**AI DRIVEN ASSESSMENT TOOL FOR CODE COMPREHENSION**

A Main Project thesis submitted in partial fulfilment of requirements for the award of degree for VIII Semester

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE AND ENGINEERING**

By

**P. SAI VENKATESH PRABHU (2181951065)**

**CH. NOOKA VENKATA PRAVEEN (2181951058)**

**R. SAKETH KUMAR (2181951069)**

**P. SUDHA KEERTHI (2181951066)**

**G. SAHITHI PRASANNA (2181951062)**

**R. LAVANYA (2181951068)**

Under the esteemed guidance of

**Dr. G. Rama Krishna** **M. Tech (Ph. D)**

**Assistant Professor**

**Department of Computer Science and Engineering**



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COLLAGE OF ENGINEERING

Dr.B.R.AMBEDKAR UNIVERSITY-SRIKAKULAM

ETCHERLA-532410,ANDHRA PRADESH,INDIA

(2021-2025)

**CERTIFICATE**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**Dr. B. R. Ambedkar University**



This is to certify that the main project entitled **“AI Driven Assessment Tool for Code Comprehension”** being submitted by

**P. SAI VENKATESH PRABHU (2181951065)**

**CH. NOOKA VENKATA PRAVEEN (2181951058)**

**R. SAKETH KUMAR (2181951069)**

**P. SUDHA KEERTHI (2181951066)**

**G. SAHITHI PRASANNA (2181951062)**

**R. LAVANYA (2181951068)**

has been carried out in partial fulfilment of requirement for the award of the degree in Bachelor of Technology in Computer science and Engineering in Dr. B. R. Ambedkar University, Srikakulam during the tenure 2021-2025. This work carried out by the above said student fraternity under my guidance and supervision.

**Signature of the Guide**

G. Rama Krishna **M. Tech (Ph. D)**

**Supervisior/Mentor External Examiner Head Of The Department**

G. Rama Krishna **M. Tech (Ph. D)**  B. Yyasa Geetha **M. Tech (Ph. D)**

**DECLARATION**

We hereby declare that this project entitled **“CodeSense - AI Driven Assessment Tool for Code Comprehension”** in submitted by us for the award of the degree of Bachelor of Technology in Computer Science and Engineering,Under the guidance of G. Rama Krishna, Dr.B.R.Ambedkar University, Srikakulam.

We also declare that this project report has not been submitted to any other university.

**Project Associates**

P. SAI VENKATESH PRABHU (2181951065)

CH. NOOKA VENKATA PRAVEEN (2181951058)

R. SAKETH KUMAR (2181951069)

P. SUDHA KEERTHI (2181951066)

G. SAHITHI PRASANNA (2181951062)

R. LAVANYA (2181951068)

Place:Srikakulam

Date:

**ACKNOWLEDGEMENT**

It is a great privilege a honour for us to work under the inspiring and meticulous guidance to express my sincere gratitude to G. Rama Krishna, project supervisor, had given continuous, abiding interest has finally evolved into this research work.

I thank B. Vyasa Geetha, Department of computer science and Engineering, Dr. B. R. Ambedkar University, Srikakulam, for her constant advice and encouragement in the completion of this work.

I thank, Prof Ch. Raja Sekhara Rao, Principle Of Dr.B.R.Ambedkar University for facilities provided to accomplish to this project.

I am grateful to all faculty members of department of computer science, Dr . B. R. Ambedkar University, Srikakulam, for their valuable suggestions and encouragement. We appreciate and thanks to all those names are not listed here. But whose affection and help, I greatly acknowledgement and cherish opportunity We would like to take this unique opportunity to acknowledge support and blessing.

**Project Associates**

P. SAI VENKATESH PRABHU (2181951065)

CH. NOOKA VENKATA PRAVEEN (2181951058)

R. SAKETH KUMAR (2181951069)

P. SUDHA KEERTHI (2181951066)

G. SAHITHI PRASANNA (2181951062)

R. LAVANYA (2181951068)

Place: Srikakulam

Date:

**ABSTRACT**

In computer science education, verifying a student's understanding of code is a critical yet challenging task, particularly in lab settings where traditional assessments often focus on code outputs without providing insight into the student’s conceptual grasp. To address this, an AI-driven assessment tool is being developed to evaluate students’ comprehension of their code submissions. By leveraging generative AI, the tool dynamically creates customized questions tailored to each student’s code, prompting them to explain their logic, predict outcomes of code modifications, and address related theoretical concepts. The AI analyzes the submitted code to generate context-specific follow-up questions, ensuring a thorough assessment of both code correctness and the depth of conceptual understanding, thereby distinguishing between rote memorization and true comprehension. Student responses are automatically evaluated, with scores integrated into lab or viva assessments, providing educators with a more accurate, efficient, and insightful means of grading. This innovative approach not only enhances the evaluation process but also fosters deeper learning and engagement among students.

**Keywords:** Generative AI, Web application, Learning, LLM API, Code Comprehension, Online Coding Platform.

**INDEX**

[CHAPTER 1. INTRODUCTION](#_Toc42230)

[1.1 Purpose 1](#_Toc42231)

[1.2 Scope 1](#_Toc42232)

[CHAPTER 2. LITERATURE SURVEY](#_Toc42233)

[2.1 Literature Survey 2](#_Toc42234)

[2.2 Literature Survey Summary 3](#_Toc42235)

[2.3 Objectives 4](#_Toc42236)

[CHAPTER 3. EXISTING SYSTEM](#_Toc42237)

[3.1 Methodology 5](#_Toc42238)

[3.2 Performance Measure 5](#_Toc42239)

[3.3 Drawbacks 6](#_Toc42240)

[CHAPTER 4. PROPOSED SYSTEM](#_Toc42241)

[4.1 System Architecture 7](#_Toc42242)

[4.2 Modules 7](#_Toc42243)

[4.3 Flow Diagram 9](#_Toc42244)

[4.4 Methodology 10](#_Toc42245)

[4.5.Advantage 13](#_Toc42246)

[CHAPTER 5. SYSTEM DESIGN](#_Toc42247)

[5.1 Introduction to UML 14](#_Toc42248)

[5.2.Bulding blocks of the UML 15](#_Toc42249)

[5.3.UML Diagrams 19](#_Toc42250)

[CHAPTER 6. SOFTWARE DESCRIPTION](#_Toc42251)

[6.1 Development Environment 21](#_Toc42252)

[6.2 System Architecture 21](#_Toc42253)

[6.3 System Workflow 22](#_Toc42254)

[CHAPTER 7. SAMPLE CODING 24](#_Toc42255)

[CHAPTER 9. TESTING 35](#_Toc42257)

[CHAPTER 10. CONCLUSION 36](#_Toc42258)

[CHAPTER 11. FUTURE SCOPE 37](#_Toc42259)

[CHAPTER 12.USER MANUAL 38](#_Toc42260)

CHAPTER13.REFERENCES

# CHAPTER 1. INTRODUCTION

In the realm of computer science education, assessing a student's true understanding of programming concepts is a complex task, often limited by traditional methods that focus solely on code outputs. To bridge this gap, our project introduces an AI-driven assessment tool designed to evaluate students' comprehension of their code submissions dynamically. This tool leverages generative AI to create customized questions based on the submitted code, prompting students to explain their logic, predict outcomes of code modifications, and address related theoretical concepts. By integrating this tool into lab settings, educators can gain deeper insights into students' understanding, moving beyond mere code correctness to assess conceptual mastery.

### 1.1. Purpose

The purpose of this project is to enhance the assessment process in computer science education by providing a more accurate and efficient means of evaluating student comprehension. By distinguishing between rote memorization and true understanding, this tool aims to foster deeper learning and engagement among students, ultimately improving educational outcomes. This innovative approach not only benefits educators by streamlining the grading process but also empowers students to develop a more profound grasp of programming concepts.

### 1.2. Scope

The scope of this project encompasses the development of a comprehensive platform that facilitates both student and teacher interactions. The platform will allow students to submit their code, answer AI-generated questions, and receive immediate feedback on their understanding. Teachers will be able to post lab questions, monitor student progress, and access detailed results. The backend will handle user authentication, question posting, and mark storage, while the integration of a Large Language Model (LLM) API will enable the generation of context-specific questions and the evaluation of student responses.

# CHAPTER 2. LITERATURE SURVEY

### 2.1. Literature Survey

**2.1.1. Enhancing Academic Performance with Generative AI-Based Quiz Platform (Chia-Kai**

**Chang, Lee-Chia-Tung Chien, 2024):**

The generative AI-based quiz platform enhances education by creating personalized quizzes that assess conceptual understanding, offering instant feedback and performance insights. It features a user-friendly interface, a robust backend, and an AI model for tailored questions. Initial testing shows improved engagement and performance, with future plans to expand subjects, integrate multimedia, and refine AI for adaptive learning.

**2.1.2. Generating Multiple Choice Questions for Computing Courses using Large Language Models (S. Liu, J. Smith, and A. Johnson, 2023)**:

The paper "Generating Multiple Choice Questions for Computing Courses using Large Language Models" explores using LLMs to automate MCQ creation for computing education. By analyzing course content, LLMs generate relevant, high-quality questions, saving educators time while ensuring pedagogical accuracy. Challenges like bias and difficulty levels are addressed, with testing showing LLM-generated MCQs effectively assess student knowledge. This approach offers a scalable, dynamic way to enhance assessments and improve learning outcomes.

**2.1.3. Code generation using the Generative AI (Time Schlippe, 2024):**

The paper "Code Generation using Generative AI" examines how AI models automate code creation, streamlining software development. Trained on programming datasets, these models generate accurate, context-aware code, reducing manual effort and errors. Challenges like accuracy and security are addressed, with experiments showing AI's potential to boost productivity, accelerate development, and make coding more accessible.

**2.1.4. AI-Bases Quiz System for Personalized Learning (P. Wang, X. Zhang, and Y. Chen, 2023):**

The AI-based quiz system (iQS) is designed for integration with popular LMS platforms like Moodle and MOOCs. It consists of four core components: quiz content creator, quiz generator, knowledge graph manager, and LMS quiz plug-in. The system's algorithms are reusable, requiring only domain-specific knowledge and learner data for adaptation to new applications.

**2.1.5. Program Code Generation with Generative Ais (Baskhad Idrisov and Tim Schlippe, 2024):**

This paper compares the correctness, efficiency, and maintainability of AI- and human-generated code using various complexity and performance metrics. Experiments on LeetCode problems show GitHub Copilot performed best, solving 50% of problems, while other AIs had lower success rates. Despite AI-generated code solving only 20.6% of problems, minor modifications could make 8.7% correct, saving up to 71.3% in development time.

### 2.2. Literature Survey Summary

The integration of generative AI in education and software development, as explored in the research papers "Generating Multiple Choice Questions for Computing Courses using Large Language Models," "Code Generation using Generative AI," and the concept of an AI-driven assessment tool, highlights the transformative potential of AI in these fields. The first paper demonstrates how large language models (LLMs) can automate the creation of multiple-choice questions (MCQs) for computing courses, ensuring pedagogical relevance and reducing educators' workload. By analyzing course materials, LLMs generate contextually appropriate questions that assess a wide range of topics, from basic programming to advanced computational theories. This approach not only enhances the efficiency of assessment creation but also ensures that questions are aligned with learning objectives, providing students with a more comprehensive evaluation of their understanding.

The second paper, "Code Generation using Generative AI," showcases the application of AI in automating code creation, which can significantly streamline software development processes. By training on extensive datasets of programming languages and frameworks, generative AI models can produce functional and contextually relevant code snippets, reducing manual effort and minimizing errors. This capability is particularly beneficial for automating repetitive tasks, prototyping, and assisting developers in writing efficient code. When combined with the AI-driven assessment tool discussed earlier, these technologies can create a holistic educational ecosystem. For instance, students can submit code for evaluation, receive AI-generated feedback, and answer dynamically created MCQs to demonstrate their conceptual understanding. This integrated approach not only enhances learning outcomes but also prepares students for real-world software development challenges, making education more interactive, efficient, and aligned with industry needs

### 2.3. Objectives

* **Develop AI-Driven Assessments**: Create a tool using generative AI to evaluate code submissions and generate tailored questions, assessing both code correctness and conceptual understanding.
* **Enhance Learning Through Feedback**: Provide instant, automated feedback on student responses to help them improve their coding skills and theoretical knowledge in real-time.
* **Streamline Educator Workload**: Automate question generation and evaluation, reducing the time educators spend on creating and grading assessments while ensuring alignment with learning objectives. ● **Promote Scalability and Accessibility**: Design the platform to be adaptable across programming languages, course levels, and class sizes, making it accessible to diverse learners and educational contexts..

# 

# CHAPTER 3. EXISTING SYSTEM

### 3.1. Methodology

**1.Requirement Analysis**: Identify the needs of educators and students, focusing on improving code assessment and conceptual understanding.

**2.System Design**: Develop a modular architecture with a user-friendly frontend for students and educators, a backend for processing and evaluation, and integration with generative AI models.

**3.AI Model Integration**: Use large language models (LLMs) to generate context-specific questions and evaluate student responses based on submitted code.

**4.Implementation**: Build the platform using modern web technologies, ensuring seamless integration with existing learning management systems (LMS).

**5.Testing and Validation**: Conduct pilot tests with real students and educators to gather feedback and refine the system.

**6.Deployment and Monitoring**: Roll out the platform for broader use and continuously monitor its performance to make iterative improvements.

### 3.2. Performance Measure

**1.Accuracy of Question Generation**: Evaluate how well the AI-generated questions align with the learning objectives and the submitted code

.

**2.Response Evaluation Precision**: Measure the accuracy of the AI in assessing student responses and providing relevant feedback.

**3.User Satisfaction**: Collect feedback from students and educators on the usability and effectiveness of the platform.

**4.Time Efficiency**: Assess the reduction in time educators spend on creating and grading assessments.

**5.Learning Outcomes**: Compare student performance and engagement levels before and after using the platform to gauge its impact on learning.

### 3.3. Drawbacks

Despite advancements, existing systems have several limitations:

1. **Limited Insight into Conceptual Understanding:** Traditional systems often focus on code output without assessing the depth of a student's understanding.

1. **Manual Effort**: Educators spend significant time creating and grading assessments, which can be inefficient and prone to human error**.**

1. **Lack of Personalization:** Existing tools may not tailor questions to individual student needs, leading to a one-size-fits-all approach**.**

1. **Delayed Feedback:** Students often receive feedback long after submitting their work, which can hinder immediate learning and correction**.**

1. **Scalability Issues:** Traditional methods may not scale well for large classes or diverse programming languages, limiting their effectiveness in varied educational contexts.

# CHAPTER 4. PROPOSED SYSTEM

### 4.1. System Architecture

CodeSense follows a three-tier architecture comprising the Frontend (React.js), Backend (Node.js/Express.js), and Database (MongoDB). This structure ensures scalability, modularity, and efficient handling of user interactions while providing real-time updates and AI-driven feedback.

**Presentation Layer (Frontend - React.js):**

* Handles user input, displays assignments, quizzes, and feedback.
* Uses Monaco Editor for an interactive code editing experience.
* Communicates with the backend via REST APIs.

**Application Layer (Backend - Node.js/Express.js):**

* Manages API requests, processes code submissions, and handles user authentication.
* Ensures secure user sessions using JWT (JSON Web Token).
* Implements AI-driven feedback and quiz generation using Google Generative AI.

**Data Layer (Database - MongoDB):**

* Stores user authentication details, assignments, submissions, and marks. • Uses Mongoose ORM for structured database interactions

### 4.2. Modules

**4.2.1. User Management Module**

* **Functionality**: Combines user authentication, role-based access, and user profile management.
* **Features**:
  + - * Secure login and signup for students and teachers. o Role-based access control (student/teacher). o Storage and management of user profiles (students and teachers).
      * Routes for /login, /signup, and profile management.

**4.2.2. Learning and Assessment Module**

* **Functionality**: Combines student and teacher functionalities, including question submission, code execution, lab posting, and evaluation.

* **Features**:
  + **Student Side**:
    - * + Submit questions and execute code.
        + View evaluated answers and feedback. o **Teacher Side**:
        + Post lab questions and assignments.
        + View student submissions and results.
        + Assign and manage marks.
  + Integrated AI-driven question generation and answer evaluation (via APIs).
  + Dynamic MCQ creation and automated feedback.

**4.2.3. Database and API Module**

* **Functionality**: Combines database management and API integration for seamless data handling and advanced functionalities.
* **Features**:
  + Storage of user data, questions, answers, and marks.
  + Integration of APIs for:
    - * + Question generation (MCQs).
        + Answer evaluation and feedback.
        + Code execution (if applicable).
  + Scalable and secure data management.

**4.2.4. Server, Routing, and Security Module**

* **Functionality**: Combines server configuration, routing, and security measures.
* **Features**:
  + Server setup to handle user requests and data flow. o Secure routing for authentication, question posting, answer submission, and marks . o Data encryption and protection against unauthorized access.
  + Efficient handling of traffic and scalability.

**4.2.5. User Interface and Notification Module**

* **Functionality**: Combines the user interface design and notification system for a seamless user experience.
* **Features**:
  + - * + Intuitive and responsive UI for students and teachers.s

* + - * + Notifications for:

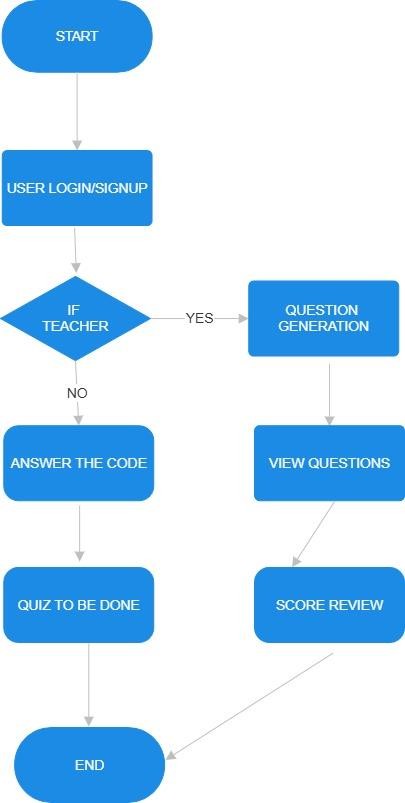
New lab questions (students).

Submitted assignments and evaluated results (teachers).

Deadlines and updates.

o Easy navigation and accessibility for all users

### 4.3. Flow Diagram



*Fig. 4.1 Flow of the CodeSense Application*

### 4.4. Methodology

**4.4.1. Project Initiation and Planning**

**Objective Definition**: The project aims to create an online platform for interactive learning and assessment, catering to students and teachers. The platform will include features like question submission, code execution, lab posting, and result evaluation.

**Requirement Gathering**: Conduct meetings with stakeholders (educators, students, and developers) to gather detailed requirements for the platform.

**Scope Finalization**: Define the scope, including user authentication, role-based access, database management, API integration, and server configuration.

**Timeline and Resource Allocation**: Create a project timeline, assign tasks to team members, and allocate resources (tools, technologies, and budget).

**4.4.2. User Authentication and Role Management**

**Design Authentication System**: Develop a secure login and signup system for students and teachers. Use encryption (e.g., bcrypt) to protect user credentials.

**Role-Based Access Control**: Implement role-based access to ensure students can only access student-specific features (e.g., question submission, code execution), while teachers can access teacherspecific features (e.g., posting lab questions, viewing results).

**Routes for Authentication**: Create routes for /login and /signup to handle user authentication and registration.

**4.4.3. Student Functionality Development**

**Question Submission Interface**: Develop a user-friendly interface for students to submit questions related to their coursework.

**Code Execution Environment**: Implement a code execution feature that allows students to write, execute, and test code directly on the platform. Use APIs or sandbox environments to ensure secure code execution.

**Feedback and Evaluation**: Integrate an AI-driven evaluation system to provide feedback on submitted questions and code. Ensure the feedback is detailed and actionable.

**4.4.4. Teacher Functionality Development**

**Lab Question Posting**: Develop a feature for teachers to post lab questions and assignments. Include options for adding instructions, deadlines, and grading criteria.

**Result Monitoring**: Create a dashboard for teachers to view student submissions, evaluate them, and assign marks. Include filters and sorting options for easy navigation.

**Marks Management**: Implement a system to store and manage student marks in the database.

Ensure teachers can update marks and generate reports.

**4.4.5. Database Design and Implementation**

**Database Schema Design**: Design a database schema to store user information (students and teachers), questions, answers, and marks. Use a relational database (e.g., MySQL, PostgreSQL) for structured data storage.

**Data Storage**: Configure the database to store:

User profiles (name, email, role, etc.).

Lab questions and student submissions.

Evaluated answers and marks.

**Data Integrity and Security**: Implement constraints and validation rules to ensure data integrity.

Use encryption to protect sensitive data.

**4.4.6. API Integration**

**Question Generation API**: Integrate an API to automatically generate multiple-choice questions (MCQs) for assessments. Ensure the questions are contextually relevant and aligned with learning objectives.

**Answer Evaluation API**: Integrate an API to evaluate student answers and provide feedback. Use natural language processing (NLP) techniques for accurate evaluation.

**Code Execution API**: If required, integrate an API to handle code execution securely and provide real-time output.

**4.4.7. Server and Routing Configuration**

**Server Setup**: Set up a server to handle user requests and manage data flow. Use frameworks like Node.js, Django, or Flask for backend development.

**Secure Routing**: Create routes for:

Authentication (/login, /signup).

Question posting and retrieval.

Marks management and result viewing.

**Security Measures**: Implement HTTPS, input validation, and protection against common vulnerabilities (e.g., SQL injection, XSS).

**4.4.8. User Interface (UI) Development**

**Student Interface**: Design an intuitive interface for students to submit questions, execute code, and view feedback. Ensure the interface is responsive and works well on all devices.

**Teacher Interface**: Develop a dashboard for teachers to post lab questions, view student submissions, and assign marks. Include visualizations (e.g., charts, graphs) for better insights.

**Accessibility and Usability**: Ensure the platform is accessible to all users, including those with disabilities. Conduct usability testing to identify and fix issues.

## 4.5. Advantage

The project offers numerous advantages, including enhanced efficiency for educators by automating question generation and answer evaluation, significantly reducing their workload. Students benefit from an interactive learning environment where they can submit questions, execute code, and receive immediate feedback, fostering better understanding and skill development. The platform’s secure authentication and role-based access ensure data privacy and tailored experiences for both students and teachers. By integrating advanced technologies like AI-driven APIs, the system provides accurate and contextually relevant assessments, improving the overall quality of education. Additionally, the platform’s scalable design and user-friendly interface make it accessible and adaptable for a growing number of users, ultimately bridging the gap between academic learning and real-world application.

# CHAPTER 5. SYSTEM DESIGN

### 5.1. INTRODUCTION TO UML

Unified Modeling Language (UML) is a general-purpose modeling language. The main aim of UML is to define a standard way to visualize the way a system has been designed. It is quite similar to blueprints used in other fields of engineering. UML is not a programming language, it is rather a visual language. We use UML diagrams to show the behavior and structure of a system. UML helps software engineers, businessmen, and system architects with modeling, design, and analysis.

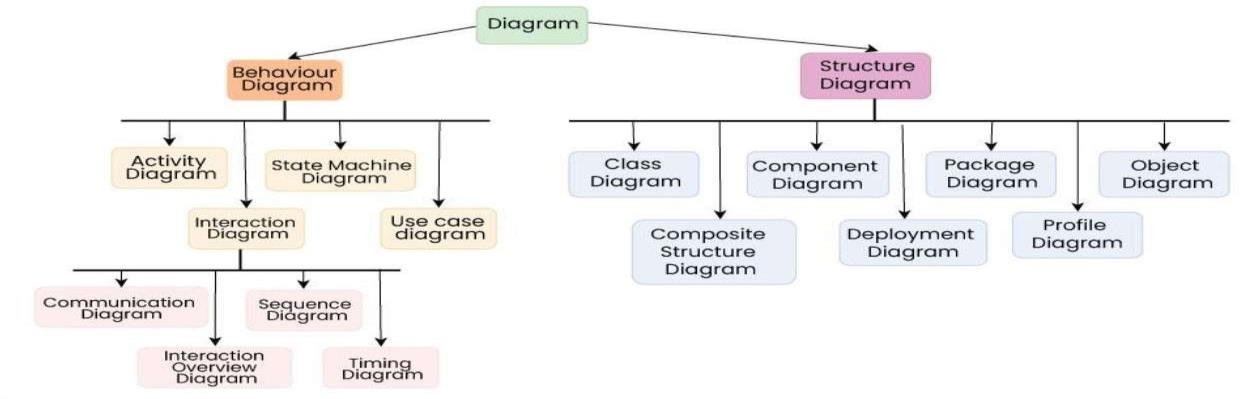
**Why do we need UML**

We need UML (Unified Modeling Language) to visually represent and communicate complex system designs, facilitating better understanding and collaboration among stakeholders. Below is why we need :

* Complex applications need collaboration and planning from multiple teams and hence require a clear and concise way to communicate amongst them.
* Businessmen do not understand code. So UML becomes essential to communicate with nonprogrammers about essential requirements, functionalities, and processes of the system.

**Types of UML Diagrams**

UML is linked with object-oriented design and analysis. UML makes use of elements and forms associations between them to form diagrams. Diagrams in UML can be broadly classified as:



*Fig 5.1: Classification of UML Diagrams*

## 5.2. BUILDING BLOCKS OF THE UML

The building blocks of UML can be defined as:

* Things
* Relationships
* Diagrams

**5.2.1.Things**

Things are the most important building blocks of UML. Things can be:

**1) Structural Things**

Structural things define the static part of the model. They represent the physical and conceptual elements.

Following are the brief descriptions of the structural things.

**Class −** Class represents a set of objects having similar responsibilities.



*Fig 5.2: Class*

**Interface −** Interface defines a set of operations, which specify the responsibility of a class.



*Fig 5.3: Interface*

**Collaboration −**Collaboration defines an interaction between elements.



*Fig 5.4: Collaboration*

**Use case −**Use case represents a set of actions performed by a system for a specific goal.



*Fig 5.5: Use case*

**Node −** A node can be defined as a physical element that exists at run time.



*Fig 5.7: Node*

***2****)***Behavioural Things**

A behavioural thing consists of the dynamic parts of UML models. Following are the behavioural things.

**Interaction −** Interaction is defined as a behaviour that consists of a group of messages exchanged among elements to accomplish a specific task.



*Fig 5.7: Interaction*

**State machine −** State machine is useful when the state of an object in its life cycle is important. It defines the sequence of states an object goes through in response to events. Events are external factors responsible for state change.



*Fig 5.8: State machine*

**3)Grouping Things**

Grouping things can be defined as a mechanism to group elements of a UML model together. There is only one grouping thing available.

**Package** − Package is the only one grouping thing available for gathering structural and behavioural things.



*Fig 5.9: Package*

**4)Annotational Things**

Annotational things can be defined as a mechanism to capture remarks, descriptions, and comments of UML model elements.

**Note** - It is the only one Annotational thing available. A note is used to render comments, constraints, etc. of an UML element.



*Fig 5.10: Note*

**5.2.2.Relationship**

Relationship is another most important building block of UML. It shows how the elements are associated with each other and this association describes the functionality of an application.

There are four kinds of relationships available.

1. **Dependency**

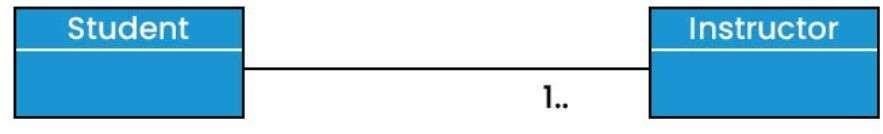
Dependency is a relationship between two things in which change in one element also affects the other.



*Fig 5.11: Dependency*

1. **Association**

Association is basically a set of links that connects the elements of a UML model. It also describes how many objects are taking part in that relationship.



*Fig 5.12: Association* **3) Generalization**

Generalization can be defined as a relationship which connects a specialized element with a generalized element. It basically describes the inheritance relationship in the world of objects.



*Fig 5.13: Generalization* **4) Realization**

Realization can be defined as a relationship in which two elements are connected. One element describes some responsibility, which is not implemented and the other one implements them. This relationship exists in case of interfaces.



*Fig 5.14: Realization*

**5.2.3.Diagrams**

UML diagrams are the ultimate output of the entire discussion. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system.

The visual effect of the UML diagram is the most important part of the entire process. All the other elements are used to make it complete.

UML includes the following nine diagrams:

* Class diagram
* Object diagram
* Use case diagram
* Sequence diagram
* Collaboration diagram
* Activity diagram
* State Chart diagram
* Deployment diagram
* Component diagram

## 5.3.UML Diagrams

**5.3.1. Use Case Diagram**

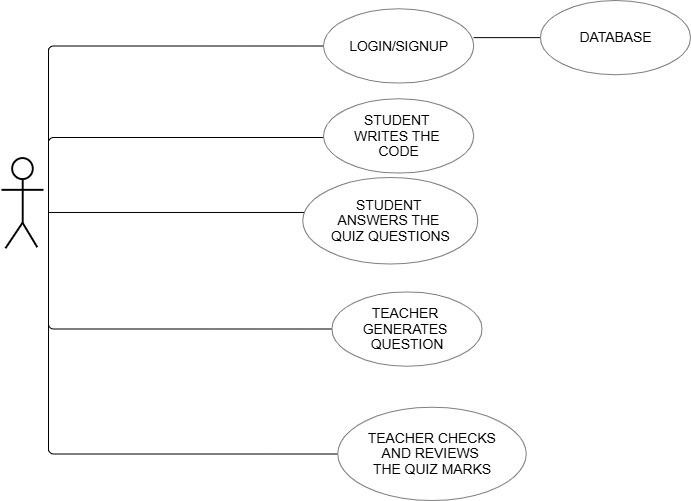


Fig .5.3.1. Use Case Diagram for CodeSense

**5.3.2. Component Diagram**

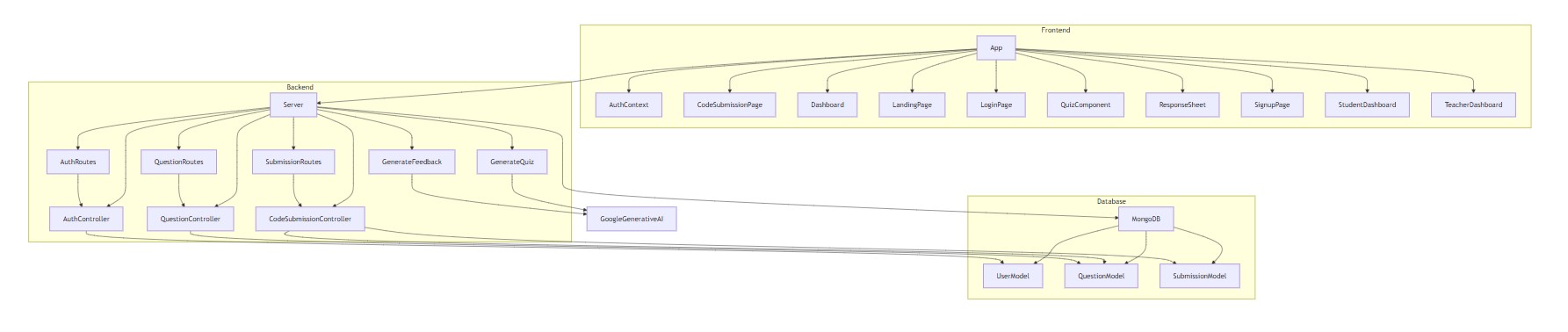


Fig.5.3.2. Component Diagram of the Application

**5.3.3. Class Diagram**

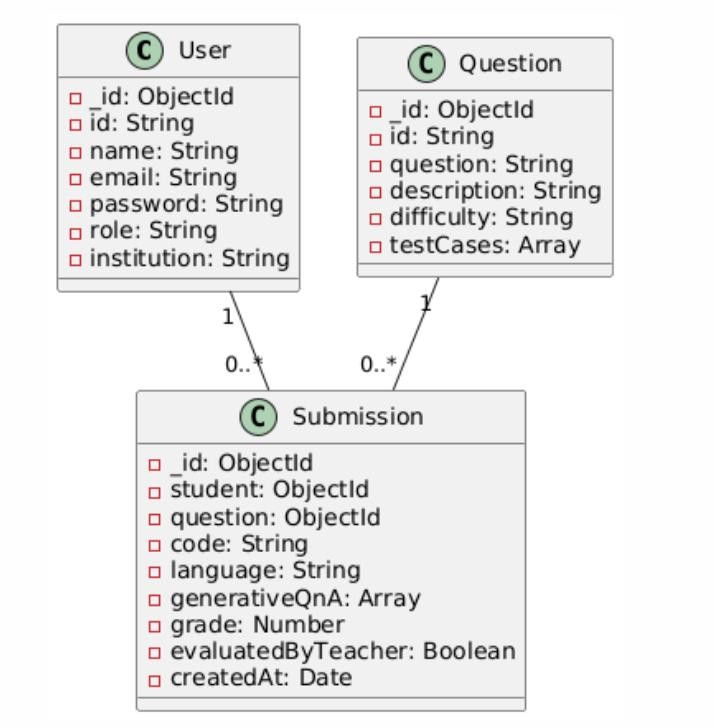


Fig.5.3.3. Class Diagram of Data Model

# CHAPTER 6. SOFTWARE DESCRIPTION

**CodeSense** is a web-based platform designed to facilitate interactive learning and assessment for students and teachers in coding education. The platform allows students to submit code, receive AI-generated feedback, and take quizzes to test their understanding. Teachers can create assignments, evaluate submissions, and monitor student progress in real-time. CodeSense integrates advanced technologies like AI and real-time updates to enhance the learning and teaching experience.

### 6.1. Development Environment

CodeSense is developed using Visual Studio Code (VS Code), leveraging its development tools and extensions for efficient coding. The project is built using the MERN stack (MongoDB, Express.js, React.js, Node.js) to ensure seamless integration between the frontend and backend, providing a scalable and high-performance platform for coding assignments and assessments.

Programming Languages Used:

* **JavaScript (React.js & Node.js):** Used for developing the frontend UI, managing user interactions, and handling API requests between the client and server.
* **HTML & CSS:** Used for structuring and styling the UI for responsiveness and visual appeal.

**Frameworks & Libraries**:

* **React.js**: Provides a component-based UI framework for efficient rendering and state management.
* **Node.js**: Executes server-side logic, handles API requests, and manages backend operations.
* **Express.js**: A lightweight backend framework for handling routing and middleware in the server.
* **MongoDB & Mongoose**: Stores user data, assignments, submissions, and marks in a NoSQL database.
* **JWT (JSON Web Token):** Provides secure user authentication and session management.
* **Google Generative AI:** Used for generating feedback and quizzes based on student submissions.

### 6.2. System Architecture

CodeSense follows a three-tier architecture comprising the Frontend (React.js), Backend (Node.js/Express.js), and Database (MongoDB). This structure ensures scalability, modularity, and efficient handling of user interactions while providing real-time updates and AI-driven feedback.

**Presentation Layer (Frontend - React.js):**

* Handles user input, displays assignments, quizzes, and feedback.
* Uses Monaco Editor for an interactive code editing experience.
* Communicates with the backend via REST APIs.

**Application Layer (Backend - Node.js/Express.js):**

* Manages API requests, processes code submissions, and handles user authentication.
* Ensures secure user sessions using JWT (JSON Web Token).
* Implements AI-driven feedback and quiz generation using Google Generative AI.

**Data Layer (Database - MongoDB):**

* Stores user authentication details, assignments, submissions, and marks.
* Uses Mongoose ORM for structured database interactions.

### 6.3. System Workflow

**User Authentication:**

* Users register/log in via the React.js UI.
* Credentials are verified and encrypted before storing in MongoDB.

**Code Submission and Feedback:**

* Students input code manually or upload files for assignments. o The frontend sends a POST request to the backend with the code. o The backend processes the code, generates feedback using AI, and returns the results to the frontend. o The frontend displays the feedback and allows students to review and improve their **code.**

**Quiz Generation:**

* After code submission, the backend generates a quiz based on the submitted code. o The frontend displays the quiz for students to complete.

* Results are evaluated and displayed in real-time.

**Assignment Management:** o Teachers create assignments with instructions and deadlines via the React.js UI. o The backend stores the assignment details in MongoDB.

* Students can view and submit assignments through their dashboard.

**Real-time Updates:**

* Students and teachers receive real-time updates on submission status, feedback, and results.
* The frontend continuously communicates with the backend to fetch the latest data.

**Performance Comparison:**

* Teachers can compare student performance across different assignments.
* The backend calculates and returns performance metrics, which are displayed on the teacher’s dashboard

# CHAPTER 7. SAMPLE CODING

##### 7.1. Front End

**7.1.1.index.html**

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<link rel="icon" type="image/png" sizes="64x64" href="/CS.png">

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>CodeSense</title>

</head>

<body>

<div id="root"></div>

<script type="module" src="/src/main.jsx"></script>

</body>

</html>

**7.2.2.package.json** {

"name": "codesense",

"private": true,

"version": "0.0.0",

"type": "module",

"scripts": {

"dev": "vite",

"build": "vite build",

"lint": "eslint .",

"preview": "vite preview"

},

"dependencies": {

"@monaco-editor/react": "^4.6.0",

"@tailwindcss/vite": "^4.0.3",

"axios": "^1.7.9",

"lucide-react": "^0.475.0",

"react": "^18.3.1",

"react-dom": "^18.3.1",

"react-hot-toast": "^2.5.2",

"react-router-dom": "^7.1.5",

"tailwindcss": "^4.0.3"

},

"devDependencies": {

"@eslint/js": "^9.17.0",

"@types/react": "^18.3.18",

"@types/react-dom": "^18.3.5",

"@vitejs/plugin-react": "^4.3.4",

"eslint": "^9.17.0",

"eslint-plugin-react": "^7.37.2",

"eslint-plugin-react-hooks": "^5.0.0",

"eslint-plugin-react-refresh": "^0.4.16",

"globals": "^15.14.0",

"vite": "^6.0.5"

}

}

**7.2.3.eslint.config.json** import js from '@eslint/js' import globals from 'globals' import react from 'eslint-plugin-react' import reactHooks from 'eslint-plugin-react-hooks' import reactRefresh from 'eslint-plugin-react-refresh'

export default [

{ ignores: ['dist'] },

{ files: ['\*\*/\*.{js,jsx}'],

languageOptions: { ecmaVersion: 2020, globals: globals.browser, parserOptions: { ecmaVersion: 'latest', ecmaFeatures: { jsx: true }, sourceType: 'module',

},

}, settings: { react: { version: '18.3' } }, plugins: { react,

'react-hooks': reactHooks,

'react-refresh': reactRefresh,

}, rules: {

...js.configs.recommended.rules,

...react.configs.recommended.rules,

...react.configs['jsx-runtime'].rules,

...reactHooks.configs.recommended.rules,

'react/jsx-no-target-blank': 'off',

'react-refresh/only-export-components': [

'warn',

{ allowConstantExport: true },

],

},

}

##### 7.2. Back End

**7.2.1.App.js**

import express from 'express'; import cors from 'cors';

import dotenv from 'dotenv'; // Load environment variables dotenv.config();

// Import routes import authRouter from './routes/authRouter.js'; import cookieParser from 'cookie-parser'; import questionRouter from './routes/questionRouter.js' import submissionRouter from './routes/submissionRouter.js';

// Initialize Express app const app = express();

const corsconfig = { origin: "http://localhost:5173", credentials: true

}

// Middleware app.use(cors( corsconfig )); // Enable CORS app.use(express.json()); // Parse JSON bodies app.use(cookieParser(process.env.COOKIE\_SECRET));

// Routes app.use('/api/auth', authRouter); // Authentication routes app.use('/api/questions', questionRouter); // Question-related routes app.use('/api/submissions', submissionRouter);

// app.use('/api/code', submissionRoutes); // Submission-related routes

// Health check endpoint app.get('/health', (req, res) => { res.status(200).json({ status: 'OK', message: 'Server is running' });

});

// 404 Handler app.use((req, res) => { res.status(404).json({ error: 'Route not found' });

});

// Error handling middleware app.use((err, req, res, next) => { console.error(err.stack); res.status(500).json({ error: 'Something went wrong!' });

});

export default app;

` **7.2.2 package.json**

{

"name": "backend",

"version": "1.0.0",

"description": "",

"main": "index.js",

"type": "module",

"scripts": {

"start": "nodemon server.js",

"test": "echo \"Error: no test specified\" && exit 1"

},

"keywords": [],

"author": "",

"license": "ISC",

"dependencies": {

"@google/generative-ai": "^0.22.0",

"bcrypt": "^5.1.1",

"bcryptjs": "^2.4.3",

"cookie-parser": "^1.4.7",

"cors": "^2.8.5",

"crypto": "^1.0.1",

"dotenv": "^16.4.7",

"express": "^4.21.2",

"express-validator": "^7.2.1",

"jsonwebtoken": "^9.0.2",

"mongodb": "^6.13.0",

"mongoose": "^8.10.0",

"together-ai": "^0.13.0"

},

"devDependencies": {

"nodemon": "^3.1.9"

}

}

**7.2.3.server.js** import app from './app.js'; import connectDB from './config/db.js'; import dotenv from 'dotenv'; // Load environment variables dotenv.config(); connectDB(); // Server setup const PORT = process.env.PORT || 5000; app.listen(PORT, () => { console.log(`Server is running on port ${PORT}`);

})

# 

#### CHAPTER 8. OUTPUTS

#### 8.1 Home Page

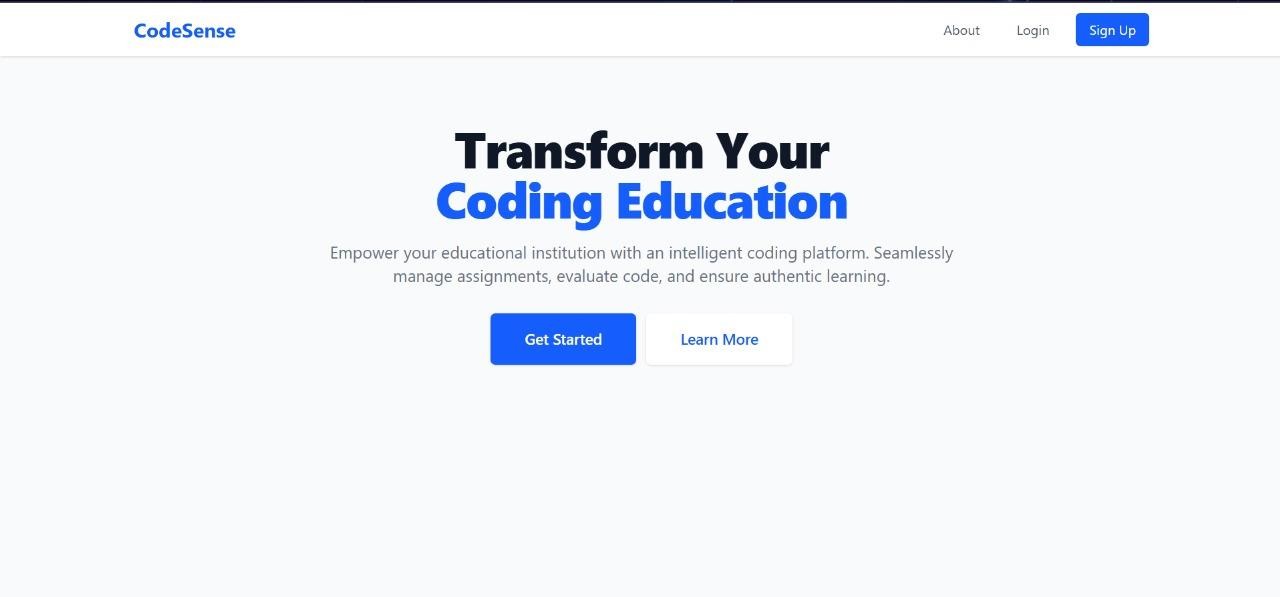


Fig. 8.1 : Home page of the UI

#### 8.2 Login& Sign Up Page

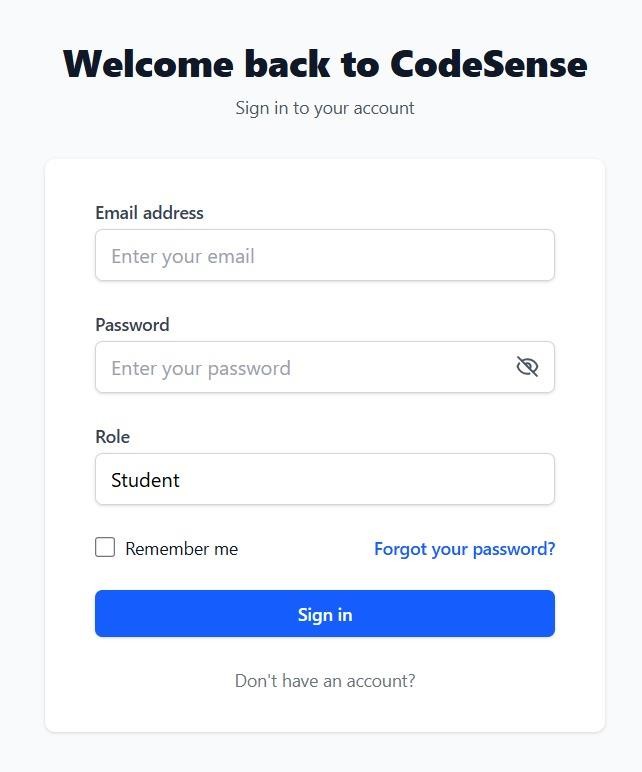


Fig. 8.2.1: Student Login Page

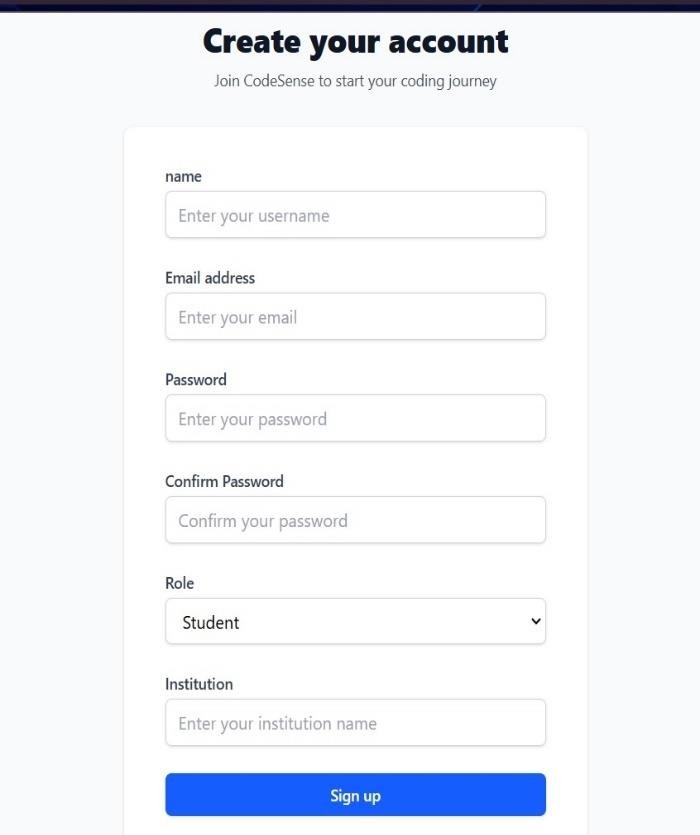
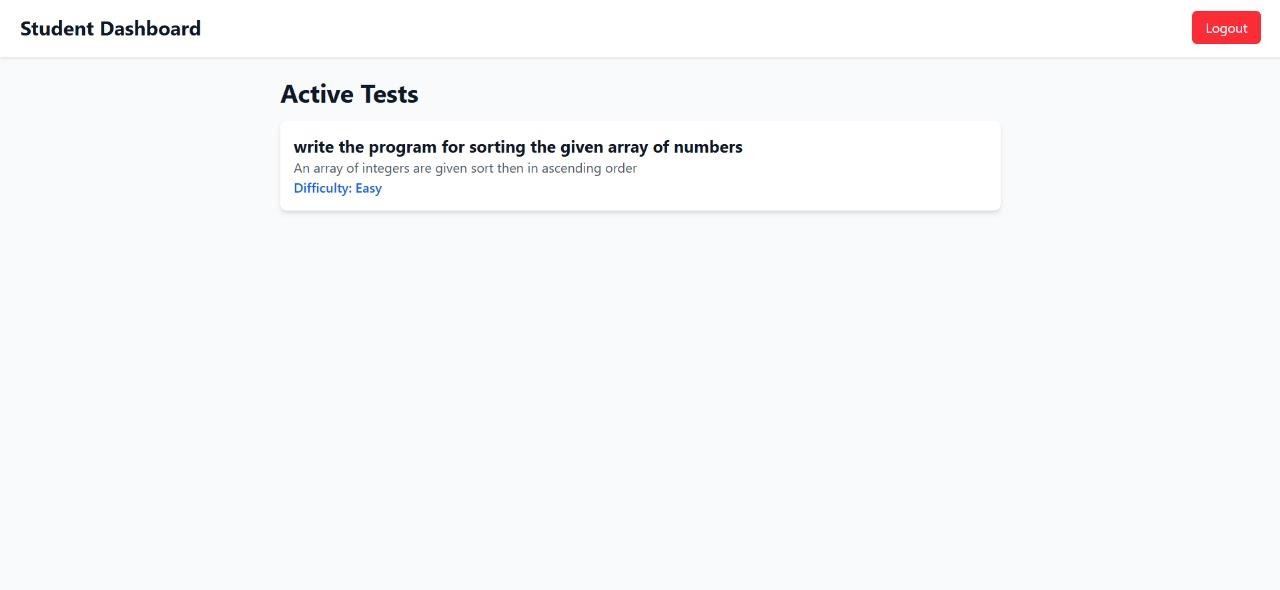


Fig. 8.2.2: Sign Up Page

#### 8.3 Login Interface and Assessment Evaluation



**Fig. 8.3.1 : Dashboard**

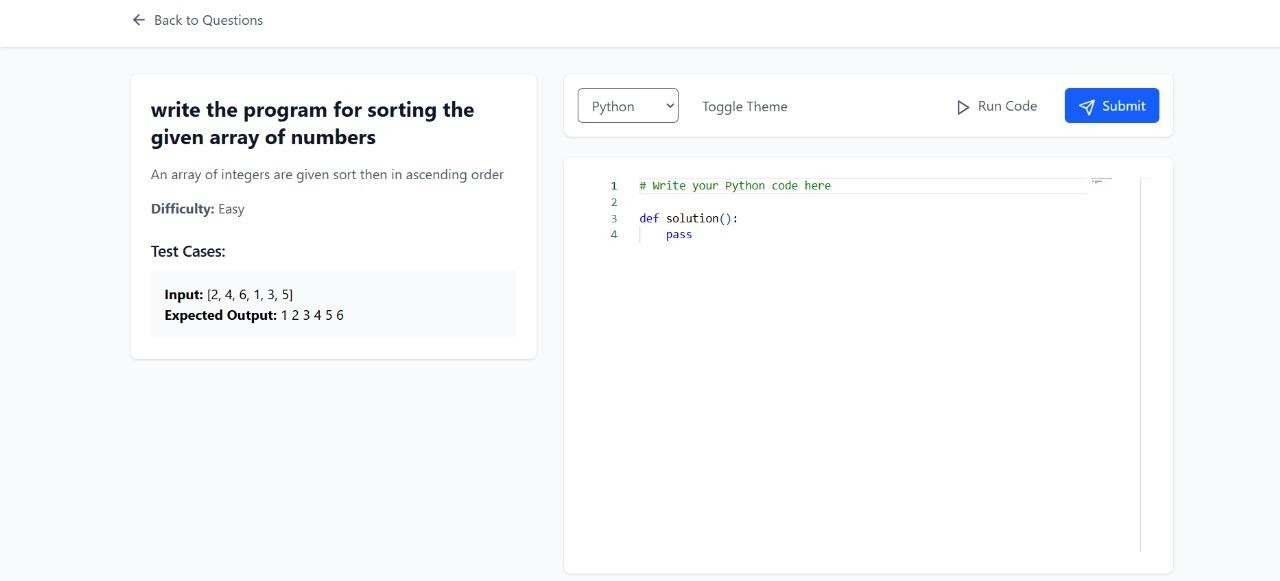


Fig. 8.3.2 :Test Interface

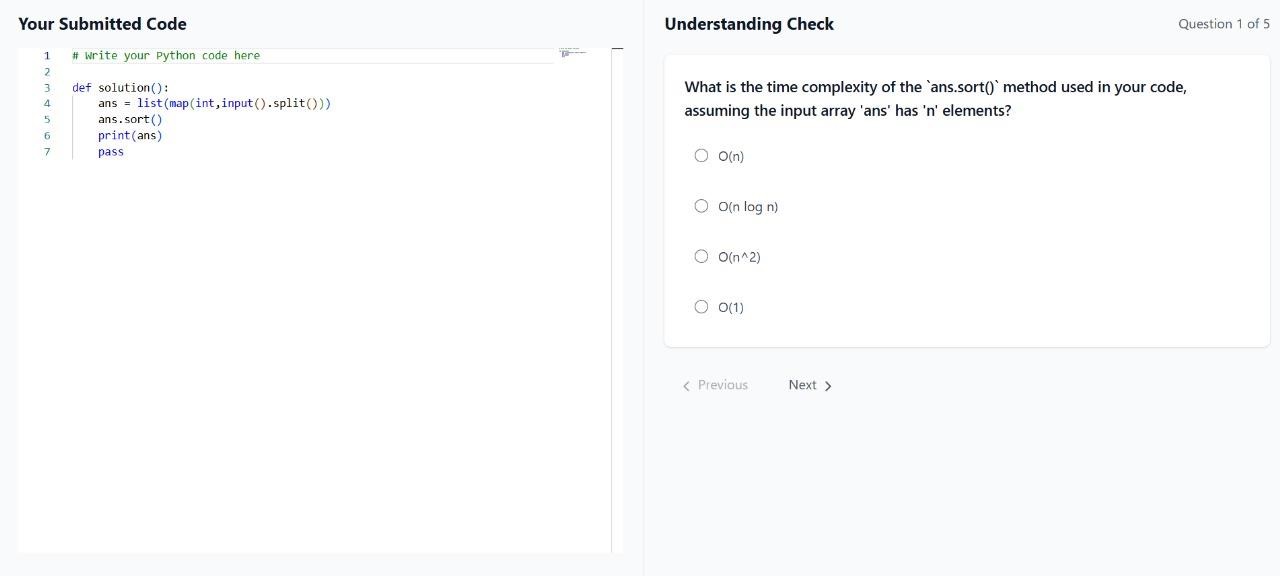


Fig. 8.3.3 :Question Generation

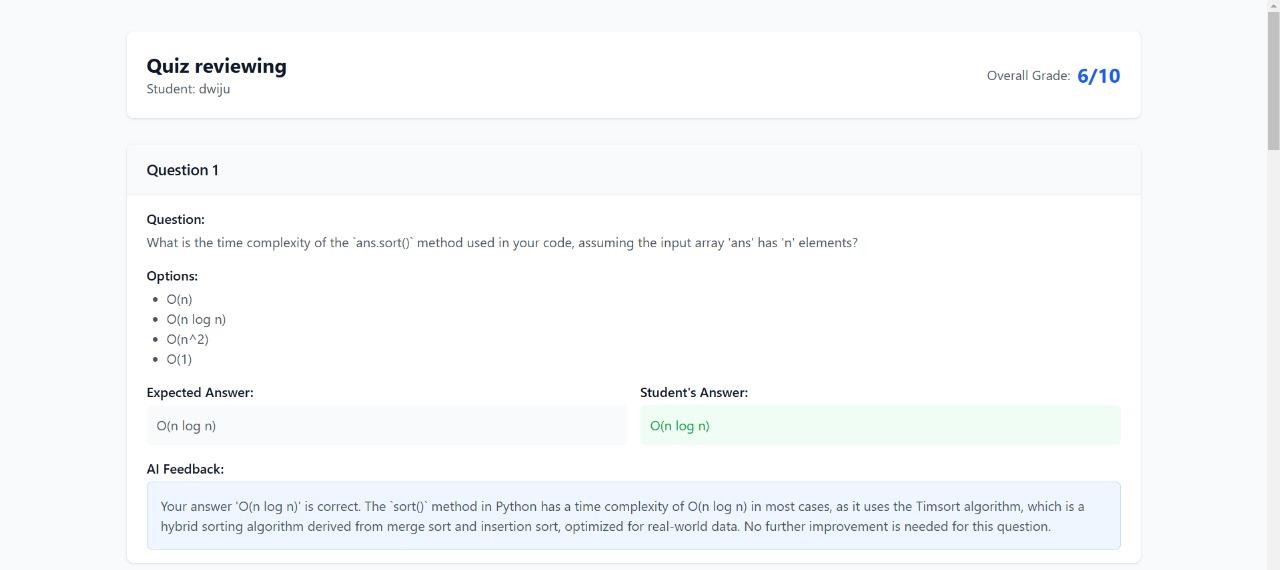


Fig. 8.3.4 : Assessment Evaluation

#### 8.4. Teacher Login

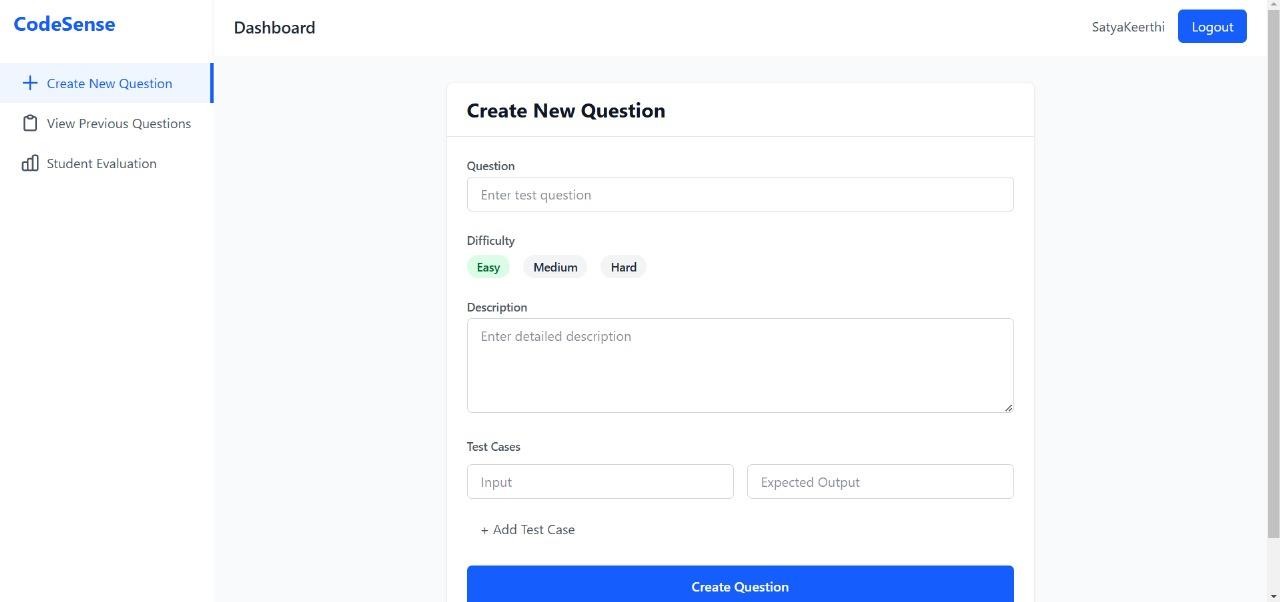


Fig.8.4.1. Teacher Login Dashboard

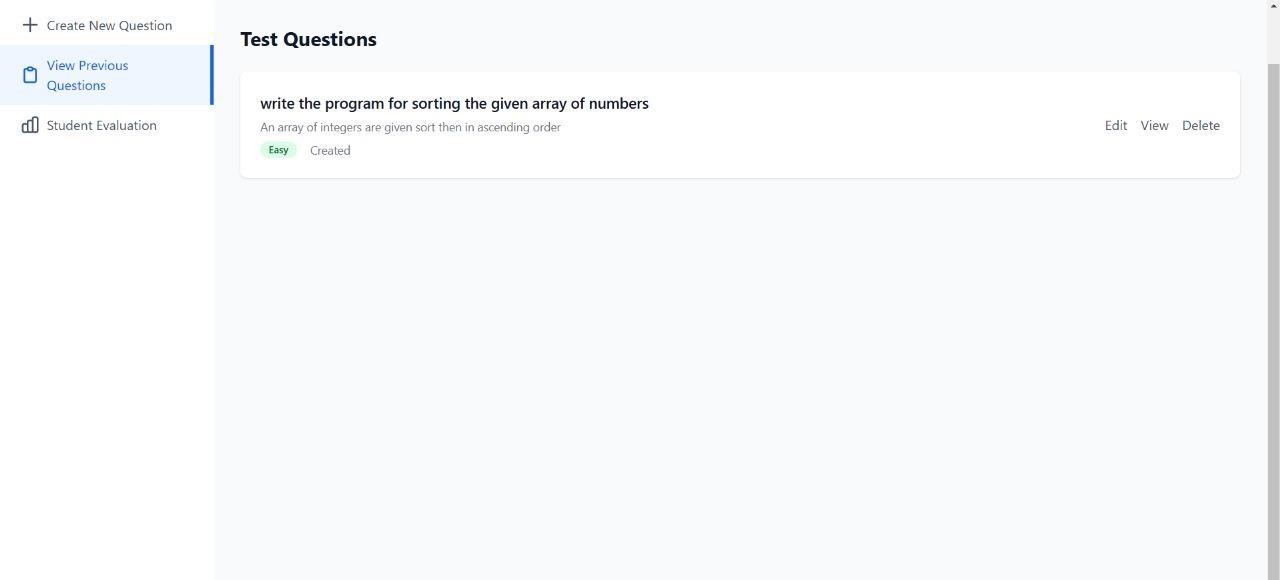


Fig.8.4.2. Previous Questions Generated

#### 8.5. Student Evaluation

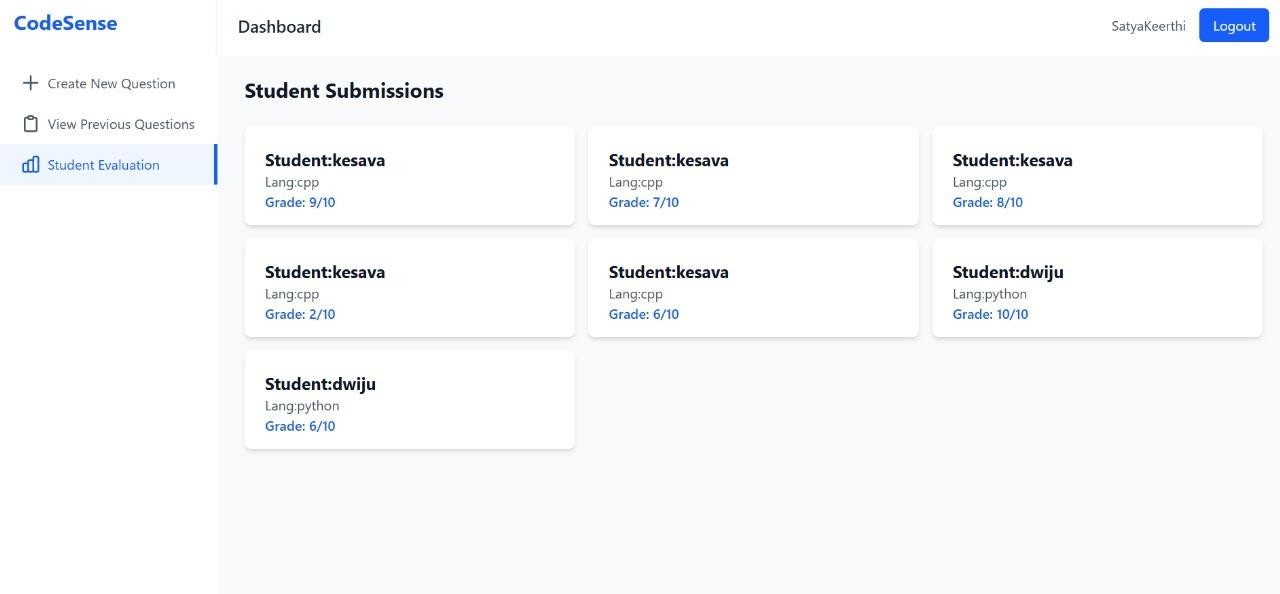


Fig.8.5. Previous Questions Generated

# 

# CHAPTER 9. TESTING

**9.1. Testing Phases**:

* **Unit Testing**: Test individual components like login, question submission, and code execution to ensure they work correctly.
* **Integration Testing**: Check how different modules (e.g., authentication, database, APIs) interact with each other to ensure seamless functionality.
* **System Testing**: Test the entire platform as a whole to confirm all features work together as expected.
* **User Acceptance Testing (UAT)**: Let real users (students and teachers) test the platform to ensure it meets their needs and expectations.

**9.2. Test Cases**: o Test student and teacher login with valid and invalid credentials to ensure secure access.

* Verify that students can submit questions, execute code, and receive feedback, while teachers can post lab questions, view results, and assign marks.
* Check if the AI generates relevant questions and evaluates answers accurately. o Ensure the database stores and retrieves user data, questions, and marks correctly. o Test API integrations for code execution and question generation to confirm they work as intended.
* Validate security by testing unauthorized access attempts and ensuring data protection.

**9.3. Testing Tools**:

* Use tools for unit testing, **Postman** for API testing, **JMeter** for performance testing, and **OWASP ZAP** for security testing.

**9.4. Testing Outputs**:

* Confirm that all components work individually (unit testing) and together

(integration testing).

* Ensure the platform handles high user loads without crashing and responds quickly to user requests.
* Validate that the platform meets user requirements and provides a smooth, errorfree experience (UAT).

# CHAPTER 10. CONCLUSION

In conclusion, our project addresses critical gaps in traditional lab evaluations by combining practical coding assessments with AI-driven conceptual understanding. The platform ensures a **holistic evaluation** by integrating coding tasks, dynamic AI-generated questions, and theoretical testing, enabling students to demonstrate both their practical skills and deeper knowledge. This approach moves beyond static assessments, providing a more comprehensive measure of student understanding and application of concepts.

The platform also emphasizes **authenticity** through features like plagiarism detection and tailored questions, ensuring that students' submissions reflect genuine effort and comprehension. By generating context-specific questions based on individual code submissions, the tool prevents rote memorization and encourages critical thinking. This not only enhances the integrity of assessments but also promotes a culture of originality and deep learning.

Furthermore, the system prioritizes **efficiency** by automating the evaluation process, significantly reducing the workload for faculty. Educators can focus on guiding students rather than spending excessive time on grading, while the AI handles question generation, response evaluation, and feedback delivery. This streamlined process ensures accuracy and consistency in assessments, benefiting both students and educators.

Finally, the project introduces **innovation** by transforming traditional lab assessments into adaptive and engaging experiences. The AI-powered question generator creates dynamic, context-aware questions that challenge students to think critically and apply their knowledge in new ways. This forward-thinking approach not only modernizes the evaluation process but also prepares students for real-world problem-solving in computing and beyond. Together, these features make our platform a transformative tool for computer science education.

# 

# CHAPTER 11. FUTURE SCOPE

While the current system demonstrates **promising results**, several enhancements can be made to:

1. **Expansion to Multiple Programming Languages**:
   * Extend the platform’s capabilities to support a wider range of programming languages, making it versatile for diverse computing courses and curricula.
2. **Integration with Learning Management Systems (LMS)**:
   * Seamlessly integrate the tool with popular LMS platforms like Moodle, Canvas, or Blackboard to streamline adoption and usage in educational institutions.
3. **Advanced Plagiarism Detection**:
   * Enhance the plagiarism detection system by incorporating AI models that can identify more sophisticated forms of code duplication and ensure academic integrity.
4. **Multimedia Question Generation**:
   * Incorporate multimedia elements (e.g., diagrams, videos, or interactive simulations) into AI-generated questions to make assessments more engaging and comprehensive.

**5 . Real-Time Collaboration Features**:

o Introduce real-time collaboration tools that allow students to work on coding problems together while being assessed individually, fostering teamwork and peer learning.

# CHAPTER 12. USER MANUAL

#### 12.1. Introduction

CodeSense is a platform that simplifies coding assignments and assessments for students and teachers. Students can submit code, receive AI-generated feedback, and take quizzes to test their understanding. Teachers can create assignments, evaluate submissions, and provide feedback. The platform is designed to enhance the learning and teaching experience through real-time updates and AI integration.

#### 12.2. Features

**User Authentication:** Sign up and log in as a student or teacher.

**Code Submission:** Submit code for assignments and receive feedback.

**Quiz Generation:** Automatically generate quizzes based on submitted code to test understanding.

**AI Feedback:** Receive AI-generated feedback on code submissions.

**Dashboard:** Separate dashboards for teachers and students to manage assignments and submissions.

**Real-time Updates:** Get real-time updates on submission status and feedback.

#### 12.3. Installation Guide

Follow these steps to set up CodeSense on your local machine:

**12.3.1. Clone the Repository:**  ```bash

git clone https://github.com/yourusername/CodeSense.git cd CodeSense

```

**12.3.2.** **Install Dependencies:**

* For the backend:

```bash cd backend npm install

```

* For the frontend: ```bash

cd ../frontend npm install

```

**12.3.3. Set Up Environment Variables:**

- Create a `.env` file in the `backend` folder and add the following:

```

MONGO\_URI=your\_mongodb\_connection\_string

JWT\_SECRET=your\_jwt\_secret

GOOGLE\_API\_KEY=your\_google\_api\_key

```

**12.3.4. Run the Backend Server:**

```bash cd backend

npm start

```

**12.3.5. Run the Frontend Server:**

```bash cd ../frontend npm run dev

```

**12.3.6. Access the Application:**

- Open your browser and go to `http://localhost:5173` to access CodeSense.

#### 12.4. User Guide

**Student Guide:**

**Sign Up/Log In:**

* Create an account or log in using your credentials.
* Select the "Student" role during signup.

**Dashboard:**

* View assigned tasks, quizzes, and feedback.
* Check real-time updates on your submissions.

**Submit Code:**

* Navigate to the assignment section.
* Write or paste your code in the Monaco Editor and submit it.

**Take Quizzes:**

* After submitting code, take the automatically generated quiz to test your understanding.

**View Feedback:**

* Receive AI-generated feedback on your code submissions.
* Review feedback and improve your code.

**Teacher Guide**

**Sign Up/Log In:**

* Create an account or log in using your credentials.
* Select the "Teacher" role during signup.

**Dashboard:**

* View all assignments, student submissions, and results.
* Manage assignments and provide feedback.

**Create Assignments:**

* Navigate to the assignments section.
* Create new assignments with instructions and deadlines.

**Evaluate Submissions:**

* Review student submissions and provide additional feedback if needed.
* Assign marks based on performance.

**Monitor Progress:**

* Track student progress through real-time updates and analytics.

**CHAPTER 13. REFERENCES**

1. **S. Liu, J. Smith, and A. Johnson**, “Generating Multiple Choice Questions for Computing Courses using Large Language Models,” *ResearchGate*, 2023. [Online]. Available:

[https://www.researchgate.net/publication/373894953\_Generating\_Multiple\_Choice\_Questions\_for\_ Computing\_Courses\_using\_Large\_Language\_Models](https://www.researchgate.net/publication/373894953_Generating_Multiple_Choice_Questions_for_Computing_Courses_using_Large_Language_Models)

1. **M. Brown and L. Taylor**, “Large Language Model-Based Artificial Intelligence in the Language Classroom: Practical Ideas For Teaching,” *ResearchGate*, 2023. [Online]. Available:

[https://www.researchgate.net/publication/368412009\_Large\_Language\_Model-](https://www.researchgate.net/publication/368412009_Large_Language_Model-Based_Artificial_Intelligence_in_the_Language_Classroom_Practical_Ideas_For_Teaching)

[Based\_Artificial\_Intelligence\_in\_the\_Language\_Classroom\_Practical\_Ideas\_For\_Teaching](https://www.researchgate.net/publication/368412009_Large_Language_Model-Based_Artificial_Intelligence_in_the_Language_Classroom_Practical_Ideas_For_Teaching)

1. **P. Wang, X. Zhang, and Y. Chen**, “AI-Based Quiz System for Personalised Learning,”

*ResearchGate*,2023.[Online].Available: [https://www.researchgate.net/publication/376099296\_AI-](https://www.researchgate.net/publication/376099296_AI-BASED_QUIZ_SYSTEM_FOR_PERSONALISED_LEARNING)

[BASED\_QUIZ\_SYSTEM\_FOR\_PERSONALISED\_LEARNING](https://www.researchgate.net/publication/376099296_AI-BASED_QUIZ_SYSTEM_FOR_PERSONALISED_LEARNING)

1. **IEEE Conference Paper**, “AI-Based Quiz System for Personalised Learning,” *IEEE Xplore*,

2023. [Online]. Available: <https://ieeexplore.ieee.org/document/10645812>

1. **React Documentation**: <https://react.dev/>

1. **Vite (React Build Tool)**: <https://vitejs.dev/>

1. **Tailwind CSS**: <https://tailwindcss.com/docs>

1. **Node.js Documentation**: <https://nodejs.org/en/docs>

1. **Express.js Documentation**: <https://expressjs.com/en/guide/routing.html>

1. **CORS Middleware**: <https://www.npmjs.com/package/cors>

1. **JSON Web Token (JWT) Authentication**: <https://www.npmjs.com/package/jsonwebtoken>

1. **MongoDB Official Docs**: <https://www.mongodb.com/docs/manual/>

1. **Mongoose (ODM for MongoDB in Node.js)**: <https://mongoosejs.com/docs/>

1. **Google Gemini API Docs**: <https://ai.google.dev/docs>

[15]**@google/generative-**

**ai(NPMPackageforGeminiAPI)**[:https://www.npmjs.com/package/@google/generative-ai](https://www.npmjs.com/package/@google/generative-ai)