A PROJECT REPORT ON

THINKCODE AI

Submitted in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

In

Computer Science and Engineering (AI)

By

Y.GOPI PRADEEP KUMAR (22A81A43D0)

P.SAI KARTHIK (22A81A43B6)

A.R.S.L HARI PRIYA (22A81A4367)

D.LOKESH ADHI REDDY (22A81A4381)

B.PARASURAM SAI KUMAR(22A81A4376)

G.SHYAM SUNDHAR(22A81A4382)

Under the Esteemed Supervision of

Mr. P.V.V SATYANARAYANA

Department of Computer Science and Engineering (Accredited by N.B.A.)
SRI VASAVI ENGINEERING COLLEGE(Autonomous)
(Affiliated to JNTUK, Kakinada)
Pedatadepalli, Tadepalligudem-534101, A.P 2025-26

SRI VASAVI ENGINEERING COLLEGE (Autonomous)

Department Of Computer Science and Engineering(AI)

Pedatadepalli, Tadepalligudem

Certificate

This is to certify that the Project Report entitled **THINK CODE AI : "An AI-Enhanced Code Mentor with Compiler Integration And Visual Execution Flow"** submitted by **Y.GOPI PRADEEP KUMAR(22A81A43D0), P.SAI KARTHIK (22A81A43B6), A.R.S.L HARIPRIYA(22A81A4367), D.LOKESH ADHI REDDY (22A81A4381), B.PARSURAM SAI KUMAR(22A81A4376), G.SHYAM SUNDHAR (22A81A4382)** for the award of the degree of Bachelor of Technology in the Department of Computer Science and Engineering (AI) during the academic year 2025-2026.

Name of Project Guide	**Head of the Department**
Mr. P.V.V SATYANARAYANA M.Tech.,(Ph.D.)	Dr. G LOSHMA M.Tech.,Ph.D..
Sr. Assistant professor.	Professor & HOD.

External Examiner

DECLARATION

We hereby declare that the project report entitled **THINK CODE AI : "An AI-Enhanced Code Mentor with Compiler Integration And Visual Execution Flow"** submitted by us to Sri Vasavi Engineering College(Autonomous), Tadepalligudem, affiliated to JNTUK Kakinada in partial fulfillment of the requirement for the award of the degree of B.Tech in Computer Science and Engineering is a record of Bonafide project work carried out by us under the guidance of Mr. P.V.V.SATYANARAYANA , Sr. Asst. professor. We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree in this institute or any other institute or University.

Project Associates

Y.GOPI PRADEEP (22A81A43D0)

P.SAI KARTHIK (22A81A43B6)

A.R.S.L HARI PRIYA (22A81A4367)

D.LOKESH ADHI REDDY (22A81A4381)

B.PARSURAM SAI KUMAR (22A81A4376)

G.SHYAM SUNDHAR (22A81A4382)

ACKNOWLEDGEMENT

First and foremost, we sincerely salute to our esteemed institute **SRI VASAVI ENGINEERING COLLEGE,** for giving us this golden opportunity to fulfill our warm dream to become an engineer.

Our sincere gratitude to our project guide **Mr. PV.V.SATYANARAYANA** , **Sr.Asst.Professor,** Department of Computer Science and Engineering**,** for her timely cooperation and valuable suggestions while carrying out this project.

We express our sincere thanks and heartful gratitude to **Dr. G.LOSHMA** , Professor & Head of the Department of Computer Science and Engineering(AI), for permitting us to do our project.

We express our sincere thanks and heartful gratitude to **Dr. G.V.N.S.R. RATNAKARA RAO**, Principal, for providing a favourable environment and supporting us during the development of this project.

Our special thanks to the management and all the teaching and non-teaching staff members, Department of Computer Science and Engineering, for their support and cooperation in various ways during our project work. It is our pleasure to acknowledge the help of all those respected individuals.

We would like to express our gratitude to our parents, friends who helped to complete this project.

Project Associates

Y.GOPI PRADEEP (22A81A43D0)

P.SAI KARTHIK (22A81A43B6)

A.R.S.L HARI PRIYA (22A81A4367)

D.LOKESH ADHI REDDY (22A81A4381)

B.PARSURAM SAI KUMAR (22A81A4376)

G.SHYAM SUNDHAR (22A81A4382)

ABSTRACT

In the rapidly evolving landscape of programming education, learners often struggle to bridge the gap between writing code and understanding its inner workings. **Think Code AI** is an advanced AI-integrated educational platform designed to transform coding education by combining Artificial Intelligence with EdTech innovation. It serves as both a learning companion and an analytical tool, capable of explaining code logic, identifying bugs, analyzing time and space complexity, and suggesting optimized solutions—all without giving away direct answers.

The system goes beyond traditional compilers by providing **step-by-step code explanations**, **execution flow visualization**, and **performance insights**, helping students grasp both the logic and efficiency of their solutions. Powered by AI-driven analysis, Think Code AI ensures that learners understand not only what went wrong, but why it happened, thereby improving conceptual understanding and long-term retention. Supporting multiple programming languages, it enables secure code execution within an interactive environment that keeps learners engaged and motivated.

Ultimately, **Think Code AI** bridges the gap between code execution and comprehension—empowering students to code smarter, learn deeper, and grow into confident programmers.

Keywords: Artificial Intelligence, Programming Education, Code Explanation, Complexity Analysis, Bug Detection, Performance Optimization, EdTech, Visual Execution Flow.

TABLE OF CONENTS

LIST OF FIGURES

CHAPTER-1
INTRODUCTION

1.INTRODUCTION

1.1 Introduction

Programming can be challenging for beginners, as traditional compilers only display the final output without explaining how the program reaches that result. Students often struggle to understand loops, recursion, and algorithmic flow due to a lack of guided reasoning.
ThinkCode AI addresses this issue by integrating Artificial Intelligence with modern web technologies to create a smart, interactive coding mentor. It provides plain-language explanations, complexity analysis, and execution visualization for every piece of code entered by the learner.

Unlike standard platforms, ThinkCode AI not only detects errors but also explains why they occur and how to improve the logic. This platform also measures performance metrics such as time and space complexity, helping students write efficient code. Supporting multiple languages and secure execution, ThinkCode AI provides an engaging, step-by-step coding experience designed to improve both conceptual clarity and problem-solving ability.

1.2 Motivation

Many students find programming difficult because most compilers focus only on outputs rather than understanding. Learners face challenges in connecting theory with practice, often becoming frustrated when they cannot identify logical or performance issues in their code.

ThinkCode AI was developed to solve this educational gap by using AI to explain coding behavior in natural language and visualize how the code runs internally. It motivates learners by turning passive code execution into an active, interactive learning process. The platform encourages experimentation, provides intelligent feedback, and nurtures curiosity—making programming more engaging and accessible for all learners.

1.3 Scope

- ➢ Provides AI-based step-by-step explanations for better understanding of code logic.
- ➢ Analyzes both **time and space complexity** to help students write efficient programs.
- ➢ Detects bugs and provides **AI-driven improvement suggestions**.
- ➢ Offers a **visual execution flow** for loops, recursion, and conditionals.
- ➢ Supports **multiple programming languages** in a secure environment.
- ➢ Suitable for use in **schools, universities, and online coding platforms**.
- ➢ Can be extended to include **voice-based explanations, personalized learning paths, and deeper algorithm insights** in future updates.

1.4 Project Outline

Chapter-1	Introduction
Chapter-2	Literature Survey
Chapter-3	System Study and Analysis
Chapter-4	System Design
Chapter-5	Technologies
Chapter-6	Implementation
Chapter-7	Testing
Chapter-8	Screenshots
Chapter-9	Conclusion and Future Work

CHAPTER 2
LITERATURE SURVEY

2.LITERATURE SURVEY

Authors	Year	Title	Key Findings	Methodology
Aditya KVS, Kishore Gowda,Sanjay KS	2025	An AI-Driven Approach to Automatic code analysis	Code analysis and Summarization	Machine Learning
Eleni Vrochidow,Theofanis,kanakaris	2024	AI-Powered Software Development	AI driven Recommendat-ions	CNN
Feng, Zhangyin; Guo, Daya; et al.	2020	A Pre-Trained Model for Programming and Natural Languages	Improves tasks like code search &documentation generation	Transformer-based Model
Eduardo Adam Navas-López	2022	Modular and Didactic Compiler Design with XML Inter-Phases Communication	Using XML to communicate between compiler phases; modular design; didactic tool for compiler courses	Python 3.8.

CHAPTER 3
SYSTEM STUDY AND ANALYSIS

3.1 Problem Statement

Traditional compilers only show the output of a program without explaining how that result is achieved. This makes it difficult for learners to understand code logic, particularly in complex areas such as recursion, loops, and condition handling. Without guidance, students often memorize syntax rather than developing true problem-solving skills.

The lack of conceptual feedback, complexity analysis, and visual aids prevents beginners from gaining a holistic understanding of programming. ThinkCode AI aims to overcome these challenges by making coding education more explanatory, interactive, and performance-aware.

3.2 Existing System

Existing online compilers and IDEs primarily focus on code execution and syntax highlighting. While some advanced tools provide debugging or autocomplete, they lack educational insight. They do not explain the underlying logic, analyze complexity, or suggest optimized solutions.

Research in AI-assisted code analysis has improved automation but remains limited in personalized learning and student engagement.

Hence, current systems fail to connect **concept comprehension** with **code correctness**, leaving a crucial gap in learning effectiveness.

3.3 Limitations of Existing System

- Only provides output, not logical explanations.
- No real-time performance or complexity analysis.
- Lacks intelligent, step-by-step code flow visualization.
- Does not provide conceptual or educational feedback.
- No AI-driven assistance for error reasoning or optimization.
- Poor engagement and motivation for learners.

3.4 Proposed System

- ThinkCode AI enhances the learning process by integrating Artificial Intelligence into a user-friendly coding environment. It explains the logic behind every code snippet in plain, easy-to-understand language, detects errors, estimates complexity, and provides optimization suggestions.
- The platform offers a visual representation of code execution, showing how variables, loops, and recursion behave step-by-step. AI modules analyze both correctness and performance, helping users write optimized and clean code. ThinkCode AI also provides secure sandbox execution for multiple programming languages, ensuring a safe learning experience.
- With its AI tutor-style feedback, interactive explanations, and performance evaluation,ThinkCode AI bridges the gap between traditional compilers and conceptual understanding—making it a powerful companion for students, teachers, and self-learners alike.

3.5 Advantages of Proposed System

- **Conceptual Clarity:** Explains each line in plain English for easy comprehension.
- **AI-Based Insights:** Detects bugs, estimates complexity, and suggests optimizations.
- **Interactive Learning:** Visualizes loops, recursion, and algorithm flow.
- **Performance Awareness:** Analyzes time and space complexity.
- **Motivational Feedback:** Encourages students through interactive guidance.
- **Scalable & Secure:** Works smoothly across multiple devices and browsers.

- **Multi-Language Support:** Enables coding in various programming languages.

3.6 Functional requirements

- Users can write/run code in browser.
- AI explains code logic.
- Visual execution tracing of loops/recursion.
- Detect algorithms & estimate complexity.
- Suggest optimizations & detect bugs.
- Generate pseudocode & visual flows.
- Save, export, share code.
- User authentication/session management.

3.7 Non-Functional Requirements

- **Accessibility:** Web-based,works on modern browsers.
- **Performance:** Fast code execution and AI response.
- **Usability:** Intuitive frontend with React.js + Monaco Editor.
- **Scalability:** Backend (Java) handles multiple concurrent users.
- **Security:**Protects user data(firebase).
- **Deployment:** Frontend on Vercel, backend on Render.

3.8 User Interface Requirements

3.8.1 System Requirements

3.8.1.1 Hardware requirements

- **Server-Side (Backend Hosting)**
  - Processor: Minimum 2 cores
  - RAM: 4 GB or higher
  - Storage: 50 GB SSD or more

- Internet: High-speed, stable connection

- **Client-Side (User System / Browser)**
  - Processor: Dual-core or better
  - RAM: 4 GB or higher
  - Storage: 10 GB free space
  - Browser: Latest Chrome, Firefox, or Edge
  - Internet: Stable connection for real-time code execution & AI responses\

3.8.1.1 Hardware requirements

- **Frontend**
  - React.js – For UI development
  - Monaco Editor – VS Code-like code editor
  - TailwindCSS – Styling
- **Backend**
  - Java – Server-side processing
  - Node.js / Flask – REST APIs (optional)
  - Pyodide – Python code execution support
- **Database & Authentication**
  - Firebase Auth– User login
  - Firestore DB – Data storage
- **AI & NLP**
  - Gemini-2.0 / CodeT5 – Code explanation & analysis
  - AST Parsing – Code structure analysis
- **Deployment Platforms**
  - Frontend → Vercel
  - Backend → Render

CHAPTER 4
SYSTEM DESIGN

4 SYSTEM DESIGN

4.1 System Architecture Design

ThinkCode AI's architecture initiates with a secure **Authentication (Auth)** module, offering both Google SSO for convenience and a demo user option for immediate access, ensuring a smooth entry into the platform. Once authenticated, users are directed to an intuitive **Frontend** interface, which acts as their interactive coding environment where they write and manage their code. User progress and project files are diligently managed within the **Storage** module, allowing learners to seamlessly save and load their coding **workspaces**. At the heart of the system, a robust **Backend** operates, powered by a **WebSocket Server** that facilitates real-time, bidirectional communication, crucial for instant feedback and dynamic content delivery. This backend intelligently routes user requests: code submitted for analysis is directed to the **AI** module, which processes it to provide step-by-step explanations, visualize execution flow, analyze complexity, and suggest optimized solutions without revealing direct answers. Alternatively, code sent for execution is handled by the **Execution** module, which runs it in a secure sandboxed environment and streams the live output or detailed error messages back to the user's frontend. This integrated design ensures a highly responsive and intelligent learning experience, bridging the gap between writing code and truly understanding its intricacies.

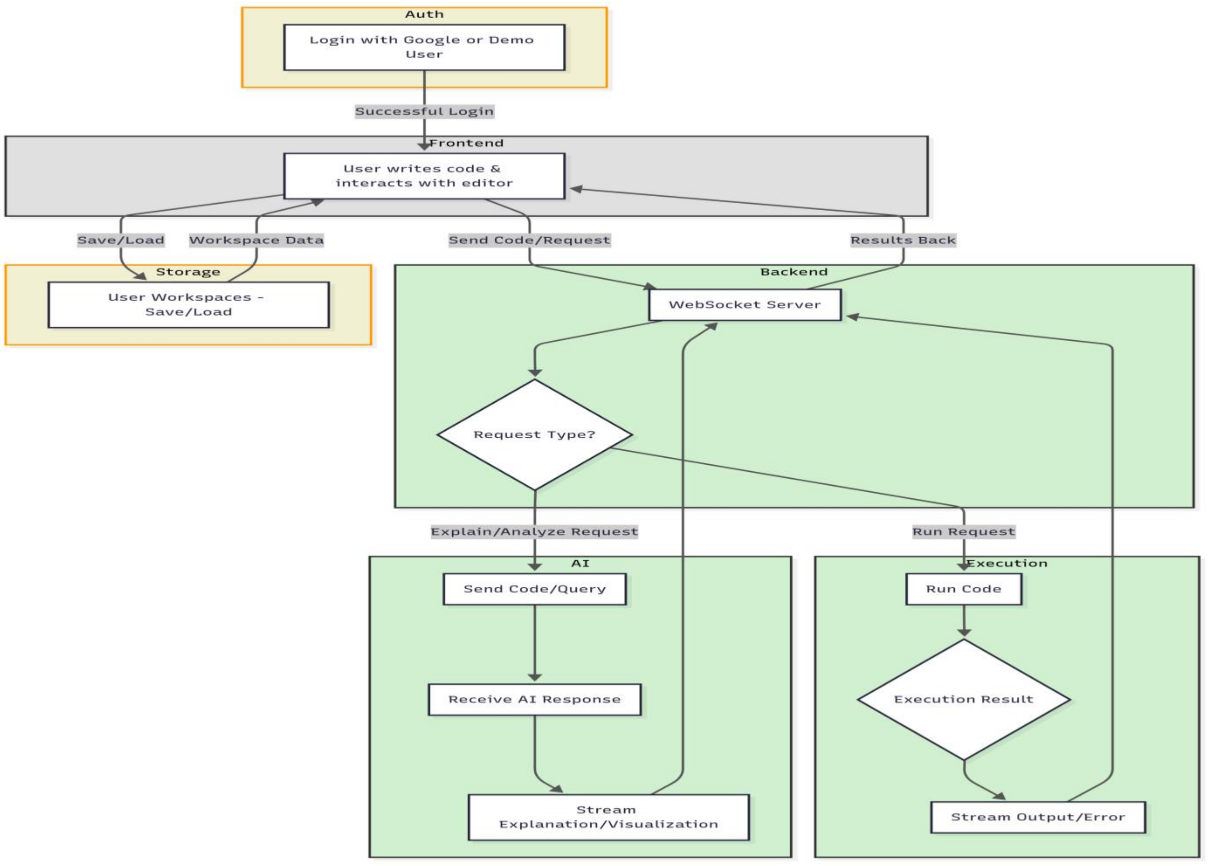


Fig 1:system architecture of Think Code AI

4.2 UML Diagrams

A **UML diagram** shows the unified visual presentation of the UML (Unified Modeling Language) system intending to let developers or business owners understand, analyze, and undertake the structure and behaviors of their system.So far, the UML diagram has become one of the most common business process modeling tools, which is also highly significant to the development of object-oriented software.UML diagrams have many benefits for both software developers and business people, and the most key advantages are:

- **Problem-Solving** - Enterprises can improve their product quality and reduce cost especially for complex systems in large scale. Some other real-life problems including physical distribution or security can be solved;
- **Improve Productivity** - By using the UML diagram, everyone in the team is on the same page and lots of time are saved down the line;
- **Easy to Understand** - Since different roles are interested in different aspects of the system, the UML diagram offers non-professional developers, for example, stakeholders, designers, or business researchers, a clear and expressive presentation of requirements, functions and processes of their system.

Class Diagram

The class diagram is the building block of all object-oriented software systems. Users can depict the static structure and identify classes relationship of a system by checking system's classes and attributes. Each class has three basic elements: the class name at the top, the class attributes in the middle, and class behaviors at the bottom. the middle, and the class behaviors at the bottom

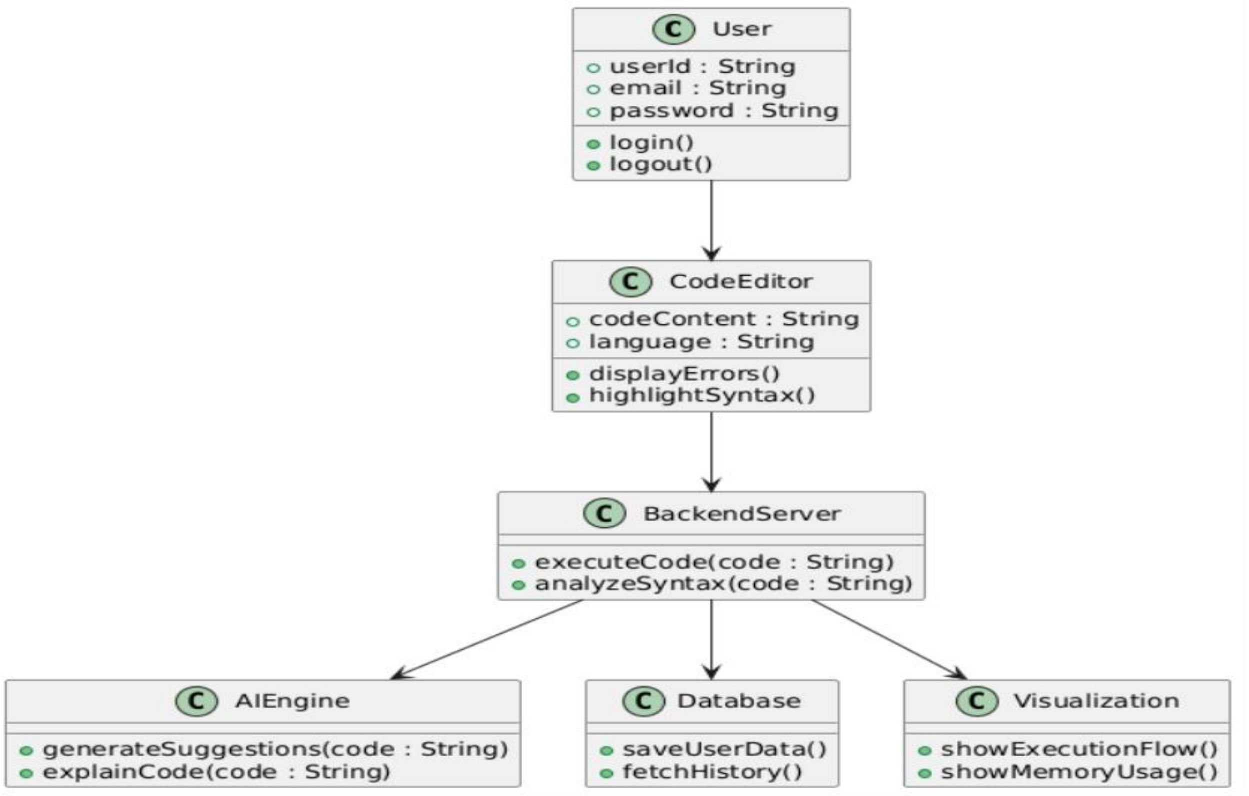


Fig no.2 Class diagram of Think Code AI

Usecase Diagram

A use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behavior, and not the exact method of making it happen. Use cases once specified can be denoted both textual and visual representation. A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.

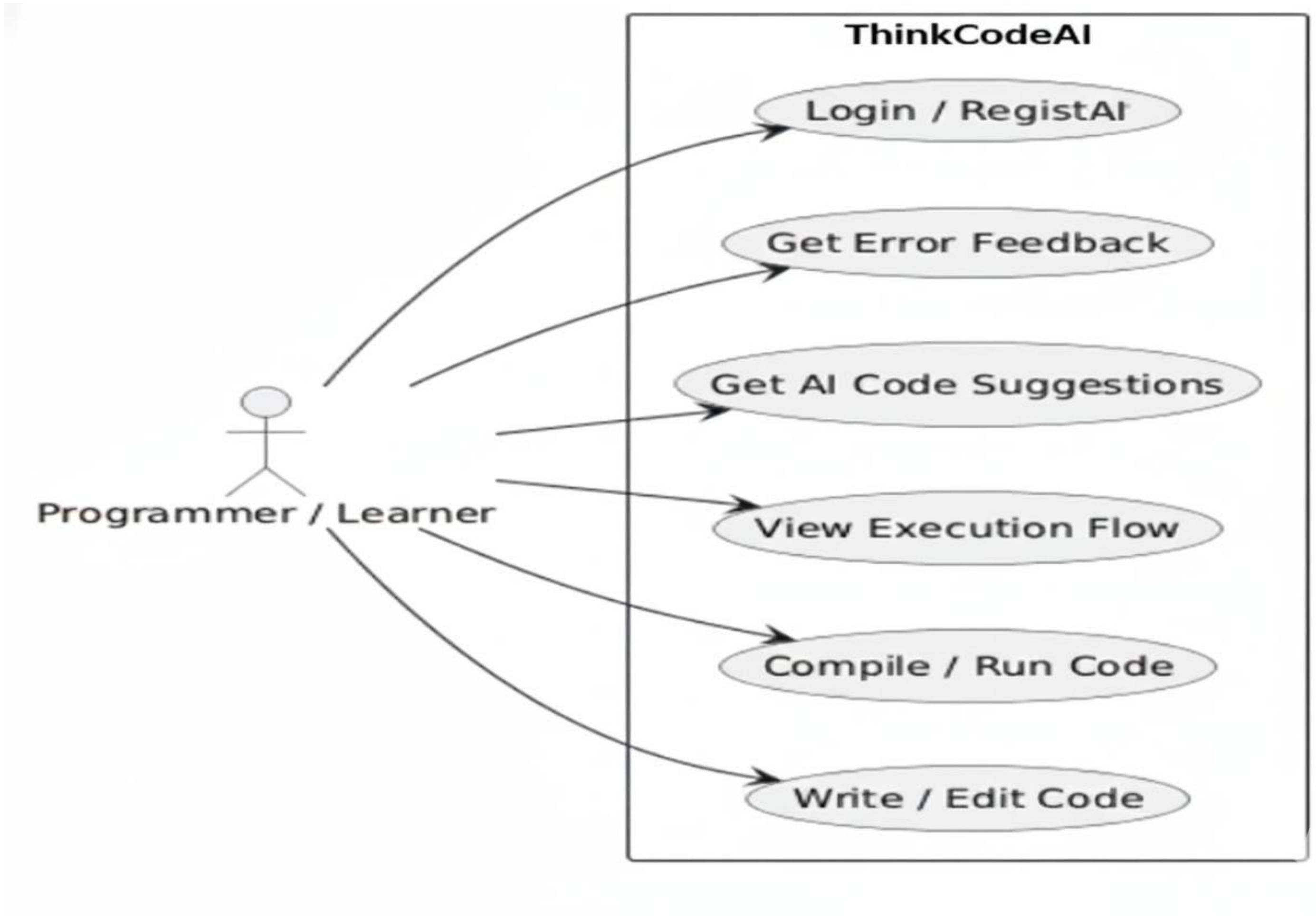


Fig no.3 Usecase diagram of Think Code AI

Sequence Diagram

A Sequence Diagram generally shows the interaction between objects in a sequential order. It is for users to document and understand requirements in a new system. In software development, this type of diagram is used to represent the architecture of a system.

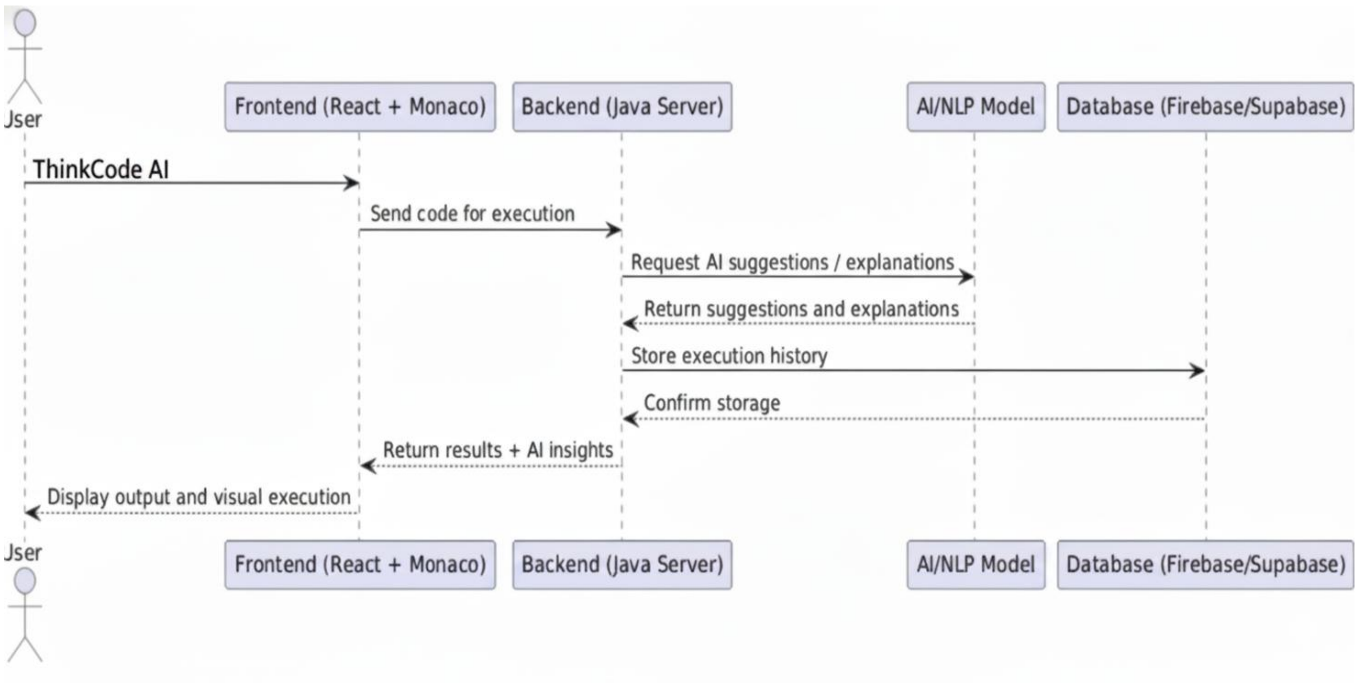

Jser
Frontend (React + Monaco)
Backend (Java Server)
AI/NLP Model
Database (Firebase/Supabase)
ThinkCode AI
Send code for execution
Request AI suggestions / explanations
Return suggestions and explanations
Store execution history
Confirm storage
Return results + AI insights
Display output and visual execution
Jser
Frontend (React + Monaco)
Backend (Java Server)
AI/NLP Model
Database (Firebase/Supabase)

Fig no.4 Sequence diagram of Think Code AI

Activity Diagram

A UML activity diagram helps to visualize a certain use case at a more detailed level. It is a behavioral diagram that illustrates the flow of activities through a system.UML activity diagrams can also be used to depict a flow of events in a business process. They can be used to examine business processes in order to identify its flow and requirements.

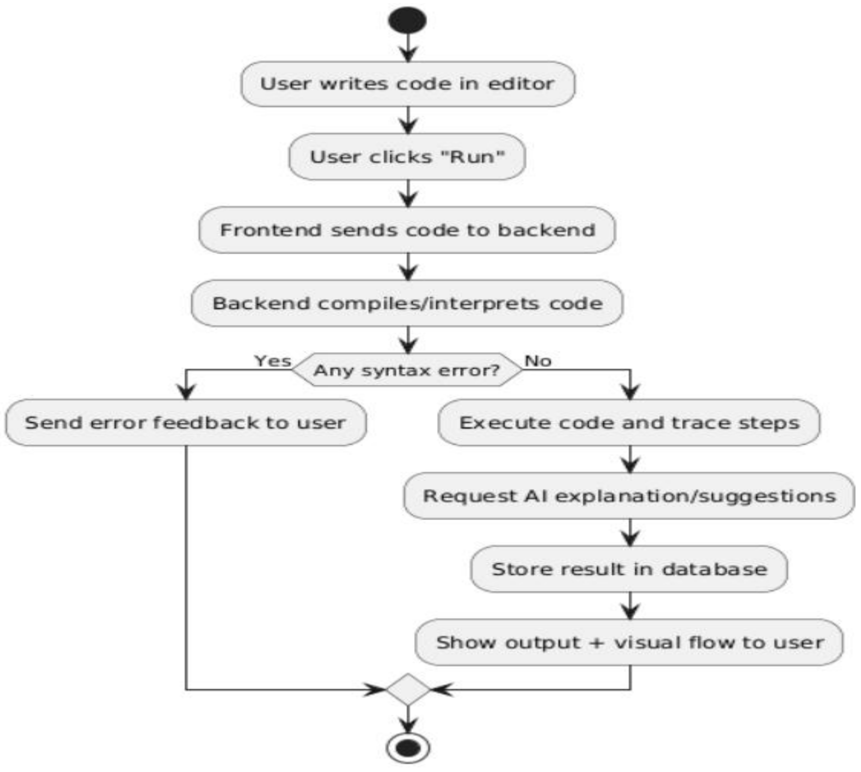

User writes code in editor
User clicks "Run"
Frontend sends code to backend
Backend compiles/interprets code
Yes
Any syntax error?
No
Send error feedback to user
Execute code and trace steps
Request AI explanation/suggestions
Store result in database
Show output + visual flow to user

Fig no.4 Activity diagram of Think Code AI

CHAPTER 5
TECHNOLOGIES

5 TECHNOLOGIES

5.1 About Java

Java is a powerful, object-oriented programming language known for its platform independence and strong security features. It is widely used for backend development, web applications, and enterprise systems. In Think Code AI, Java is used for building the backend logic, managing server requests, and ensuring smooth communication between the frontend and database. Its robustness and scalability make it suitable for handling multiple users and real-time code compilation tasks efficiently.

5.1.2 Required Java Libraries

Spring Boot:

Spring Boot is a Java framework that simplifies backend development by providing pre-configured tools for creating APIs, managing databases, and handling user authentication. It helps the system process requests quickly and securely.

Maven:

Maven is a build automation tool used for managing dependencies and compiling Java projects easily. It ensures that all the required libraries are correctly configured.

Servlets:

Servlets are used to handle HTTP requests and responses. In Think Code AI, they manage communication between the user interface and backend logic.

JSP (Java Server Pages):

JSP is used to dynamically generate web pages and connect frontend content with backend data.

JDBC (Java Database Connectivity):

JDBC enables the backend to connect and interact with databases for storing and retrieving user information.

5.1.3 Frontend Technologies

React.js:

React.js is a JavaScript library for building dynamic and responsive user interfaces. In Think Code AI, it is used to create an interactive frontend where users can type, edit, and execute their code in real time.

Monaco Editor:

Monaco Editor is the same code editor used in Visual Studio Code. It provides syntax highlighting, auto-completion, and debugging features, giving users a professional coding experience within the browser.

Tailwind CSS:

Tailwind CSS is a utility-first CSS framework that allows fast and responsive UI design. It ensures the Think Code AI interface looks modern, clean, and works well on all devices.

5.1.4 AI & NLP Technologies

Gemini 2.5 :

These AI models are used for natural language processing and code understanding. They explain code logic in plain English, detect errors, and suggest optimizations.

AST Parsing (Abstract Syntax Tree):

AST parsing helps the system understand the structure of the code, allowing it to identify algorithms, estimate complexity, and visualize the flow of execution.

5.1.5 Database and Authentication

Firebase Authentication:

Firebase Auth handles user registration, login, and secure access control, protecting user data.

Firestore Database:

Firestore is a NoSQL cloud database used to store user data, project history, and feedback in real time. It ensures fast read/write operations for multiple users.

5.2 Development and Deployment Tools

Visual Studio Code:

Visual Studio Code (VS Code) is a free and powerful source-code editor used for writing and debugging both frontend and backend code. It supports extensions and IntelliSense for auto-suggestions, making development faster and easier.

Render:

Render is a cloud platform where the Java backend of Think Code AI is deployed. It ensures that the backend runs smoothly and is always accessible.

Vercel:

Vercel hosts the React.js frontend of Think Code AI, providing fast performance, scalability, and automatic deployment updates.

CHAPTER 6

IMPLEMENTATION

6 IMPLEMENTATION

The implementation of Think Code AI is based on a modern, distributed system architecture, integrating a high-performance Java backend with an interactive React-based frontend and multiple external AI and database services. The goal of the implementation is to create a seamless, real-time learning environment that delivers code execution, visual flow tracing, and intelligent AI feedback.

6.1 System Components and Technologies

The system is constructed using the following primary technologies, as outlined in the System Requirements:

Component	**Technology**	**Role in the Project**
Frontend (UI/Code Editor)	React.js	Building the dynamic and responsive user interface.
	Monaco Editor	Providing a professional, VS Code-like code editor with syntax highlighting and auto-completion.
	Tailwind CSS	A utility-first CSS framework for a modern, clean, and responsive UI design[99].
Backend (Server & Logic)	Java	Server-side processing, managing requests, and ensuring communication between the frontend and database.
	Spring Boot	Simplifies backend development,

Component	**Technology**	**Role in the Project**
		API creation, and handles user authentication.
	Servlets	Used to handle HTTP requests and responses, managing communication between the user interface and backend logic.
AI & NLP	Gemini-2.5	Explaining code logic in plain English, detecting errors, and suggesting optimizations.
	AST Parsing	Analyzing code structure to identify algorithms, estimate complexity, and visualize execution flow.
Database & Auth	Firestore Database	NoSQL cloud database for storing user data, project history, and feedback.
	Firebase Authentication	Handling user registration, login, and secure access control.
Deployment	Vercel (Frontend) / Render (Backend)	Hosting the React.js frontend for fast performance and deploying java backend for smooth access.

Implementation Steps:

The core functionality of Think Code AI is realized through an integrated, real-time sequence that processes the user's code and delivers intelligent feedback. This process is orchestrated by a WebSocket Server on the backend for bidirectional communication.

1. User Authentication and Entry

- The system begins with a secure Authentication (Auth) module.
- Users can log in using Google SSO or utilize a Demo User option for immediate access.

2. Code Submission

- The authenticated user interacts with the Frontend (React + Monaco Editor) to write or edit code.
- Upon clicking "Run," the Frontend sends the code to the Backend (Java Server) for execution and analysis.

3. Backend Processing (Request Handling)

The Java Backend routes the request based on whether the user wants to Run the code or Explain/Analyze the code.

A. Code Execution and Tracing (Run Request)

- The code is passed to the Execution module which runs it in a secure sandboxed environment.
- The system traces the steps of execution to prepare for visual flow.
- Output (or detailed error messages) is streamed back to the frontend.

B. AI Analysis and Explanation (Explain/Analyze Request)

- The code is sent to the AI/NLP Model (Gemini-2.0 / CodeT5) for processing.
- The AI provides step-by-step explanations, analyzes time and space complexity, detects bugs, and suggests optimizations.
- AST Parsing is utilized to analyze the code structure and generate data for the Visual Execution Flow.
- The AI response (insights and visualization data) is streamed back to the backend.

4. Data Storage and Feedback Delivery

- The Backend stores the execution history and user workspace data in the Firestore Database.
- The Backend aggregates the execution results (output) and the AI analysis (insights, visualizations).
- The complete data package is returned to the Frontend.

5. Output Display

- The Frontend receives the results and displays the output, visual execution flow, and AI insights to the user.

CHAPTER 7
TESTING

7 TESTING

7.1 Introduction to Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

7.2 Test Objectives

• All field entries must be filled properly.

• Pages must be activated at every level.

• The messages and responses must not be delayed.

7.3 Test Strategies

7.3.1Unit Testing

Unit testing is a method of testing individual units or components of a software application. It is typically done by developers and is used to ensure that the individual units of the software are working as intended. Unit tests are usually automated and are designed to test specific parts of the code, such as a particular function or method. Unit testing is done at the lowest level of the software development process, where individual units of code are tested in isolation.

7.3.2 Integration Testing

Integration testing is a method of testing how different units or components of a software application interact with each other. It is used to identify and resolve any issues that may arise when different units of the software are combined. Integration testing is typically done after unit testing and before functional testing and is used to verify that the different units of the software work together as intended.

7.3.3 Functional Testing:

Functional testing is a type of system software testing that validates the software against the functional requirements . The purpose of Functional tests is to test each function of the software application, by providing appropriate input and verifying the output against the Functional requirements.

7.3.4   System Testing:

System testing is a type of software testing that evaluates the overall functionality and performance of a complete and fully integrated software solution. It tests if the system meets the specified requirements and if it is suitable for delivery to the end-users. This type of testing is performed after the integration testing and before the acceptance testing.

In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behavior of a component or a system when it is tested. **System Testing** is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both.

White Box testing:

At the system testing level, white-box testing may involve examining the overall structure and interactions within the system, ensuring that internal components work together seamlessly.

Black Box Testing:

In system testing, black-box testing assesses the entire integrated system's external behavior, focusing on inputs, outputs, and overall functionality. Testers are concerned with whether the system meets specified requirements.

7.3.5 Acceptance Testing

Acceptance testing is the final phase of software testing where the system is evaluated to ensure it meets user requirements and business needs. Conducted by end-users or stakeholders, this testing phase validates that the software is ready for deployment. Acceptance testing includes user acceptance testing (UAT), where users confirm the system's alignment with their expectations, ensuring its readiness for production use. Successful acceptance testing indicates that the software is fit for its intended purpose and satisfies the criteria set forth during requirements definition

CHAPTER 8
RESULTS

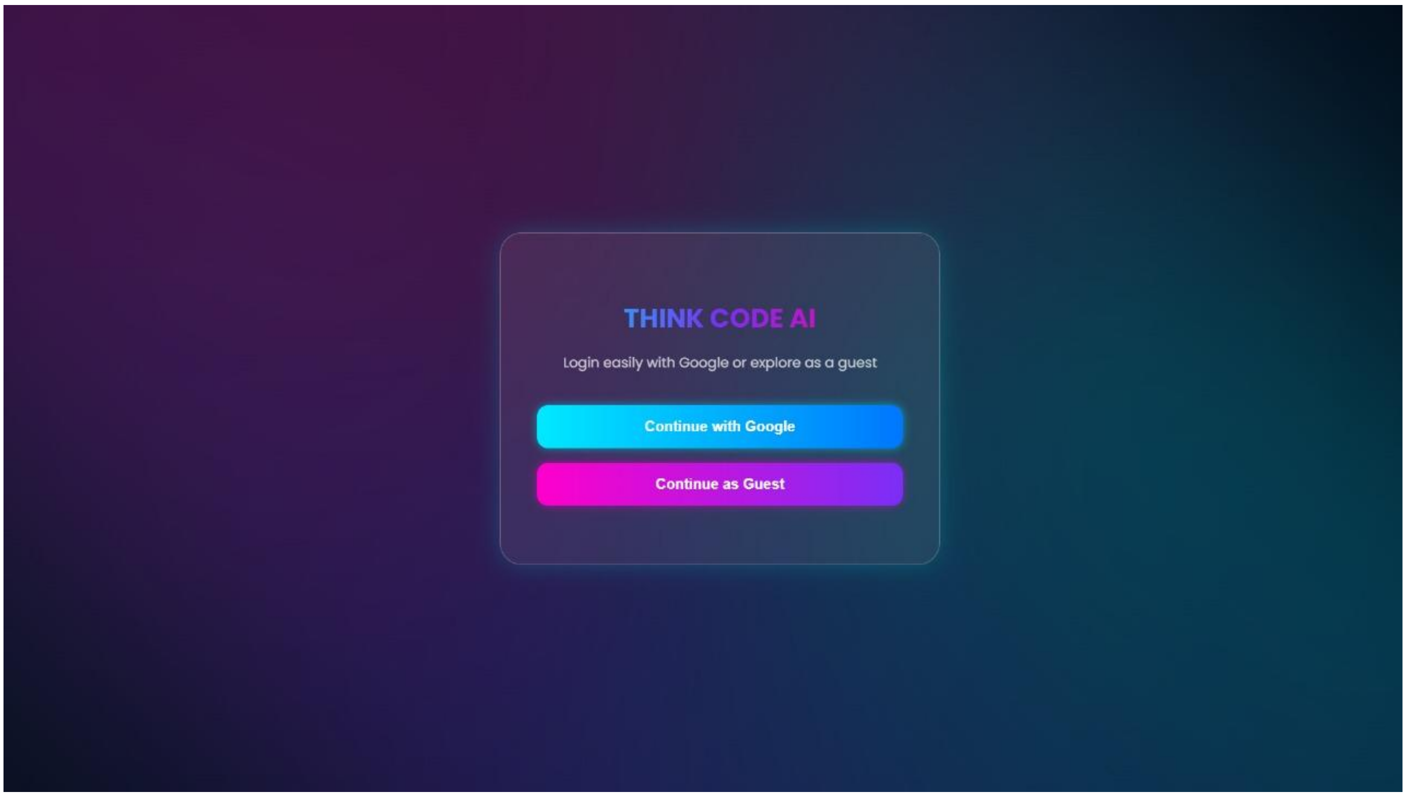
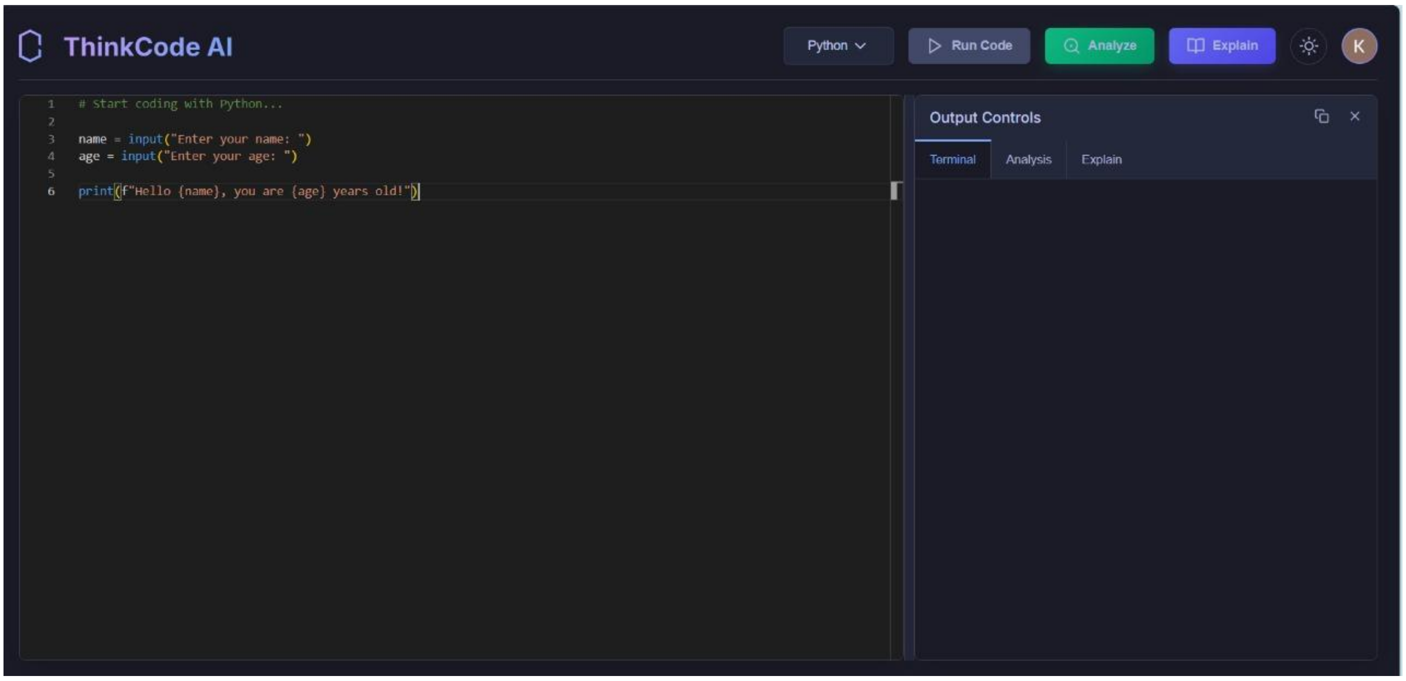


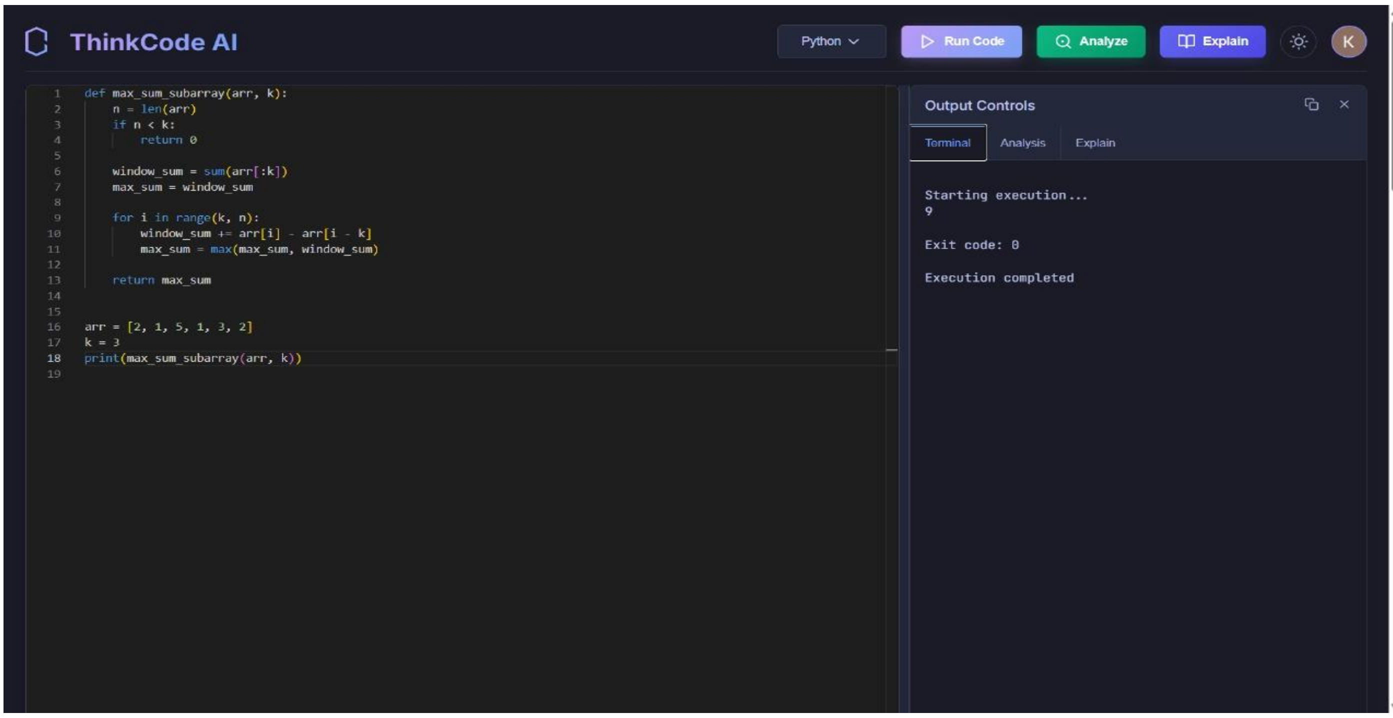
Fig:8.1 Login Page of ThinkCode AI



```python
# Start coding with Python...

name = input("Enter your name: ")
age = input("Enter your age: ")

print(f"Hello {name}, you are {age} years old!")
```

Fig:8.2 Home Page of ThinkCode AI

Students write code in the editor, which is sent to the backend for execution. The AI analyzes the results and instantly returns output, errors, and step-by-step explanations on the screen.

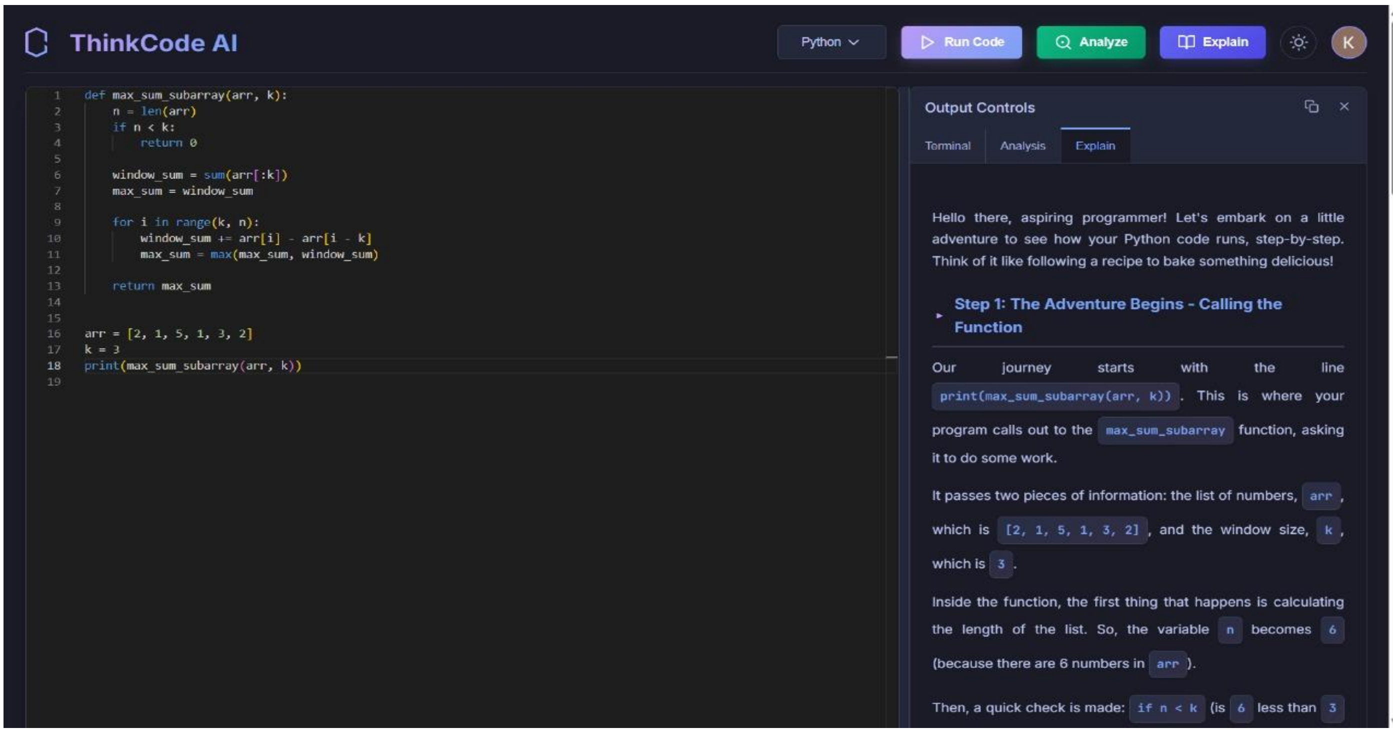


```python
def max_sum_subarray(arr, k):
    n = len(arr)
    if n < k:
        return 0

    window_sum = sum(arr[:k])
    max_sum = window_sum

    for i in range(k, n):
        window_sum += arr[i] - arr[i - k]
        max_sum = max(max_sum, window_sum)

    return max_sum


arr = [2, 1, 5, 1, 3, 2]
k = 3
print(max_sum_subarray(arr, k))
```

```
Starting execution...
9

Exit code: 0

Execution completed
```

Fig;8.3 Execution of Code In ThinkCode AI

The AI interprets the code's logic and structure, then generates clear, step-by-step explanations. This helps students understand how the program works instead of just seeing the final output



```python
def max_sum_subarray(arr, k):
    n = len(arr)
    if n < k:
        return 0

    window_sum = sum(arr[:k])
    max_sum = window_sum

    for i in range(k, n):
        window_sum += arr[i] - arr[i - k]
        max_sum = max(max_sum, window_sum)

    return max_sum


arr = [2, 1, 5, 1, 3, 2]
k = 3
print(max_sum_subarray(arr, k))
```

Hello there, aspiring programmer! Let's embark on a little adventure to see how your Python code runs, step-by-step. Think of it like following a recipe to bake something delicious!

Step 1: The Adventure Begins - Calling the Function

Our journey starts with the line `print(max_sum_subarray(arr, k))`. This is where your program calls out to the `max_sum_subarray` function, asking it to do some work.

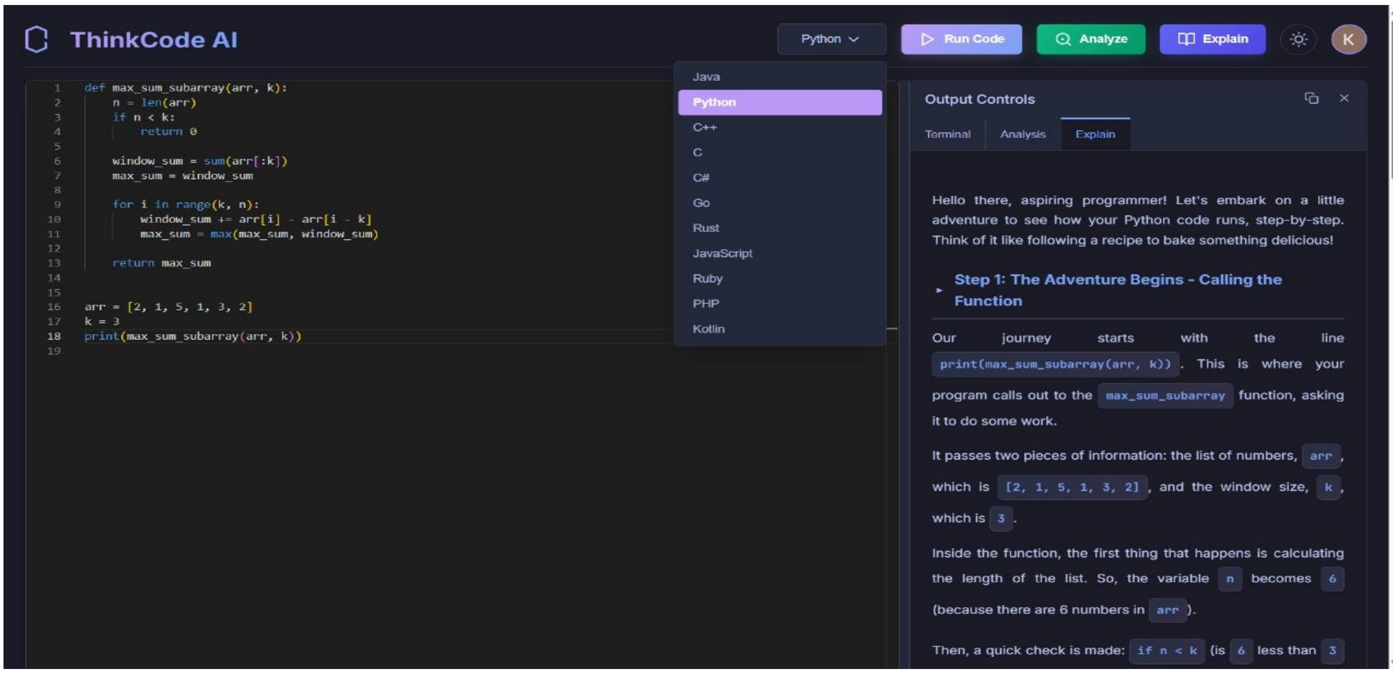
It passes two pieces of information: the list of numbers, `arr`, which is `[2, 1, 5, 1, 3, 2]`, and the window size, `k`, which is `3`.

Inside the function, the first thing that happens is calculating the length of the list. So, the variable `n` becomes `6` (because there are 6 numbers in `arr`).

Then, a quick check is made: `if n < k` (is `6` less than `3`

Fig:8.4 Explanation of Code in ThinkCode AI

Allowing students to practice and learn in C, C++, Java, Python, and more. This flexibility helps learners adapt to different languages and broaden their coding skills.



```
def max_sum_subarray(arr, k):
    n = len(arr)
    if n < k:
        return 0

    window_sum = sum(arr[:k])
    max_sum = window_sum

    for i in range(k, n):
        window_sum += arr[i] - arr[i - k]
        max_sum = max(max_sum, window_sum)

    return max_sum


arr = [2, 1, 5, 1, 3, 2]
k = 3
print(max_sum_subarray(arr, k))
```

Hello there, aspiring programmer! Let's embark on a little adventure to see how your Python code runs, step-by-step. Think of it like following a recipe to bake something delicious!

▸ **Step 1: The Adventure Begins - Calling the Function**

Our journey starts with the line `print(max_sum_subarray(arr, k))`. This is where your program calls out to the `max_sum_subarray` function, asking it to do some work.

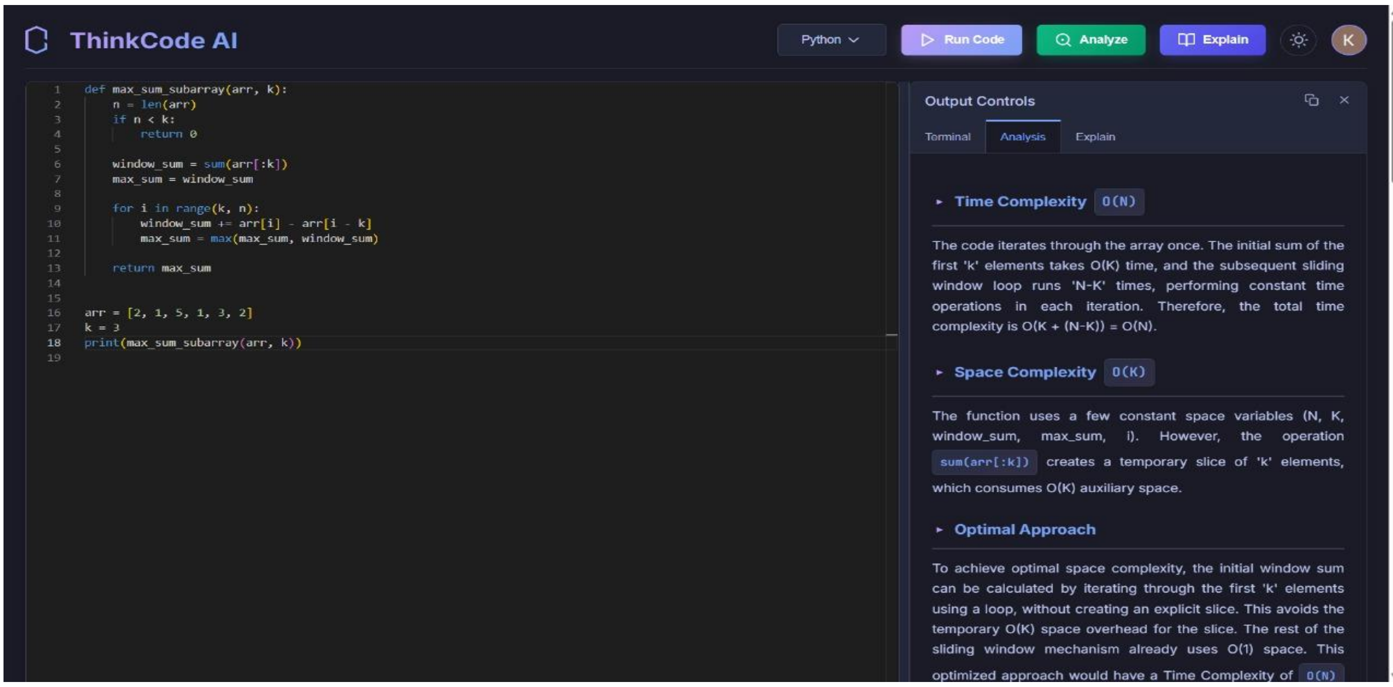
It passes two pieces of information: the list of numbers, `arr`, which is `[2, 1, 5, 1, 3, 2]`, and the window size, `k`, which is `3`.

Inside the function, the first thing that happens is calculating the length of the list. So, the variable `n` becomes `6` (because there are 6 numbers in `arr`).

Then, a quick check is made: `if n < k` (is `6` less than `3`

Fig:8.5 Multiple Languages supported in ThinkCode AI

The AI inspects the submitted code to detect syntax and logical errors. It then provides step-by-step explanations, highlights mistakes, and suggests improvements for better understanding.



```
def max_sum_subarray(arr, k):
    n = len(arr)
    if n < k:
        return 0

    window_sum = sum(arr[:k])
    max_sum = window_sum

    for i in range(k, n):
        window_sum += arr[i] - arr[i - k]
        max_sum = max(max_sum, window_sum)

    return max_sum


arr = [2, 1, 5, 1, 3, 2]
k = 3
print(max_sum_subarray(arr, k))
```

▸ **Time Complexity** `O(N)`

The code iterates through the array once. The initial sum of the first 'k' elements takes O(K) time, and the subsequent sliding window loop runs 'N-K' times, performing constant time operations in each iteration. Therefore, the total time complexity is O(K + (N-K)) = O(N).

▸ **Space Complexity** `O(K)`

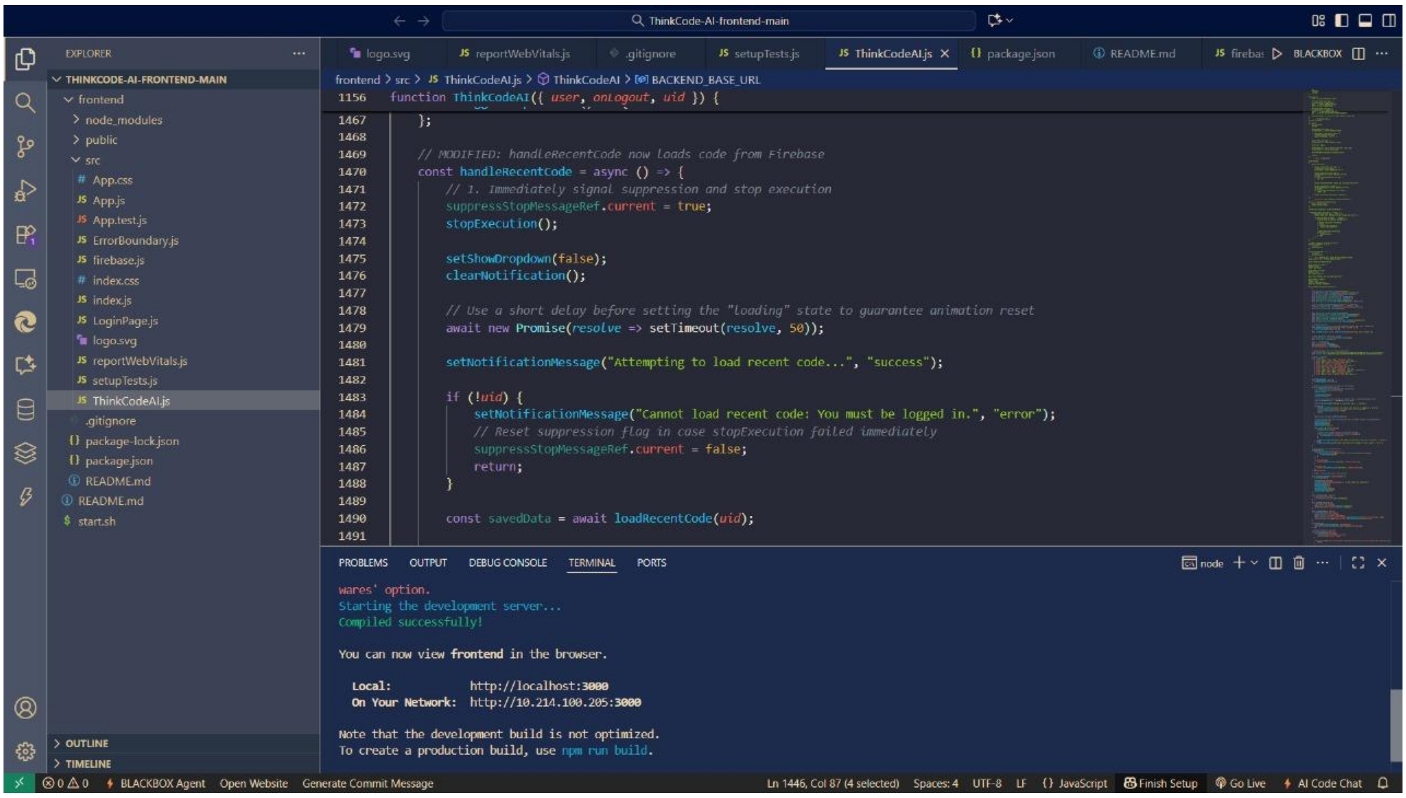
The function uses a few constant space variables (N, K, window_sum, max_sum, i). However, the operation `sum(arr[:k])` creates a temporary slice of 'k' elements, which consumes O(K) auxiliary space.

▸ **Optimal Approach**

To achieve optimal space complexity, the initial window sum can be calculated by iterating through the first 'k' elements using a loop, without creating an explicit slice. This avoids the temporary O(K) space overhead for the slice. The rest of the sliding window mechanism already uses O(1) space. This optimized approach would have a Time Complexity of `O(N)`

Fig:8.6 Analysis of Code in ThinkCode AI

Started the local development server for ThinkCode AI frontend project using the npm start command. This successfully compiles code, and the application is now running and accessible for testing at http://localhost:3000.

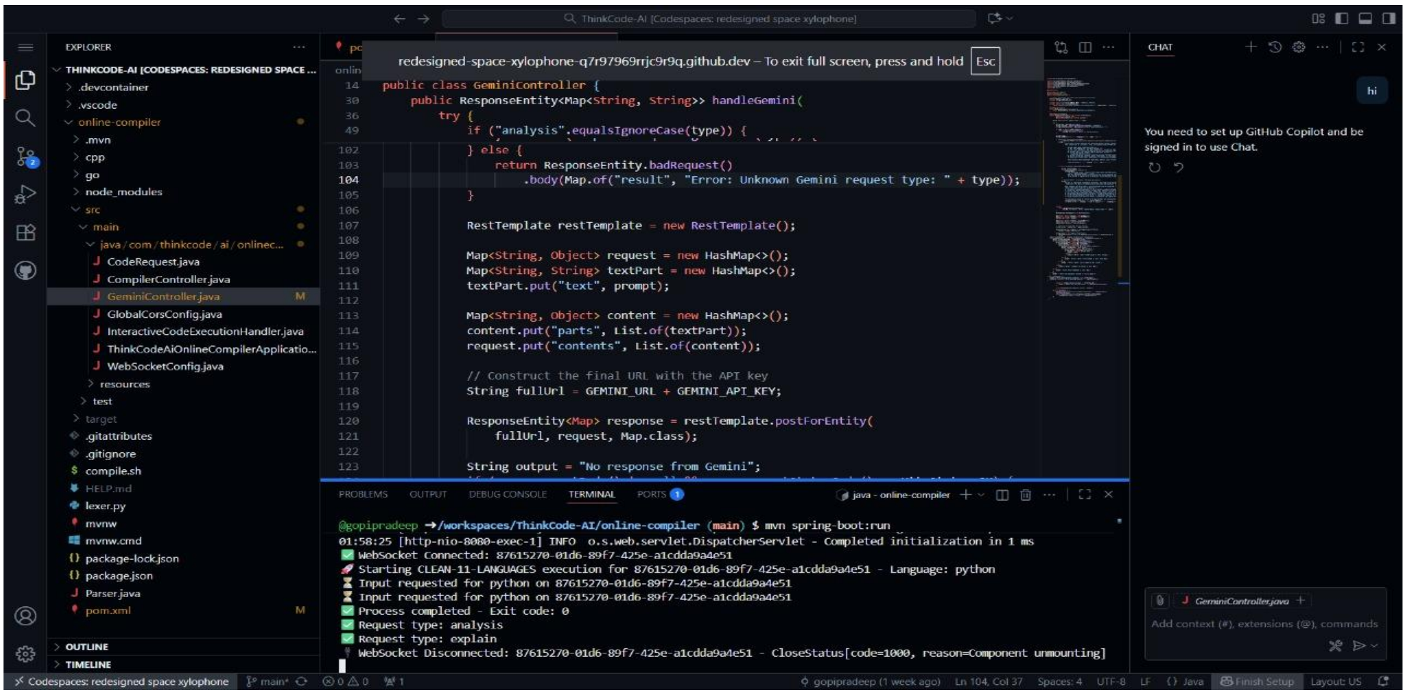


```
function ThinkCodeAI({ user, onLogout, uid }) {
    };

    // MODIFIED: handleRecentCode now loads code from Firebase
    const handleRecentCode = async () => {
        // 1. Immediately signal suppression and stop execution
        suppressStopMessageRef.current = true;
        stopExecution();

        setShowDropdown(false);
        clearNotification();

        // Use a short delay before setting the "loading" state to guarantee animation reset
        await new Promise(resolve => setTimeout(resolve, 50));

        setNotificationMessage("Attempting to load recent code...", "success");

        if (!uid) {
            setNotificationMessage("Cannot load recent code: You must be logged in.", "error");
            // Reset suppression flag in case stopExecution failed immediately
            suppressStopMessageRef.current = false;
            return;
        }

        const savedData = await loadRecentCode(uid);
```

```
wares' option.
Starting the development server...
Compiled successfully!

You can now view frontend in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://10.214.100.205:3000

Note that the development build is not optimized.
To create a production build, use npm run build.
```

Fig:8.7 Frontend of ThinkCode AI

Started ThinkCode AI backend service using the Maven command mvn spring-boot:run. The server is running successfully, and it's handles requests to analyze and explain code.

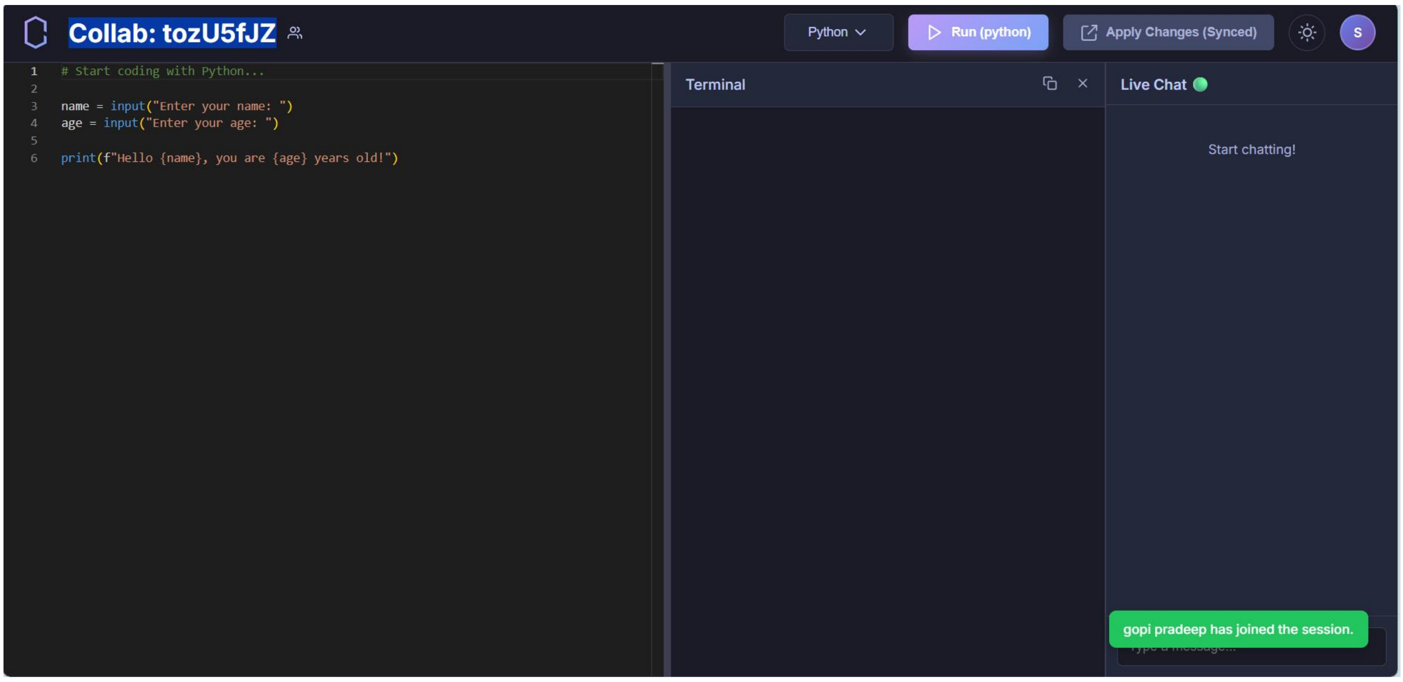


```
public class GeminiController {
    public ResponseEntity<Map<String, String>> handleGemini(
        try {
            if ("analysis".equalsIgnoreCase(type)) {
            } else {
                return ResponseEntity.badRequest()
                    .body(Map.of("result", "Error: Unknown Gemini request type: " + type));
            }

            RestTemplate restTemplate = new RestTemplate();

            Map<String, Object> request = new HashMap<>();
            Map<String, String> textPart = new HashMap<>();
            textPart.put("text", prompt);

            Map<String, Object> content = new HashMap<>();
            content.put("parts", List.of(textPart));
            request.put("contents", List.of(content));

            // Construct the final URL with the API key
            String fullUrl = GEMINI_URL + GEMINI_API_KEY;

            ResponseEntity<Map> response = restTemplate.postForEntity(
                fullUrl, request, Map.class);

            String output = "No response from Gemini";
```

```
@gopipradeep →/workspaces/ThinkCode-AI/online-compiler (main) $ mvn spring-boot:run
01:58:25 [http-nio-8080-exec-1] INFO  o.s.web.servlet.DispatcherServlet - Completed initialization in 1 ms
✅ WebSocket Connected: 87615270-01d6-89f7-425e-a1cdda9a4e51
🚀 Starting CLEAN-11-LANGUAGES execution for 87615270-01d6-89f7-425e-a1cdda9a4e51 - Language: python
⌛ Input requested for python on 87615270-01d6-89f7-425e-a1cdda9a4e51
⌛ Input requested for python on 87615270-01d6-89f7-425e-a1cdda9a4e51
✅ Process completed - Exit code: 0
✅ Request type: analysis
✅ Request type: explain
🔌 WebSocket Disconnected: 87615270-01d6-89f7-425e-a1cdda9a4e51 - CloseStatus[code=1000, reason=Component unmounting]
```

Fig:8.8 Backend of ThinkCode AI

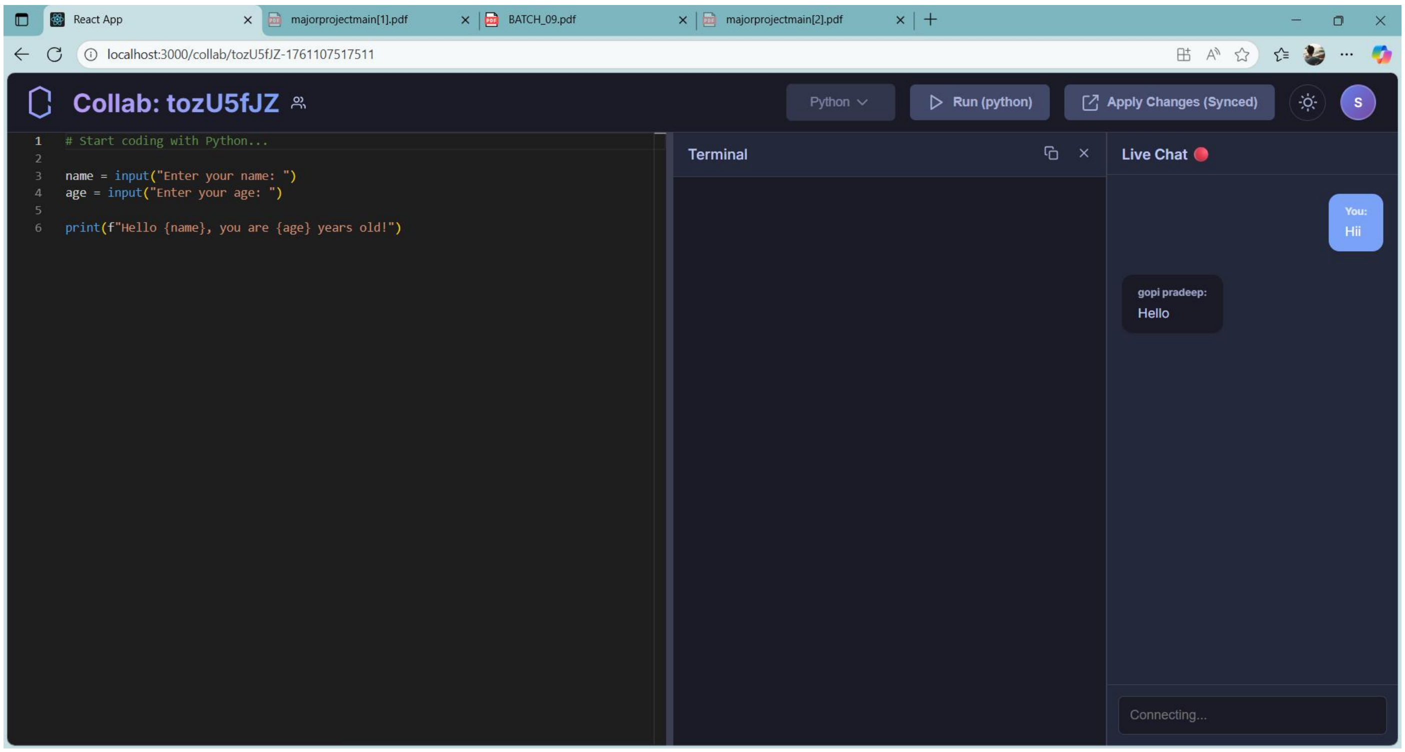
The user can be interact with the another user in the codeing environment by using the colabrative option in the thinkcode AI the two users are interact at a time and made the changes in the code that can be appear in the real time



```
# Start coding with Python...

name = input("Enter your name: ")
age = input("Enter your age: ")

print(f"Hello {name}, you are {age} years old!")
```

Fig:8.9 Colabrative Codeing in ThinkCode AI

The connected users can directly chart with each other at the run time so they can clarify the douths with each other



```
# Start coding with Python...

name = input("Enter your name: ")
age = input("Enter your age: ")

print(f"Hello {name}, you are {age} years old!")
```

You: Hii

gopi pradeep: Hello

Fig:8.10 Live Conversation in ThinkCode AI

CHAPTER 9

CONCLUSION AND FUTURE PLAN

9 CONCLUSION AND FUTURE PLAN

Conclusion:

The proposed system, **ThinkCode AI**, effectively transforms coding education by merging Artificial Intelligence with interactive learning. It explains programming concepts in plain language, visualizes execution flow, and measures performance efficiency. Learners not only understand the logic behind their code but also receive actionable feedback to improve their skills. By integrating educational AI, ThinkCode AI helps bridge the gap between theory and practice, turning complex programming challenges into clear, structured learning experiences.

Future Plan:

- Integrate **voice-based code explanations** for accessibility.
- Add **personalized AI mentors** to track learner progress.
- Support **additional languages** like Java, Python, and C++.
- Implement **AI-driven performance dashboards** for educators.
- Enable **cloud-based code history and collaborative learning**.

CHAPTER 10

REFERENCES

10 REFERENCES

[1] A. Ahmad, M. Khan, and R. Gupta, *"AI-Powered Code Understanding: A Survey on Code Summarization and Explanation Techniques,"* IEEE Access, vol. 9, pp. 112345–112359, 2021.

[2] T. Chen, Y. Liu, and M. Sun, *"CodeBERT: A Pre-Trained Model for Programming and Natural Languages,"* in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 1536–1547.

[3] M. Allamanis, E. T. Barr, P. Devanbu, and C. Sutton, *"A Survey of Machine Learning for Big Code and Naturalness,"* ACM Computing Surveys, vol. 51, no. 4, pp. 1–37, 2018.

[4] J. R. Cordy, *"Comprehending Programs through Dynamic Visualization,"* in *IEEE International Workshop on Program Comprehension*, 2019, pp. 12–19.

[5] A. Vaswani et al., *"Attention Is All You Need,"* in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008.

[6] K. Li, S. Feng, and J. Huang, *"AI-Assisted Code Review and Bug Detection using Transformer Models,"* IEEE Transactions on Software Engineering, vol. 48, no. 11, pp. 4210–4222, 2022.

[7] S. Ray and N. Kumar, *"Interactive Visual Debugging Tools for Learning Programming Concepts,"* in *Proceedings of the IEEE Conference on Learning and Teaching in Computing and Engineering (LaTiCE)*, 2020, pp. 45–52.

[8] P. Jain and R. Singh, *"Integrating Compiler Visualization and AI Feedback for Programming Education,"* International Journal of Educational Technology in Higher Education, vol. 19, no. 5, 2023.

[9] D. Li and K. S. Chan, *"Real-Time Code Execution Analysis and Visualization Framework for Teaching Programming,"* in *IEEE Global Engineering Education Conference (EDUCON)*, 2021, pp. 1340–1348.

[10] M. Chen et al., *"Evaluating Large Language Models Trained on Code,"* arXiv preprint arXiv:2107.03374, 2021.