****

**COMSATS University, Islamabad Pakistan**

**Codable – Code Learning Platform**

***By***

**Khawaja Muhammad Rafay CIIT/FA22-BCS-038/ISB**

**Abdul Rafay CIIT/FA22-BCS-003/ISB**

***Supervisor*Prof. Majid Iqbal Khan**

***Bachelor of Science in Computer Science (2022-2026)***

**The candidate confirms that the work submitted is their own and appropriate  
 credit has been given where reference has been made to the work of others**.

****

**COMSATS University, Islamabad Pakistan**

**Codable – Code Learning Platform**

**A project presented to**

**COMSATS University, Islamabad**

**In partial fulfillment**

**of the requirement for the degree of**

***Bachelor of Science in Computer Science (2022-2026)***

**By**

**Khawaja Muhammad Rafay CIIT/FA22-BCS-038/ISB**

**Abdul Rafay CIIT/FA22-BCS-003/ISB**

**DECLARATION**

We hereby declare that this software, neither whole nor as a part, has been copied out from any source. It is further declared that we have developed this software and accompanied report entirely on the basis of our personal efforts. If any part of this project is proved to be copied out from any source or found to be a reproduction of some other, we will stand by the consequences. No Portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

Khawaja Muhammad Rafay Abdul Rafay

--------------------------- ---------------------------

**CERTIFICATE OF APPROVAL**

It is to certify that the final year project of BS (CS) Dukan was developed by   
**KHAWAJA MUHAMMAD RAFAY (CIIT/FA22-BCS-038/ISB)** and **ABDUL RAFAY (CIIT/FA22-BCS-003/ISB)** under the supervision of PROF. MAJID IQBAL KHAN and that in his opinion; it is fully adequate, in scope and quality for the degree of Bachelors of Science in Computer Sciences.

---------------------------------------

**Supervisor**

---------------------------------------

**External Examiner**

---------------------------------------

**Head of Department**

**(Department of Computer Science)**

**EXECUTIVE SUMMARY**

In today’s world, learning programming has become a crucial skill, but the traditional methods of teaching and practicing coding often fall short in terms of engagement, personalization, and effectiveness. Beginners face problems like struggling to build logic, being overwhelmed with syntax, lack of guided practice, and limited feedback on their mistakes. On the other hand, instructors face challenges in managing classrooms, tracking student progress, and providing timely support.

To address these issues, **Codable** is developed. It is an intelligent, all-in-one coding learning platform designed to guide learners through every phase of becoming a programmer, from understanding the fundamentals of programming through logic building, to structured learning, skill polishing, and finally moving towards real development. The platform integrates AI-driven feedback, adaptive question generation, and progress tracking to make the learning experience personalized and effective.

**Codable** includes interactive modules such as a **Logic Building Canvas** for flowcharts to help beginners understand the core concepts behind programming before diving into syntax. It further provides structured documentation-based lessons, quizzes, and a smart practice environment where coding questions are generated based on previous mistakes. A VS Code–like compiler is also embedded, allowing learners to experiment with real coding under the guidance of an AI assistant.

The system also supports an **Instructor/Class Management Module**, enabling teachers to create classrooms, assign content, track student performance, and evaluate submissions with ease. This dual approach ensures that the platform can be used both for self-learning and guided, instructor-led learning.

Codable is a scalable platform with the potential to support multiple programming languages in the future, making it a complete ecosystem for coding education.

# Abstract

The rationale for developing **Codable** is to provide an intelligent and structured platform that makes learning programming more effective and accessible. Existing systems such as static tutorials, coding editors, and generic practice platforms primarily focus on syntax and lack interactive, adaptive, and personalized support for learners. While these approaches provide basic exposure, they often overlook concept-building, fail to adapt to individual needs, and do not offer effective classroom management for instructors. The proposed project aims to address challenges such as the absence of logic-building before syntax, limited personalized feedback, inadequate progress tracking, and the lack of integration between self-learning and instructor-led environments. The gap lies in the need for a holistic platform that bridges conceptual understanding, coding practice, and guided teaching in a single ecosystem. The objectives of the project are to introduce a Logic Building Module with interactive flowcharts for strengthening problem-solving, provide structured lessons with quizzes for systematic learning, and integrate AI-driven practice and feedback for skill polishing. It will also include a VS Code–like compiler for real-world development and an Instructor/Class Management Module for guided learning. Ultimately, the project’s significance lies in offering a comprehensive, adaptive, and scalable platform that transforms coding education for both individual learners and classrooms.

**Acknowledgement**

All praise is to Almighty Allah who bestowed upon us a minute portion of His boundless knowledge by virtue of which we were able to accomplish this challenging task.

We are greatly indebted to our project supervisor Prof. Majid Iqbal Khan. Without their personal supervision, advice and valuable guidance, completion of this project would have been doubtful. We are grateful to them for their encouragement and continual help during this work.

And we are also thankful to our parents and family who have been a constant source of encouragement for us and brought us with the values of honesty & hard work.

Khawaja Muhammad Rafay Abdul Rafay

--------------------------- ---------------------------

**Abbreviations**

|  |  |
| --- | --- |
| **SRS** | Software Require Specification |
| **PC** | Personal Computer |
|  |  |
|  |  |
|  |  |

Table of Contents

[Abstract v](#_Toc211354896)

[1 Chapter 1: Introduction and Problem Definition 3](#_Toc211354897)

[1.1 Overview of the Project 3](#_Toc211354898)

[1.2 Vision Statement 3](#_Toc211354899)

[1.3 Problem Statement 4](#_Toc211354900)

[1.4 Problem Solution 4](#_Toc211354901)

[1.5 Objectives of the Proposed System 4](#_Toc211354902)

[1.6 Scope 5](#_Toc211354903)

[1.6.1 Limitations/Constraints 5](#_Toc211354910)

[1.7 Modules 6](#_Toc211354911)

[1.7.1 Module 1: Logic Building Module 6](#_Toc211354914)

[1.7.2 Module 2: Skill Improvement Module 6](#_Toc211354915)

[1.7.3 Module 4: Interactive Code Workspace 7](#_Toc211354916)

[1.7.4 Module 5: Class & Instructor Management 7](#_Toc211354917)

[1.7.5 Module 6: Progress Tracking and Analytics 7](#_Toc211354918)

[1.7.6 Module 7: Personalized AI Tutor 8](#_Toc211354919)

[1.7.7 Module 8: Smart Question Generator 8](#_Toc211354920)

[1.7.8 Module 9: Feedback & Rating System 8](#_Toc211354921)

[1.8 Related System Analysis/Literature Review 9](#_Toc211354922)

[1.8.1 Literature Review 9](#_Toc211354924)

[1.8.2 Related System Analysis 9](#_Toc211354925)

[1.9 Tools and Technologies 10](#_Toc211354926)

[1.10 Project Contribution 11](#_Toc211354927)

[1.11 Relevance to Course Modules 12](#_Toc211354928)

[2 Chapter 2: Requirement Analysis 13](#_Toc211354929)

[2.1 User Classes and Characteristics 13](#_Toc211354931)

[2.1.1 Use case Diagram 13](#_Toc211354932)

[2.1.2 Requirement Identifying Technique 15](#_Toc211354933)

[2.2 Functional Requirements 16](#_Toc211354934)

[2.2.1 Backend Process Requirements 22](#_Toc211354935)

[2.3 Non-Functional Requirements 24](#_Toc211354936)

[2.3.1 Reliability 24](#_Toc211354937)

[2.3.2 Usability 24](#_Toc211354938)

[2.3.3 Performance 25](#_Toc211354939)

[2.3.4 Security 25](#_Toc211354940)

[2.4 External Interface Requirements 26](#_Toc211354941)

[2.4.1 User Interfaces Requirements 26](#_Toc211354942)

[2.4.2 Software interfaces 27](#_Toc211354943)

[2.4.3 Hardware interfaces 28](#_Toc211354944)

[2.4.4 Communications interfaces 28](#_Toc211354945)

**List of Figures**

[Figure 4.3 Assignee Dashboard 29](#_Toc204256460)

[Figure A‑1: Context diagram of the Cafeteria Ordering System. 36](#_Toc204256461)

[Figure A-2: Use Case Diagram of an Appointment System 38](#_Toc204256462)

# Chapter 1: Introduction and Problem Definition

**Codable** is an intelligent, web-based platform aimed at transforming the way programming is taught and learned. The system caters to two key user groups: self-learners who wish to build programming skills independently, and instructors who need tools to manage classes, assign learning material, and evaluate student progress. Unlike traditional methods that focus mainly on syntax and static tutorials, **Codable** emphasizes logical understanding, adaptive practice, AI-driven feedback, and classroom management, making programming education more structured, interactive, and scalable.

Current programming education platforms face several limitations. Most rely on syntax-based teaching and lack a strong foundation in logic-building, leaving beginners struggling to understand core concepts. Learners often receive little to no personalized feedback, making it difficult to identify and correct mistakes effectively. At the same time, instructors face challenges in monitoring student performance, assigning targeted exercises, and providing timely support in large classrooms. These gaps highlight the need for an integrated, intelligent system that combines logic-building, structured learning, personalized practice, real-world development, and class management features under one unified platform.

## Overview of the Project

**Codable** is an intelligent, web-based platform designed to modernize and enhance the process of learning programming. Operating in the domain of computer science education, the system targets two main user groups: self-learners who want to build programming skills independently, and instructors who require effective tools to create classes, assign content, and track student performance. Traditional coding education relies heavily on syntax memorization, static tutorials, and manual practice, which often fails to ensure logical understanding, adaptive learning, or personalized feedback. Additionally, instructors face difficulties in monitoring large groups, identifying common problem areas, and providing timely evaluations. The proposed system addresses these challenges by introducing automation in learning through AI-driven feedback, adaptive question generation, progress tracking dashboards, and integrated class management features. By combining modules for logic building with flowcharts, structured lessons, coding practice, and a VS Code–like environment with AI assistance, **Codable** provides a comprehensive ecosystem that supports both independent learners and instructor-led classrooms, ensuring scalability, efficiency, and effectiveness in programming education.

## Vision Statement

For aspiring programmers and educators who seek an engaging, adaptive, and efficient platform for learning and teaching programming, Codable is an interactive web-based code learning system that provides personalized tutorials, real-time AI-driven code evaluation, and seamless class management tools. Unlike static, one-size-fits-all platforms or traditional classroom setups, Codable empowers users with adaptive learning paths, real-time feedback, and intelligent task recommendations, ensuring continuous growth and mastery of any programming language. Our product stands out by offering a dynamic, user-centered approach that combines structured learning with hands-on practice, making coding education more interactive, scalable, and accessible to all learners.

## Problem Statement

In the fast-evolving tech landscape, coding has become an indispensable skill for success, yet the journey to mastering it remains overwhelming for many learners. Aspiring programmers often struggle to find reliable resources that not only explain concepts but also provide structured guidance and personalized feedback. Existing platforms typically offer isolated tutorials or exercises, which fail to ensure interactive engagement or step-by-step progression. Beginners frequently feel lost when trying to build logical foundations, while intermediate learners face challenges in identifying and correcting mistakes effectively. At the same time, instructors encounter difficulties in managing diverse groups of learners, tracking individual progress, and providing timely, tailored support. These challenges highlight the gaps in current programming education systems and the pressing need for a more comprehensive and adaptive approach.

## Problem Solution

Our proposed platform, Codable, provides a comprehensive solution for learning programming through structured, adaptive, and interactive modules. Learners begin with logic-building exercises using flowcharts, progress into structured lessons with practice tasks, and reinforce their knowledge with quizzes. To support continuous improvement, learners are assessed on their coding skills and then provided with tailored exercises to strengthen weaker areas. The integrated AI model evaluates how learners write code, how they approach problems, and how they perform, ensuring a truly personalized learning journey. A progress tracking system allows both learners and instructors to monitor growth, while the instructor module supports class management, task assignment, and AI-powered analytics of student performance. Additionally, Codable offers a workspace with a real coding environment, enabling users to practice, run, and debug code interactively with AI assistance. This unified approach ensures effective, engaging, and scalable programming education for both self-learners and instructors*.*

## Objectives of the Proposed System

**BO-1:** Provide a step-by-step learning path starting from logic-building through flowcharts, progressing into structured programming tutorials, hands-on practice, and integrated quizzes to guide learners from basics to advanced concepts.

**BO-2:** Implement an AI-powered evaluation system that not only checks syntax, logic, and efficiency but also analyzes how learners write code, how they learn, and how they perform, offering tailored feedback and growth insights.

**BO-3**: Deliver a personalized and adaptive learning experience by recommending skill-improvement tasks and challenges based on learner performance, ensuring continuous skill development and confidence building.

**BO-4:** Enable instructors to manage classes effectively, assign coding or logic-building tasks, and access AI-driven analytics with detailed progress tracking to monitor and support individual student growth.

**BO-5:** Integrate a dedicated coding workspace with a smart editor that provides real-time AI assistance for coding, debugging, and logic-building, ensuring an interactive and engaging coding experience.*.*

## Scope

The scope of Codable is to provide a comprehensive and intelligent coding education platform that caters to both learners and instructors by combining structured learning, interactive practice, and AI-driven evaluation. The platform begins with a logic-building module, where learners develop problem-solving skills using flowcharts and visual representations before moving on to actual coding. This ensures that users build strong foundations in computational thinking before learning programming syntax. Once learners transition into programming, they follow a structured curriculum with tutorials, practice exercises, and quizzes, enabling step-by-step skill development. To further enhance the learning process, an AI-powered evaluation engine assesses not only the correctness of code but also its logic, efficiency, and the way learners approach problem-solving. Based on this evaluation, the system provides personalized feedback and adaptive tasks to help learners strengthen weak areas and continuously improve their skills. Learners also have access to a dedicated coding workspace with a smart code editor, offering real-time AI assistance for writing, debugging, and optimizing code. The platform includes progress tracking features, enabling learners to visualize their journey, identify growth patterns, and stay motivated. On the instructor side, the system provides a class management module that allows mentors to create and manage classes, assign coding or logic-building tasks, and monitor the performance of individual students or groups. AI-driven analytics assist instructors by automatically generating performance reports and insights, reducing their workload while improving the quality of personalized feedback they can provide. The platform also supports adaptive learning recommendations, where learners are assigned coding challenges and improvement tasks based on their prior performance and learning style. Unlike traditional static learning systems, Codable emphasizes hands-on, interactive, and guided learning that evolves with the learner’s progress. The scope covers integration of quizzes, coding practice, AI evaluation, instructor tools, and real-time collaboration features, making it suitable for both self-learners and classroom-based learning. It does not aim to replace professional integrated development environments (IDEs), but rather to provide a simplified, learning-focused workspace that simulates real coding environments in an educational context. The system will initially be launched for a single programming language but is designed to be scalable for multiple languages in the future. In summary, the scope of Codable lies in creating an end-to-end solution that bridges the gap between traditional coding tutorials and modern, intelligent learning systems, providing structured learning for beginners, skill enhancement for intermediates, and effective class management tools for instructors.



### Limitations/Constraints

**LI-1:** The platform will initially support only the Java programming language; support for other languages may be considered in future versions.

**LI-2:** Real-time AI evaluation and feedback are dependent on external LLM APIs or hosted models, which may introduce latency or usage limits due to API rate restrictions or server load.

**LI-3:** The accuracy and quality of AI-generated suggestions and evaluations may not always be perfect and can vary based on model limitations or vague user code.

**LI-4:** Users must have a stable internet connection to interact with the platform features, especially those relying on cloud-based LLMs and the Monaco editor.

**LI-5:** The initial version of the system will be web-only and will not include a mobile application.

**LI-6:** Users attempting to run malicious, infinite loops, or heavy resource-consuming code may affect system performance; sandboxing and code execution limits will be applied.

**LI-7:** Instructor access to student performance will be limited to task and topic-related metrics and will not provide full code traceability due to privacy and fairness concerns.

## Modules



### Module 1: Logic Building Module

This module focuses on helping absolute beginners develop problem-solving and logical thinking skills before diving into coding. Users create flowcharts to visualize algorithms, gradually learning how to break problems into smaller steps.

FE-1: Provide either visual canvas or non-technical questionnaires to create flowcharts

Beginners are given an easy-to-use interface with standard flowchart symbols (start, input/output, process, decision, loops).

FE-2: Offer guided exercises for basic problem-solving

The system provides simple tasks (e.g., calculate the sum of two numbers, find the largest of two values) that must be solved using flowcharts.

FE-3: Convert flowcharts into pseudo-code/code for validation

Once a flowchart is complete, it can be translated into pseudo-code or real code (Java/Python/etc.) to check correctness and execution flow.

FE-4: Evaluate logic correctness and provide hints

AI evaluates whether the constructed flowchart correctly solves the problem and highlights errors (e.g., missing conditions, incorrect loops) with helpful hints.

FE-5: Track progress in logic-building tasks

The system saves completed exercises, performance, and improvement areas, ensuring the learner masters problem-solving before moving to coding.

### Module 2: Skill Improvement Module

This module is designed for users who already know the basics and want to improve. It uses AI to evaluate their current level and adaptively assign challenges that strengthen weak areas.

FE-1: Allow users to select difficulty level: Beginner, Intermediate, or Advanced

Users start by selecting their perceived skill level. This initial choice helps tailor the assessment experience and sets expectations for the upcoming challenges.

FE-2: Conduct a pre-assessment test to evaluate current skill level

Before assigning learning content, the platform verifies the user’s actual proficiency with a short test. This test includes logic, OOP, arrays, and control structures.

FE-3: Assign topic-specific coding challenges

Based on assessment results, users receive coding problems that target their weakest areas. Each challenge is designed to improve specific concepts.

FE-4: Evaluate code submissions for syntax, logic, and time complexity using LLM

The submitted code is analyzed by a language model that checks for correctness, logic, and efficiency. Feedback is provided accordingly.

FE-5: Provide AI-generated tips and feedback for better logic application

When logic or structure is weak, users receive step-by-step tips and alternate approaches. This feedback helps them think better and refactor their code.

### Module 4: Interactive Code Workspace

This module is a real-time code playground with an AI-powered coding assistant. It mimics a simplified version of VS Code in the browser.

FE-1: Integrate Monaco Editor for code editing

Users write and format code in the Monaco Editor, a lightweight but powerful editor used in VS Code.

FE-2: Enable code execution with output display

Users can instantly compile and run code. Output is displayed in a console area, just like in traditional IDEs.

FE-3: Support syntax highlighting and auto-completion

Enhances user experience by making the code easier to read and reducing syntax errors with smart suggestions.

FE-4: Integrate AI assistant (CodeT5+ or Copilot-like) for real-time help

As users type, an AI model assists by suggesting completions, fixing errors, or providing code explanations.

FE-5: Allow users to save code snippets and view edit history

Users can bookmark useful code, revisit past edits, and track how their code evolved over time.

### Module 5: Class & Instructor Management

This module empowers mentors to manage classes, assign tasks, and monitor students. Ideal for bootcamps, coding workshops, or formal teaching environments.

FE-1: Instructor registration and dashboard access

Instructors sign up separately and gain access to a dashboard tailored for class and student management.

FE-2: Create and manage classes with enrollment codes

Instructors can create classes, generate unique codes, and invite students to join with a single link or code.

FE-3: Assign coding tasks to students based on selected topics

Instructors pick topics from the curriculum and let the system generate coding tasks accordingly.

FE-4: View, review, and evaluate student submissions and progress

Teachers get access to all student submissions, detailed feedback reports, and progress analytics.

FE-5: Set task deadlines and grading rubrics

Helps structure the learning process by setting deadlines and evaluating submissions according to defined rubrics.

### Module 6: Progress Tracking and Analytics

This module tracks learner performance and provides insights into their coding journey. It helps users and instructors monitor growth, identify weak areas, and adapt learning paths accordingly.

FE-1: Track chapter, topic, and quiz completion

The system records which lessons, topics, and assessments the learner has completed. This creates a clear picture of progress through the curriculum.

FE-2: Monitor quiz and assessment performance

Stores detailed performance data such as quiz scores, pass/fail rates, number of attempts, and time taken per assessment.

FE-3: Capture coding practice activity

Logs coding problems attempted, solved, failed, and time spent per problem. Identifies common error patterns (syntax vs logic).

FE-4: Provide visual analytics and progress reports

Displays dashboards and charts showing completion percentages, score trends, and skill growth over time for learners and instructors.

### Module 7: Personalized AI Tutor

This module provides real-time, AI-driven guidance tailored to each learner’s coding journey. It analyzes mistakes, explains concepts, and suggests personalized practice to strengthen weak areas.

FE-1: Offer instant feedback on code submissions

The AI reviews user code step by step, identifies syntax and logic errors, and explains them in simple terms with hints for correction.

FE-2: Suggest personalized practice questions

Based on the learner’s mistakes and weak concepts, the AI generates or recommends tailored coding problems for reinforcement.

FE-3: Provide code improvement suggestions

The tutor highlights opportunities to make code more efficient, readable, or optimized (e.g., better time complexity or cleaner syntax).

FE-4: Enable interactive Q&A support

Learners can ask the AI tutor conceptual or debugging questions, and receive context-aware explanations with examples.

### Module 8: Smart Question Generator

This module dynamically generates coding questions tailored to the learner’s skill level and progress. It ensures variety, increasing difficulty, and alignment with the concepts being studied.

FE-1: Generate adaptive coding questions based on progress

The system analyzes completed topics, quiz results, and past mistakes to create relevant practice questions.

FE-2: Offer multiple difficulty levels

Questions are automatically categorized into beginner, intermediate, and advanced levels to match the learner’s growth.

FE-3: Provide diverse question formats

Includes MCQs, output prediction, debugging tasks, and full coding challenges to build well-rounded problem-solving skills.

FE-4: Avoid repetition with intelligent variation

The generator ensures that each new question is unique, preventing redundancy and keeping practice engaging.

### Module 9: Feedback & Rating System

Enables a feedback loop that improves both the platform and its content through user ratings and reviews.

FE-1: Allow learners to provide feedback on tutorials and challenges

After completing content, users can submit reviews or suggestions to help content creators

FE-2: Enable instructors to give feedback on individual student submissions

Instructors can comment on a student's code in addition to the AI-generated feedback.

FE-3: Let users rate AI feedback for quality improvement

Learners can mark whether the AI’s feedback was helpful, guiding future improvements.

FE-4: Store and display average feedback scores for content

Content with high ratings can be promoted, while lower-rated items may be revised.*.*

## Related System Analysis/Literature Review



### Literature Review

Previous research in intelligent programming education platforms and AI-based tutoring systems (e.g., Li et al., 2021; Kumar & Singh, 2022) highlights the role of adaptive feedback and automated code evaluation in enhancing learner engagement and coding accuracy. However, most of these systems focus solely on syntax correction or code output verification, lacking deeper evaluation of logic-building and problem-solving skills. Similarly, studies on visual learning environments for beginners (e.g., Ahmed et al., 2023) demonstrate that flowchart-based learning improves conceptual understanding but often operate as isolated modules without integration into a full coding curriculum.

Recent advancements in machine learning–driven assessment tools (e.g., Zhao et al., 2022) and personalized learning systems (e.g., Chen et al., 2023) emphasize the benefits of AI-powered recommendations to tailor learning paths based on user performance. Despite these efforts, existing systems rarely provide continuous skill improvement mechanisms that assess how a learner codes, learns, and progresses over time. Moreover, instructor management tools in prior solutions remain limited, offering little to no automation in performance tracking and analytics.

**Codable** addresses these gaps by integrating AI-driven learning analysis, flowchart-based logic-building, and a structured programming curriculum into a unified, interactive environment. It evaluates not only code correctness but also the learner’s logic, structure, and coding behavior, delivering personalized tasks for continuous improvement. The platform also empowers instructors through AI-powered class management, automated performance reports, and student progress visualization. By combining structured learning, adaptive feedback, and intelligent analytics, **Codable** extends the existing research by offering a complete, data-driven ecosystem for both learners and instructors—bridging the gap between conceptual understanding and real-world coding proficiency*.*

### Related System Analysis

|  |  |  |
| --- | --- | --- |
| **Application Name** | **Weakness** | **Proposed Project Solution** |
| W3Schools | Offers static tutorials with limited interactivity; lacks code evaluation and personalized feedback. | Codable provides interactive topic-wise lessons with instant quizzes, real-time code practice, and AI-generated feedback, helping users correct mistakes and improve their logic as they learn. |
| LeetCode | Focused mostly on coding challenges; lacks beginner-level structured learning paths and real-time feedback on logic or efficiency of code. | Codable offers beginner-to-advanced guided learning modes, with AI evaluating the correctness and time complexity of each code submission and providing suggestions for improvement. |

Table 1.1 Related System Analysis with proposed project solution

## Tools and Technologies

|  |  |  |  |
| --- | --- | --- | --- |
| **Tools**  **And**  **Technologies** | **Tools** | **Version** | **Rationale** |
| Visual Studio Code | 1.89.1 | Code editor for both frontend and backend development |
| Postman | 10.x | API testing and debugging |
| Figma | Latest | UI/UX design and mockup creation |
| MongoDB Compass | 1.40.x | GUI for managing MongoDB database |
| Git & GitHub | Latest | Version control and collaboration |
| **Technology** | **Version** | **Rationale** |
| React JS | 19.x | Front-end Development |
| Node.js | 22.x | Server-side JavaScript runtime |
| Express.js | 5.x | RESTful API backend development |
| MongoDB | 8.x | NoSQL database for storing user and class data |
| Monaco Editor | Latest | Embedded browser-based code editor |
| Machine Learning (Python/Scikit-learn) | Latest | For question recommendation system |
| CodeT5+ | Latest | LLMs for code evaluation and feedback |
| React Flow or Fabric JS | Latest | To build canvas for flowcharts |
| OpenAI API *(optional)* | GPT-4 | AI-powered suggestions and analysis |

Table 1.2 Tools and Technologies for Proposed Project

## Project Contribution

The Codable – Code Learning Platform introduces several technical and conceptual innovations aimed at transforming the way programming, specifically Java, is taught and practiced online. Unlike conventional learning platforms that separate theory from practice, Codable unifies structured learning, AI-driven evaluation, and personalized improvement in one adaptive environment.

**Technical and Conceptual Contributions**

* **AI-Driven Code Evaluation Engine**
  + Uses Natural Language Processing (NLP) and Machine Learning (ML) models (e.g., CodeT5+) to evaluate user-submitted Java code not only for correctness but also for readability, conciseness, and time complexity.
  + Provides human-like explanations for logic errors, helping learners understand *why* their solution failed rather than just *that* it failed.
* **Smart Question Recommendation System**
  + Implements an ML-based question generator that analyzes a learner’s previous mistakes, weak areas, and coding style to automatically generate **personalized practice problems**.
  + This makes practice adaptive and continuous, unlike fixed question banks on existing coding platforms.
* **Personalized AI Tutor Integration**
  + Embeds an interactive AI tutor that maintains contextual memory across user sessions, offering step-by-step guidance, debugging tips, and explanations during problem-solving.
  + Enables a human-tutor-like learning experience that adapts to each learner’s pace.
* **Hybrid Learning Model (Textbook + Practice Integration)**
  + Introduces a book-based learning module that merges structured theoretical content (scraped and adapted from trusted sources) with interactive coding exercises and quizzes.
  + Allows learners to move from reading to coding seamlessly within one interface, reducing context switching.
* **Instructor-Led and Self-Learning Dual Mode**
  + Supports both self-paced learners and instructor-managed classes, where instructors can assign tasks, deadlines, and grading rubrics while monitoring student progress in real-time.
  + Blends academic-style course management with automated code evaluation.
* **VS Code-like Workspace Environment**
  + Provides an embedded sandboxed compiler with a user interface similar to Visual Studio Code, enabling learners to write, execute, and test Java programs in a secure environment.
  + The workspace integrates with the AI engine to provide instant, context-aware code feedback.
* **Integrated Progress Tracking and Analytics**
  + Continuously tracks each user’s learning progress, evaluation results, and topic mastery across all modules.
  + Aggregates this data into a visual dashboard that displays trends and improvement areas, helping learners identify specific weaknesses.
* **Logic-Building and Concept Reinforcement Module**
  + Simplifies programming logic for beginners through non-technical learning methods, including conceptual flow breakdowns, logic-based MCQs, and guided examples.
  + Strengthens foundational thinking before code writing, improving long-term learning retention.

**Contribution Impact**

By combining AI, ML, compiler technology, and structured pedagogy, Codable redefines the traditional online coding experience into an intelligent, adaptive, and feedback-driven ecosystem.  
These innovations collectively:

* Increase learning efficiency through personalization and automation.
* Enhance usability by integrating all learning stages (reading, coding, feedback) in one unified platform.
* Provide scalable instructor tools for academic or institutional use.
* Improve performance and engagement through adaptive learning and real-time feedback loops.

Compared to existing coding platforms, Codable stands out for its AI-tutored learning flow, ML-based problem generation, and real-time personalized evaluation, making it a smart, data-driven learning companion rather than just a code compiler or question bank.

## Relevance to Course Modules

* **Programming Fundamentals (PF):** Applied core programming concepts, logical problem-solving, and control structures in building platform functionalities.
* **Database Systems:** Designed and managed structured data storage for users, quizzes, and progress tracking using a cloud-based database.
* **Web Technologies:** Applied modern web frameworks like React and Node.js to develop an interactive frontend and a scalable backend.
* **Software Engineering Concepts:** Followed SDLC phases including requirement gathering, UML-based design modeling, development, testing, and documentation.
* **Artificial Intelligence (AI):** Integrated AI techniques for automated code evaluation, logical error detection, and personalized learning recommendations.
* **Machine Learning (ML):** Applied ML models to identify user learning behaviors, recommend suitable coding challenges, and improve adaptive learning.
* **DevOps:** Implemented version control, automated deployment, and continuous integration using tools like GitHub and cloud hosting environments

# Chapter 2: Requirement Analysis



## User Classes and Characteristics

A diagram of a software company

AI-generated content may be incorrect.

Fig 1. Context Diagram

### Use case Diagram

A diagram of a coder

AI-generated content may be incorrect.

Fig 1.1 System Centric Use Case Diagram

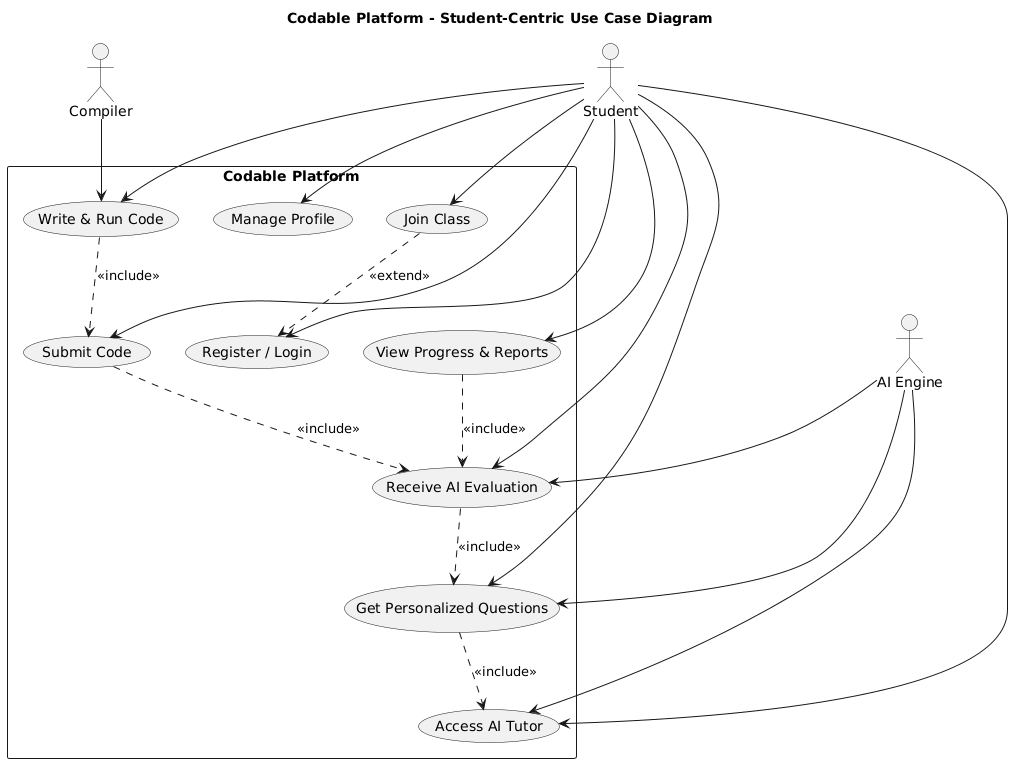


Fig 1.2 Student Centric Use Case Diagram

A diagram of a student

AI-generated content may be incorrect.

Fig 1.3 Instructor Centric Use Case Diagram

### Requirement Identifying Technique

This section describes the technique used to identify and document the functional requirements of the system.

For this project, [[1]](#footnote-1)**Mockup-Based Requirement Analysis** is used. In this approach, user interface mockups are created using UI/UX tools such as Figma, Adobe XD, or Balsamiq. These mockups visually represent how users will interact with the system. Functional requirements are then derived from each screen’s elements and behaviors.

#### Steps to Follow:

* Identify key features and break them down into individual **screens**.
* Assign a **Mockup ID** to each screen (e.g., M1 for Login Page, M2 for Dashboard).
* Create **mockups** of the screens using any UI/UX design tool.
* Identify **interactive elements** on each screen (buttons, forms, menus, etc.).
* Derive a functional requirement from each element, describing what the system should do in response.
* Organize all functional requirements using the table format provided in Section 3.3.

## Functional Requirements

**Module 1:** **User Profiling & Tracking Module**

|  |  |  |
| --- | --- | --- |
| ***Feature (derived from UI)*** | ***Functional Requirement (FR-ID: Statement)*** | ***Business Rule*** |
| ***User Registration Form*** | ***FR1.1:*** *The system shall automatically create a user profile upon registration, storing details such as name, email, signup date, and selected learning path.* | *Each email address must be unique; profile creation requires valid user credentials.* |
| ***User Dashboard*** | ***FR1.2:*** *The system shall maintain a detailed activity log for each user, recording lessons accessed, quizzes attempted, code submissions, and time spent.* | *Activity data shall be timestamped and stored securely.* |
| ***Progress Tracker*** | ***FR1.3:*** *The system shall track user progress across all learning paths (logic building, quizzes, and coding practice) and update progress percentages in real time.* | *Progress updates trigger only upon successful completion of an activity.* |
| ***Performance Analytics Panel*** | ***FR1.4:*** *The system shall analyze topic-wise performance to identify user strengths and weaknesses based on quiz scores and AI feedback.* | *Performance analytics are updated after every quiz or code evaluation.* |
| ***AI Evaluation Data Viewer*** | ***FR1.5:*** *The system shall store AI evaluation data (logic quality, efficiency, and improvement scores) to dynamically refine the user’s learning profile.* | *AI evaluation data shall remain read-only for users but editable by system processes.* |
| ***Level Indicator*** | ***FR1.6:*** *The system shall automatically update the user’s learning level (Beginner* ***→*** *Intermediate → Advanced) based on progress and performance trends****.*** | *Promotion between levels is based on reaching a**defined score threshold.* |
| ***Progress Visualization Dashboard*** | ***FR1.7:*** *The system shall visualize progress using graphs and charts showing performance over time, topic mastery, and coding efficiency.* | *Visualization updates are triggered in real time with every progress change.* |
| ***Task Manager*** | ***FR1.8:*** *The system shall maintain a list of completed and pending tasks linked to corresponding topics and modules.* | *Tasks marked as “completed” cannot be edited by the user****.*** |
| ***System Integration API*** | ***FR1.9:*** *The system shall share profile data with AI Evaluation, AI Tutor, Question Generator, and Instructor Management modules for personalized learning and reporting.* | *Data sharing occurs through secure internal APIs with authentication tokens.* |
| ***Session Summary Panel*** | ***FR1.10:*** *The system shall generate a performance summary report after each session, summarizing time spent, completed activities, and progress updates.* | *Reports are stored for a minimum of 30 days.* |
| ***Instructor Dashboard*** | ***FR1.11:*** *The system shall allow instructors to view and monitor student profiles, including progress history, strengths, and weaknesses.* | *Only assigned instructors can access specific student profiles.* |
| ***Data Privacy Controller*** | ***FR1.12:*** *The system shall enforce role-based access control, ensuring that only authorized users or instructors can view or modify profile data.* | *User data shall comply with data privacy standards and encryption protocols****.*** |

**Module 2: AI Evaluation Module**

|  |  |  |
| --- | --- | --- |
| ***Feature (derived from UI)*** | ***Functional Requirement (FR-ID: Statement)*** | ***Business Rule*** |
| ***Code Submission Panel*** | ***FR2.1:*** *The system shall receive and analyze user-submitted code from the coding workspace.* | *Code must be submitted in supported programming languages only (starting with Java).* |
| ***Secure Execution Environment*** | ***FR2.2:*** *The system shall execute user code in a secure, sandboxed environment to prevent unauthorized system-level access.* | *Sandbox limits CPU and memory usage to predefined thresholds.* |
| ***Code Evaluation Engine*** | ***FR2.3:*** *The system shall evaluate code based on multiple parameters, including correctness, logic accuracy, time complexity, memory efficiency, and readability.* | *Evaluation parameters are weighted based on difficulty level.* |
| ***Error Detection Interface*** | ***FR2.4:*** *The module shall detect and classify errors such as syntax errors,**logical errors, and runtime exceptions.* | *All detected errors must be labeled with error type and**line reference.* |
| ***Feedback Display Panel*** | ***FR2.5:*** *The system shall generate detailed feedback explaining identified issues and provide step-by-step guidance or hints for correction.* | *Feedback must be generated within 5 seconds after evaluation.* |
| ***Performance Scoring Component*** | ***FR2.6:*** *The system shall calculate and display an overall performance score or rating derived from evaluation metrics.* | *Score ranges from 0–100; performance <50 is flagged as “Needs Improvement.”* |
| ***Code Comparison Viewer*** | ***FR2.7:*** *The system shall compare the user’s latest submission with previous attempts to show performance and logic improvement trends.* | *Comparison results must be stored in the user’s learning history.* |
| ***Integration API with User Profile*** | ***FR2.8:*** *The module shall update the user’s profile data (progress, weaknesses, and skill metrics) after each evaluation.* | *Only the latest evaluation data overwrites previous performance metrics****.*** |
| ***Evaluation History Viewer*** | ***FR2.9:*** *The system shall store evaluation reports for future reference by learners and instructors.* | *Reports must be retained for a minimum of 90 days.* |
| ***AI Model Optimizer*** | ***FR2.10:*** *The system shall continuously improve feedback accuracy by training AI models on historical user performance data and common coding patterns.* | *Model retraining shall occur periodically without disrupting user sessions.* |

**Module 3: Performance-Based Question Generator**

|  |  |  |
| --- | --- | --- |
| ***Feature (derived from UI)*** | ***Functional Requirement (FR-ID: Statement)*** | ***Business Rule*** |
| ***Performance Data Interface*** | ***FR3.1:*** *The system shall retrieve user feedback, detected weaknesses, and performance metrics directly from the AI Evaluation Module after each coding session.* | *Data retrieval occurs automatically after every code submission.* |
| ***Weakness Interpretation Engine*** | ***FR3.2:*** *The system shall interpret user weaknesses (e.g., loop logic errors, recursion misuse, poor variable handling) to determine learning areas needing reinforcement.* | *Each weakness is tagged under a predefined topic or subtopic.* |
| ***Question Generation Engine*** | ***FR3.3:*** *The system shall generate new coding questions targeting the specific weaknesses identified by the AI Evaluation Module.* | *Each generated question must target at least one identified weak area.* |
| ***Difficulty Adjustment Mechanism*** | ***FR3.4:*** *The system shall adjust the difficulty level of generated questions based on the user’s current skill level and progress trend.* | *Difficulty levels are defined as Beginner, Intermediate, and Advanced.* |
| ***Question Variation System*** | ***FR3.5:*** *The system shall ensure diversity by generating question variations with different constraints or input conditions to avoid repetition.* | *A maximum of two identical questions per topic is allowed.* |
| ***Topic Mapping Panel*** | ***FR3.6:*** *The system shall map each identified weakness to its relevant topic or subtopic (e.g., loops, arrays, recursion) for structured learning progression.* | *Each topic must have a unique topic ID for database mapping.* |
| ***Question Queue Manager*** | ***FR3.7:*** *The system shall automatically queue generated questions into the user’s next practice or learning sessions.* | *Queued questions appear in the user’s dashboard in priority order.* |
| ***Data Sync with Profile Module*** | ***FR3.8:*** *The system shall send data of newly generated and attempted questions back to the User Profiling Module to update user progress and skill metrics.* | *Synchronization occurs in real time after question completion.* |
| ***Instructor Review Panel*** | ***FR3.9:*** *In instructor-led mode, the system shall allow instructors to view, approve, or modify AI-generated questions before assigning them to students.* | *Only authorized instructors can edit or publish AI-generated questions.* |

.

**Module 4: Class Management Module**

|  |  |  |
| --- | --- | --- |
| ***Feature (derived from UI)*** | ***Functional Requirement (FR-ID: Statement)*** | ***Business Rule*** |
| ***Class Creation Form*** | ***FR4.1:*** *The system shall allow instructors to create new classes by specifying class name, subject (e.g., Java), duration, and description.* | *Each class must have a unique class ID and an assigned instructor.* |
| ***Student Management Interface*** | ***FR4.2:*** *The system shall allow instructors to add, remove, or approve students within a class. Students may join via a class code or invitation link.* | *A student can be enrolled in multiple classes but only one instructor per class.* |
| ***Assignment Manager*** | ***FR4.3:*** *The system shall allow instructors to assign coding problems, quizzes, or practice sets to a class and define deadlines for submission.* | *Deadlines must be set within the course duration; overdue tasks are marked automatically.* |
| ***Class Dashboard*** | ***FR4.4:*** *The system shall provide instructors with a dashboard showing class-wide statistics such as average scores, completion rates, and progress distribution.* | *Statistics refresh automatically every 24 hours or upon manual refresh.* |
| ***Student Performance Viewer*** | ***FR4.5:*** *The system shall allow instructors to view AI Evaluation results and detailed feedback for each student to assess weaknesses and learning gaps.* | *Only authorized instructors can access their assigned class performance data.* |
| ***Announcement Board*** | ***FR4.6:*** *The system shall provide a communication panel where instructors can post announcements, share guidance, or pin important messages for the class.* | *Announcements remain visible to all enrolled students until removed by the instructor.* |
| ***Analytical Insights Panel*** | ***FR4.7:*** *The system shall generate class-level analytical insights, such as most common errors, top-performing students, and topics requiring revision.* | *Insights are generated weekly or upon completion of major assignments.* |
| ***Data Integration API*** | ***FR4.8:*** *The system shall fetch performance and progress data from the User Profiling and AI Evaluation Modules to display student-specific insights in the class dashboard.* | *Data integration follows secure API calls with role-based access validation.* |

.

Module 5: Interactive Coding Workspace

|  |  |  |
| --- | --- | --- |
| ***Feature (derived from UI)*** | ***Functional Requirement (FR-ID: Statement)*** | ***Business Rule*** |
| ***Code Editor Interface*** | ***FR5.1:*** *The system shall provide a web-based code editor with syntax highlighting, indentation, auto-completion, and real-time error detection.* | *The editor must support multiple programming languages, starting with Java.* |
| ***Code Execution Console*** | ***FR5.2:*** *The system shall allow users to write, compile, and execute code directly within the workspace.* | *Code execution is limited to a sandboxed environment for safety.* |
| ***Output & Error Display Panel*** | ***FR5.3:*** *The system shall display compilation results, runtime errors, and output in a separate console integrated within the workspace.* | *Output must appear instantly after execution completion.* |
| ***AI Coding Assistant*** | ***FR5.4:*** *The system shall provide real-time AI-based suggestions (similar to GitHub Copilot) for code completion, logic correction, and debugging.* | *Suggestions are non-intrusive and can be accepted or ignored by the user.* |
| ***Code Submission Button*** | ***FR5.5:*** *The system shall allow users to submit written code to the AI Evaluation Module for automated assessment and feedback generation.* | *Submissions are allowed only for supported question types or assignments.* |
| ***Auto-Save System*** | ***FR5.6:*** *The system shall automatically save user code progress, including unsaved drafts, enabling resumption of work without data loss.* | *Auto-save triggers every 10 seconds or upon code execution.* |
| ***Version History Viewer*** | ***FR5.7:*** *The system shall maintain multiple versions of each user’s submitted code, allowing comparison of changes and improvements.* | *A maximum of 10 recent versions per problem are stored.* |
| ***Integrated Practice Loader*** | ***FR5.8:*** *The system shall integrate with the Question Generator Module to load assigned coding problems directly into the workspace.* | *Questions load dynamically based on user’s learning path or instructor assignment.* |
| ***AI Tutor Integration*** | ***FR5.9:*** *The system shall connect with the AI Tutor Module to provide in-editor learning guidance, code explanations, and hints.* | *Tutor hints appear contextually based on user code behavior.* |
| ***Collaborative Coding Mode*** | ***FR5.10:*** *The system shall allow collaborative sessions between instructors and students, enabling shared editing and live feedback.* | *Collaboration is enabled only within active instructor-led classes.* |
| ***Secure Sandbox Environment*** | ***FR5.11:*** *The system shall execute user-submitted code in an isolated sandbox to prevent unauthorized system-level access or file manipulation.* | *Sandbox resource usage (CPU, memory) is monitored and limited per user session.* |

.

Module 6: Logic Building Module

|  |  |  |
| --- | --- | --- |
| ***Feature (derived from UI)*** | ***Functional Requirement (FR-ID: Statement)*** | ***Business Rule*** |
| ***Logic Learning Interface*** | ***FR6.1:*** *The system shall allow users to learn fundamental programming logic through simplified, non-technical explanations, visual examples, or guided lessons.* | *Content should be beginner-friendly and avoid direct programming syntax to focus on conceptual understanding.* |
| ***Logic Practice Exercises*** | ***FR6.2:*** *The system shall provide interactive exercises to test users’ logical thinking using step-by-step reasoning tasks.* | *Exercises should cover sequence, conditions, and loops without requiring code writing.* |
| ***Scenario-based MCQs*** | ***FR6.3:*** *The system shall present scenario-based MCQs that challenge users to predict the correct logical flow or output.* | *Each scenario must have one correct answer and a detailed explanation for learning reinforcement.* |
| ***Adaptive Difficulty*** | ***FR6.4:*** *The system shall automatically adjust question difficulty based on user performance.* | *The system should increase complexity only if the user consistently scores above 70% on previous exercises.* |
| ***Feedback and Hints*** | ***FR6.5:*** *The system shall display feedback and hints after each question to guide users toward correct reasoning.* | *Hints should not reveal the correct answer directly but provide conceptual guidance.* |
| ***Link with Coding Module*** | ***FR6.6:*** *The system shall map completed logic exercises to related programming examples in the coding module.* | *Each logic concept should correspond to at least one simple code example in Java to build conceptual continuity.* |

**Module 7: Personalized AI Tutor**

|  |  |  |
| --- | --- | --- |
| ***Feature (derived from UI)*** | ***Functional Requirement (FR-ID: Statement)*** | ***Business Rule*** |
| ***AI-driven Doubt Assistant*** | ***FR7.1:*** *The system shall allow users to ask programming-related questions and receive AI-generated explanations or solutions.* | *The AI must only respond to questions within the context of Java programming and learning content available on the platform.* |
| ***Context-aware Feedback*** | ***FR7.2:*** *The system shall analyze user code submissions and provide step-by-step feedback highlighting logical, syntactical, or structural errors.* | *The AI feedback must include the reason for the error and a suggested correction, without directly providing full solutions.* |
| ***Personalized Learning Suggestions*** | ***FR7.3:*** *The system shall recommend personalized lessons, topics, or exercises based on user mistakes and performance trends.* | *Recommendations should be generated using the user’s learning history, accuracy rate, and difficulty level of attempted questions.* |
| ***Hint Generation*** | ***FR7.4:*** *The system shall provide hints to help users solve a problem instead of giving direct answers.* | *Hints must be concise and progressively more detailed upon repeated user requests.* |
| ***Real-time Code Analysis*** | ***FR7.5:*** *The system shall evaluate user-written code in real-time and detect errors as they type.* | *Real-time feedback should only highlight the specific line or logic causing the issue, not alter or auto-correct the user’s code.* |
| ***Learning Progress Tracking*** | ***FR7.6:*** *The system shall monitor user interactions with the AI tutor to identify areas of improvement.* | *The collected data should be stored securely and used exclusively for learning personalization and analytics.* |

### Backend Process Requirements

|  |  |  |
| --- | --- | --- |
| ***Event*** | ***System Response / Backend Process*** | ***Condition / Business Rule*** |
| ***User registration request received*** | *System validates input → encrypts password using bcrypt → stores record in User\_Accounts → sends verification email.* | *Email must be unique; password ≥ 8 chars.* |
| ***User completes verification*** | *Token validated → activates user account → initializes default progress and profile in User\_Profiles.* | *Verification token expires in 15 minutes.* |
| ***User starts beginner learning module*** | *Backend fetches lesson content from Learning\_Content → initializes tracking record in Learning\_Progress.* | *Next topic unlocked only after previous one is completed.* |
| ***Quiz submission received*** | *Backend compares answers with answer key → stores score in Quiz\_Results → triggers evaluation update.* | *Pass score ≥ 60% to unlock next topic.* |
| ***Code submitted for evaluation*** | *Code sent to compiler microservice → executes in sandbox → output logged → AI Evaluation API called for feedback → stores report in Code\_Evaluations.* | *Execution limited to 5 seconds; memory ≤ 256MB.* |
| ***AI Evaluation completed*** | *AI feedback analyzed → updates Evaluation\_Summary → adjusts user’s accuracy metrics and weak areas in profile.* | *Feedback text length capped to 1000 tokens.* |
| ***Weak area detected*** | *Smart Question Generator fetches user’s past errors → ML model generates related new question set → saves in Recommended\_Questions.* | *At least 3 similar mistakes required to trigger generation.* |
| ***Instructor assigns a task*** | *Backend validates instructor role → stores assignment in Class\_Tasks → sends notifications to enrolled students.* | *Only verified instructors can create tasks.* |
| ***Student submits class task*** | *Submission stored in Submissions → auto-evaluated using AI Evaluation module → grading rubric applied → result stored in Grades.* | *Late submissions flagged after due date.* |
| ***Workspace code run triggered*** | *Code sent to containerized compiler service → backend retrieves output → returns to frontend.* | *Only Java code allowed; malicious commands blocked.* |
| ***Workspace session saved*** | *Code snapshot stored in Workspace\_Saves with timestamp → version control record updated.* | *Maximum 10 saved sessions per user.* |
| ***User interacts with AI Tutor*** | *Query routed to AI Tutor microservice → context from session loaded → AI generates personalized explanation → logs stored in Tutor\_History.* | *Conversation context retained for 7 days.* |
| ***Logic-building question attempted*** | *Backend executes user input → compares with reference logic → generates feedback using AI → stores attempt in Logic\_Records.* | *Minimum one valid output required for evaluation.* |
| ***Instructor views class analytics*** | *Backend aggregates Submissions, Grades, and Activity\_Logs → generates analytics JSON response.* | *Only accessible to class creator.* |
| ***User progress updated*** | *System calculates updated completion % → adjusts level, badges, and recommended content → stores updated metrics in User\_Progress.* | *Updates triggered only after successful lesson or quiz submission.* |
| ***Logout request received*** | *Backend invalidates JWT token in Active\_Sessions → clears temporary cache data.* | *Logs retained for audit for 30 days.* |

## Non-Functional Requirements

### Reliability

The reliability of **Codable** is ensured through robust architecture, consistent testing, and automated recovery mechanisms. The system is designed to achieve a **Mean Time Between Failures (MTBF)** of at least **1,000 hours**, ensuring stable operation during continuous use. A failure is defined as any system event that disrupts user activity such as loss of unsaved progress, inability to execute code, or interruption in AI-based feedback processing.

To minimize failures, all major modules (learning, instructor management, AI evaluation, and code editor) are independently containerized and monitored for exceptions. Error detection strategies include logging and real-time anomaly tracking through monitoring tools. In case of minor software faults, automatic retries and recovery scripts are executed; in critical cases, fail-safe mechanisms temporarily switch to cached data to preserve user progress.

Instructors and learners are notified of system disruptions through alerts, ensuring transparency and controlled recovery. Regular backups, version control, and continuous testing further enhance reliability, maintaining consistent system behavior even under heavy concurrent usage.

### Usability

**USE-1:** The system shall provide a clean, intuitive, and responsive user interface to ensure ease of navigation for both learners and instructors.

**USE-2:** The system shall allow new users to complete onboarding (account setup and feature walkthrough) within **5 minutes** without requiring prior technical knowledge.

**USE-3:** The system shall include an integrated tutorial and guided tooltips to help learners understand how to use the coding workspace, AI feedback, and learning modules effectively.

**USE-4:** The system shall provide meaningful error messages and recovery suggestions when a user encounters a syntax, logic, or compilation error in the coding environment.

**USE-5:** The interface shall maintain consistent design patterns, color schemes, and component placement across all modules (learning, instructor, and AI feedback) for familiarity and reduced cognitive load.

**USE-6:** The system shall support accessibility features such as adjustable text size, high contrast mode, and keyboard navigation to accommodate users with visual or motor impairments.

**USE-7:** The system shall minimize the number of clicks required to perform key actions (e.g., starting a lesson, submitting code, or viewing feedback) to ensure interaction efficiency.

**USE-8:** The system shall automatically save user progress and restore the last active session upon login to prevent data loss and enhance user experience.

### Performance

**PER-1:** 95% of system pages, including dashboards, lessons, and instructor panels, shall load completely within **4 seconds** over a **20 Mbps or faster** internet connection.

**PER-2:** The AI evaluation engine shall generate syntax and logic feedback for user-submitted code within **5 seconds** for programs up to **150 lines of code**.

**PER-3:** The system shall support at least **500 concurrent users** performing learning, code submission, and feedback retrieval operations without noticeable performance degradation.

**PER-4:** Code execution and output generation in the integrated coding workspace shall complete within **3 seconds** for standard test cases under normal load conditions.

**PER-5:** User progress updates, such as quiz results or coding assessments, shall be reflected in the database and visible on the user dashboard within **2 seconds** of submission.

**PER-6:** The instructor module shall generate performance reports for a class of up to **100 students** within **10 seconds** of request initiation.

**PER-7:** The system shall maintain a server uptime of **99.5%,** ensuring reliable access for learners and instructors at all times.

**PER-8:** Background processes such as AI model training updates or data synchronization shall not affect the response time of user-facing components.

### Security

**SEC-1:** The system shall ensure that only authorized users can access protected resources such as user dashboards, instructor panels, and progress data.

**SEC-2:** All user data, including code submissions and learning progress, shall be protected from unauthorized viewing, modification, or deletion.

**SEC-3:** The system shall maintain the confidentiality and integrity of all data transmitted between client and server, preventing interception or tampering.

**SEC-4:** User roles (learner, instructor, admin) shall be strictly enforced to prevent privilege escalation or unauthorized access to restricted functionalities.

**SEC-5:** The system shall detect and restrict repeated unauthorized access attempts, ensuring protection against brute-force or automated attacks.

**SEC-6:** Sensitive information, such as performance analytics and AI evaluation data, shall be stored and handled securely to prevent misuse or exposure.

**SEC-7:** The system shall log all user and administrative activities for traceability, ensuring accountability in case of security incidents.

**SEC-8:** The platform shall maintain resilience against data corruption or injection attacks, ensuring system stability under all operational conditions.

**SEC-9:** Access to internal system operations and AI models shall be restricted to designated administrative entities with verified authorization.

## External Interface Requirements

### User Interfaces Requirements

**UI-1:** The user interface will follow a modern, minimalistic design consistent with the **Codable** platform’s theme, featuring blue and white as primary colors with accent tones for highlights and buttons.

**UI-2:** All screens will support a minimum resolution of **1366x768 pixels** and will be fully responsive across desktop, tablet, and mobile devices for accessibility.

**UI-3:** Navigation will remain consistent throughout the platform, with main sections including **Dashboard, Learn, Practice, AI Tutor, Workspace, and Profile.**

**UI-4:** Shortcut keys will be provided for frequently used actions (for example, **Ctrl+S** to save code, **Ctrl+Enter** to run code, and **Ctrl+K** to open the AI assistant).

**UI-5:** Each learning and coding interface will include **help icons, tooltips, and interactive walkthroughs** to guide new users.

**UI-6:** The user interface will comply with **internationalization (i18n)** and **localization (l10n)** standards, allowing for future **multi-language support.**

**UI-7:** The platform will use **consistent UI components** (buttons, cards, modals, inputs) across modules for uniform user experience.

**UI-8:** The **Monaco Editor** will be integrated for code writing, featuring syntax highlighting, auto-completion, and error highlighting.

**UI-9:** The system will include **dark and light mode options** to enhance usability and user comfort.

**UI-10:** Visual feedback such as **loading indicators, progress bars, and success/error alerts** will be displayed for all user actions.

### Software interfaces

**SI-1: MongoDB Database**  
**SI-1.1:** The system shall connect to the **MongoDB** (version 6.0 or later) or MySQL database through the **Mongoose ORM** or Prisma for storing and retrieving user data, progress, assessments, and generated questions.

**SI-2: Node.js Runtime Environment**  
**SI-2.1:** The backend shall operate in a **Node.js (v18+)** environment using **Express.js** to manage API routing, request handling, and integration with frontend and AI modules.

**SI-3: AI Evaluation Service**  
**SI-3.1:** The system shall integrate with **OpenAI API (GPT-based models)** or a fine-tuned **LLM model (e.g., CodeT5+)** through HTTPS requests for AI-based code evaluation, feedback generation, and tutoring.

**SI-4: Frontend Framework**  
**SI-4.1:** The user interface shall communicate with the backend through RESTful APIs using **React.js (v18+)** for dynamic content rendering and interaction handling.

**SI-5: Code Editor Integration**  
**SI-5.1:** The platform shall integrate the **Monaco Editor** (used in VS Code) via its JavaScript API for real-time code writing, syntax highlighting, and error detection.

**SI-6: Cloud Storage and Deployment**  
**SI-6.1:** The system shall integrate with **AWS S3** for file and code storage and **AWS EC2 / Elastic Beanstalk** for deployment and hosting of backend and frontend services.

**SI-7: Authentication and User Management**  
**SI-7.1:** The system shall utilize **JWT (JSON Web Tokens)** for secure user authentication and **bcrypt** for password encryption.

**SI-8: Analytics and Tracking Tools**  
**SI-8.1:** The system shall interface with **Google Analytics API** (or an equivalent tracking system) for monitoring user activity and engagement.

**SI-9: Email and Notification Services**  
**SI-9.1:** The platform shall use **Nodemailer** integrated with **SMTP** or a cloud email service (such as SendGrid) for sending verification emails, password resets, and instructor notifications.

**SI-10: Version Control and CI/CD Tools**  
**SI-10.1:** The project shall integrate with **GitHub** for version control and **GitHub Actions** for automated testing and deployment workflows.

### Hardware interfaces

No direct hardware interfaces are required beyond standard client devices (PCs, laptops, and mobile devices) and servers running Ubuntu 64-bit.

### Communications interfaces

**CI-1:** The system shall communicate with users through **email notifications** for account verification, password reset, and important updates such as class invitations or performance reports.

**CI-2:** The system shall use **HTTPS** protocol for all client–server communications to ensure secure data exchange.

**CI-3:** The system shall support **WebSocket-based real-time communication** for collaborative coding sessions, AI tutor chat interactions, and live instructor feedback.

**CI-4:** The system shall integrate with **SMTP servers** to handle outgoing emails such as verification links, progress summaries, and instructor announcements.

**CI-5:** The system shall be accessible through all major **web browsers** (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari) ensuring compatibility and consistent performance.

**CI-6:** The system shall log all communication-related activities (e.g., sent notifications, API requests) for monitoring and debugging purposes.

1. For further reading on mockup-driven functional requirement elicitation, see: [↑](#footnote-ref-1)