WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

NEXAURA: The Missing Notebook — A Story-Driven 3D Environment for Learning Data Structures and Algorithms

Mrs. Haritha M [#1], Harini R [#2], Harisroshan M [#3], Harish Raja K [#4], Harithaa G [#5]

[#1] Assistant Professor, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India
[#2] Student, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.
[#3] Student, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.
[#4] Student, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.
[#5] Student, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.

ABSTRACT :

NEXAURA: The Missing Notebook is a browser-based educational detective game designed to connect Data Structures and Algorithms (DSA) learning with an interactive investigation. The player investigates the disappearance of Maya's research notebook at the Maple Park Science Fair, using programming challenges to unlock clues and advance the story. The project combines a React and Vite interface with a Three.js-based 3D environment implemented through React Three Fiber and Drei. Its learning structure is organized into six DSA chapters: Arrays, Linked Lists, Stacks and Queues, Trees, Graphs, and Dynamic Programming. The project description identifies a bank of 40 Java challenges and a backend judging workflow that compiles submitted programs with javac, executes them with java, and evaluates outputs against sample and hidden test cases. Successful completion is connected to progression mechanisms such as experience points, coins, and chapter unlocking. A proposed AI assistant, IRI, is intended to provide contextual hints and explain programming errors; this integration is treated as future work rather than a completed capability. The paper describes the project architecture, learning workflow, implementation scope, limitations, and evaluation needs. No controlled user study or quantitative learning-gain result is claimed.

Keywords: NEXAURA, Educational Game, Data Structures and Algorithms, Java Programming, Serious Games, React, Three.js, Gamified Learning.

INTRODUCTION

Data Structures and Algorithms are foundational topics in computer science, but learners may find it difficult to connect abstract concepts with practical problem-solving. Conventional exercises often present a problem, request code, and return a verdict without providing a meaningful narrative context. This can make repeated practice feel disconnected from a learner's broader goals.

Game-based learning can provide a structured context for practice by combining objectives, feedback, progress, and challenges. NEXAURA: The Missing Notebook applies this approach through a detective story in which solving programming challenges advances an investigation. The player explores the Maple Park Science Fair after Maya's research notebook disappears before judging begins at 9:00 AM.

The project combines narrative progression with Java coding tasks. Its six chapter topics are Arrays, Linked Lists, Stacks and Queues, Trees, Graphs, and Dynamic Programming. A challenge is intended to reinforce the topic represented by the current chapter, while successful completion can unlock further investigation content and rewards.

To develop a browser-based detective game that introduces DSA concepts through a connected storyline.

To provide Java programming challenges with executable evaluation rather than relying only on answer matching.

To connect challenge completion with clues, rewards, and sequential chapter progression.

1) To use a 3D environment for spatial exploration and evidence interaction in the opening chapter.
2) To establish a foundation for contextual AI hints and beginner-friendly error explanations as future enhancements.
3) The paper describes the architecture and functionality reported for the project. Since a complete NEXAURA source-code archive was not attached with the current document, code-level statements are based on the project details already supplied and should be verified against the final repository before publication. No measured improvement in learning outcomes is claimed.

II. LITERATURE SURVEY

Educational games and serious games use structured goals, feedback, and interaction to support learning activities. Game elements such as progression, challenge, rewards, and narrative can help organize practice, but their effectiveness depends on instructional alignment and the quality of the learning tasks.

Gamification and Learning. Gamification applies selected game-design elements in non-game contexts. In an educational programming environment, progress indicators and rewards may encourage continued practice, but rewards should support rather than replace understanding of the underlying concepts.

Programming Education. Novice programmers commonly need repeated opportunities to trace logic, test hypotheses, interpret errors, and revise solutions. Immediate execution feedback can help make the relationship between source code and program behaviour more visible. A judging system must nevertheless provide useful feedback and should not be treated as a complete explanation of a learner's mistake.

Narrative-Based Learning. A story can provide a reason for completing individual tasks and can connect separate exercises into a coherent sequence. In NEXAURA, clues and evidence provide the narrative link between DSA challenges and the missing-notebook investigation.

Interactive 3D Environments. Browser-based 3D technologies allow users to navigate a scene and interact with objects. Such interaction can increase the sense of exploration, although the 3D environment must remain usable and must not distract from the programming objectives.

Research Gap. The project addresses the design opportunity of combining a detective narrative, topic-based Java challenges, executable code judging, and 3D evidence collection in one learning experience. Its educational effectiveness, usability, and long-term retention still require systematic evaluation.

III. PROPOSED METHODOLOGY

NEXAURA is organized as a browser-based game with a user interface, chapter and narrative state, 3D exploration components, coding challenges, and a Java execution-and-judging service. The player moves from the introduction to the case map, selects an available chapter, investigates relevant evidence where supported, and submits a solution to a programming challenge.

System Architecture. The frontend is built with React and Vite. Three.js, React Three Fiber, and Drei support the 3D scene and related interaction components. The coding workflow communicates with a backend service that compiles and runs Java submissions. The game state tracks challenge completion and progression. The proposed IRI assistant is a separate future integration and is not presented as an already deployed AI feature.

Learning Workflow. The player is introduced to the missing-notebook case, reviews the case map, enters an available chapter, reads the narrative context, and completes the corresponding Java challenge. The judge compiles and executes the submitted code and compares output with test cases. A successful result can award experience points or coins and enable the next stage, subject to the implemented progression rules.

Chapter Structure. The project is organized into six DSA chapters: Arrays, Linked Lists, Stacks and Queues, Trees, Graphs, and Dynamic Programming. The project description reports a challenge bank of 40 Java problems across these topics. The exact distribution of problems per chapter should be taken from the final challenge data rather than inferred.

3D Investigation. The opening Array Fair environment is described as a third-person exploration scene. The player can move through the environment and interact with evidence points. The initial evidence sequence includes a sign-in sheet, a fair office terminal, and a torn page. The reported scope places the most developed 3D exploration in Array Fair; full 3D exploration across all chapters remains a future extension.

Java Judging. Submitted Java code is compiled using javac and executed using java. The judge evaluates program output against sample and hidden test cases and returns a verdict. Compilation errors and incorrect answers should be communicated separately where supported by the implementation. Execution of untrusted code requires isolation, time limits, memory limits, and controlled file and network access; these safeguards must be confirmed in the actual backend before making security claims.

Progression and Rewards. The game connects successful challenge completion with experience points, coins, and chapter unlocking. The system should ensure that repeated submissions do not unintentionally duplicate rewards and that the displayed progress reflects the stored game state. Persistence across refreshes or sessions should only be claimed if it is implemented and tested.

IV. SYSTEM IMPLEMENTATION

The implementation is divided into the frontend, 3D environment, challenge presentation, Java judging workflow, and progression logic. This modular arrangement separates the user experience from the code execution process and allows individual components to be tested independently.

Frontend Module. React and Vite provide the browser application structure. Reusable interface components support the landing page, introductory screens, case map, dialogue, chapter challenges, and progression indicators. The interface presents the investigation context and gives the player a clear path from exploration to coding tasks.

3D Scene Module. Three.js with React Three Fiber and Drei supports the 3D environment. Character movement, camera-follow behaviour, scene objects, evidence markers, and interaction panels form the described exploration workflow. The extent and reliability of each interaction should be confirmed against the final source code.

Challenge Module. The challenge bank organizes Java problems by DSA topic. Each problem is described as containing a statement, input and output requirements, starter code, sample cases, hidden cases, and hints. The actual challenge definitions should be reviewed to confirm that all 40 problems and their test cases are present in the release version.

Judge Module. The backend compiles the submitted Java program, runs the resulting program, and compares the output with expected results. It should handle compilation failures, runtime errors, timeouts, malformed submissions, and unavailable execution resources through controlled responses. Hidden test data should not be exposed to the client.

Progression Module. Successful completion updates the player's progress and may award experience points and coins. Chapter unlocking is based on completion state. Reward calculations, duplicate-completion handling, and refresh behaviour should be covered by automated tests.

IRI Assistant. IRI is planned as an AI-powered investigation assistant. The intended functions include contextual hints, explanations of programming errors, and guidance on algorithmic reasoning. Since the available project description identifies this integration as planned work, this paper does not claim

that AI-generated hints are currently implemented or evaluated.

Implementation Scope. The described prototype includes the main investigation interface, a six-chapter DSA structure, Java challenges, a Java judging workflow, progression rewards, and a 3D Array Fair environment. Full 3D exploration in every chapter, persistent progress where not yet implemented, the complete case-ending sequence, and IRI integration require confirmation or further development.

V. ADVANTAGES

1. Contextual Learning. The detective storyline gives programming exercises a shared objective and makes challenge completion part of the investigation.
2. Practical Coding. Java submissions are intended to be compiled and executed, providing a more authentic evaluation path than static answer selection.
3. Topic-Based Organization. The six chapters organize practice around major DSA topics and support sequential learning.
4. Immediate Feedback. Judge verdicts can help learners identify whether a solution compiles and passes the supplied test cases.
5. Motivation Through Progression. Clues, experience points, coins, and chapter unlocking provide visible progress indicators.
6. Interactive Exploration. The 3D Array Fair environment connects evidence collection with a programming challenge.
7. Extensible Architecture. The frontend, 3D scene, challenge bank, judge, and progression components can be developed and tested separately.

VI. RESULTS AND ANALYSIS

The project description identifies a prototype that connects detective-themed navigation, DSA challenges, Java execution, and progression rewards. It reports a challenge bank of 40 Java problems across six DSA areas and a 3D investigation environment focused on Array Fair. These are project-scope details, not results from a controlled educational experiment.

Functional Analysis. The main workflow to verify is navigation from the landing page to the introduction and case map, chapter access, challenge loading, Java submission, verdict display, reward allocation, and chapter unlocking. The final manuscript should report only the test cases that were actually executed against the submitted code.

Judge Evaluation. Test cases should include accepted solutions, incorrect answers, compilation errors, runtime exceptions, timeouts, empty submissions, and malformed inputs. Hidden test cases should be checked without exposing their contents to the frontend. No pass rate or execution-time benchmark is reported here because verified test logs were not provided.

3D Interaction Analysis. The Array Fair scene should be tested for character movement, camera behaviour, evidence accessibility, interaction prompts, and transition from evidence collection to the associated challenge. Additional chapters should not be described as fully explorable in 3D unless their scenes and interactions are implemented.

Progression Analysis. Tests should verify that successful submissions update the correct challenge state, rewards are granted according to the defined rules, locked chapters remain inaccessible, and repeat submissions do not produce unintended duplicate rewards. Persistence should be tested separately if the game is expected to preserve state after a refresh.

Educational Evaluation. A future study could measure task completion, time on task, challenge success rate, usability, learner confidence, and retention. A suitable comparison group or pre-test/post-test design would be needed before claiming that NEXAURA improves DSA learning outcomes.

Limitations. The current paper does not include a verified quantitative user study, a complete performance benchmark, or a security audit of the Java execution service. The described 3D implementation is concentrated on Array Fair, and IRI remains planned work. These limitations should be addressed before making broader claims about effectiveness, scalability, or security.

VII. CONCLUSION

This paper presented NEXAURA: The Missing Notebook, a browser-based educational detective game that connects DSA practice with a story-driven investigation. The project combines a React and Vite interface, a Three.js-based 3D environment, six DSA chapters, a reported bank of 40 Java challenges, and a backend workflow for compiling and evaluating Java submissions. Narrative clues and progression rewards are intended to connect coding tasks with the missing-notebook case.

The project provides a foundation for combining interactive exploration and executable programming practice in one learning environment. However, the educational impact and operational reliability cannot be established from feature descriptions alone. Final claims should be validated against the released code, execution logs, and user testing. Future work includes broader 3D chapter coverage, completion of the story arc, improved progress persistence, robust isolation of submitted code, and evaluation of the planned IRI assistant.

VIII. FUTURE ENHANCEMENTS

IRI AI Assistant. Integrate an AI service to provide progressive hints, explain compilation and runtime errors, and guide learners through algorithmic reasoning without immediately revealing complete solutions. The feature should be evaluated for correctness, usefulness, latency, and safe handling of user submissions.

Expanded 3D Chapters. Extend interactive environments beyond Array Fair so that each DSA chapter can use relevant spatial or narrative interactions. The 3D elements should reinforce the concept being taught rather than function only as decoration.

Complete Investigation Arc. Add and test transitions between all evidence items, suspects, chapter outcomes, and the final resolution of the missing-

notebook mystery.

Progress Persistence. Store chapter completion, earned rewards, and evidence state reliably if cross-session continuity is required. Define how progress is restored and how duplicate rewards are prevented.

Secure Code Execution. Isolate Java submissions in a restricted execution environment with CPU and memory limits, execution timeouts, controlled filesystem access, and disabled or restricted network access. Conduct a security review before exposing the judge to untrusted public submissions.

Learning Analytics. Record privacy-conscious measures such as attempts per challenge, completion rates, common compilation errors, and time on task. Use these measures to identify confusing challenges and improve the learning sequence.

Usability and Learning Study. Conduct structured testing with representative learners, using defined tasks and validated questionnaires where appropriate. Report participant numbers, procedure, metrics, limitations, and results transparently.

Accessibility and Compatibility. Test keyboard navigation, readable contrast, responsive layouts, reduced-motion options, and performance on commonly used desktop and mobile browsers.

IX. REFERENCES

[1]. Deterding, S., Dixon, D., Khaled, R., and Nacke, L. (2011). From Game Design Elements to Gamefulness: Defining Gamification. Proceedings of the 15th International Academic MindTrek Conference, 9–15. https://doi.org/10.1145/2181037.2181040

[2]. Gee, J. P. (2003). What Video Games Have to Teach Us About Learning and Literacy. Palgrave Macmillan.

[3]. Prensky, M. (2001). Digital Game-Based Learning. McGraw-Hill.

[4]. ACM/IEEE-CS Joint Task Force. Computer Science Curricula 2023. Association for Computing Machinery and IEEE Computer Society. https://csed.acm.org/

[5]. React Documentation. https://react.dev/

[6]. Vite Documentation. https://vite.dev/

[7]. Three.js Documentation. https://threejs.org/docs/

[8]. React Three Fiber Documentation. https://r3f.docs.pmnd.rs/

[9]. Drei Documentation. https://drei.docs.pmnd.rs/

[10]. Oracle. Java Development Kit Documentation. https://docs.oracle.com/en/java/

manuscript.