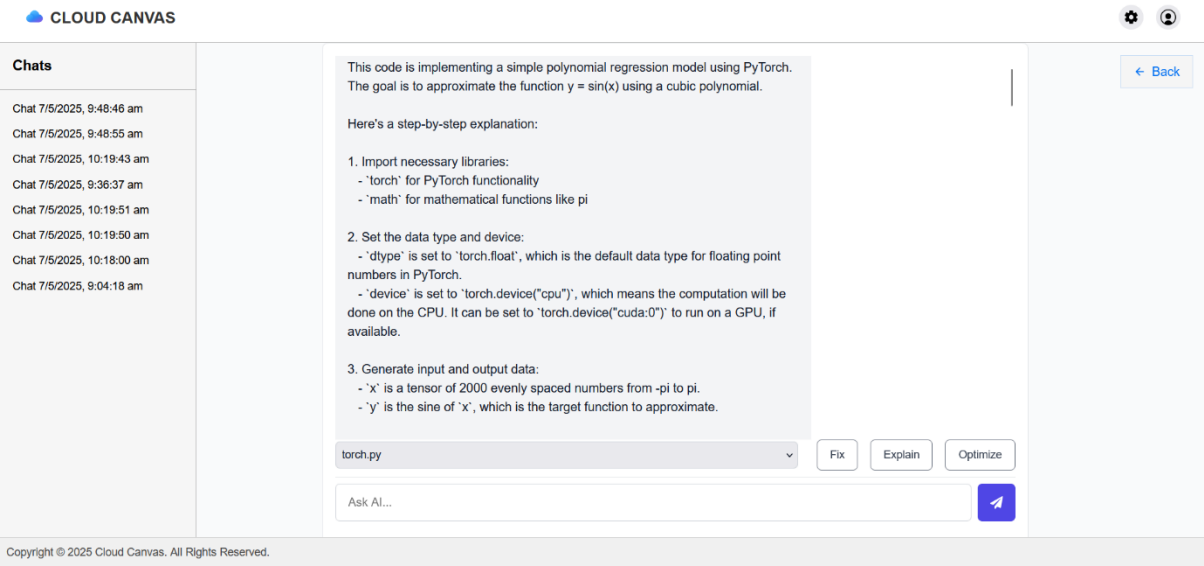


Fig A.2.6 AI Chatbot

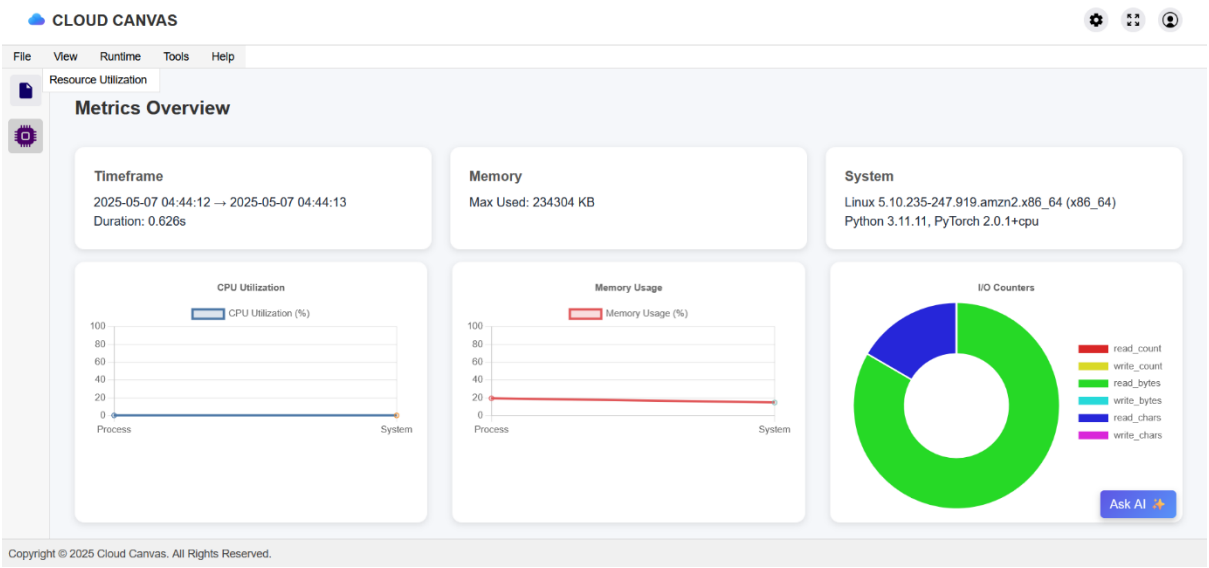


Fig A.2.7 Metrics

## REFERENCES


1) Y. Li, J. Huang, F. Tian, H.-A. Wang, and G.-Z. Dai, “Gesture interaction in virtual reality,” Virtual Reality & Intelligent Hardware, vol. 1, no. 1, pp. 84–112, Jan. 2019.
2) V. Pavlovic, R. Sharma, and T. Huang, “Visual interpretation of eye gestures for human-computer interaction: a review,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 7, pp. 677–695, 1997.
3) H. Hasan and S. Abdul-Kareem, “Human–computer interaction using vision-based eye gesture recognition systems: a survey,” Neural Computing and Applications, vol. 25, no. 2, pp. 251–261, 2013.
4) D. Crankshaw, P. Bailis, J. Gonzalez, et al., “The missing piece in complex analytics: Low latency, scalable model management and serving with Velox,” in Proc. Conf. on Innovative Data Systems Research (CIDR), 2015.
5) A. Sato, M. Tanaka, and S. Okamoto, “Design of a serverless architecture for scalable and maintainable cloud applications,” International Journal of Cloud Computing and Services Science (IJ-CLOSER), vol. 9, no. 2, pp. 136–144, 2020.
6) M. Zaharia, R. S. Xin, P. Wendell, et al., “Apache Spark: A unified engine for big data processing,” Communications of the ACM, vol. 59, no. 11, pp. 56–65, 2016.
7) B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, “Borg, Omega, and Kubernetes,” Communications of the ACM, vol. 59, no. 5, pp. 50–57, 2016.
8) G. C. Fox, S. Jha, J. Qiu, and A. Luckow, “Towards a comprehensive cyberinfrastructure for scalable data science,” Future Generation Computer Systems, vol. 79, pp. 3–17, 2018.