The Development of Game-Based Learning System to Enhance C++ Programming Education

Nur Aina Liana Kamarulnazri, *Nor Shahida Mohamad Yusop

Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Shah Alam Selangor

***Corresponding author**

DOI: https://dx.doi.org/10.47772/IJRISS.2025.922ILEIID0028

Received: 22 September 2025; Accepted: 30 September 2025; Published: 22 October 2025

ABSTRACT

Learning to program in C++ presents considerable 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. CodeQuest main features include interactive exercises with immediate feedback, progress monitoring dashboards, a learning materials repository, and various gamification elements such as badges and achievements, leaderboards, points and rewards systems. The system aims increase student engagement and provide educators with a data-driven view of student performance, enabling them to offer timely and targeted support.

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

INTRODUCTION

Introductory programming courses are a fundamental component of the science, technology, engineering, and mathematics (STEM) curriculum in higher education, providing students with their first exposure to computational thinking and programming languages (Sobral, 2021). While languages such as Python and C++ are commonly taught to first-year undergraduates, a significant number of students encounter persistent difficulties that impede their progress and mastery of the subject (Islam, 2019). The challenges are particularly pronounced in C++, where students report having difficulty in understanding syntax, debugging code, and, most critically, developing algorithms to solve problems. Moreover, advanced concepts like pointers and file handling present additional challenges, even for individuals with prior programming experience.

The Faculty of Computer and Mathematical Sciences (FSKM) at UiTM has observed that its mandatory Programming II (CSC404) course, which uses a hybrid teaching model of online lectures and in-person laboratory sessions, has one of the highest failure rates among all programming courses offered. Performance data from a student survey further highlights a systemic issue, with a notable disparity in grades that emphasizes the challenges inherent in the course's current delivery methods (Suhaimi, 2024). Alghamdi (2025) identifies three primary challenges that contribute to student underperformance: disengagement from traditional teaching, a lack of adaptive feedback, and the difficulty of mastering abstract concepts.

The first challenge arises from the disengaging nature of conventional pedagogical approaches. The current hybrid model, while leveraging technology, relies on static resources such as e-books and pre-recorded videos that fail to foster active and immersive learning. Cheah (2020) indicates that textbook-driven methodologies are often not well-suited for the dynamic, hands-on nature of programming. The passive consumption of these materials can result in students being unprepared to independently address coding problems, which can result in frustration and detachment from the learning objectives.

Secondly, the existing framework suffers from an absence of adaptive feedback mechanisms. Traditional classroom settings lack the systems to provide real-time, personalized guidance during coding exercises. Research has demonstrated that Adaptive Immediate Feedback (AIF) tools are transformative, enabling learners to iteratively refine their work based on targeted insights, thereby improving task completion rates and boosting confidence (Marwan, 2022). The delayed or generic feedback prevalent in the current course delivery hinders the mastery of complex topics and contrasts sharply with the potential of AIF to clarify errors and motivate continuous progress.

Finally, the abstract nature of C++ presents a significant cognitive gap for beginners. As a language with evolving complexity, C++ demands strong logical reasoning and familiarity with advanced features that are often beyond the grasp of novices. Students with limited coding experience report feeling overwhelmed by the transition from theoretical concepts to practical application, particularly when faced with debugging or algorithmic design tasks (Cyganek, 2022). This highlights a pressing need for a pedagogical strategy that bridges this cognitive gap with interventions such as visual aids and scaffolded problem-solving exercises.

In recent years, information technology (IT) has markedly improved programming education through multiple dimensions. Digital tools for interactive accessibility and engagement have increased knowledge retention rates in programming courses (Alshammari, 2024; Alghamdi, 2025). According to Asgari (2024), mobile learning tools have facilitated 85% of programming students to effectively visualize abstract programming concepts using simulations and visual debugging interfaces. In a different study, Tsai (2023) found that online peer-facilitated learning models enhance the programming skill development by 31% when compared to solo learning. This improvement is measurable in the areas of code optimization capabilities and problem-solving strategies. Additionally, by incorporating gamification elements into teaching and learning programming courses, students can be actively engaged in their learning.

While information technology has improved programming education by increasing knowledge retention and enhancing the visualization of abstract concepts through digital tools and mobile learning, there remains a specific need for a targeted, localized, and engaging system to address the challenges at FSKM. The project was thus initiated with three core objectives: to design an intuitive dashboard for educators to track student performance, to develop an interactive C++ teaching and learning platform, and to enhance student engagement and motivation. By addressing these objectives, the CodeQuest system seeks to transform the educational experience and improve student outcomes.

LITERATURE REVIEW

Challenges in Programming Course

The programming course in computer science education is often reported to have high failure and dropout rates (Cheah, 2020). This problem remains prevalent despite the large number of learning tools, as effective programming education requires a continuous, multi-faceted approach that exceeds traditional methods. One of the primary contributing factors to these difficulties is the ineffectiveness of conventional teaching methods, such as lectures, slide presentations, and paper-based books, which are ill-equipped to handle the dynamic nature of programming (Hainey & Baxter, 2024).

In addition to methodological challenges, students often struggle with the fundamental understanding of the course content. Studies have revealed that a significant percentage of students have only a low to medium grasp of programming logic and concepts, finding it particularly difficult to master syntax, semantics, and reasoning procedures. The inability to comprehend fundamental concepts, write problem-solving code, and identify errors has been identified as a key challenge faced by students. In 2023, Lovrencic and Sekovanic conducted research on students' understanding of the logic programming course. The biggest challenge is mastering the Prolog such as syntax, semantics and reasoning procedure (Bosse & Gerosa, 2017; Islam, 2019; Savage & Piwek, 2020; Lovrencic & Sekovanic, 2023). Moreover, student attitude and a lack of motivation are critical challenges, as many students exhibit minimal effort and a negative perspective toward programming, a factor that profoundly impacts their learning outcomes (Kadar et al., 2022). For instance, Liu, Shaikh and Gazizova (2020) reported that 80.23% of students who participated in their research studied with their own motivation for at least two hours while 6.98% spent less than an hour for classes.

Game-based Learning (GBL)

Game-Based Learning (GBL) is a pedagogical approach that uses full-fledged video games to teach specific skills or deliver learning objectives (Dimitra, Konstantinos, Christina & Katerina, 2020). It is distinct from gamification, which is the application of game-design elements and principles in non-game contexts to increase user engagement and motivation. While gamification focuses on external motivators like leaderboards and rewards to encourage the completion of tasks, GBL provides educational value through an actual game, thereby enhancing the learning experience and imparting knowledge (Findlay, 2016).

GBL has been shown to facilitate learning on cognitive, emotional, and social-cultural levels, providing an experience that is far more interactive and engaging than traditional media (Dimitra et al., 2020). Research into this field has identified various types of game-based learning, including memory games, simulation games, quizzes, puzzles, and strategy games, each contributing to different aspects of skill development such as critical thinking, decision-making, and memory. Incorporating games into the learning process has been shown to improve learning effectiveness and induce higher levels of student motivation (Chang, Chin & Hsieh, 2019; Plass, 2015).

Gamification in Education

Gamification in education has emerged as a prevalent strategy to augment student engagement. According to Mikrouli, Tzafilkou & Protogeros, (2024), there are seven types of game-based learning which are memory games, simulation games, interactives, quiz games, puzzles, strategy games and reality testing games. Memory games are particularly useful in improving and maintaining memory abilities as they offer opportunities to strengthen memory skills. Simulation games will present users with various scenarios that stimulate real-life situations. Students can learn and practice key concepts, procedures and decision-making skills in an interactive way. Interactive games is a popular gaming website which offers a wide range for students. It involves active participation such as answering questions, making decisions or solving puzzles.

Another type of game-based learning is quiz, it is a game that is quite effective tools that can facilitate educational tools in various ways of process. Puzzles provide opportunities to learn and mental exercise. It helps develop critical thinking and problem-solving skills. Besides, strategy games are the same as puzzle, it helps to develop critical thinking problem solving and decision-making skills. Lastly, reality testing games provide a unique and immersive learning experience to engage students more. Areas of application of VR games can vary design, second language, and chemistry to cultural studies and medicine.

METHODOLOGY

The CodeQuest was developed following Rapid Application Development (RAD), which emphasizes an iterative and agile process. This methodology allowed 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. All the articles reviewed by ACM, IEEE and ScienceDirect provide more relevant and accurate information. The researcher conducted an online survey and interview with students and lecturers at FSKM to gather specific requirements. An analysis of existing platforms like Sololearn and Codecademy helped to define key functionalities. By benchmarking these existing platforms, the researcher was able to gather initial requirements that are relevant and potentially applicable to the game-based learning by using requirement mapping technique. Each of the features from similar system and applications has been mapped with Requirement ID. This approach not only helped to develop the prototype but also ensured that the initial requirements were grounded in real-world practices and user expectations. 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.

RESULTS AND DISCUSSION

Initial Requirements

The development of CodeQuest, an online C++ programming learning system, began with a comprehensive analysis of existing platforms such as Sololearn, Mimo, Codecademy, and Ozaria. This review led to the identification of initial requirements for the system as shown in Figure 1. The initial use cases for CodeQuest encompass essential functionalities such as login, signup, user profile management, viewing learning materials, answering exercises, managing exercises, and accessing student progress reports. There are two users of the system, student and lecturer. For the student, they can view learning materials and take the exercises. Lecturers, on the other hand, manage the exercises, view students answer and view student progress reports.

The first version of the mid-fidelity prototype includes 16 designed interfaces, although not all have been fully developed at this stage. Notably, the User Profile and Answer Exercise interfaces have been implemented with game features.

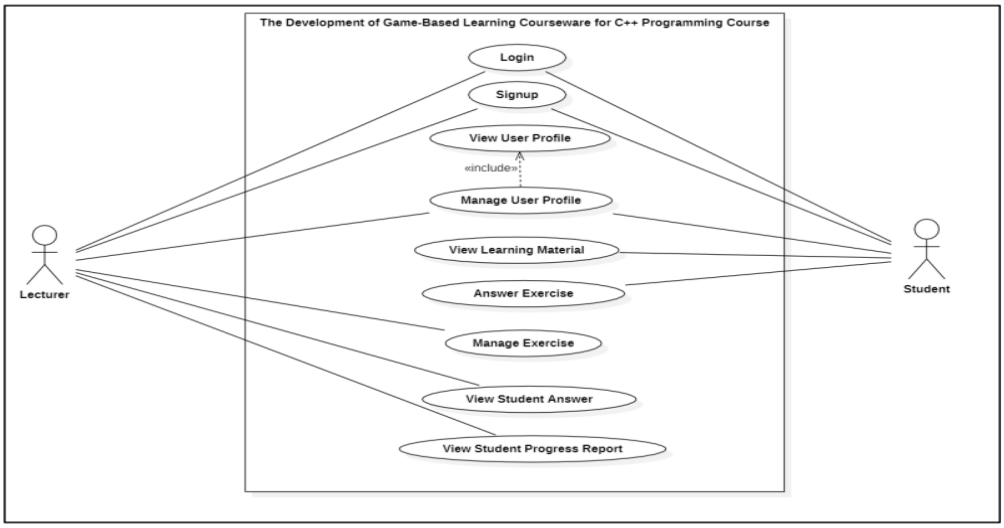

Figure 1 Initial use case diagram of CodeQuest

User Feedback and Refinement

Feedback from the first prototype demonstration was gathered from two lecturers, both of whom acknowledged the clarity and informativeness of the displayed information. During the demonstration, they provided several enhancement suggestions. Lecturer 1 recommended adding video explanations, flexible exercise options, and dual leaderboards, along with specific business rules regarding exercise submissions and profile visibility. Similarly, Lecturer 2 suggested additional reference materials, download options, and modifications to the login and signup processes, while also proposing similar business rules. List of business rules are summarized in Table 1.

Table 1 List of business rules suggested by lecturers

No	Business Rules
1	Once a student submits an exercise, they are not allowed to retake it.
2	Students can only view the overall leaderboard after completing the corresponding exercise.
3	Only lecturers and the respective student can view their profile.
4	Lecturers can only remove a topic or exercise if no student has attempted it yet.
5	Each topic should have multiple exercises.
6	Lecturers should be able to manually mark student’s coding answers.

In response to the lecturers’ feedback, several changes were implemented. Three new use cases were added: Manage Learning Material, Download Learning Material, and Give Feedback, with the existing View Student Answer use case being redefined to align with these new requirements. Enhancements were made to the View User Profile, View Learning Material, Manage Exercise, and View Student Progress Report use cases. The leaderboard feature was refined to include personal and overall categories, allowing students to access their personal leaderboard at any time while restricting overall leaderboard access until after exercise completion. Additionally, the Manage Exercise use case was improved to facilitate more efficient marking of student answers, and the View Learning Material use case was expanded to include downloadable lecture notes. Finally, the View Student Progress Report use case was refined to allow reports to be viewed by class, specific exercises, and topics, ensuring a more tailored experience for both students and lecturers.

Final Development

Based on the feedback from lecturers, the final requirements of CodeQuest are depicted in use case diagram in Figure 2. From these use cases, key features of CodeQuest are summarized as follow.

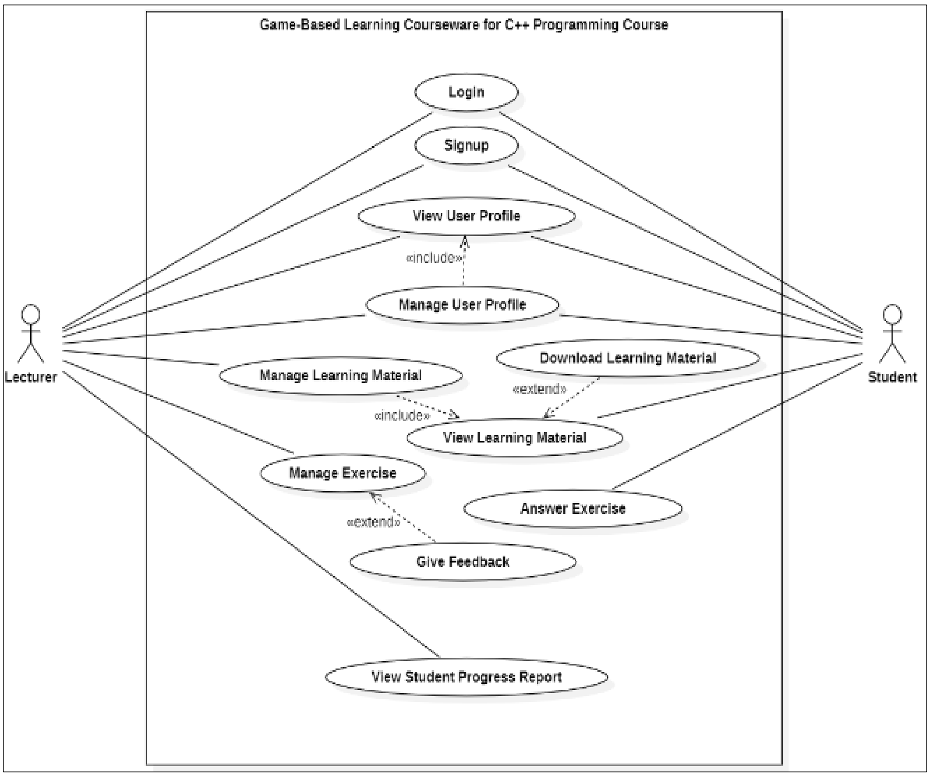

Figure 2: Final use case diagram for CodeQuest

- **User Management and Dashboards** - The system employs 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.

- **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.

- **Student Progress Monitoring** – Lecturer can view student progress report by exercise, topic and class.

A domain class diagram was created, as illustrated in Figure 3, featuring 10 tables, each with specific attributes. The diagram highlights various associations and multiplicities, including one-to-many relationships. A generalization relationship exists between the users table and the specialized classes for students and lecturers, indicating that both are extensions of the users table.

Additionally, the diagram shows aggregation relationships between courses and topics, topics and notes, and topics and exercises, signifying whole-part relationships where the components can exist independently. In contrast, there are composition relationships between the answers and exercises tables, as well as between topics and notes. These composition relationships indicate that answers and notes are dependent on their respective exercises and topics, meaning they cannot exist independently.

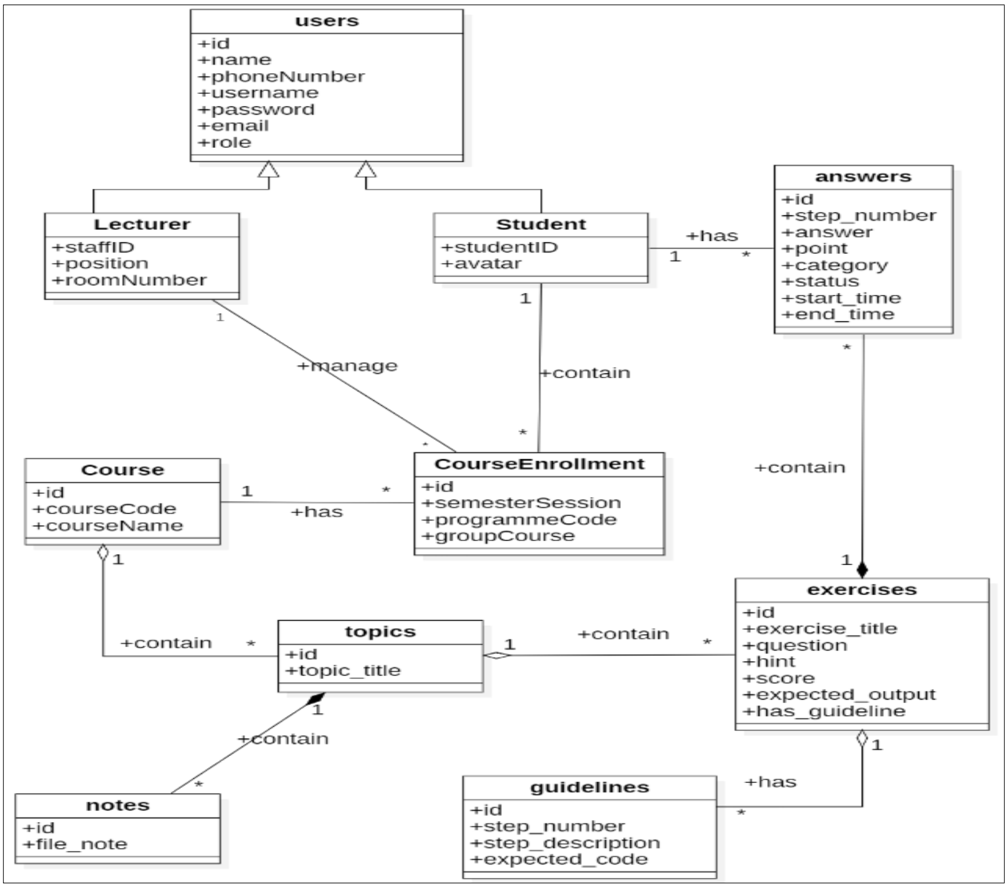

Figure 3 Domain Class Diagram

Figure 4 represents the physical model for databases using MySQL Workbench. The physical model shows the blueprint of the actual implementation for the database. The physical model extends the domain class diagram by adding the type of column and the primary keys and foreign keys.

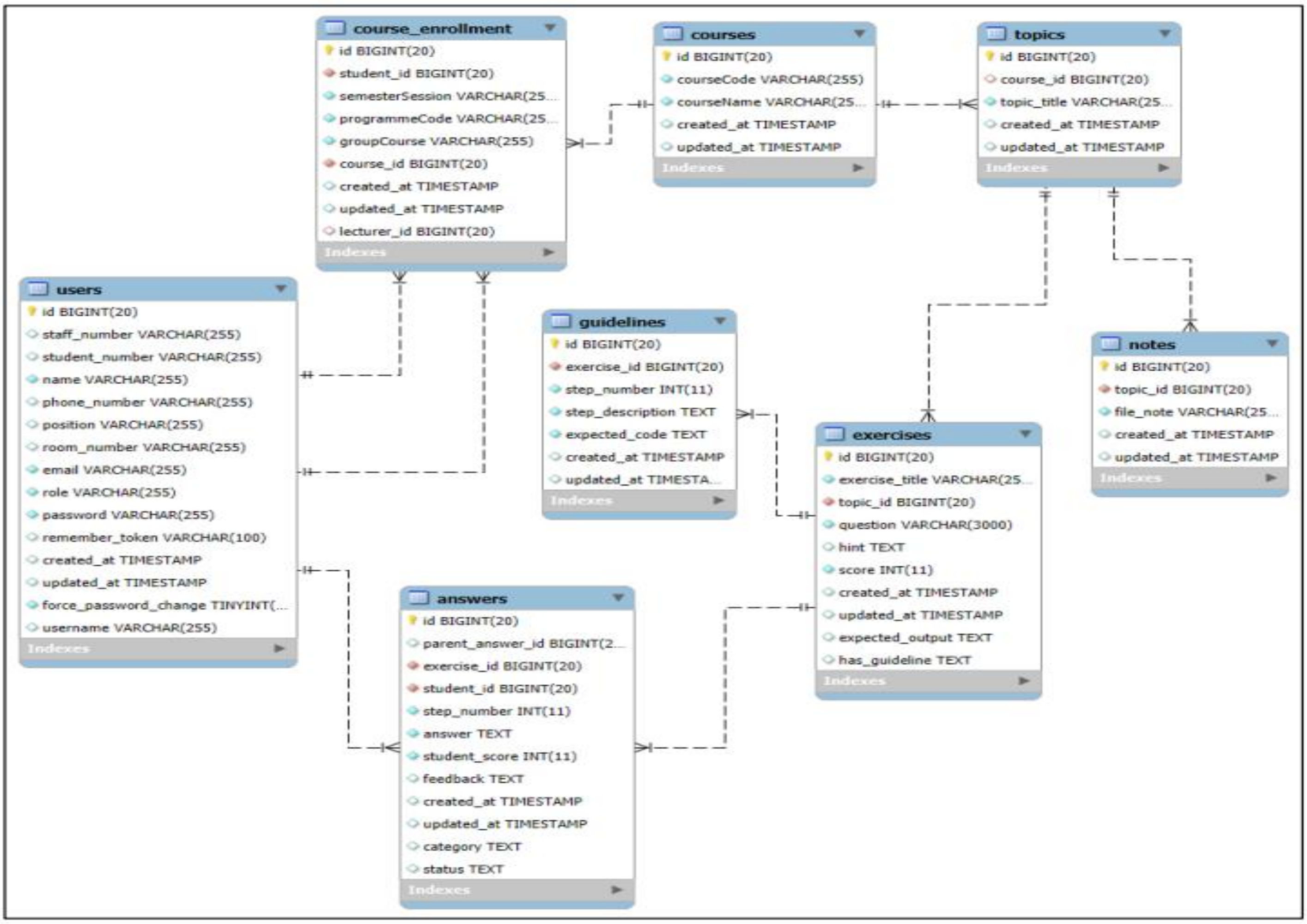

Figure 4 Physical model of database design

For the user interface (UI) design, the UI was designed and implemented using HTML, CSS, Tailwind CSS and Chart,js. Compared to the earlier mock-ups during mid-fidelity prototype, this system includes a specific aerospace theme which makes it more gamified UI.

In term of gamification elements, the game design of CodeQuest incorporates several engaging features aimed at enhancing the learning experience for students. One of the key elements is the **Interactive Progress Map**, which visually represents a student's overall learning journey. As shown in Figure 5, an avatar progresses to the next stage, referred to as a planet on the map, only when the student achieves a certain level of overall completion. This approach emphasizes cumulative progress rather than individual topic completion, allowing students to see their advancement in a more holistic manner. The progress map serves as a motivational tool, encouraging students to complete more exercises to advance their avatars to new stages.

Another significant feature is the **Real-time Leaderboard** (see Figure 6), which displays the rankings of all students within the same class for each specific exercise based on their points or achievements in the system. This leaderboard updates instantly whenever a student earns points, providing immediate feedback on their current standing relative to their peers. Additionally, it includes a time display that indicates the total time spent on each exercise. This competitive aspect fosters engagement among students, motivating them to improve their performance and climb higher in the rankings.

Lastly, the design incorporates a **Game Timer** for each exercise, which adds an element of urgency and challenge. If a student completes an exercise in less than 10 minutes, they are rewarded with double the usual points. This feature encourages students to solve problems efficiently, rewarding those who demonstrate both speed and accuracy. However, it is important to note that lecturers cannot manually input marks, as the system automatically assigns fixed points based on performance. This structured approach to game design not only enhances the learning experience but also instills a sense of competition and achievement among students (see Figure 6).

Figure 5 Interactive student dashboard – using avatars to show learning progress

Figure 6 Overall Leaderboard - overall ranking of students within the same class along with their interactive avatars

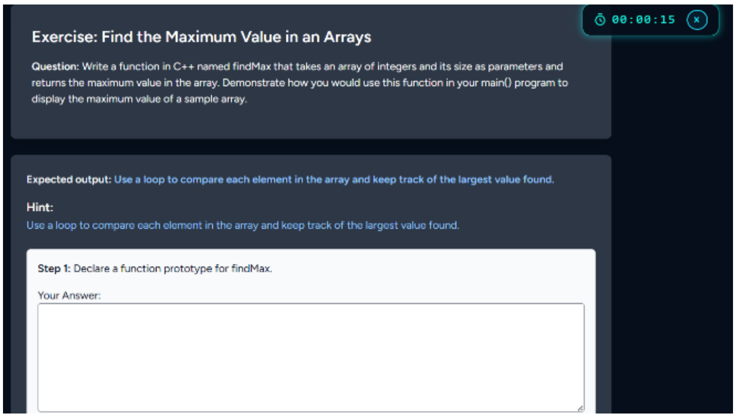

Figure 7 Game Timer in Exercise – Double point will be rewarded for student answering questions less than 10 minutes

ILEIID 2025 | International Journal of Research and Innovation in Social Science (IJRISS)
ISSN: 2454-6186 | DOI: 10.47772/IJRISS
Special Issue | Volume IX Issue XXII October 2025

The system architecture, as depicted in Figure 8, is constructed using the Laravel framework, which adheres to the Model-View-Controller (MVC) architectural pattern. This design facilitates a clear separation of concerns, enhancing the maintainability and scalability of the application. The View Layer serves as the interface between the user and the system, providing a means for users to interact with the application. This layer is implemented using PHP, HTML, and Tailwind CSS, which collectively enable the display of information and the collection of user input. Communication between the user's browser and the internet server occurs through this layer, allowing for the transmission of requests and the reception of responses.

The Domain Layer plays a critical role in processing the requests received from the View Layer. This layer is essential within the MVC architecture, as it encompasses the application server, business logic classes, and response pages, all developed in PHP. Upon receiving a request, the Domain Layer processes it according to the defined business logic and subsequently sends an appropriate response back to the View Layer using the designated response pages.

Finally, the Data Access Layer is responsible for managing the storage and retrieval of data from the database. This layer includes a MariaDB database and a data access class, which facilitate communication between the database and the Domain Layer. The data access class interacts with the database to input and output data as required. Additionally, the Model classes within this layer manage the stored data and provide a user interface for the Controller, enabling it to access and manipulate the data effectively.

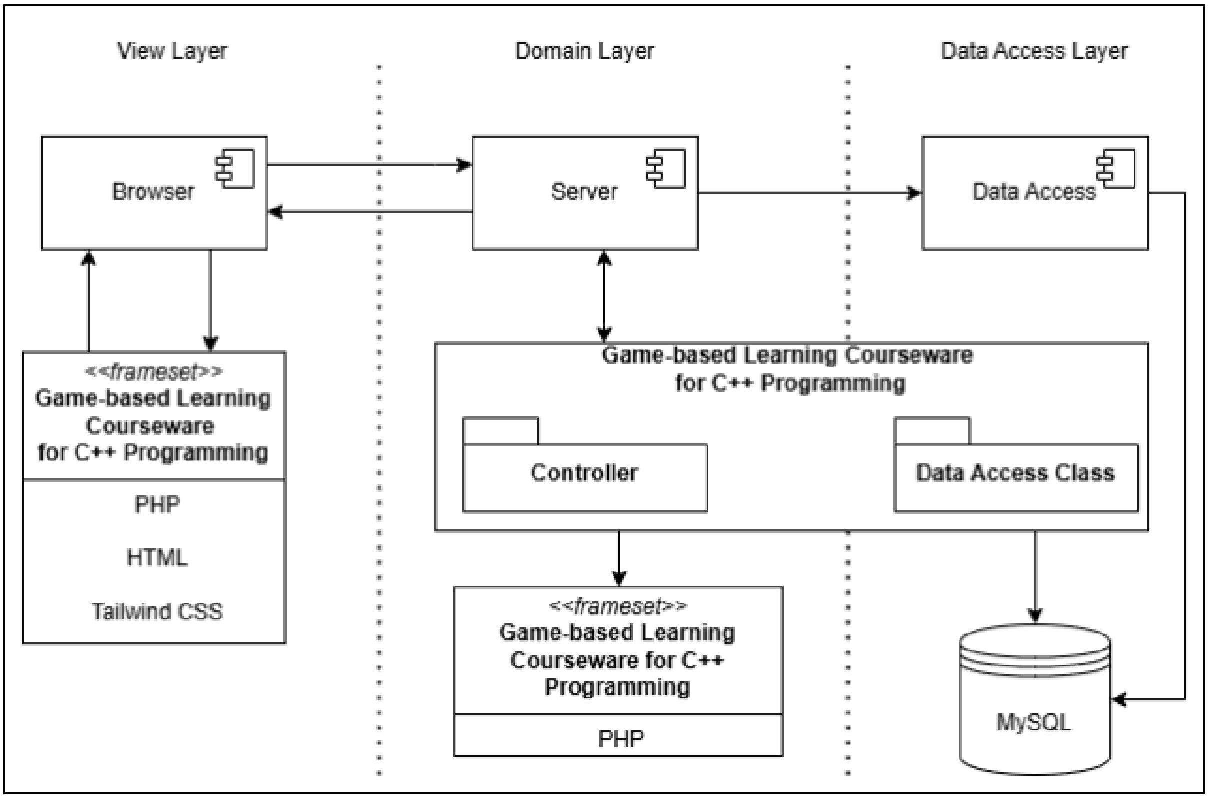

Figure 8 CodeQuest system architecture

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 CodeQuest 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.

CONCLUSION AND FUTURE RESEARCH RECOMMENDATION

This project reports the development of CodeQuest, a game-based learning system designed for C++ lecturers and students at the Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA. CodeQuest addresses the challenges students have in learning programming courses, particularly the difficulties in understanding abstract concepts and maintaining engagement through traditional teaching methods. The game-based learning system provides an alternative educational solution by offering a platform for lecturers to monitor student performance, thereby creating a learning environment that is more interactive and effective. The completion of this project not only validates the system through user feedback but also establishes a framework for future educational innovations in programming courses at UiTM.

To establish the implementation of CodeQuest, future research will expand its testing with a more extensive and varied student demographic to measure actual learning improvements and engagement levels. This extensive testing is essential to understand the impact of the system across various demographics, especially given the high failure rates in programming courses (Suhaimi, 2024). Additionally, incorporating quantifiable results, such as reductions in course failure rates and increases in exercise completion rates, will provide conclusive evidence of the system's effectiveness in enhancing student performance. These measurements are important for validating the educational benefits of game-based learning systems, as previous studies have demonstrated the significance of adaptive feedback in improving student outcomes (Marwan, 2022). Besides focusing on testing functional features and usability, the security and privacy of students' data needs to be considered. Although the current implementation uses role-based access control and encryption to protect sensitive information and comply with data protection regulations, further penetration testing is needed to ensure the system is secure and reliable.

Furthermore, the scalability of CodeQuest is also a long-term aspect that needs to be prioritized, particularly in terms of cloud infrastructure that support monitoring and automated updates that could ensure the system remains current with educational advancements. Finally, exploring integration with existing Learning Management Systems (LMS) will facilitate adoption more readily by universities, enhancing user experience through seamless access to learning materials and progress tracking. By aligning with established educational frameworks, CodeQuest can better support educators and students, resulting in improved learning outcomes and engagement in programming courses.

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. ALshammari, F. L. (2024). Video-Based Microlearning and the Impact on Programming Skills and Technology Acceptance. Journal of Education and e- Learning Research, 11(1), 155-165.
3. Asgari, M., Tsai, F.-C., Mannila, L., Strömbäck, F., & Sadique, K. M. (2024). Students' perspectives on using digital tools in programming courses. Discover Education, 3(1). https://doi.org/10.1007/s44217-024-00144-4
4. Bosse, Y., & Gerosa, M. A. (2017). Why is programming so difficult to learn? ACM SIGSOFT Software Engineering Notes, 41(6), 1–6. https://doi.org/10.1145/3011286.3011301
5. Chang, Y. C., Chin, K. Y., & Hsieh, H. C. (2019). Development of Digital Game- Based Learning System in Social Curriculums of Primary Schools. Proceedings - 2019 8th International Congress on Advanced Applied Informatics, IIAI-AAI 2019, 1065–1066. https://doi.org/10.1109/IIAI-AAI.2019.00228
6. 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

7. 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
8. Dimitra, K., Konstantinos, K., Christina, Z., & Katerina, T. (2020). Types of Game-Based Learning in Education: A brief state of the art and the implementation in Greece. The European Educational Researcher, 3(2), 87–100. https://doi.org/10.31757/euer.324
9. Findlay, J. (2016,). Game-Based Learning vs. Gamification: Do You Know the Difference? Retrieved from Training Industry: https://trainingindustry.com/articles/learningtechnologies/game-based-learning-vs-gamification-do-you-know-the-difference/
10. Hainey, T., & Baxter, G. (2024). A Serious game for programming in higher education. Computers & Education: X Reality, 4, 100061. https://doi.org/10.1016/j.cexr.2024.100061
11. 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
12. Kadar, R., Mahlan, S. B., Shamsuddin, M., Othman, J., & Wahab, N. A. (2022). Analysis of Factors Contributing to the Difficulties in Learning Computer Programming among Non-Computer Science Students. 2022 12th IEEE Symposium on Computer Applications and Industrial Electronics, ISCAIE 2022, 89–94. https://doi.org/10.1109/ISCAIE54458.2022.9794546
13. Liu, Z. Y., Shaikh, Z. A., & Gazizova, F. (2020). Using the concept of game- based learning in education. International Journal of Emerging Technologies in Learning, 15(14), 53–64. https://doi.org/10.3991/ijet.v15i14.14675
14. Lovrenčić, S., & Sekovanić, V. (n.d.). How Well Students Perceive Their Understanding of Logic Programming Course Content?
15. 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
16. Mikrouli, P., Tzafilkou, K., & Protogeros, N. (2024). Applications and Learning Outcomes of Game Based Learning in Education. International Educational Review, 25–54. https://doi.org/10.58693/ier.212
17. Plass, J. L., Homer, B. D., & Kinzer, C. K. (2015). Foundations of Game-Based Learning. Educational Psychologist, 50(4), 258–283. https://doi.org/10.1080/00461520.2015.1122533
18. Piwek, P., & Savage, S. (2020). Challenges with learning to program and problem solve: An analysis of student online discussions. SIGCSE 2020 - Proceedings of the 51st ACM Technical Symposium on Computer Science Education, 494–499. https://doi.org/10.1145/3328778.3366838
19. 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
20. 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
21. Tsai, C. W., Lin, M. Y. C., Cheng, Y. P., Lee, L. Y., Chyr, W. L., Lin, C. H., ... & Tsai, M. C. (2023). The effects of online peer-facilitated learning and distributed pair programming on students' learning. Computers & education, 203, 104849.