UNIVERSITY OF LEICESTER

School of Computing and Mathematical Sciences

CO7201 Individual Project

Preliminary Report

An AI Multiple-Choice Programming Exercises Tutor

Sai Raghava Vemuri

Srv10@student.le.ac.uk

249034443

Project Supervisor: Dr. Wentao Li
Principal Marker: Dr. Martina Palusa

Word Count: 1938
27-06-2025

DECLARATION
All sentences or passages quoted in this report, or computer code of any form whatsoever used and/or submitted at any stages, which are taken from other people's work have been specifically acknowledged by clear citation of the source, specifying author, work, date and page(s). Any part of my own written work, or software coding, which is substantially based upon other people's work, is duly accompanied by clear citation of the source, specifying author, work, date and page(s). I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this module and the degree examination as a whole.

Name: Sai Raghava Vemuri
Date: 27-06-2025

Contents

1. Aims and Objectives

1.1 Motivation

The growing demand for accessible, effective, and engaging programming education highlights the need for scalable learning tools that adapts to individual learning styles. Existing modern learning environments, such as online educational platforms or static video tutorials, often lack real-time feedback and adaptability. These gaps can significantly slow down a student's learning curve and reduce engagement.

This project addresses these limitations by offering a self-paced, interactive, and AI-enhanced multiple-choice tutor that helps students learn programming in a more personalized and convenient manner [1]. Unlike traditional educational system like classroom learning, this is accessible anytime and from anywhere, providing comfort and flexibility to users, whether they are full-time students, professionals upskilling after work, or self-taught programmers preparing for interviews.

By integrating adaptive question difficulty, AI-generated explanations, and gamified elements, this system aims to simulate a real-time intelligent learning assistant enhancing both comprehension and motivation [2].

1.2 Aim

The aim of this project is to design and develop an AI-powered web-based system that delivers programming Multiple Choice Questions (MCQ) quizzes, dynamically adapts to the learner's performance, and explains concepts using AI-generated feedback, thereby enhancing engagement, personalization, and learning outcomes. This project also offers user a self-paced, comfortable and flexible experience.

1.3 Objectives

Objectives	Corresponding Requirements
Build a web-based quiz system with structured MCQ workflows.	E1, E2, E3
Integrate GPT-4 API to generate dynamic questions and explanations.	E4, E5
Implement adaptive difficulty based on user's ongoing performance.	E6
Develop a user-friendly interface using Flask and HTML/CSS.	R1
Enable optional user profiles and session history logging.	R2, R3, R4
Add gamification features: reward points and performance tracking.	O1, O2

2. Requirements

2.1 Essential

These are the minimum features required for the core functionality of the system. These ensure that users can take programming quizzes, receive AI-generated explanations, and experience adaptive learning behaviour.

Id	Requirement	Description
E1	MCQ Presentation	Display questions and options to the user.
E2	Answer Checking	Validate user's answer and compute score.
E3	Topic Selection	Allow users to choose programming Languages or topics.
E4	Questions Generation	Use OpenAI API to generate MCQ's.
E5	AI Explanation	Use OpenAI API to generate explanations.
E6	Adaptive Difficulty	Modify question difficulty in real time based on performance.

2.2 Recommended

These features enhance the user experience, personalize learning, and increase engagement. While not strictly required for a functional product, they are expected in a polished system.

Id	Requirement	Description
R1	Web Interface	Flask-based frontend for user interaction.
R2	User Profiles	Track scores and progress across sessions.
R3	Recommendation Engine	Suggest content recommendation based on weak areas.
R4	Log incorrect	Store incorrect responses for future review.

2.3 Optional

These are advanced, innovative features that can significantly enhance the educational value and interactivity of the system. They are not required for the core system but can distinguish the project academically.

Id	Requirement	Description
O1	Reward System	Assign points for correct answers to boost motivation.
O2	Personalized Dashboard	Dynamic personalized dashboard to visualize progress.
O3	Chat Bot	Allows users to follow up and understand topics interactively.

3. Technical Specification

3.1 System Architecture

The project will follow a modular architecture, with clear separation of concerns between data handling, AI integration, core logic, and user interface:

- Frontend: HTML/CSS templates rendered via Flask.
- Backend: Python (Flask framework), responsible for quiz logic, AI integration, and session management.
- Database (optional for user profiles): SQLite or simple flat-file JSON storage.
- AI Component: OpenAI GPT-4o API for generating explanations and optionally generating MCQs.
- Data Storage: Initial question sets stored in Excel/CSV files, later expandable via AI.

3.2 Key Technologies

Technology	**Purpose**
Python	Core backend logic
Flask	Web framework
HTML/CSS	Frontend user interface
OpenAI GPT API	AI-powered explanation generation
Pandas	Data loading and manipulation
Matplotlib	Data visualization
SQLite/JSON	Persistent storage for profiles
Git/GitHub	Version control

3.3 Key Functional Modules

- Question Loader: Loads MCQs from Excel or CSV files.
- Quiz Engine: Random question selection, answer checking, scoring.
- AI Questions Module: Calls GPT API to generate MCQ dynamically.
- Adaptive Difficulty Module: Adjusts question complexity based on performance.
- AI Explanation Module: Calls GPT API to generate explanations dynamically.
- User Profile Module: Tracks user history and achievements.
- Gamification Module: Reward points system for motivation.
- Evaluation Module: Logs incorrect answers for later review.

3.4 Development Methodology

The project will be developed using an iterative, Agile approach, starting with a minimal viable product (MVP) that includes a command-line quiz system. This early prototype will establish core functionality, including MCQ delivery and AI-generated explanations via the OpenAI GPT API.

Once the CLI version is validated, the system will be transitioned into a web application using Flask, with a focus on building a clean user interface. New features such as adaptive difficulty, gamification, and user tracking will be added incrementally in weekly sprints.

Throughout the project, continuous testing and regular supervisor feedback will guide development. This flexible methodology ensures early results, manageable risk, and progressive refinement toward a polished final system.

4. Requirements Evaluation Plan

To ensure the system meets its functional and educational objectives, a multi-faceted evaluation approach will be adopted. This includes technical testing, AI output validation, and user-focused usability assessments.

4.1 Evaluation Criteria

The success of the project will be measured against the following criteria:

Aspect	Evaluation Method
Functionality	Verify that all core features (quiz flow, AI explanations, scoring) work as required.
Accuracy	Ensure that answer checking and adaptive difficulty logic operate reliably.
AI Quality	Manually review AI-generated questions and feedback for clarity, correctness, and relevance.
Usability	Gather peer feedback on user interface clarity, ease of use, and engagement.
Stability	Test system response under incorrect inputs or heavy usage conditions.
Engagement	Evaluate effectiveness of reward points and gamification in motivating users.
Extendibility	Assess how easily new features (topics, languages, dashboard) can be added.

4.2 Testing Methods

- Unit Testing: Each function/module will be tested individually.
- Integration Testing: Test overall system flow end-to-end.
- Usability Testing: Conduct small-scale user testing with peers or volunteers.

- AI Output Review: Evaluate AI-generated explanations for correctness, clarity, and helpfulness.
- Supervisor Review: Periodic meetings with the supervisor for feedback.
- Self-assessment: Continuous tracking against milestones and objectives.

4.3 Evaluation Involvement

- Developer (Myself): Responsible for all technical testing, validation, and debugging.
- Supervisor: Will provide ongoing guidance, milestone review, and feedback on functionality and quality.
- Test Users: A small group of peer students will provide informal feedback on usability and educational value.

5. Background Research and Reading list

This project intersects multiple domains, including artificial intelligence, education technology, natural language processing, and web development. Understanding the theoretical and technical foundations of these areas is essential to guide the design of a robust and educationally effective system. The reading list includes foundational books and online API documentations for practical guidance.

5.1 Background Topics

- **Intelligent Tutoring Systems (ITS):** These are computer-based learning environments that provide immediate and personalized instruction. Key principles such as adaptive learning, learner modelling, and formative feedback inform the project's architecture [2].
- **Natural Language Processing (NLP):** Enables AI to generate human-readable explanations. Recent advancements through large language models (LLMs) like GPT-4 make it possible to offer dynamic, context-aware educational feedback [7].
- **Adaptive Learning Algorithms:** Techniques such as rule-based adaptation, reinforcement learning, or performance-based adjustment help in customizing the difficulty level to each user's learning curve [6].
- **Gamification in Education:** Incorporating elements like rewards, points, and leaderboards has been shown to boost motivation and engagement in online learning platforms [4].
- **MCQ Design Principles:** Effective multiple-choice questions must test conceptual understanding, not just memorization. Balancing difficulty, clarity, and distractor design is key to educational value.

5.2 Reading List

The reading list includes foundational books such as *Artificial Intelligence: A Modern Approach* by Russell [1] and Norvig and *Machine Learning* by Mitchell [2], offering core

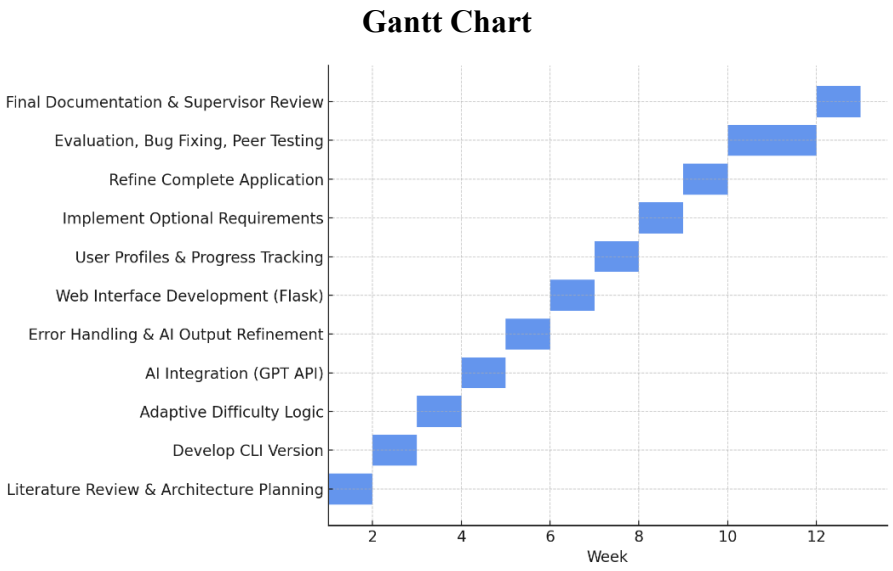
principles in AI and learning algorithms that support this project's technical foundation. Recent academic papers by Huang et al., Khan et al., and Chen et al. contribute contemporary insights into adaptive programming systems, reinforcement learning, and AI-generated educational content directly relevant to the system's explanation and personalization modules [4][5][6].

Additionally, practical guidance is drawn from online resources such as the OpenAI API documentation[7] and Flask web framework[8] tutorials. Educational platforms like Coursera's AI for Everyone[9] and research archives like ACM Digital Library ensure both implementation-level understanding and exposure to current academic discourse in AI education[10].

6. Time-plan and Risk Plan

6.1 Time Plan

The project is structured across a 12-week timeline, with each week focused on a specific module or development stage. The plan ensures progressive implementation, testing, and final documentation, as illustrated in the Gantt chart.



Timeline	Planned Activity
Week 1	Literature review and architecture planning
Week 2	Development of CLI-based quiz prototype
Week 3	Implementation of adaptive difficulty logic
Week 4	Integration of OpenAI GPT API for explanations
Week 5	Testing and refining AI outputs and error handling
Week 6	Development of Flask-based web interface
Week 7	User profile and session tracking
Week 8	Implement Optional Requirements

Week 9	Refine Complete Application
Week 10	Evaluation, Bug Fixing, Peer Testing and feedback
Week 11	
Week 12	Final Documentation & Supervisor Review

6.2 Risk Plan

To ensure the project stays on track, a proactive risk management approach will be followed. Regular risk reviews will be conducted weekly to assess technical and integration challenges. Each identified risk will be assigned a mitigation plan, such as fallback explanations for AI outages or simplified UI alternatives. Backup plans include prioritizing essential features over optional ones to manage scope under time constraints.

Risk	**Impact**	**Mitigation Strategy**
GPT API limitations	Medium	Use static fallback explanations if API fails or limit usage.
Time overrun	High	Prioritize core features (MVP-first), optional features only if time allows.
UI/UX issues	Medium	Gather early feedback, use simple and proven design patterns.
Unexpected bugs	Medium	Conduct weekly testing and maintain modular code.
Feature creep	High	Stick to clearly defined requirement tiers (essential, recommended, optional).

This structured approach will help address issues early, avoid scope creep, and ensure a stable and functional system is delivered within the project timeline.

7. References

[1] Russell, S. J., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.
[2] VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, 46(4), 197–221. https://doi.org/10.1080/00461520.2011.611369
[3] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.
[4] Huang, T.-H., Lin, Y.-W., & Cheng, H.-N. (2021). An adaptive programming practice system with feedback based on learning style and performance. *Educational Technology & Society*, 24(1), 180–192.
[5] Khan, M., Ahmed, F., & Sarwar, M. (2023). Leveraging GPT models for intelligent tutoring: A case study in MCQ explanation generation. *Journal of Artificial Intelligence in Education*, 33(2), 311–326.
[6] Chen, W., Wu, Y., & Zhao, L. (2022). Adaptive learning paths using reinforcement learning in intelligent tutors. *IEEE Transactions on Learning Technologies*, 15(1), 78–88. https://doi.org/10.1109/TLT.2021.3055893
[7] OpenAI. (n.d.). *API documentation*. https://platform.openai.com/docs
[8] Flask Documentation. (n.d.). *Flask web framework*. https://flask.palletsprojects.com/en/latest/
[9] Ng, A. (2019). *AI for Everyone* [MOOC]. Coursera. https://www.coursera.org/learn/ai-for-everyone
[10] ACM Digital Library. (n.d.). https://dl.acm.org