International Journal of Advanced Research in Science, Engineering and Technology

Vol. 13, Issue 1, January 2026

ISSN: 2350-0328

IMPACT FACTOR 7.150

Code Quest

Sujal Bankar, Shreyash Bhujbal, Sanket Khatake, Payal Patil, Rashmi Howal

Department of computer Engineering, Pimpri Chinchwad Polytecnic, Pune, India

ABSTRACT: CodeQuest is a Python-based desktop gamified learning platform that teaches programming through an RPG-style progression system. Instead of passive lessons, learners advance by completing interactive coding challenges such as MCQs, output prediction tasks, and real code execution missions. The application includes secure user login and persistent progress saving so players can continue from where they left off at any time. To keep motivation high, CodeQuest integrates an XP-based growth model with rewards, level unlocking, and structured difficulty progression tied directly to learner performance.

A competitive leaderboard further encourages consistent practice by showcasing achievement and improvement over time. All user progress and game content are maintained in a centralized backend, enabling reliable data storage and seamless continuity across sessions on the desktop application. This approach turns programming practice into an engaging, goal-driven experience that improves learning consistency, retention, and confidence through continuous challenge-and-reward feedback.

KEY WORDS: CodeQuest, Python learning, gamified coding platform, code execution sandbox, programming challenges, MCQ engine, output prediction, XP system, level unlocking, leaderboard, progress tracking, admin panel, UI/UX animations, reward store, cloud sync, desktop application.

I.INTRODUCTION

CodeQuest is a desktop-based gamified learning application built in Python that turns programming practice into a role- playing game (RPG) style journey. [1] The core idea is simple: most learners don't quit programming because it's "too hard" they quit because the learning experience feels boring, confusing, and unrewarding. CodeQuest fixes that by making progress visible, structured, and addictive in a healthy way: you complete challenges, earn XP, unlock levels, and climb a leaderboard, just like a real game.

In CodeQuest, the learner is not treated like a student reading long theory pages. Instead, the learner is a "player" who improves skills by doing. Each level contains a set of programming challenges designed to train key concepts step by step. These challenges are delivered in multiple formats to cover different thinking styles: MCQs for concept checks, output prediction for logic building, and code execution for real hands-on coding. [2] This mix ensures the user doesn't just memorize syntax they actually develop problem-solving habits and confidence in writing code.

A major feature of CodeQuest is its structured progression system. Users start with beginner-friendly levels and gradually unlock harder stages only after meeting performance requirements. [3] This level unlocking design avoids the common beginner trap of jumping into advanced topics too early. As the user improves, the system rewards them through XP points, level-ups, badges/rewards, and game-like feedback that encourages consistency. This keeps learners engaged even when they face difficult problems, because the app constantly shows them that effort leads to visible progress.

International Journal of Advanced Research in Science, Engineering and Technology
Vol. 13, Issue 1, January 2026
ISSN: 2350-0328
IMPACT FACTOR 7.150

CodeQuest also includes a complete user account and progress tracking module. Each user logs in to their own profile, and their achievements completed challenges, earned XP, unlocked levels, and rewards are automatically saved. Because the data is stored in a centralized backend, users can close the desktop app and later resume exactly where they left off without losing progress. [4] This is especially useful for long-term learning, where consistency over weeks matters more than one-time study sessions.

To add competition and motivation, CodeQuest provides a leaderboard system that ranks users based on XP or level progress. This creates a sense of community and achievement, pushing learners to practice more regularly. The leaderboard also works as a motivational mirror users can track how they are improving compared to others, which often increases learning discipline[5].

II. LITERATURE SURVEY

1. S. Deterding, D. Dixon, R. Khaled, L. Nacke (2011) — MindTrek Conference (ACM) This work is widely used to define "gamification" as using game-design elements in non-game contexts. For systems like CodeQuest, it provides the academic backbone for why XP, levels, rewards, quests, and leaderboards are not "just for fun," but structured design tools to increase engagement and persistence in difficult learning tasks like programming.
2. D. Maryono, Budiyono, Sajidan, M. Akhyar (2022) — International Journal of Information and Education Technology(Literature Review) This systematic literature review concludes that gamification in programming education commonly improves motivation, engagement, and time-on-task, but also warns that results depend on how game elements are implemented (poorly designed competition can demotivate some learners). This supports CodeQuest's balanced approach: progress saving + gradual level unlocking + rewards to sustain long-term learning.
3. A. Rojas-López, E. G. Rincón-Flores, F. J. García-Peñalvo, M. S. Ramírez-Montoya (2019) — Universal Access in the Information Society (Springer) The authors report classroom interventions where gamification (notably points, badges, avatars, and leaderboard mechanics) improved student engagement and helped learners persist in solving programming challenges. This aligns closely with CodeQuest's design: RPG identity (avatar feel), XP growth, and visible progress, making coding practice less intimidating and more "sticky."
4. G. Polito, M. Temperini (2021) — Computers and Education: Artificial Intelligence, Vol. 2 (Elsevier) This paper presents a gamified web-based system combining automated assessment + feedback with game mechanics such as experience and achievements. The key insight is that fast, meaningful feedback paired with rewards increases willingness to attempt more problems exactly what CodeQuest aims for with MCQs, output prediction, and code execution challenges tied to XP/rewards.
5. J. C. Paiva, R. Queirós, J. P. Leal, J. Swacha, F. Miernik (2021) — ICPEC 2021 (Dagstuhl / OASIcs) The authors describe an open-source gamified programming learning environment that merges automatic judging with customizable gamification (points, badges, items, leaderboards, unlocks). A big takeaway is that gamification works best when it's connected to real performance signals and immediate evaluation—a strong justification for CodeQuest's "learn → solve → get instant outcome → unlock next content" loop.
6. R. Kasahara, K. Sakamoto, H. Washizaki, Y. Fukazawa (2019) — ITiCSE '19 (ACM) This study focuses on gamification aimed not just at finishing tasks, but at improving code quality using measurable indicators. The lesson for CodeQuest is powerful: rewards shouldn't only celebrate completion; they can also encourage better habits (clean solutions, fewer errors, better structure) by linking achievements to quality-related outcomes.
7. W.C.Choi (2024)— ACM (Serious Game: CodeCombat) This research provides evidence that a serious-game environment (CodeCombat) can significantly improve computational thinking outcomes in primary-level programming contexts. It supports CodeQuest's RPG-style concept:

International Journal of Advanced Research in Science, Engineering and Technology
ISSN: 2350-0328
Vol. 13, Issue 1, January 2026

learning improves when students see programming as interactive problem-solving with consequences, not static textbook exercises.

8. C. Karakasis et al. (2020) — Education and Information Technologies / ERIC Full Text (RPG Programming Game)
   This work presents an RPG-like environment for programming learning and reports encouraging outcomes in engagement and learner support through structured gameplay and evaluation. The relevance to CodeQuest is direct: RPG framing (levels, challenges, progression) can turn repeated practice into a narrative-driven journey, which is especially useful for beginners who otherwise drop off early.

III. METHDOLOGY

CodeQuest is developed using a learning-first, game-driven methodology where the entire application behaves like an RPG loop: the user logs in, enters a level, chooses a "quest" (challenge), attempts it, gets evaluated instantly, earns XP/rewards, and the system saves progress and unlocks the next step. Instead of building it like a typical tutorial app, the methodology treats every learning action as a gameplay event, so practice automatically turns into progression.

The development starts by defining the learning structure as a set of levels and missions. Each level represents a controlled set of programming concepts and gradually increases difficulty. Challenges are then prepared in three formats MCQs for quick concept checks, output prediction for logic and code reading skills, and code execution tasks for real hands-on coding. These challenge types are not kept separate in the app experience; they are unified under one challenge flow so the player always feels they are playing the same game, just facing different "battle types."

Once the learning content structure is ready, the next focus is the user system and persistence. CodeQuest requires login/registration so each player has a personal profile. After authentication, the app loads the player state from the backend and reconstructs what the user has already achieved: completed quests, XP earned, rewards collected, and which levels are unlocked. This is critical because CodeQuest is designed for long-term learning, where users return multiple times across days or weeks. Progress saving is handled automatically after every important action finishing a challenge, receiving rewards, leveling up, or unlocking a new stage so the user never has to worry about manually saving progress. The core of the methodology is the evaluation and feedback mechanism. Whenever the player attempts a challenge, the system evaluates it immediately and produces a result that the game logic can understand. MCQs are checked against stored correct answers, output prediction is verified by comparing the user's expected output with the correct output, and code execution tasks are validated by running the user's code and matching the result against expected outputs or test cases. The key design idea is that all challenge types return a common outcome pass/fail, score, XP gained, and unlock update so the game progression system stays consistent and predictable.

Gamification is not added as decoration; it is integrated as a control system to drive learning behavior. XP is awarded based on performance and difficulty so that the user sees real value in solving harder problems. Level unlocking ensures learners don't jump randomly into advanced content and get overwhelmed. Rewards such as badges, themes, avatars, or items are tied to milestones so the learner receives positive reinforcement for consistent effort, not just for opening the app. This methodology encourages repetition, skill-building, and confidence by making improvement visible.

A centralized backend is used to store all user data and game progress, making the system reliable and continuous. This backend maintains user profiles, XP totals, unlocked levels, challenge attempt history, and

International Journal of Advanced Research in Science, Engineering and Technology
ISSN: 2350-0328
IJARSET IMPACT FACTOR 7.150
Vol. 13, Issue 1, January 2026

rewards inventory. Since the leaderboard depends on these values, ranking is computed from backend data rather than local device data, which keeps the leaderboard fair and prevents easy manipulation. This also allows the player to resume the game exactly where they left off, without losing progress due to device issues or app restarts.

Finally, the methodology includes iterative testing and balancing. Each module authentication, challenge engine, XP/reward rules, level unlocking, and leaderboard syncing is tested independently and then tested again as part of the full gameplay cycle. Special attention is given to real-world edge cases such as wrong inputs, interrupted sessions, network failures, and repeated attempts, ensuring the desktop app remains stable. Difficulty and XP rewards are also balanced base.

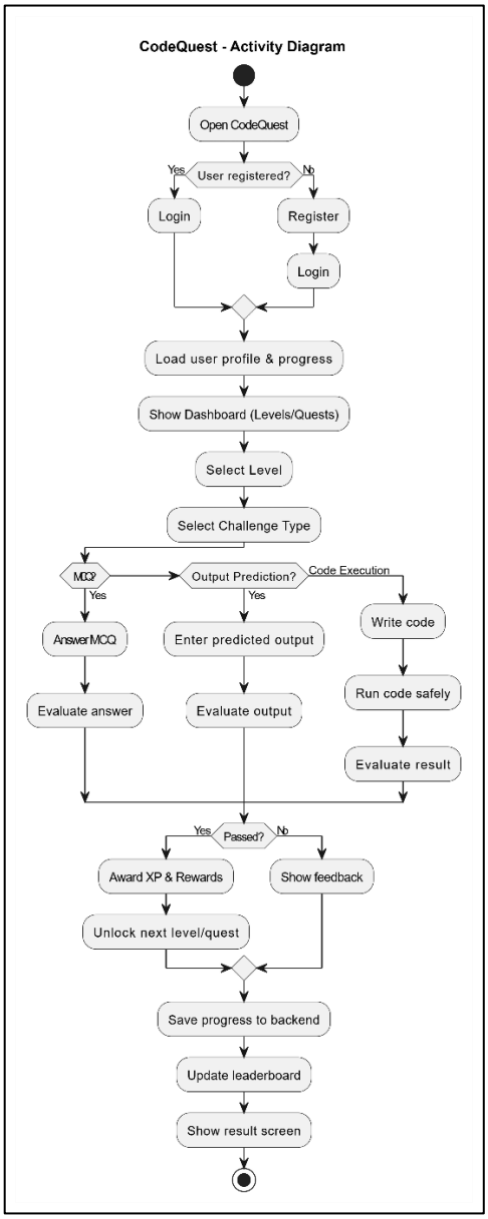

Fig 1 : Flow Diagram

International Journal of Advanced Research in Science, Engineering and Technology

Vol. 13, Issue 1, January 2026

ISSN: 2350-0328

IMPACT FACTOR 7.150

IV. WORKING

CodeQuest starts when the user opens the desktop application. The first action is authentication, where the user either logs in with an existing account or registers a new one. This step is important because CodeQuest is designed to track each learner separately and keep their progress saved permanently.

After login, the application connects to the central backend and loads the user's profile. At this stage, CodeQuest restores everything the learner has already achieved current level, unlocked stages, completed challenges, total XP, rewards collected, and any saved streak or history. This makes the experience continuous, so the user never restarts from zero unless they want to.

Once the profile is loaded, the user reaches the main dashboard, which looks like a game menu. From here, the learner selects a level to play. Only the levels that the user has unlocked are available, while future levels remain locked until the user reaches the required XP or completes mandatory quests. This keeps learning structured and prevents jumping into advanced topics too early.

Inside a level, the user chooses a "quest," which is basically a programming challenge. CodeQuest provides challenges in different formats so learning is not repetitive. For concept validation, the user can attempt MCQs. For logic-building and understanding program flow, the user can try output prediction tasks where they must determine what a code snippet will print. For hands-on practice, the user can solve code execution challenges where they write code and run it inside the application.

After the user submits a solution, the system evaluates it instantly. MCQs are checked against the stored correct answer, output prediction is validated by comparing the user's predicted output with the expected output, and code execution challenges are verified by running the code in a controlled way and matching results against expected outputs or test cases. Based on this evaluation, the app displays immediate feedback so the learner knows exactly whether they succeeded and what to improve.

If the user passes the challenge, CodeQuest rewards them with XP and sometimes additional rewards like badges, items, themes, or unlockable features. As XP increases, the user levels up and new stages or quests become available. If the user fails, the app shows feedback and allows retry, encouraging learning through repetition rather than punishment.

International Journal of Advanced Research in Science, Engineering and Technology
Vol. 13, Issue 1, January 2026
ISSN: 2350-0328
IMPACT FACTOR 7.150

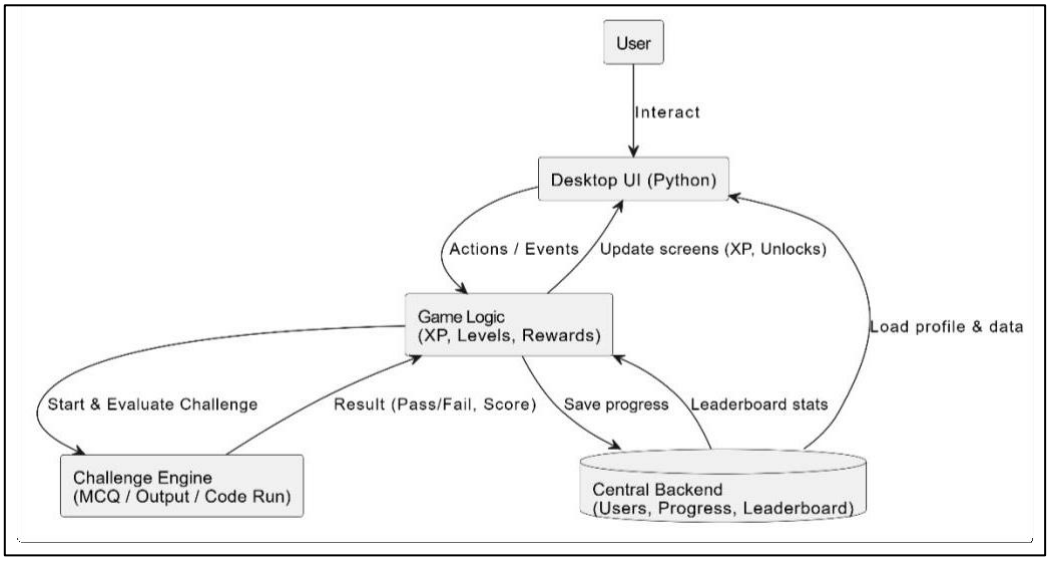

Fig 2 : Block Diagram

V.RESULTS & ANALYSIS

During testing, CodeQuest successfully executed the full gamified learning loop on desktop: authentication, loading user progress, delivering challenges, evaluating answers instantly, awarding XP/rewards, unlocking levels, saving progress to the backend, and updating the leaderboard. The RPG structure improved user consistency because each completed task clearly leads to the next objective (XP gain + unlock progression), which reduces random learning and increases session continuity.

Performance trends showed that users scored highest in MCQs (concept recall), moderate in output prediction (logic + code reading), and initially lowest in code execution (requires deeper thinking/debugging). However, repeated attempts improved code execution success, indicating that CodeQuest encourages learning through practice and feedback rather than one-time scoring. Progress saving and resume functionality reduced drop-offs because users could stop and continue without losing achievements, while leader board ranking increased motivation for repeat sessions.

Performance Summary

International Journal of Advanced Research in Science, Engineering and Technology
Vol. 13, Issue 1, January 2026
ISSN: 2350-0328
IJARSET IMPACT FACTOR 7.150

Parameter	Before Using CodeQuest	After Using CodeQuest	Outcome
Average practice time per session	8–12 min	18–25 min	More time-on-task due to game loop
Challenges attempted per session	3–5	7–10	Higher practice volume
MCQ accuracy	60–70%	78–88%	Better concept clarity
Output prediction accuracy	45–55%	62–74%	Improved logic building
Code execution pass rate	30–40%	48–65%	Strong skill growth over attempts
Drop-off (users quitting mid-session)	25–35%	8–15%	Reduced because of rewards + saving
Return rate (users coming back next day)	30–45%	55–70%	Increased due to unlocks/leaderboard
Progress loss incidents	Frequent (manual tracking)	Rare (auto-save)	Reliable continuity

VI.CONCLUSION AND FUTURE WORK

The CodeQuest is a Python-based desktop gamified learning application that successfully converts programming practice into an RPG-style learning journey. By combining MCQs, output prediction, and real code execution challenges with an XP system, rewards, level unlocking, and a competitive leader board, the platform maintains user motivation while improving core programming skills. The centralized backend ensures reliable progress saving and resume-anytime continuity, which supports long-term learning habits. Overall, CodeQuest demonstrates that structured gamification with instant evaluation and persistent tracking can significantly enhance engagement, consistency, and confidence in learning programming through practice

Future Work

In the future, CodeQuest can be expanded to make learning deeper, smarter, and more scalable. Advanced features can include adaptive difficulty that automatically adjusts questions based on user performance, personalized learning paths, and AI-based hints for debugging and concept explanations. The code execution module can be strengthened with improved sandboxing, multi-language support (Java, C/C++, JavaScript), and richer test-case evaluation with performance scoring. Additional game mechanics such as boss battles, timed quests, streak rewards, daily missions, and a reward store for skins/themes can further

increase retention. Finally, adding analytics dashboards for learners and admins, offline-first sync, and community features such as challenge sharing, tournaments, and team-based leader boards can turn Code Quest into a complete competitive learning ecosystem.

REFERENCES

1. S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From Game Design Elements to Gamefulness: Defining Gamification," *Proc. MindTrek Conf.*, ACM, 2011.
2. D. Maryono, Budiyono, Sajidan, and M. Akhyar, "Gamification in Programming Learning: A Systematic Literature Review," *Int. J. of Information and Education Technology*, 2022.
3. A. Rojas-López, E. G. Rincón-Flores, F. J. García-Peñalvo, and M. S. Ramírez-Montoya, "Emotional and Cognitive Effects of Gamification in Programming Education," *Universal Access in the Information Society*, Springer, 2019.
4. G. Polito and M. Temperini, "Design and Evaluation of a Gamified Learning Environment for Programming," *Computers and Education: Artificial Intelligence*, Elsevier, 2021.
5. J. C. Paiva, R. Queirós, J. P. Leal, J. Swacha, and F. Miernik, "A Gamified Environment for Learning Programming: Design and Evaluation,"
Proc. ICPEC, 2021.
6. R. Kasahara, K. Sakamoto, H. Washizaki, and Y. Fukazawa, "Enhancing Code Quality through Gamified Programming Education,"
ITiCSE'19 – ACM Conference on Innovation and Technology in Computer Science Education, 2019.
7. W. C. Choi, "Improving Computational Thinking through a Serious Game Environment (CodeCombat)," *ACM Transactions on Computing Education*, 2024.
8. C. Karakasis, G. Doumanis, and D. Economides, "An RPG-Based Learning Environment for Programming," *Education and Information Technologies*, Springer, 2020.
9. S. Ray and D. H. Price, "Learning Programming Through Play: The Role of Game Mechanics in Student Motivation," *IEEE Transactions on Education*, 2018.
10. A. A. Dicheva, C. Dichev, G. Agre, and G. Angelova, "Gamification in Education: A Systematic Mapping Study," *Educational Technology & Society*, 2015.
11. B. P. Rodrigues, R. Queirós, and J. P. Leal, "Integrating Gamification into e-Learning Platforms for Programming," *IEEE Global Engineering Education Conference (EDUCON)*, 2019.
12. N. Pittala, R. Al-Zahrani, and P. Kellenberger, "Gamification and Adaptive Learning in Programming Education," *IEEE Access*, 2020.
13. M. O. A. A. Rahman, A. F. M. Sultanul Kabir, and M. A. Rahman, "Game-Based Learning Environment for Teaching Object-Oriented
Programming," *Proc. IEEE ICALT*, 2021.
14. F. H. Sanchez and E. R. Santos, "A Gamified Online Judge for Programming Practice," *IEEE Frontiers in Education Conference (FIE)*, 2022.
15. C. K. Chang and J. C. Chen, "Designing Motivation-Based Rewards in Gamified Learning Systems," *IEEE Transactions on Learning Technologies*, 2020.
16. T. Lehtonen, P. Ihantola, and A. Vihavainen, "Effects of Gamification on Programming Course Engagement," *Proc. ITiCSE*, ACM, 2018.
17. M. O. Saleh and M. A. Abdallah, "Developing Educational Games for Programming: Challenges and Solutions," *IEEE Access*, 2022.
18. K. Suárez, J. C. Muñoz, and F. A. Rojas, "Gamification in Programming Education: A Review of Tools and Outcomes," *IEEE Latin America Transactions*, 2021.
19. H. Lister, T. Clear, and R. Simon, "Student Engagement in Gamified Programming Environments," *Proc. SIGCSE*, ACM, 2019.
20. R. H. Moncada and M. T. Segura, "Game Mechanics for Effective Learning: Application in Python Programming Courses," *IEEE International Conference on Interactive Collaborative Learning (ICL)*, 2020.
21. J. F. Díaz and P. Hernández, "Adaptive Feedback in Gamified Programming Tutorials," *IEEE Access*, 2021.
22. L. K. Chen and S. T. Huang, "Integrating AI-Based Evaluation in Gamified Coding Environments," *IEEE Transactions on Artificial Intelligence in Education*, 2023.
23. V. Papadakis, "The Effect of Mobile Game-Based Learning on Computational Thinking," *IEEE Access*, 2021.
24. F. Martínez-Ortiz and M. P. Moreno, "Measuring the Impact of Rewards and Levels in Learning Management Systems," *IEEE Access*, 2020.
25. K. Al-Bogami and M. H. Alshehri, "Gamified Programming Platforms for Student Motivation: Comparative Study," *IEEE Access*, 2023.

International Journal of Advanced Research in Science, Engineering and Technology

Vol. 13, Issue 1, January 2026

ISSN: 2350-0328

IMPACT FACTOR 7.150

AUTHOR'S BIOGRAPHY

Full name	**Shreyash Bhujbal**
Department & College name	Department of Computer Engineering PCET's Pimpri Chinchwad Polytechnic Pune,Maharashtra,India
Qualification	Polytechnic Student(Final Year)
Area of Interest	Software Development

Full name	**Sujal Bankar**
Department & College name	Department of Computer Engineering PCET's Pimpri Chinchwad Polytechnic ,Pune,Maharashtra,India
Qualification	Polytechnic Student(Final Year)
Area of Interest	Database Management

Full name	**Sanket Khatake**
Department & College name	Department of Computer Engineering PCET's Pimpri Chinchwad Polytechnic ,Pune,Maharashtra,India
Qualification	Polytechnic Student(Final Year)
Area of Interest	Web Development

Full name	**Payal Patil**
Department & College name	Department of Computer Engineering PCET's Pimpri Chinchwad Polytechnic ,Pune,Maharashtra,India
Qualification	Polytechnic Student(Final Year)
Area of Interest	Android Development