CodeQuest: A Game-Based Learning System to Enhance C++ Programming Education

[1]Nur Aina Liana Kamarulnazri, [2]Nor Shahida Mohamad Yusop

[1 2] Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA

DOI: https://dx.doi.org/10.47772/IJRISS.2025.925ILEIID000071

Received: 23 September 2025; Accepted: 30 September 2025; Published: 07 November 2025

ABSTRACT

Learning to program in C++ poses significant challenges for students, particularly beginners, due to the abstract nature of concepts and lack of engaging, adaptive feedback in traditional educational settings. This paper introduces CodeQuest, a game-based learning system designed to enhance C++ programming education at the Faculty of Computer and Mathematical Sciences (FSKM), Universiti Teknologi MARA (UiTM). The system was developed using the Rapid Application Development (RAD) approach, with requirements gathered from both educators and students. The system's primary features include user management, learning materials, interactive exercises with immediate feedback, and progress monitoring. This system incorporates gamification elements such as badges and achievements, leaderboards, and points and rewards systems. CodeQuest aims to increase student engagement and provide educators with a clear view of student progress, enabling them to offer timely support

Keywords: game-based learning, C++ programming, rapid application development

INTRODUCTION

Introductory programming courses are fundamental to science, technology, engineering, and mathematics (STEM) education, equipping students with fundamental computational thinking and problem-solving abilities. This programming courses, often taught using Python, C, or C++ during the first year of undergraduate studies. However, many students, particularly beginners, have considerable challenges in mastering syntax, debugging errors, and constructing logical algorithms (Sobral, 2021; Lovrenčić, 2023). Among these, algorithm design emerges as the most persistent obstacle, while advanced concepts such as pointers and file handling further compound difficulties—even for those with prior programming exposure (Islam, 2019).

Gamification has emerged as a promising tool to enhance engagement and mitigate learning barriers in programming education. By integrating elements like instant feedback, progress tracking, and interactive problem-solving, instructors can foster motivation and reduce the frustration associated with abstract concepts. Cheah (2020) emphasizes the importance of timely support in this context, noting that delayed feedback in courses like C++ programming often demoralizes learners. Gamified environments address this by providing real-time guidance, enabling students to iteratively refine their code and grasp complex topics through trial and error. Such approaches not only sustain interest but also align with the dynamic, practice-driven nature of programming.

At Faculty of Computer and Mathematical Sciences (FSKM), Universiti Teknologi MARA, the challenges of teaching C++ programming are particularly pronounced. The C++ programming course employs a hybrid model combining two-hour in-person laboratory sessions with two-hour online lectures, yet it continues to report one of the highest failure rates among programming courses. Lecturers attribute this to the inherent complexity of C++ syntax and abstract concepts like memory management. Suhaimi's (2024) research reinforces these observations, revealing that students at UiTM achieve better comprehension through individualized learning methods. For instance, personalized practice on functions and data structures has proven effective in reinforcing foundational skills, suggesting that adaptability in teaching strategies could alleviate recurring difficulties. Additionally, Alghamdi (2025) reports that students often struggle with three

core areas: grasping fundamental programming principles, translating problem-solving logic into syntactically correct code, and identifying errors during debugging. These challenges are exacerbated by the rapid pace of coursework, which leaves limited room for remedial practice. To address these gaps, experts advocate for a blended approach combining gamification's engagement benefits with personalized instruction. By fostering incremental skill development and providing scaffolded support, educators can empower students to navigate the steep learning curve of programming—ultimately transforming initial struggles into long-term competency.

Problem Statement

At FSKM, Programming II (CSC404) stands as a mandatory programming course for computing students, designed to instill foundational skills in coding and computational logic. Despite its essential role, many students struggle to achieve high grades, as evidenced by recent performance data. A survey of CDCS266 students revealed that only 18 out of 22 participants earned grades between A+ and B+, while four scored below B+. This disparity underscores systemic challenges in course delivery, which students attribute to three primary obstacles: disengagement in traditional teaching methods, insufficient adaptive feedback, and difficulties in mastering abstract programming concepts.

The first challenge stems from low engagement in conventional pedagogical approaches. The course employs a hybrid model combining online lectures and physical laboratory sessions, yet students report that static resources—such as e-books, lecture notes, and pre-recorded videos—fail to foster active learning. According Cheah (2020), textbook-driven methodologies are ill-suited for teaching dynamic subjects like programming. Passive consumption of materials often leaves students unprepared to tackle coding problems independently, exacerbating frustration and disconnection from course objectives. Without interactive or immersive elements, learners struggle to internalize concepts, perpetuating a cycle of underperformance.

A second challenge lies in the absence of adaptive feedback mechanisms. Traditional classrooms lack systems to provide real-time, personalized guidance during coding exercises, leaving students without timely corrections or support. Marwan's (2022) research highlights the transformative potential of Adaptive Immediate Feedback (AIF) tools, which enable learners to iteratively refine their work based on targeted insights. In his study, AIF not only improved task completion rates but also boosted confidence, with participants praising its role in clarifying errors and motivating progress. Such findings contrast sharply with FSKM's current framework, where delayed or generic feedback hinders mastery of complex topics like loop structures or memory management.

Finally, the abstract nature of C++ programming poses a significant challenge for beginners. The language's evolving complexity, as noted by Cyganek (2022), demands strong logical reasoning and familiarity with modern features such as templates or object-oriented principles—skills often absent in novices. Students with limited coding experience report feeling overwhelmed by the leap from theoretical concepts to practical application, particularly when tasked with debugging or designing algorithms. This cognitive gap highlights the need for pedagogical strategies that bridge abstraction and practicality, such as visual aids or scaffolded problem-solving exercises. Without such interventions, the course risks alienating learners, further entrenching the perception of C++ as an insurmountable challenge.

Objectives

The objectives of this project are:

1. To design intuitive dashboard for educators to track students' performance.
2. To develop interactive C++ teaching and learning platform.
3. To enhance student engagement and motivation in learning C++.

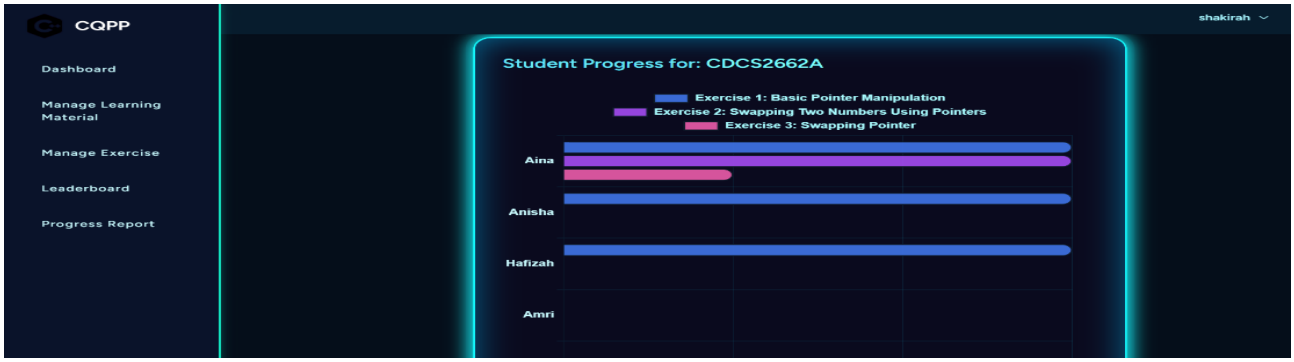
PRODUCT DESCRIPTION & METHODOLOGY

Product Description

The key features of Code Quest are:

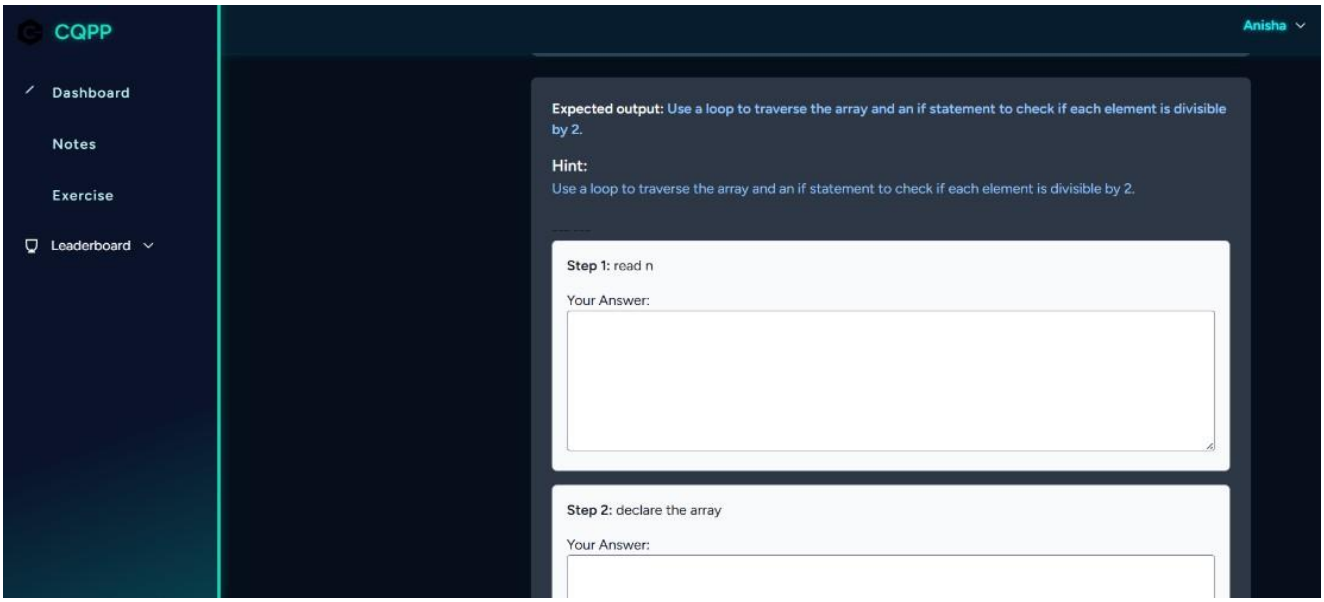
- **User Management and Dashboards** - The system employs a role-based architecture to separate dashboards for students and lecturers. Students can track their progress, view performance analytics, and submit assignments. Lecturers can monitor class progress through a comprehensive dashboard. This feature enables them to identify students who may need additional support and provides a data-driven approach to teaching.

Figure 1: Lecturer dashboard – allowing lectures to monitor student progress



- **Learning Materials Repository** – Lecturers can upload tutorials, code examples, and multimedia content through an intuitive content management system (CMS), complete with version control to track updates and maintain consistency. Students gain access to downloadable resources, curated reading lists, and modular tutorials that adapt to diverse skill levels.
- **Interactive Coding Exercises** - Exercises provide step-by-step guidance, and an integrated compiler offers real-time syntax checks, error diagnostics, and instant feedback. This adaptive feedback mechanism allows students to correct errors instantly and reinforces learning through practice. The exercises are self-paced, allowing students to learn at their own speed.

Figure 2: Answer *e*xercise *p*age - students can attempt coding exercises with step-by-step guidance provided



- **Gamification Elements** - The platform includes a reward system with badges and achievement points for completing milestones. A leaderboard fosters healthy competition among students, and a planetary

progression metaphor provides a visual representation of their learning journey. This system transforms the often-frustrating process of learning to code into an interactive and rewarding game.

Figure 3: Interactive student dashboard – using avatars to show learning progress

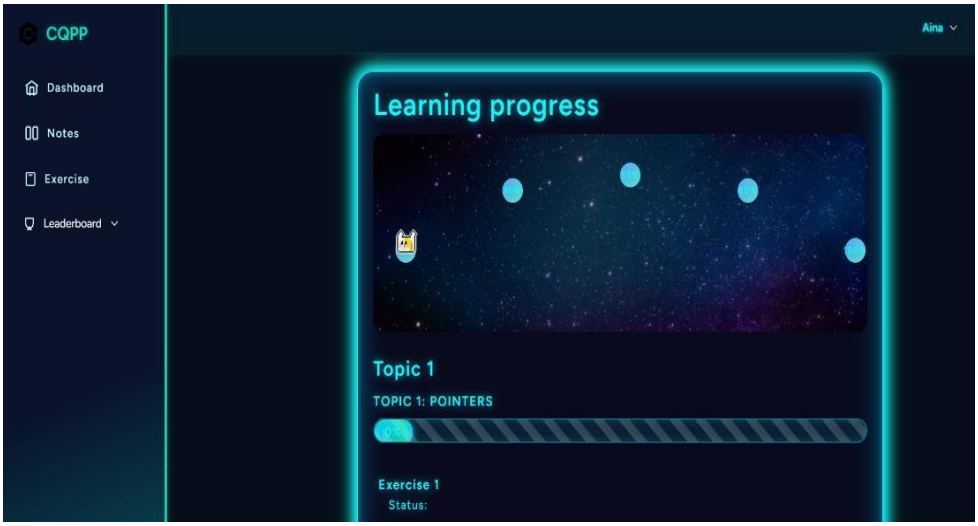
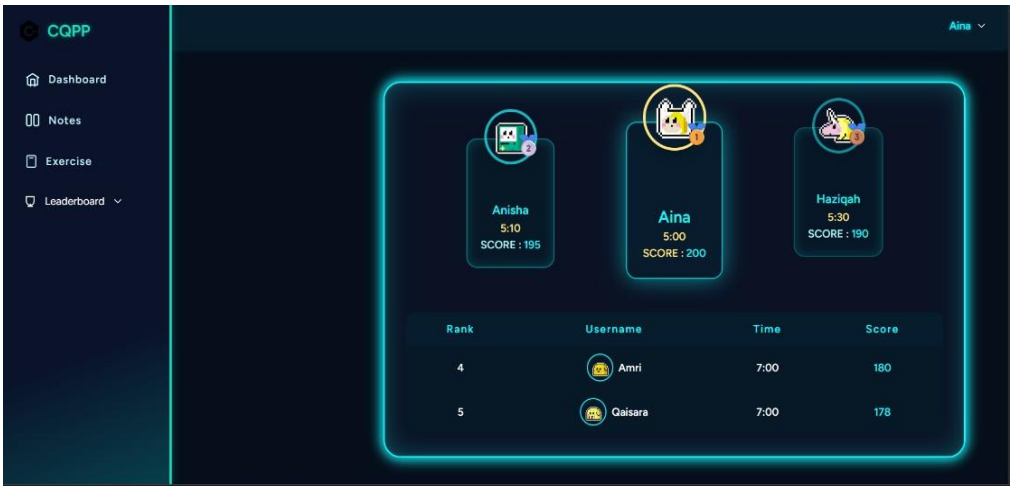


Figure 4: Overall Leaderboard - overall ranking of students within the same class along with their interactive avatars



METHODOLOGY

The CodeQuest was developed following the Rapid Application Development (RAD), which emphasizes an iterative and agile process. This methodology allowed for continuous refinement based on user feedback. The development process was structured into three phases:

Phase 1: Preliminary Investigation and Requirement Analysis – This phase began with a comprehensive literature review to identify the core challenges in C++ programming education. We conducted an online survey and interview s with students and lecturers at FSKM to gather specific requirements. An analysis of existing platforms like Sololearn and Codecademy helped to define key functionalities. The collected requirements were modeled in a use case diagram using StarUML to outline the system's core features.

Phase 2: Iterative Prototyping and Testing – This phase involved three cycles of user-centered design and prototyping:

- Iteration 1: A mid-fidelity prototype was created using PowerPoint to visualize the student and lecturer interfaces. Feedback from initial interviews was used to refine the user experience.

- Iteration 2: Adaptive features were integrated, and the system's domain class diagram was finalized. The prototype was refined based on a second round of feedback.
- Iteration3: The system transitioned to a full-scale development using Laravel and was coded in Visual Studio Code. This final prototype was evaluated through user testing to validate its usability and effectiveness.

Phase 3: Refinement and Development – User feedback from the testing phases was systematically incorporated to enhance interactivity, specifically by refining gamification elements and finalizing the database structure. The completed system was deployed via Nixpacks, ensuring accessibility for all users.

POTENTIAL FINDINGS AND COMMERCIALISATION

Potential Findings

The preliminary findings from CodeQuest highlight the significant impact of gamification on student engagement and motivation in C++ education. The real-time leaderboard and interactive progress map have been shown to encourage active participation, leading to a deeper understanding of programming concepts. The platform's interactive nature allows students to learn at their own pace and caters to diverse learning styles, a critical element in programming education. The embedded immediate feedback mechanisms also represent a significant improvement over traditional methods, enabling students to quickly identify and correct errors.

Commercialisation

CodeQuest holds significant market potential as a game-based learning platform tailored for C++ programming, addressing both educational and professional development needs. Its primary customer segments include programming novices, intermediate learners, and educational institutions seeking interactive curriculum tools, as well as game development enthusiasts leveraging C++ in industry-standard engines like Unreal or Unity. The platform's unique value proposition lies in its immersive gamification framework—avatars, planetary progression systems, and leaderboards—that transforms abstract programming concepts into engaging challenges. Unlike traditional tutorials, CodeQuest combines real-time coding feedback with self-paced learning, while offering lecturers granular progress analytics to identify knowledge gaps. This dual focus on engagement and pedagogy positions it as a versatile tool for academia and hobbyist developers alike.

To maximize reach, CodeQuest will deploy via a web-based platform with future expansion into mobile apps (iOS/Android), ensuring accessibility across devices. Strategic GitHub integration will foster community-driven content sharing and open-source collaboration, enhancing platform utility. Revenue will be generated through tiered subscriptions for advanced courses, micro-credentials for module completion, and enterprise licenses for institutions. Partnerships with game engine developers (e.g., Unity Technologies) and universities will ensure curriculum relevance and provide cross-promotional opportunities, while in-house C++ experts and instructional designers will maintain academic rigor. Scalable cloud hosting and a dedicated support team will ensure reliability as user bases grow.

The cost structure prioritizes sustainable growth, with investments in game engine licensing, cloud infrastructure, and dynamic content creation. While initial development and maintenance costs are substantial, recurring revenue from subscriptions and institutional licenses will ensure long-term viability. By aligning cutting-edge gamification with practical skill-building, CodeQuest is poised to disrupt programming education, offering a scalable solution that bridges classroom instruction and industry demands.

NOVELTY AND RECOMMENDATIONS

The proposed game-based learning system introduces an innovative pedagogical framework that distinguishes it from conventional programming education tools. While existing platforms often rely on passive, formulaic exercises (e.g., multiple-choice questions or code rearrangement), this system mimics human instruction through **structured, adaptive guidance**, enabling learners to iteratively build code while deepening conceptual mastery. A key innovation lies in its **real-time API compiler integration**, which provides instant

feedback on syntax, logic, and efficiency—transforming static exercises into dynamic problem-solving experiences. This approach not only replicates instructor-led mentorship but also empowers students to experiment, debug, and refine solutions autonomously, bridging the gap between theoretical knowledge and practical application.

However, the system's current limitations highlight opportunities for refinement. The compiler's inability to tokenize diverse coding responses—particularly for problems with multiple valid solutions—restricts its evaluative precision. Future recommendations should prioritize integrating **advanced parsing algorithms** capable of recognizing semantic equivalence and supporting line-by-line error diagnosis. Additionally, expanding language compatibility to include Python, Java, and PHP would broaden its applicability across academic curricula and industry domains, fostering interdisciplinary relevance.

ACKNOWLEDGEMENTS

The authors wish to thank Dr Shakirah Hashim and the students of CDCS266 program for their invaluable contributions to this project.

REFERENCES

1. Alghamdi, M. (2025). Dealing with Coding Challenges Through Digital Platforms: Assessing Their Effectiveness in Skill Development. CLEI Electronic Journal, 28(1), 9- 1.
2. Cheah, C. S. (2020). Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. Contemporary Educational Technology, 12(2), 1–14. https://doi.org/10.30935/cedtech/8247
3. Cyganek, B. (2022). Modern C++ in the era of new technologies and challenges – why and how to teach modern C++? Proceedings of the 17th Conference on Computer Science and Intelligence Systems, FedCSIS 2022, 35–40. 116 https://doi.org/10.15439/2022F308
4. Gananjaya, I., Chandra, J. O. T., Christanto, J. F. A., Widianto, M. H., & Audrey, J. (2022). "a Lone Burglar" Stealth Game Development Using Rapid Application Development. 2022 4th International Conference on Cybernetics and Intelligent System, ICORIS 2022. https://doi.org/10.1109/ICORIS56080.2022.10031499
5. Islam, N., Shafi Sheikh, G., Fatima, R., & Alvi, F. (2019). A Study of Difficulties of Students in Learning Programming. Journal of Education & Social Sciences, 7(2), 38–46. https://doi.org/10.20547/jess0721907203
6. Lovrenčić, S., & Sekovanić, V. (n.d.). How Well Students Perceive Their Understanding of Logic Programming Course Content?
7. Marwan, S., Akram, B., Barnes, T., & Price, T. W. (2022). Adaptive Immediate Feedback for Block-Based Programming: Design and Evaluation. IEEE Transactions on Learning Technologies, 15(3), 406–420. https://doi.org/10.1109/TLT.2022.3180984
8. Sobral, S. R. (2021). Teaching and learning to program: Umbrella review of introductory programming in higher education. Mathematics, 9(15). https://doi.org/10.3390/math9151737
9. Suhaimi A., Kapi, A., Y., Hasmy, H., Jabal, M., F., A., (2024). SPARK: Simplified Practices, Analogies, and Resources for Knowing C++ Functions. International Jasin Multimedia & Computer Science Invention and Innovation Exhibition. https://ir.uitm.edu.my/id/eprint/94395/1/94395.pdf