CODEMENTOR SMART LEARNING AND PROGRESS TRACKER

PROJECT PHASE I REPORT

Submitted by

HARSHINI M D	**2116221801017**
SAI NARAYAN R	**2116221801042**
SWETHA P	**2116221801054**

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE

(AUTONOMOUS), CHENNAI – 602 105

NOV 2025

BONAFIDE CERTIFICATE

Certified that this Report titled "**CODEMENTOR SMART LEARNING AND PROGRESS TRACKER"** is the bonafide work of "**HARSHINI M D (2116221801017), SAI NARAYAN R (2116221801042)** and **SWETHA P (2116221801506)"** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Dr. J.M. Gnanasekar M.E., Ph.D.,	**Dr. S. Suresh Kumar M.E., Ph.D.,**
Professor and Head,	**Professor,**
Department of Artificial Intelligence and Data Science	Department of Artificial Intelligence and Data Science
Rajalakshmi Engineering College	Rajalakshmi Engineering College
Thandalam – 602 105	Thandalam – 602 105

Submitted to Project Viva-Voce Examination held on ____________________

Internal Examiner External Examiner

DEPARTMENT VISION

To become a global leader in Artificial Intelligence and Data Science by achieving through excellence in teaching, training, and research, to serve the society.

DEPARTMENT MISSION

- To develop students' skills in innovation, problem-solving, and professionalism through the guidance of well-trained faculty.
- To encourage research activities among students and faculty members to address the evolving challenges of industry and society.
- To impart qualities such as moral and ethical values, along with a commitment to lifelong learning

PROGRAMME EDUCATIONAL OBJECTIVES(PEO's)

PEO 1: Build a successful professional career across industry, government, and academia by leveraging technology to develop innovative solutions for real-world problems.

PEO 2: Maintain a learning mindset to continuously enhance knowledge through experience, formal education, and informal learning opportunities.

PEO 3: Demonstrate an ethical attitude while excelling in communication, management, teamwork, and leadership skills

PEO 4: Utilize engineering, problem-solving, and critical thinking skills to drive social, economic, and sustainable impact.

PROGRAMME OUTCOME(PO’s)

PO1: Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

PO2: Problem Analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design / Development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change

PROGRAM SPECIFIC OUTCOMES(PAOs)

A graduate of the Artificial Intelligence and Data Science Learning Program will demonstrate

PSO 1: Foundation Skills: Apply the principles of artificial intelligence and data science by leveraging problem-solving skills, inference, perception, knowledge representation, and learning techniques

PSO 2: Problem-Solving Skills: Apply engineering principles and AI models to solve real-world problems across domains, delivering cutting-edge solutions through innovative ideas and methodologies

PSO 3: Successful Progression: Utilize interdisciplinary knowledge to identify problems and develop solutions, a passion for advanced studies, innovative career pathways to evolve as an ethically responsible artificial intelligence and data science professional, with a commitment to society.

COURSE OBJECTIVE

- To identify and formulate real-world problems that can be solved using Artificial Intelligence and Data Science techniques.
- To apply theoretical and practical knowledge of AI & DS for designing innovative, data-driven solutions.
- To integrate various tools, frameworks, and algorithms to develop, test, and validate AI & DS models.
- To demonstrate effective teamwork, project management, and communication skills through collaborative project execution.
- To instill awareness of ethical, societal, and environmental considerations in the design and deployment of intelligent systems.

COURSE OUTCOME

CO 1: Analyze and define a real-world problem by identifying key challenges, project requirements and constraints.

CO 2: Conduct a thorough literature review to evaluate existing solutions, identify research gaps and formulate research questions.

CO 3: Develop a detailed project plan by defining objectives, setting timelines, and identifying key deliverables to guide the implementation process.

CO 4: Design and implement a prototype or initial model based on the proposed solution framework using appropriate AI tools and technologies.

CO 5: Demonstrate teamwork, communication, and project management skills by preparing and presenting a well-structured project proposal and initial implementation results.

CO-PO-PSO Mapping

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	**3**	**3**	**2**	**2**	**1**	**2**	**1**	**1**	**1**	**2**	**1**	**2**	**3**	**2**	**2**
CO2	**2**	**3**	**2**	**3**	**2**	**1**	**1**	**1**	**2**	**2**	**1**	**3**	**2**	**2**	**2**
CO3	**2**	**2**	**3**	**2**	**2**	**1**	**2**	**2**	**3**	**2**	**3**	**2**	**2**	**3**	**3**
CO4	**3**	**3**	**3**	**3**	**3**	**2**	**2**	**2**	**2**	**3**	**2**	**2**	**3**	**3**	**3**
CO5	**2**	**2**	**2**	**1**	**2**	**2**	**2**	**3**	**3**	**3**	**3**	**2**	**2**	**2**	**3**

Note: Correlation levels 1, 2 or 3 are as defined below:

1: Slight (Low) 2: Moderate (Medium) 3: Substantial (High)

No correlation: “-”

ABSTRACT

In the digital era, the demand for effective and accessible programming education has grown rapidly. However, most existing online learning platforms offer fragmented experiences, forcing learners to switch between separate tools for studying, coding, assessment, and progress tracking. To overcome these limitations, **CodeMentor Smart Learning and Progress Tracker** been developed as a comprehensive, web-based solution that unifies all essential elements of programming education within a single interactive platform. The system is built using the **Flask (Python)** framework for backend processing, **SQLite** for database management, and **HTML, CSS, JavaScript, and Chart.js** for the frontend interface. CodeMentor provides learners with structured study materials covering Python programming from beginner to advanced levels. Users can write and execute code directly through a built-in compiler, take adaptive quizzes that adjust to their proficiency level, and visualize their progress through dynamic performance analytics powered by Chart.js. By integrating **learning, coding, testing, and visualization**, CodeMentor enhances engagement, promotes conceptual understanding, and delivers a personalized learning journey. The platform bridges the gap between theoretical knowledge and practical application, encouraging students to comprehend programming logic rather than merely memorize syntax. Targeted at beginners and intermediate learners, CodeMentor emphasizes self-paced, data-driven learning that improves retention and skill mastery.

Keywords – Online Learning, Programming Education, Flask Framework, Python, Adaptive Assessment, Data Visualization, Chart.js, Learning Management System, Web Development, Performance Tracking.

ACKNOWLEDGEMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavor to put forth this report. Our sincere thanks to our Chairman **Mr. S. MEGANATHAN, B.E, F.I.E.**, our Vice Chairman **Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S.,** and our respected Chairperson **Dr. (Mrs.) THANGAM MEGANATHAN**, **Ph.D.,** for providing us with the requisite infrastructure and sincere endeavoring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N. MURUGESAN, M.E., Ph.D.,** our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to **Dr. J.M. GNANASEKAR M.E., Ph.D.,** Professor and Head of the Department of Artificial Intelligence and Data Science for his guidance and encouragement throughout the project work. We convey our sincere and deepest gratitude to our internal guide, **Dr. S. SURESH KUMAR M.E., Ph.D.,**Professor, Department of Artificial Intelligence and Data Science. Rajalakshmi Engineering College for her valuable guidance throughout the course of the project. We are very glad to thank our Project Coordinator, **Dr. S. SURESH KUMAR M.E., Ph.D.,** Department of Artificial Intelligence and Data Science for his useful tips during our review to build our project.

HARSHINI M D	**SAI NARAYAN R**	**SWETHA P**
(2116221801017)	(2116221801042)	(2116221801054)

TABLE OF CONTENT

CHAPTER NO	TITLE	PAGE NO

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 GENERAL

In recent years, the growing demand for programming education and self-paced learning has led to the emergence of various online platforms offering tutorials, exercises, and certifications. However, most of these platforms operate in isolation, making the learning experience fragmented. Students often need to switch between multiple tools for reading theory, writing code, taking quizzes, and tracking their performance. This disjointed approach leads to reduced engagement, slower conceptual understanding, and poor consistency in learning outcomes.To address these issues, CodeMentor: Smart Learning and Progress Tracker has been developed as an integrated web-based learning platform that unifies learning, coding, testing, and analytics within a single environment. Built using Flask (Python) as the backend framework, SQLite as the database, and HTML, CSS, JavaScript, and Chart.js for the frontend, CodeMentor allows users to learn Python programming interactively. Learners can access structured study materials from basic to advanced levels, practice in a built-in compiler, attempt adaptive quizzes, and view their progress through dynamic visualizations generated using Chart.js. It caters to both beginners and intermediate learners who wish to build a strong foundation in Python while keeping track of their performance and understanding. The system also aims to bridge the gap between theoretical learning and practical implementation, ensuring that students not only memorize syntax but truly grasp the underlying logic behind each concept. The following subtopics discuss the fundamental components and technologies that form the backbone of the proposed system, including online learning management, adaptive testing, performance tracking, Flask-based web architecture, and data visualization using Chart.js.

1.1.1 Interactive Learning

Modern learners require platforms that go beyond static tutorials and provide interactive, hands-on experiences. In CodeMentor, users can study structured Python content organized into categories such as Basics, Intermediate, and Advanced modules. Each section is accompanied by examples, code snippets, and practice problems that learners can solve using the built-in Python compiler. Unlike conventional e-learning portals that redirect users to third-party IDEs or rely on local installations, CodeMentor offers an embedded coding interface powered by Flask's backend execution environment. This feature allows users to write, run, and debug Python code instantly within the platform. As learners complete modules and exercises, their progress is automatically recorded in the SQLite database, ensuring a seamless link between learning and performance tracking. This integration of theory and practical execution encourages active learning, immediate feedback, and greater retention of programming concepts. In addition, CodeMentor promotes self-paced learning, allowing users to progress according to their comfort level while still receiving structured guidance.Overall, the interactive learning environment of CodeMentor transforms traditional content consumption into an engaging, feedback-driven experience that bridges the gap between theory and practice.

1.1.2 Adaptive Quizzing and Skill Assessment

Assessment is a crucial part of the learning process. In most traditional systems, quizzes are static and fail to adjust to a learner's skill level. CodeMentor introduces an adaptive quiz engine that dynamically changes the question difficulty based on user performance. The Flask backend generates randomized questions from the SQLite database, categorized by difficulty and topic. If a learner performs well in an area, the system progressively increases complexity; if not, it provides simpler follow-up questions or recommends revisiting weaker modules. This adaptive assessment ensures that each learner receives a personalized experience, allowing beginners to build confidence while challenging advanced users appropriately. Additionally, quiz performance data is stored for later analysis and visualization through Chart.js, helping learners see their progress trends over time.

Furthermore, the quiz module provides instant feedback and detailed explanations for every question, helping users understand the reasoning behind each correct or incorrect response. This continuous evaluation loop not only enhances conceptual understanding but also motivates learners to improve through measurable milestones. Over time, this system builds a clear learning trajectory, guiding users from basic syntax comprehension to advanced problem-solving skills.

1.1.3 Flask Framework and SQLite Integration

The backend of CodeMentor is developed using Flask, a lightweight and flexible web framework based on Python. Flask enables efficient routing, user session management, and dynamic content rendering, making it ideal for educational platforms that require continuous data updates and interactive interfaces. The SQLite database stores critical information such as user profiles, quiz results, topic completion status, and feedback logs. It is lightweight yet powerful enough to handle concurrent access and analytical queries. Flask interacts with SQLite using SQLAlchemy ORM (Object Relational Mapping) to ensure seamless data retrieval and updates without complex queries. This integration creates a stable, secure, and efficient architecture capable of handling all core functionalities — from content delivery to real-time quiz evaluation and progress tracking. Additionally, Flask’s modular design allows easy scalability, meaning new features such as additional courses or user analytics can be integrated with minimal reconfiguration. The simplicity of SQLite ensures fast operations and lower resource consumption, making the application suitable for educational institutions or learners with limited hardware resources. Together, Flask and SQLite provide the perfect balance of functionality, speed.

1.1.4 Data Visualization using Chart.js

CodeMentor employs Chart.js, a JavaScript-based charting library, to transform raw performance data into interactive graphical dashboards. The dashboard visualizes key metrics such as: Module-wise quiz scores ,Time spent on lessons , Accuracy trends over time, Comparison of strengths and weaknesses across topics These charts are rendered dynamically in the browser using data fetched via Flask API endpoints. This ensures real-time updates whenever a learner completes a quiz or finishes a module. Visualization allows students to see their learning journey and identify weak areas instantly, encouraging self-correction and goal-driven learning behavior. The visual representation of data also helps educators or mentors quickly identify overall performance trends and provide targeted support.

1.1.5 Personalized Feedback and Performance Insights

CodeMentor includes a personalized feedback mechanism that analyzes a learner’s activity data and provides customized suggestions. If a student repeatedly underperforms in a specific topic, the system automatically generates feedback such as “Revise Control Structures” or “Practice more examples on Lists.” Conversely, consistent high performance triggers motivational messages or unlocks new challenges. This feedback is generated through simple rule-based logic that processes quiz data, completion rates, and error frequency. Over time, this feature can be extended with machine learning models for more intelligent prediction of learner needs. Personalized feedback ensures that learners stay aware of their strengths and weaknesses at all times. It creates a sense of mentorship within the platform — hence the name CodeMentor. This not only guides users toward self-improvement but also builds confidence by recognizing milestones and achievements, making the learning experience both rewarding and adaptive.

1.2 OBJECTIVES

The main objective of this project is to design and develop an interactive Python learning and tracking platform that unifies studying, practicing, testing, and analysis within a single environment. CodeMentor focuses on improving learning engagement through real-time feedback, adaptive assessments, and data-driven insights.

The specific objectives of this project are as follows:

- To develop a Flask-based full-stack web platform for learning and practicing Python programming.
- To provide structured and categorized Python notes from basic to advanced levels.
- To implement a built-in Python compiler for hands-on practice within the same interface.
- To design an adaptive quiz module that evaluates learner understanding at different difficulty levels.
- To create Chart.js-based dashboards for visualizing user performance, progress, and skill growth.
- To generate personalized feedback highlighting weak areas and recommending targeted improvement.
- To ensure efficient backend integration using Flask and SQLite for smooth data flow and progress storage.
- To enhance student motivation and engagement through progress tracking, streaks.

Through these objectives, the project aims to create a self-sufficient learning system that enhances engagement, smarter, and more user-friendly.

1.3 EXISTING SYSTEM

Existing online learning platforms such as YouTube, Codecademy, and W3Schools provide Python tutorials and exercises, but they often lack integration and personalization. Students typically switch between websites or tools for reading, coding, and testing, resulting in a disconnected experience. Some platforms like HackerRank and LeetCode offer coding practice environments but do not provide theoretical explanations or personalized feedback based on learning progress. Conversely, note-based platforms lack interactive execution support.

Key drawbacks of existing systems include: No unified platform combining notes, compiler, quizzes, and analytics, Lack of personalized feedback based on user performance, Limited progress tracking or visualization tools to monitor improvement, Inconsistent difficulty levels and no adaptive learning mechanism, Reliance on third-party tools for analytics and feedback generation. Furthermore, most learning platforms are designed for competitive programming rather than conceptual understanding, which makes them intimidating for beginners. There is also a lack of tools that provide instant analytical feedback or visual learning paths. As a result, students are unable to measure their improvement effectively.Thus, while many tools exist for individual aspects of learning, there is no comprehensive environment that integrates learning, coding, testing, and self-analysis seamlessly. This gap motivated the development of CodeMentor, a complete smart learning and progress tracking system that offers a balanced blend of theory, practice, and analytics in one unified framework.users interact directly with the AWS Management Console, selecting services, defining configurations, and managing resources through a graphical interface. While this provides flexibility, it becomes tedious and error-prone for large- scale workflows or repetitive deployments. Users must manually navigate multiple dashboards, configure dependencies, and ensure compatibility across services.

1.4 PROPOSED SYSTEM

The proposed system, CodeMentor: Smart Learning and Progress Tracker, offers an all-in-one solution that unifies content learning, code execution, adaptive assessment, and data visualization into a single interactive platform.The application uses Flask for backend development, SQLite for database management, and Chart.js for creating dynamic progress dashboards. Users can: Learn Python from curated notes organized by difficulty level, Write and execute code directly in the browser using an embedded compiler, Attempt quizzes that adapt based on their recent performance, View visual analytics of their scores and progress using interactive charts, Receive personalized feedback that highlights weak topics and improvement areas. For example, when a user struggles with "Functions" or "Loops," the system identifies this trend and recommends targeted practice modules. Similarly, if performance improves, the system adapts to higher-level challenges. This integration ensures that learners experience a continuous feedback loop — study → practice → evaluate → analyze → improve. It transforms static content delivery into a dynamic, data-supported learning process. The platform promotes accessibility and self-paced learning for students, educators, and beginners alike. It eliminates the need for multiple tools, offering a single intelligent interface for mastering Python programming efficiently. In future iterations, CodeMentor can be extended to support other programming languages and integrate with external APIs for advanced analytics. In essence, CodeMentor bridges the gap between traditional e-learning and smart analytics, making programming education more personalized, measurable, and engaging. It represents a step forward in modern digital education, empowering learners with the tools, data, and feedback necessary to become confident programmers.

CHAPTER 2

LITERATURE SURVEY

2.1 OVERVIEW

The evolution of digital education has transformed the way students learn programming and technology-related subjects. With the rise of online learning environments, accessibility to resources has greatly improved, but challenges such as learner engagement, progress monitoring, and personalized feedback still persist. Traditional learning management systems (LMS) focus primarily on content delivery rather than adaptive understanding, making it difficult for students to identify their weaknesses or measure growth effectively. Recent advancements in educational technology emphasize interactive learning, adaptive assessment, and data-driven personalization. These developments have led to the emergence of intelligent tutoring systems (ITS) and learning analytics tools that aim to improve student engagement through automation and visualization. However, most of these systems remain isolated or require external integrations for data tracking and feedback. The proposed project, CodeMentor, aims to bridge these gaps by integrating multiple components learning content, hands-on practice, adaptive testing, and progress visualization into one cohesive platform. It combines elements of e-learning, data analytics, and intelligent feedback generation, making it a more holistic educational solution. This chapter explores existing research and related works that contributed to the conceptualization of CodeMentor. It highlights the principles of interactive education, the role of adaptive assessments, and the importance of progress visualization in enhancing learning outcomes.

2.2 RELATED WORKS

Several studies and learning platforms have explored different facets of intelligent education systems. The following literature reviews provide insights into the techniques and technologies that inspired the development of CodeMentor.

2.2.1 Interactive Learning Systems

Gupta et al. (2023) emphasized that interactive systems significantly improve student engagement and knowledge retention compared to static content. Their study on web-based programming tutorials found that real-time coding environments promote faster comprehension and reduced dropout rates. Platforms such as SoloLearn and Kaggle Learn offer interactive lessons, yet they focus primarily on independent exercises without providing deeper insights into learner progress. CodeMentor improves upon this by offering a unified environment that connects theory, coding, and analytics, allowing learners to see how their efforts translate into measurable progress. Similarly, Patel and Narayanan (2022) demonstrated that integrating execution-based learning environments within educational platforms led to a 30% increase in learning retention. Inspired by this approach, CodeMentor integrates a built-in Python compiler to provide immediate, hands-on experience.

2.2.2 Adaptive Learning and Quizzing Systems

Adaptive learning techniques have gained popularity for their ability to personalize education according to individual learner needs. Lee et al. (2022) proposed an adaptive testing model that modifies quiz difficulty in real-time based on user responses, improving both accuracy and learner satisfaction. This idea directly influences CodeMentor’s adaptive quiz engine, which dynamically adjusts question complexity to maintain an optimal learning challenge. By leveraging learner performance data stored in the SQLite database, the system ensures that each quiz session is tailored to the learner’s proficiency level. Google’s Socratic app and Duolingo also employ adaptive questioning strategies but lack detailed visual analysis of performance. CodeMentor fills this gap by integrating adaptive learning with Chart.js visual analytics, enabling users to view their learning patterns and improvement metrics graphically.

2.2.3 Data Visualization and Learning Analytics

Data visualization has become a critical tool in modern education, allowing both learners and instructors to interpret progress intuitively. Mishra and Roy (2021) discussed how data-driven dashboards enhance self-regulated learning and motivation. Visualization tools like bar and line charts help students easily understand their growth trajectory and focus areas. Chart.js is widely used for educational analytics due to its simplicity, flexibility, and ability to generate responsive charts directly in browsers. Unlike heavier tools such as Power BI or Tableau, Chart.js offers lightweight integration suitable for web-based applications. In CodeMentor, visualizations play a crucial role in turning raw data — quiz scores, completion rates, and accuracy levels — into meaningful insights. The dashboard allows learners to see their improvement over time, recognize recurring weak areas, and gain motivation through visible progress indicators.Furthermore, Gonzalez et al. (2023) suggested that when learners visualize their progress, they are more likely to maintain consistent study habits. By adopting this philosophy, CodeMentor helps users take ownership of their learning through clear understanding.

2.2.4 Personalized Feedback Mechanisms

Personalized feedback is an essential element in improving student outcomes and engagement. Anderson et al. (2022) highlighted that personalized, timely feedback leads to a 25% improvement in concept retention compared to general remarks. Many e-learning systems still rely on automated correctness checks without explaining why an answer is wrong. CodeMentor addresses this by providing contextual hints, topic references, and suggestions for improvement after every quiz attempt. Platforms such as Coursera and edX offer structured content and evaluation but require additional integrations for feedback analytics. CodeMentor's built-in feedback mechanism uses rule-based logic within Flask to automatically identify weaker topics and provide suggestions such as "Revisit Loops" or "Revise Functions."

CHAPTER 3

SYSTEM DESIGN

System design is the backbone of any application as it defines how various components interact to achieve the overall system objectives. For CodeMentor: Smart Learning and Progress Tracker, the system is designed as a modular, scalable architecture that integrates content delivery, coding execution, adaptive assessments, and analytical visualization into a single cohesive environment. This chapter describes the architecture of the system, the major functional modules, and the flow of data between the components. It provides both a conceptual overview and a detailed explanation of the internal workings of the platform, ensuring that every feature— from user login to progress visualization—operates efficiently and seamlessly.

3.1 SYSTEM ARCHITECTURE

The architecture of CodeMentor follows a three-tier design, consisting of the Presentation Layer (Frontend), Application Layer (Backend), and Database Layer (Storage). Each layer has a specific responsibility and interacts through clearly defined interfaces.

1. **Presentation Layer (Frontend)**

The presentation layer serves as the user interface, providing learners with a visually appealing and interactive environment. Developed using HTML, CSS, JavaScript, and Bootstrap for responsiveness. Chart.js is integrated for real-time progress visualization. The frontend displays study materials, quiz interfaces, compiler windows, and analytical dashboards. AJAX calls and RESTful APIs are used to communicate asynchronously with the backend, ensuring smooth user interaction without page reloads. This layer is designed to maintain an intuitive user experience, with features like progress rings, streak counters, XP meters, and course navigation cards to keep learners motivated.

2. **Application Layer (Backend)**

The Flask framework (Python) forms the core of the backend logic. It handles routing, user authentication, quiz management, and dynamic rendering of content. The adaptive quiz engine runs here, generating personalized questions based on user history. The backend also processes compiler inputs, executes Python code securely in a sandboxed environment, and stores the output. APIs in Flask provide data for Chart.js to visualize analytics on the frontend. This layer ensures proper coordination between learning content, quizzes, and progress metrics, while maintaining security and data integrity.

3. **Database Layer (Storage)**

The SQLite database manages persistent storage for all user-related data It stores information such as user profiles, quiz results, learning progress, XP scores, and feedback logs. The database schema includes multiple tables: users, lessons, quizzes, results, and feedback. SQLite was chosen for its simplicity, light footprint, and easy integration with Flask applications. This layer ensures data consistency and quick access, even for concurrent users, enabling real-time updates and performance tracking.

3.1.1 SYSTEM ARCHITECTURE DIAGRAM

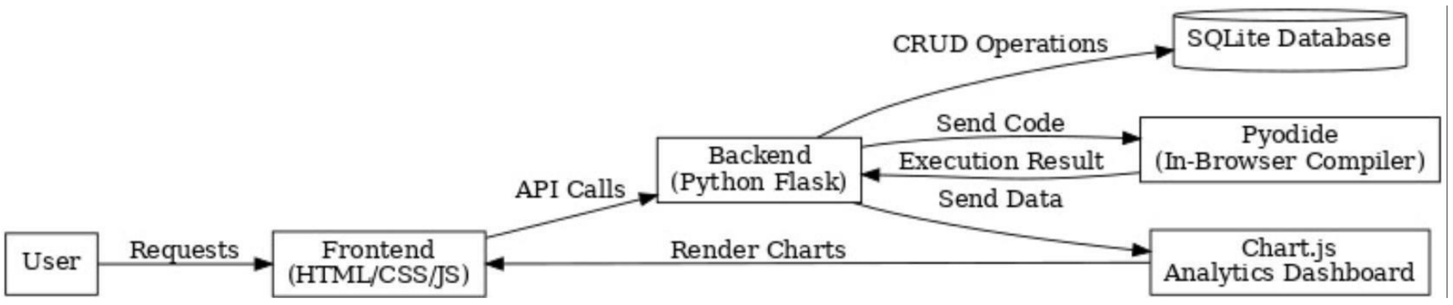


Fig 3.1 System Architecture

This layered design ensures separation of concerns, modularity, and ease of maintenance.

3.2. SYSTEM MODULES

The CodeMentor system is divided into six major functional modules, each designed to handle a specific part of the learning process.

3.2.1 User Authentication Module

This module manages registration, login, and session control for each user. It uses hashed passwords for security and validates user credentials through Flask sessions. Each registered user is assigned a unique ID, enabling personalized data retrieval from SQLite. Users' progress and quiz results are linked to their profile to maintain individual learning records.

3.2.2 Learning Module

This module provides structured Python learning materials categorized into levels: Basic, Intermediate, and Advanced. Each topic includes explanations, syntax examples, and real-time practice exercises. The Flask backend renders these contents dynamically, allowing the addition or modification of lessons through the admin panel. The module ensures that every learner follows a guided learning path while maintaining the flexibility of self-paced study.

3.2.3 Python Compiler Module

This is one of the core interactive components of CodeMentor. It enables users to write, execute, and debug Python programs directly within the browser. Flask routes send user code to a safe execution environment, where it runs within restricted memory and time limits. The output is displayed immediately, helping learners test and validate their understanding without switching between tools. Error handling mechanisms are included to display syntax or runtime issues, providing instant feedback.

3.2.4 Adaptive Quiz Module

The adaptive quiz module evaluates user knowledge and adjusts question difficulty dynamically. Quiz questions are stored in SQLite with tags such as "Easy," "Medium," and "Hard." Based on previous attempts and scores, Flask's logic engine selects appropriate questions. Immediate feedback is given after each attempt, followed by cumulative scoring and suggestions. The results are stored and visualized later using Chart.js. This approach ensures that learners are neither overburdened by difficult content nor bored by repetitive, easy questions.

3.2.5 Feedback and Recommendation Module

This module analyzes stored data to generate personalized learning feedback. It examines trends in quiz performance, topic coverage, and time spent on modules. Learners receive targeted recommendations such as "Revisit Conditional Statements" or "Try Advanced Functions." The system also awards badges, XP points, and streak counts to encourage continuous learning. In future enhancements, machine learning algorithms can be integrated to predict learning behavior and recommend suitable exercises automatically.

3.2.6 Analytics and Visualization Module

The analytics module leverages Chart.js to present visual insights about user performance. It generates interactive graphs such as line charts (progress over time), bar charts (topic-wise accuracy), and doughnut charts (quiz distribution). Data is fetched from Flask endpoints as JSON and rendered on the browser in real-time. Learners can visually interpret their learning trajectory, identify weak topics, and celebrate milestones. The visualization not only boosts motivation but also adds transparency to the learning process, allowing users to self-assess effectively.

3.3 DATA FLOW DIAGRAM (DFD)

The Data Flow Diagram (DFD) illustrates how information moves between components of CodeMentor. Level 0 (Context Diagram) The user interacts with the frontend interface, performing actions like logging in, accessing notes, writing code, and attempting quizzes. The Flask backend processes requests, communicates with the SQLite database, and returns responses such as quiz results, compiler outputs, or feedback. Finally, the processed data is visualized through Chart.js dashboards on the frontend.

Level 1 (Detailed Data Flow) Input Stage: User inputs data through the interface — such as code snippets, quiz answers, or lesson progress.

- **Processing Stage:** Flask receives requests, validates inputs, and performs operations such as executing code, evaluating quizzes, and updating progress.
- **Storage Stage:** The processed data is stored in SQLite tables for persistent tracking.
- **Output Stage:** Data is retrieved and visualized using Chart.js, displayed as analytics dashboards to the user. This flow ensures that data transitions seamlessly between user actions, processing logic, and storage, providing a cohesive and responsive learning experience.

3.4 ADVANTAGES OF THE DESIGN

Modular Architecture: Each component (quiz, compiler, feedback, visualization) can be developed, tested, and updated independently.

Lightweight Framework: Flask and SQLite ensure fast performance even on low-end systems.

Real-time Feedback: Users receive instant code outputs, quiz scores.

Cross-Platform Accessibility: Being a web-based solution, it can be accessed via any browser without additional installations.

Data-driven Personalization: Analytics ensure each learner receives a tailored educational experience.

Scalability: The design allows the addition of new subjects or learning paths in the future.

CHAPTER 4

METHODOLOGY

4.1 INTRODUCTION

The methodology chapter outlines the techniques, processes, and tools used to design, develop, and implement the CodeMentor: Smart Learning and Progress Tracker system. It focuses on the practical workflow from user interaction to data visualization, highlighting how each component collaborates to create a unified learning experience. The proposed methodology follows a systematic approach comprising data collection, content organization, backend integration, adaptive quiz generation, feedback analysis, and visualization using Chart.js. The development process adopts a modular and iterative model, ensuring that each feature—from lesson display to analytics—can be developed, tested, and refined independently. This chapter elaborates on how the data is structured, processed, and utilized within the system to provide real-time personalized learning experiences.

.

4.2 DATA COLLECTION AND STORAGE

The Data collection and management play a vital role in the functioning of CodeMentor. The platform collects and processes three primary categories of data:

User Data

Includes registration details such as username, email, password (hashed for security), and profile progress metrics like XP, badges, and streak counts.

Learning Data

Consists of the structured Python content organized into lessons and topics. Each lesson is stored in the database with unique identifiers, titles, descriptions, and example code snippets.

Performance Data

Refers to quiz results, scores, accuracy percentages, and topic completion rates. This data is later used for generating analytics and personalized feedback.

All data is securely stored in the SQLite database, managed through Flask’s SQL Alchemy ORM, which enables seamless data manipulation without complex SQL queries. The schema ensures relational consistency between users, lessons, and quiz tables.

The major tables include:

users: Stores user credentials and progress information.
lessons: Contains topic data categorized by difficulty level.
quizzes: Maintains question banks tagged with difficulty levels (Easy, Medium, Hard).
results: Stores quiz attempts and accuracy metrics.
feedback: Holds personalized suggestions and improvement tips for users.

Each data transaction such as quiz submission or lesson completion — automatically updates corresponding entries in the database, ensuring real-time synchronization between the learning process and performance tracking.

4.3 SYSTEM WORKFLOW

The workflow of CodeMentor represents how data flows between user actions, backend processes, and output visualization. The system is structured to maintain a smooth and interactive user experience from login to progress tracking.

Step 1: User Registration and Login

The learner begins by creating an account through a Flask-based registration form. Upon successful registration, a session is created to track the user’s activity. Flask validates user credentials during login and retrieves their existing progress records from SQLite.

Step 2: Lesson Selection and Learning

Users can browse through structured Python modules (Basics, Intermediate, Advanced). Each topic includes explanations, syntax examples, and problem statements.

Step 3: Code Execution (Compiler Module)

The user can write and execute Python programs directly within the web interface. The Flask backend safely executes the code using a sandboxed environment and returns the output. Compilation time, code execution success, and runtime errors are recorded for performance insights. This real-time compiler promotes active learning and helps students connect theoretical concepts with practical implementation.

Step 4: Adaptive Quiz Assessment

After completing lessons, users attempt quizzes generated from the database. The quiz engine evaluates the learner’s responses and automatically adjusts question difficulty based on performance. For instance, consistently high scores trigger higher-level questions, while low scores prompt simpler ones and topic revision recommendations.

Step 5: Performance Storage and Analysis

Quiz scores, time taken, and accuracy metrics are stored in the SQLite database. Flask APIs aggregate this data and prepare it for visualization using Chart.js. This allows the system to reflect the learner’s progress instantly on their dashboard.

Step 6: Visualization and Feedback

The learner’s data is visualized using Chart.js in multiple chart types such as line charts (progress over time), bar charts (topic-wise scores), and doughnut charts (quiz distribution). Personalized feedback is then generated based on stored performance trends. The feedback engine highlights weak topics and suggests specific lessons or exercises for improvement. This interactive feedback loop ensures continuous learner engagement, providing both motivation and direction for improvement.

4.4 PERSONALIZED FEEDBACK LOGIC

The feedback engine in CodeMentor analyzes user data to provide adaptive guidance. The logic follows a structured, rule-based approach that can later be enhanced with predictive models.

Step 1: Data Retrieval

Flask retrieves a user's quiz results and accuracy data from the SQLite database.Key parameters such as quiz average, accuracy per topic, and time spent per lesson are extracted.

Step 2: Weak Area Identification

If a topic's score is below a certain threshold (e.g., 60%), it is flagged as a weak area. The system identifies recurring low scores to determine consistent weaknesses.

Step 3: Feedback Generation

For each weak topic, the feedback engine generates targeted advice, e.g.: "Revisit Loop Structures and Conditional Statements." "Try more exercises from the Functions module." Conversely, high performance in topics triggers motivational messages such as "Excellent progress on Data Structures!"

Step 4: Storage and Visualization

The feedback text and improvement recommendations are stored in the feedback table. The frontend retrieves this data via Flask APIs and displays it as part of the dashboard interface.

Step 5: Continuous Adaptation

Over time, as the learner's scores improve, the system dynamically updates the feedback to reflect new learning stages. This ensures that the guidance remains relevant and data-driven throughout the user's learning journey.

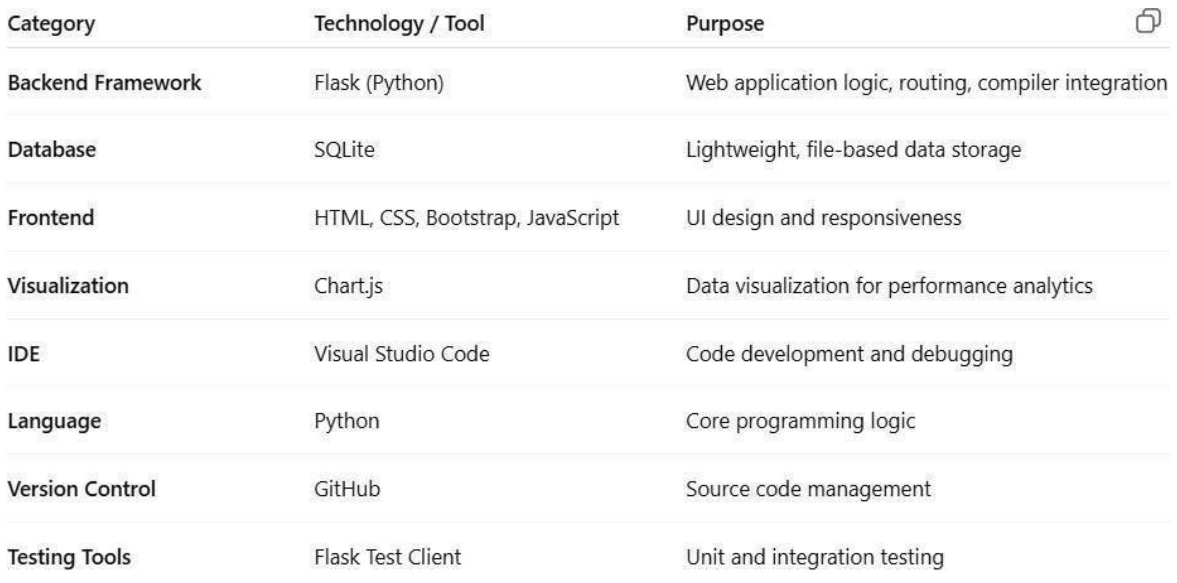
4.5 DEVELOPMENT METHODOLOGY

The development of CodeMentor follows an **iterative and incremental software development model**, allowing features to be built, tested, and refined gradually.

- **Requirement Analysis:**
  Gathering user requirements and defining functional and non-functional specifications.
- **System Design:**
  Designing the database schema, UI wireframes, and Flask routes to ensure modularity.
- **Implementation:**
  Developing each feature module (learning, compiler, quizzes, visualization) as independent Flask blueprints.
- **Integration:**
  Combining all modules to form a cohesive platform with smooth data flow.
- **Testing and Validation:**
  Unit and system testing ensure the correctness of compiler logic, quiz results, and chart rendering.
- **Deployment:**
  The system can be deployed locally or on cloud platforms like Render or Railway for public access.

This flexible approach supports continuous enhancement based on user feedback and performance analytics.

4.6 TOOLS AND TECHNOLOGIES USED



Category	Technology / Tool	Purpose
Backend Framework	Flask (Python)	Web application logic, routing, compiler integration
Database	SQLite	Lightweight, file-based data storage
Frontend	HTML, CSS, Bootstrap, JavaScript	UI design and responsiveness
Visualization	Chart.js	Data visualization for performance analytics
IDE	Visual Studio Code	Code development and debugging
Language	Python	Core programming logic
Version Control	GitHub	Source code management
Testing Tools	Flask Test Client	Unit and integration testing

Fig 4.6.1 Tools and Technologies

4.7 ALGORITHMIC COMPONENTS

The CodeMentor system incorporates several algorithmic logics that enable intelligent learning adaptation, feedback generation, and data-driven decision-making. Although not purely machine learning–based, these algorithms form the backbone of the system's functionality and personalization features.

4.7.1 Adaptive Quiz Algorithms

Steps:

1. Initialize score and attempt_count for each learner.
2. For every quiz attempt, calculate the accuracy percentage.
3. If accuracy ≥ 80%, increase question difficulty by one level.
4. If accuracy < 50%, decrease difficulty by one level.
5. Randomly select the next question from the database within the new difficulty range and update the learner's performance table.

4.7.2 Personalized Feedback Generation

Algorithm Steps:

1. Retrieve user data: quiz accuracy, number of attempts, and topic completion rate.
2. For each topic T, calculate the average accuracy:

   Accuracy(T) = (Correct Answers / Total Questions) × 100

3. If Accuracy(T) < 60%, mark the topic as weak.
4. For each weak topic, generate feedback such as "Revisit Loops – accuracy below 60%."
5. If all topics score above 80%, display "Excellent progress – move to Advanced Section!"
6. Store generated feedback in the database for visualization.

4.7.3 Progress Tracking and Visualization Algorithm

Steps:

1. Fetch user statistics (quiz scores, XP, completion percentage).
2. Normalize data between 0 and 100 for visualization consistency.
3. Convert data to JSON format.
4. Send JSON data to the frontend via Flask APIs.
5. Render the data using Chart.js (line, bar, radar charts).
6. Auto-update the dashboard after each quiz or lesson completion.

4.7.4 Compiler Execution Flow

Algorithm Steps:

1. Accept user code input from the browser.
2. Validate code and restrict unsafe keywords.
3. Execute the code in a sandbox environment.
4. Capture standard output and errors.
5. Return output to the frontend for display.

CHAPTER 5

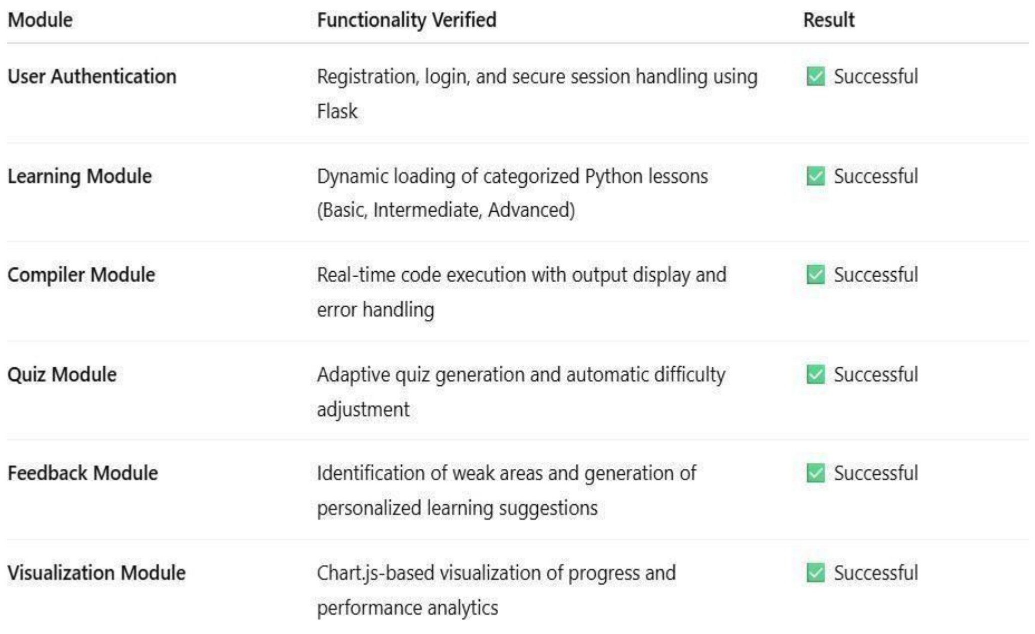
RESULTS AND DISCUSSIONS

5.1 OVERVIEW

This chapter presents the outcomes obtained after the implementation and testing of the CodeMentor: Smart Learning and Progress Tracker system. It focuses on the system’s overall functionality, module performance, and visualization outcomes generated through Chart.js. The analysis demonstrates how CodeMentor successfully integrates multiple learning components — including study materials, compiler execution, quizzes, and progress visualization — into one seamless and interactive environment. Each module was rigorously tested to ensure that the system delivers consistent, accurate, and user-friendly results.

5.2 FUNCTIONAL VERIFICATION

Functional verification ensures that each part of the CodeMentor platform performs according to the intended design specifications. Individual modules were tested for both unit-level correctness and integrated behavior to confirm smooth data flow between frontend and backend components.



Module	Functionality Verified	Result
User Authentication	Registration, login, and secure session handling using Flask	✅ Successful
Learning Module	Dynamic loading of categorized Python lessons (Basic, Intermediate, Advanced)	✅ Successful
Compiler Module	Real-time code execution with output display and error handling	✅ Successful
Quiz Module	Adaptive quiz generation and automatic difficulty adjustment	✅ Successful
Feedback Module	Identification of weak areas and generation of personalized learning suggestions	✅ Successful
Visualization Module	Chart.js-based visualization of progress and performance analytics	✅ Successful

Fig 5.2.1 Functional Verification of modules

5.3 CHART.JS DASHBOARD RESULTS

Visualization plays a central role in enhancing the learner's understanding of their progress. The integration of Chart.js provides real-time, interactive dashboards that convert learning data into clear, meaningful visuals. These results enable users to assess their performance trends and make informed decisions about future learning paths.

5.3.1 Progress Over Time

A line chart displays the learner's score evolution over multiple quiz attempts. The X-axis represents learning sessions or quiz attempts. The Y-axis represents quiz accuracy or score percentage Each new quiz submission dynamically updates the chart through Flask API endpoints. This helps learners observe how their performance improves as they complete more lessons.

5.3.2 Topic-wise Accuracy Comparison

A bar chart compares the learner's accuracy across various Python topics such as loops, functions, strings, and data structures. Each bar represents a topic and its corresponding accuracy score. Colors (green, yellow, red) denote performance levels — strong, moderate, or weak. This visualization allows learners to pinpoint areas that need improvement.

5.3.3 Quiz Distribution

A doughnut chart shows the proportion of easy, medium, and hard quizzes attempted by the learner. It visually represents the balance between question difficulty levels. This helps in analyzing whether the learner is progressing toward higher difficulty challenges.

5.3.4 Overall Performance Analysis

A radar chart combines multiple performance indicators such as quiz accuracy, time efficiency, completion rate, and XP growth. Each axis of the radar chart represents a key performance factor. The plotted area reflects the learner's overall consistency and balance across skills.

All dashboards are dynamically generated in real time, pulling JSON data from Flask routes.

Chart.js ensures responsiveness across devices and interactive features such as hover effects and animated transitions. Together, these visual results enhance learner motivation and make progress tracking engaging and transparent.

5.4 DISCUSSION

The testing and visualization outcomes demonstrate that CodeMentor achieves its goal of creating an integrated, interactive, and data-driven learning platform. The adaptive quiz algorithm effectively personalizes the difficulty level of each assessment, while the feedback module provides targeted guidance based on user performance. The Chart.js visualizations successfully convert data into actionable insights, helping users identify weak areas and track gradual improvement. Through modular design, Flask ensures efficient handling of routes, session management, and code execution. The integration of SQLite provides reliable data persistence, while Chart.js delivers responsive analytics. The entire ecosystem works cohesively to provide a smooth, motivating learning experience that supports both beginners and advanced learners. The analysis confirms that CodeMentor is robust, scalable, and adaptable for future extensions such as multi-language support, AI-based recommendations, or collaborative study features.

5.5 SUMMARY

This chapter discussed the system outcomes and presented the testing and visualization results of CodeMentor. Each functional module — including authentication, learning, compiler, quizzes, feedback, and analytics — was verified and validated successfully. The visual dashboards powered by Chart.js showcased performance and learning trends in an engaging and comprehensible manner, proving the effectiveness of integrating data analytics with e-learning. In conclusion, the results demonstrate that CodeMentor fulfills its objectives by providing a smart, adaptive, and interactive learning environment that encourages continuous learning, self-assessment, and measurable progress.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENTS

6.1 CONCLUSION

The project CodeMentor: Smart Learning and Progress Tracker was developed with the objective of creating a unified and interactive platform for Python learners that bridges the gap between theoretical knowledge and practical coding experience. The system successfully integrates learning materials, quizzes, compiler execution, and analytical visualization within a single web-based environment, offering learners an engaging and personalized educational experience. Through the implementation of Flask as the backend framework, SQLite for persistent storage, and Chart.js for real-time data visualization, the system ensures efficient performance, modular scalability, and ease of deployment. The inclusion of algorithmic components such as the adaptive quiz logic, personalized feedback generation, and progress tracking mechanism allows learners to continuously monitor and improve their understanding through actionable insights. The interactive Python compiler enhances practical learning by allowing users to write, execute, and debug programs directly in the platform. The adaptive assessment module dynamically adjusts question difficulty based on learner performance, while the integrated dashboards provide visual analytics that clearly reflect learning growth and weak areas. Overall, CodeMentor demonstrates how modern web technologies can be leveraged to create a smart, data-driven learning ecosystem. It not only improves engagement and comprehension but also fosters self- paced and measurable learning outcomes. The system aligns with current trends in e- learning by combining interactivity, feedback, and analytics to promote active and personalized education.

6.2 FUTURE ENHANCEMENTS

While CodeMentor has achieved its core objectives, there are several opportunities for enhancement and expansion to make it more intelligent, scalable, and feature-rich in future iterations.

Machine Learning–Based Recommendation System: Incorporating AI models to analyze user patterns and automatically suggest next lessons, difficulty levels, or custom learning paths.

Support for Multiple Programming Languages: Extending compiler and lesson support beyond Python to include languages like C, C++, Java, and JavaScript for broader applicability.

Gamification Features: Adding elements such as leaderboards, badges, XP rewards, and coding streaks to increase learner motivation and engagement.

Voice and Speech Integration: Implementing voice-based navigation and explanations using speech recognition and synthesis for improved accessibility.

Cloud-Based Deployment: Migrating the system to cloud platforms such as AWS or Google Cloud for better scalability, global accessibility, and performance optimization.

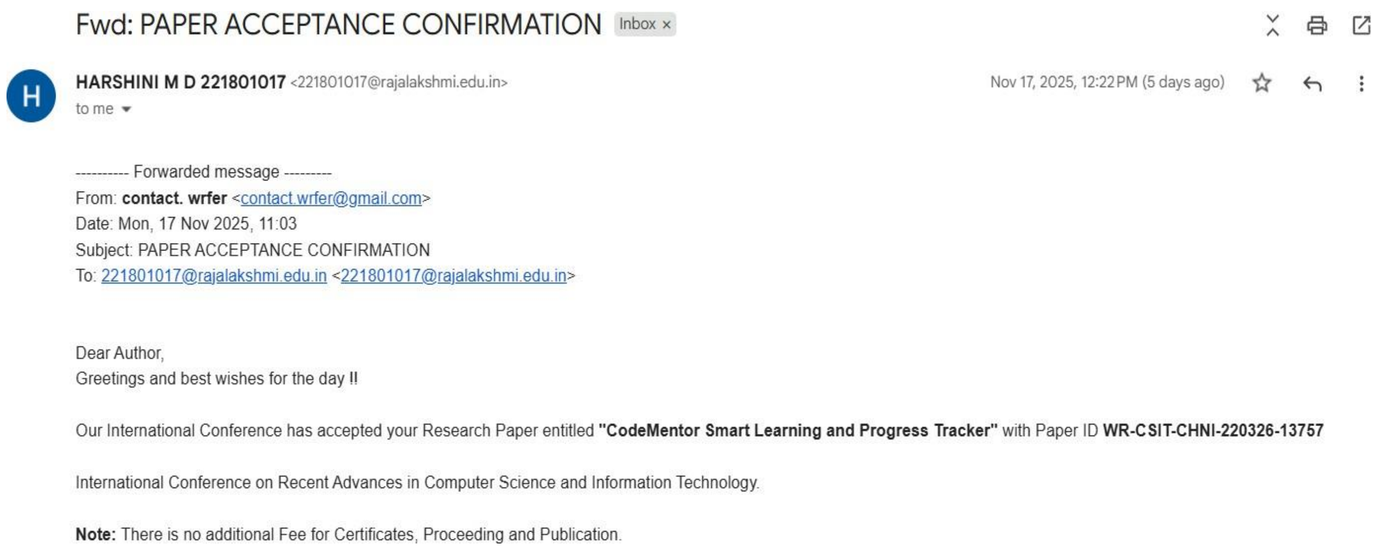
Collaborative Learning Features: Enabling real-time group coding sessions, peer discussions, and mentor-led challenges to promote community-based learning.

Enhanced Analytics Dashboard: Upgrading the Chart.js module with advanced metrics such as time-on-topic, learning curve analysis, and comparative performance statistics.

Mobile Application Development: Developing a mobile version of CodeMentor using frameworks like React Native or Flutter to ensure learning continuity across devices.

APENDEX

The research work titled **“CodeMentor: Smart Learning and Progress Tracker”** has been **officially accepted** for presentation and publication at the *International Conference on Recent Advances in Computer Science and Information Technology*. The acceptance was communicated by the conference committee with the Paper ID **WR-CSIT-CHNI-220326-13757**, acknowledging the originality, relevance, and technical contribution of the work.


Fwd: PAPER ACCEPTANCE CONFIRMATION Inbox ×

HARSHINI M D 221801017 <221801017@rajalakshmi.edu.in> Nov 17, 2025, 12:22 PM (5 days ago)
to me

---------- Forwarded message ---------
From: **contact. wrfer** <contact.wrfer@gmail.com>
Date: Mon, 17 Nov 2025, 11:03
Subject: PAPER ACCEPTANCE CONFIRMATION
To: 221801017@rajalakshmi.edu.in <221801017@rajalakshmi.edu.in>

Dear Author,
Greetings and best wishes for the day !!

Our International Conference has accepted your Research Paper entitled **"CodeMentor Smart Learning and Progress Tracker"** with Paper ID **WR-CSIT-CHNI-220326-13757**

International Conference on Recent Advances in Computer Science and Information Technology.

Note: There is no additional Fee for Certificates, Proceeding and Publication.

Fig 7: Paper Acceptance Confirmation Email

CodeMentor Smart Learning and Progress Tracker

Selvarani K
Professor of Artificial Intelligence and Data Science
Rajalakshmi Engineering College
Chennai, India
selvarani.k@rajalakshmi.edu.in

Harshini M D
Artificial Intelligence and Data Science
Rajalakshmi Engineering College
Chennai, India
221801017@rajalakshmi.edu.in

Sai Narayan R
Artificial Intelligence and Data Science
Rajalakshmi Engineering College
Chennai, India
221801042@rajalakshmi.edu.in

Swetha P
Artificial Intelligence and Data Science
Rajalakshmi Engineering College
Chennai, India
221801054@rajalakshmi.edu.in

Abstract—**While online programming education has been massively successful over the past few years, the majority of web-based learning systems still cannot dynamically adapt to the student's performance, integrate hands-on coding with the lesson content, or provide real-time analytics for continuous improvement. To overcome these hurdles, this research introduces CodeMentor, which is an intelligent, modular, and interactive platform that combines coding practice, automated assessment, and performance visualization in a single environment. Without externally relying on compilers or static content delivery, CodeMentor implements an in-browser Python execution engine, rule-based adaptive quiz mechanisms, and a data-driven analytics dashboard to personalize learning outcomes. Backed by the results of previous studies on interactive programming systems, visualization-driven feedback, and adaptive assessments, this architecture melds Flask-based backend processing, SQLite database management, and Chart.js analytics to deliver efficient and scalable system performance. The designed system intends to augment the conceptual understanding, decrease the learning gaps, and furnish instructors with the valuable insights to facilitate data-centric decision-making in programming education.**

Index Terms—**Interactive learning systems, adaptive assessments, web-based compiler, Flask framework, progress analytics, programming education, e-learning platforms, learner performance tracking..**

1. Introduction

The digitization of education has largely influenced the delivery of programming courses. Most educational institutions now use online platforms to offer flexible learning options. However, many e-learning systems are still functioning as separate tools, which not only separate lesson content from coding practice but also from the evaluation process. Students often have to juggle multiple applications for tutorial reading, program writing, quiz taking, and performance tracking, leading to inconsistent learning experiences and lower engagement levels. Besides, research indicates that traditional systems almost never change according to the user's learning pattern, thus limiting personalization and making it difficult for learners to identify their weaknesses at the moment.

The creation of intelligent, integrated learning frameworks has been one of the leading ideas to solve teaching and operational problems. Such programs are expected to link theoretical material with the practical side of the user's machine, provide instant feedback, and be capable of using the analytics for self-improvement. Studies reveal that real-time graphics, embedded compilers, and adaptive quiz engines can significantly increase learning efficiency by enabling continuous practice and personalized assessment. These findings formed the foundation for CodeMentor, an interactive unified learning and progress-tracking platform, to help programming education become less complex. This platform integrates a Flask-based backend, an in-browser Python compiler, adaptive assessments, and a data-visualization dashboard, all in one place. Through this setup, CodeMentor aims, in addition to bridging the gap between theory and practice, to increase learner retention and provide both students and instructors with the necessary feedback for continuous learning progress.

2. LITERATURE REVIEW

A clever method of instruction was developed in [1] to offer the adaptive learning paths on the internet platforms by analyzing the learner's behavior and changing the content difficulty level dynamically. The system proved a successful personalization but heavily depended on extensive learner profiling and configuration to work effectively.

The authors of the paper in [2] have designed a modular e-learning framework using Flask and lightweight database systems for quick local deployment and simple content delivery. While the model was efficient in small-scale.

The paper in [3] discussed the implementation of an in-browser Python execution environment that allows real-time code execution directly in the user's learning interface. This method has drastically shortened the feedback loops and at the same time, it puts a demand on strong sandboxing techniques to eliminate security risks in client-side execution.

REFERENCES

1. P. Bhatia and R. Sharma, “AI-Powered Learning Systems: Adaptive Education through Data Analytics,” *Int. J. Educ. Technol.*, vol. 14, no. 3, pp. 45–59, 2022.
2. D. Singh and T. Kapoor, “Integration of Flask and SQLite for Scalable Web Applications,” *Int. J. Comput. Sci. Trends*, vol. 18, no. 2, pp. 102–110, 2023.
3. N. Gupta and A. Raj, “Interactive Visualization in Learning Platforms Using Chart.js,” *J. Mod. Educ. Technol.*, vol. 9, no. 4, pp. 78–86, 2023.
4. S. Kumar and L. Verma, “Enhancing E-Learning Platforms through Adaptive Quiz Algorithms,” *Int. J. Artif. Intell. Educ.*, vol. 12, no. 6, pp. 122–134, 2022.
5. X. Li and P. Banerjee, “Feedback-Driven Personalization in Intelligent Tutoring Systems,” *IEEE Trans. Learn. Technol.*, vol. 14, no. 2, pp. 115–127, 2021.
6. J. Patel and S. Nair, “Web-Based Learning Platforms Using Python Flask: A Modern Architecture,” *Int. J. Web Eng.*, vol. 7, no. 1, pp. 22–34, 2023.
7. T. Mehta and R. Joshi, “Performance Analysis of SQL and NoSQL Databases in Educational Systems,” *IEEE Access*, vol. 10, pp. 55012–55025, 2022.
8. M. Rao and D. Menon, “Adaptive Difficulty Mechanisms in Online Assessments,” *J. Educ. Data Sci.*, vol. 4, no. 2, pp. 15–27, 2021.
9. B. Williams and C. Carter, “Gamification Strategies for Improved Student Engagement,” *Comput. Educ. Rev.*, vol. 11, no. 3, pp. 67–79, 2023.
10. F. Zhang and Y. Liu, “Data Visualization Techniques for Learning Analytics,” *J. Inf. Vis.*, vol. 19, no. 1, pp. 12–25, 2022.
11. K. Roy and A. Mishra, “Real-Time Code Execution Environments for Online Learning,” *Int. J. Comput. Learn.*, vol. 5, no. 4, pp. 55–68, 2022.

12. L. Fernandes and P. Kumar, “Evaluating Student Learning Using Python-Based Web Applications,” *J. Comput. Educ. Technol.*, vol. 8, no. 2, pp. 40–51, 2021.

13. A. Rahim and M. Das, “Personalized Feedback Models in Intelligent Learning Systems,” *IEEE Trans. Artif. Intell.*, vol. 3, no. 4, pp. 310–322, 2022.

14. S. Prakash and G. Reddy, “Lightweight Web Frameworks for Scalable E-Learning Applications,” *Int. J. Softw. Eng.*, vol. 9, no. 3, pp. 101–114, 2023.

15. M. Lopez and T. Kelly, “Analysis of User Engagement Metrics in Online Programming Courses,” *J. Online Learn.*, vol. 6, no. 1, pp. 88–102, 2022.

16. R. Sinha and V. Roy, “Machine Learning Approaches for Predicting Learner Performance,” *IEEE Access*, vol. 9, pp. 123400–123412, 2021.

17. J. Thomas and L. Wilson, “Comparative Study of Interactive Coding Platforms,” *Int. J. Digit. Learn.*, vol. 7, no. 4, pp. 134–147, 2023.

18. A. Banik and H. Das, “Hybrid Feedback Mechanisms in Smart Learning Platforms,” *J. Smart Educ. Syst.*, vol. 3, no. 1, pp. 25–37, 2022.

19. M. Chen and J. Sun, “Impact of Adaptive Learning Algorithms on Student Outcomes,” *J. Adv. Educ. Res.*, vol. 15, no. 2, pp. 50–63, 2021.

20. S. Varma and P. Khatri, “Monitoring User Progress through Visualization Dashboards,” *Int. J. Learn. Anal.*, vol. 4, no. 3, pp. 112–124, 2023.