PRESIDENCY UNIVERSITY

Itgalpura, Rajanakunte, Bengaluru - 560064

School of Computer Science & Engineering

A Project Report on Mini-Project

"ALGORITHM ARCADE USING ANDROID STUDIO"

Submitted in partial fulfillment of the requirement for the course

Mobile Application Development Lab (**CSE2506**)

Submitted by

Student Name	Roll No
SIDDHARTH KUMAR	20231CSE0113
MUSKAN KUMARI	20231CSE0117
MUSKAN NAYAK	20231CSE0067
SINCHANA R	20231CSE0107

Under the supervision of

Dr. Saurabh Sarkar
Assistant professor
Department of Computer Science and Engineering

ABSTRACT

"**Algorithm Arcade**" is an Android application designed as a unique, engaging, and educational platform for the interactive exploration of fundamental computer science concepts. The project successfully consolidates six distinct algorithmic challenges—MindMap, Tower of Hanoi, N-Queens, Rat in a Maze, Sudoku, and Taxi Tycoon—into a single, unified experience.

Each mini-game serves as a practical, visual demonstration of key computational problem-solving techniques, primarily focusing on Recursion and Backtracking. The application is developed entirely on the Android platform Android Studio using Java and adheres strictly to Material Design principles, ensuring a modern, intuitive, and visually consistent user interface that features interactive cards and dynamic gradient backgrounds.

A core feature is an integrated AI-powered Chatbot to provide users with immediate assistance, explanations, and hints. By gamifying complex logic, Algorithm Arcade effectively bridges the gap between theoretical knowledge and practical application, offering students and enthusiasts an enjoyable way to achieve a deeper understanding of efficient problem-solving algorithms.

INTRODUCTION

The study of computer science algorithms is fundamental, yet the theoretical nature of subjects like recursion and backtracking often makes them challenging for students and enthusiasts to fully grasp. Traditional learning methods often rely on static diagrams and pseudocode, which can fail to convey the dynamic, step-by-step process of how these algorithms execute complex problem-solving. This project addresses that gap by presenting a practical, interactive, and entertaining learning solution.

The "**Algorithm Arcade**" is a modern, single-application Android platform designed to make complex algorithms intuitive and engaging. Instead of passive study, users actively interact with these concepts through a curated collection of mini-games. This approach transforms abstract mathematical and logical challenges into tangible, playable experiences, thereby deepening user understanding through direct, iterative feedback.

The application's core features center around usability and educational depth. It integrates six classic algorithm problems—including the Tower of Hanoi, N-Queens, and Sudoku—each serving as a powerful demonstration of recursion and backtracking in action. The application is built using Java on the Android SDK, adhering to modern Material Design principles to ensure a seamless and aesthetically pleasing user experience. Furthermore, an integrated AI-powered Chatbot provides real-time support, hints, and detailed explanations, acting as an ever-present tutor.

This report will proceed by first detailing the comprehensive requirements and design goals of the Algorithm Arcade. It will then provide an in-depth exploration of the architectural decisions and the Java-based implementation of the core game logic and algorithms. Finally, the report will conclude with a discussion of the project's key contributions, the challenges encountered during development, and potential avenues for future enhancements.

RELATED WORK

The "**Algorithm Arcade**" project is fundamentally driven by pedagogical research indicating that active, visual, and gamified learning significantly enhances the comprehension and retention of complex computer science concepts, such as recursion and backtracking.

Despite the recognized value of interactive tools in Data Structures and Algorithms (DSA) education, a clear gap exists in the form of a consolidated, mobile-first, and game-centric application.

Existing solutions are often restricted to desktop-based visualizers or competitive coding platforms that bypass the crucial step-by-step conceptual visualization.

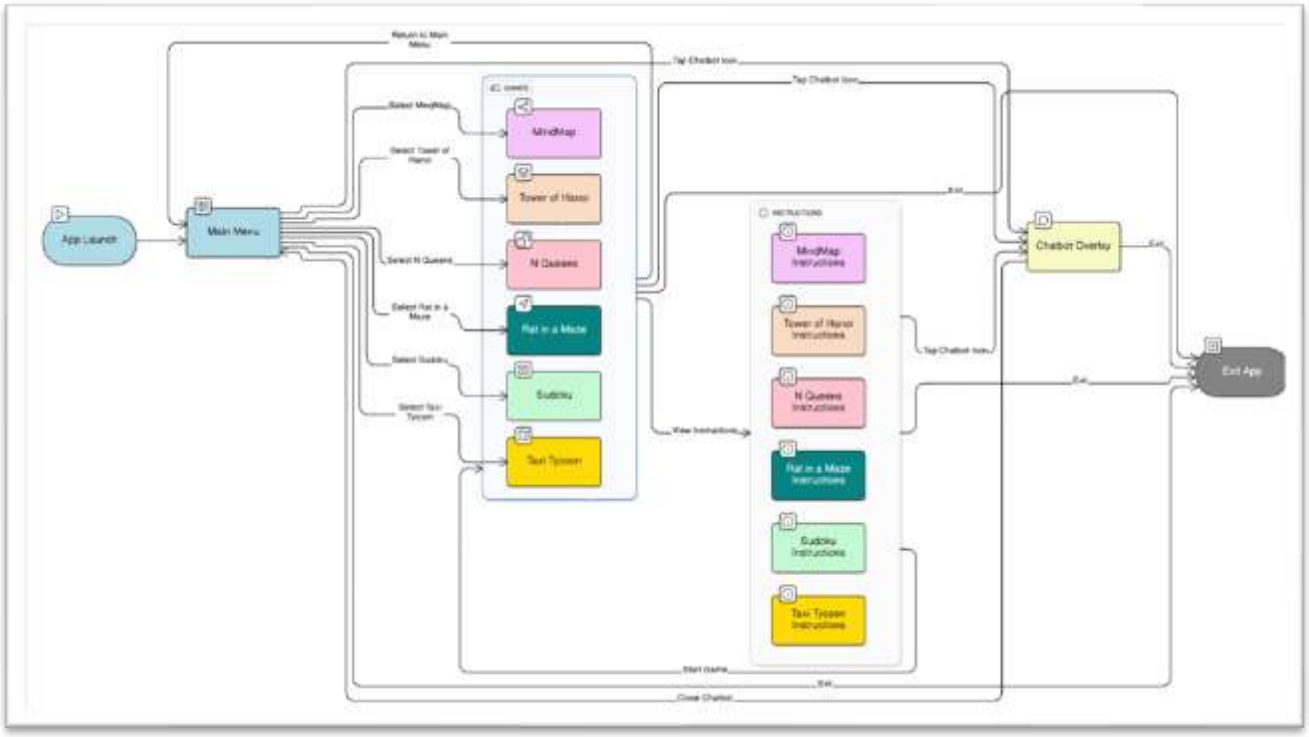
The core objective of this mini-project was therefore to bridge this divide by developing a unified Android application that ensures algorithmic fidelity across six distinct mini-games (e.g., Tower of Hanoi, N-Queens, Sudoku), prioritizes an intuitive user experience utilizing Material Design, and integrates an AI-powered Chatbot for real-time tutorial assistance.

When benchmarked against existing solutions, such as traditional online visualizers or pure logic puzzle apps, the "Algorithm Arcade" differentiates itself by providing a persistent, casual, and academically grounded learning environment on a mobile platform.

While the current implementation successfully achieves the demonstration of core concepts, the future scope is robust, encompassing the integration of multiplayer modes, the expansion to cover more advanced algorithms like Dynamic Programming, the implementation of personalized learning paths, and potential evolution into an open-source project for wider educational adoption.

IMPLEMENTATION

The "**Algorithm Arcade**" application was implemented as a native Android application, utilizing a robust structure to separate the user interface layer from the core algorithmic logic.



DESCRIPTION

The "**Algorithm Arcade**" application was engineered as a native mobile solution, ensuring high performance, deep system integration, and a consistent user experience.

Development Environment and Core Technologies

- **Android Studio**: The official Integrated Development Environment (IDE) was used for all stages of development, including coding, debugging, and testing.

- **Java**: This was the primary programming language, used for implementing all game logic, handling UI interactions, and managing backend functionalities, providing a robust foundation for the application's complex algorithmic structures.
- **Android SDK**: The necessary set of development tools, APIs, and libraries were utilized to build and target modern Android operating systems.
- **Game Logic Implementation**: Dedicated Java classes were established to house the efficient, runnable code for algorithms such as Recursion (Tower of Hanoi) and Backtracking (N-Queens, Sudoku), instrumented to trigger visual updates on the UI.

UI/UX Framework

The application's interface was designed around modern Material Design principles to achieve an intuitive, engaging, and consistent user experience:

- **Material Design**: Implemented extensively across all screens for visual appeal. This includes standardized components, navigation patterns, and typography.
- **androidx.cardview.widget.CardView**: Utilized on the main menu to display each of the six games as distinct, interactive cards, enhancing navigation and visual modularity.
- **Gradient Backgrounds**: Dynamically applied across various screens using drawable resources to enhance visual appeal and provide thematic distinction between different game modules.
- **Material Components for Android**: Employed for standard UI elements like responsive buttons, text fields, and integrated bottom navigation to maintain consistency and accessibility.

Chatbot Integration

- **Chatbot Integration**: The application utilizes the Groq API to power its integrated chatbot. This integration allows for extremely fast, low-latency responses. User queries are sent to the Groq LLM endpoint along with contextual metadata (e.g., the current game being played), allowing the chatbot to provide explanations of the underlying algorithm, and steps toward a solution without requiring the user to exit the game environment.

SYSTEM REQUIREMENT SPECIFICATION

Game Instructions

The "Algorithm Arcade" provides a consistent user experience for accessing and understanding each game. Before playing, users can view detailed instructions.

General Navigation

Main Menu: Upon launching the app, users are presented with a main menu displaying all available games as interactive cards. Each card features the game's name and a visual icon.

Game Selection: Tapping on a game card navigates the user to that specific game's instruction screen.

Chatbot Access: A dedicated icon is available across the app to summon the integrated chatbot for assistance.

Game-Specific Design Elements and Algorithms

1. MindMap

Concept: A visual puzzle where users connect ideas or concepts, often involving graph traversal or tree structures.

Algorithm Integration: Potentially uses algorithms like Depth-First Search (DFS) or Breadth-First Search (BFS) for validating connections or finding paths between nodes, or simply representing hierarchical data structures.

Gameplay: Users drag and drop nodes, draw connections, or categorize items based on a given problem.

2. Tower of Hanoi

Concept: A classic mathematical puzzle involving moving disks between three pegs.

Algorithm Integration: A direct and powerful demonstration of Recursion. The game visually depicts the recursive steps needed to move disks.

Gameplay: Users move disks one at a time, adhering to the rule that a larger disk cannot be placed on top of a smaller disk.

3. N-Queens

Concept: Placing N non-attacking queens on an N×N chessboard.

Algorithm Integration: A quintessential example of Backtracking. The game allows users to manually attempt placements and observe the backtracking process or solve it automatically.

Gameplay: Users place queens on a chessboard, with the system highlighting invalid placements (queens attacking each other) and potentially guiding them through valid solutions.

4. Rat in a Maze

Concept: Finding a path for a 'rat' from a source to a destination in a given maze.

Algorithm Integration: Another strong demonstration of Backtracking and Depth-First Search (DFS). The game visually traces the rat's path, showing where it explores and when it backtracks.

Gameplay: Users navigate a maze, or the game can animate the algorithmic solution, allowing observation of the pathfinding process.

5. Sudoku

Concept: Filling a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 subgrids contains all the digits from 1 to 9.

Algorithm Integration: Solved efficiently using Backtracking. The game can either challenge the user to solve it or demonstrate the algorithm finding a solution step-by-step.

Gameplay: Users input numbers into a grid, with the app validating moves and potentially offering a solver feature powered by the backtracking algorithm.

6. Taxi Tycoon

Concept: A simulation or strategy game where the player manages a taxi fleet, optimizes routes, and maximizes profit.

Algorithm Integration: Can involve a variety of algorithms, such as:

Shortest Path Algorithms: (e.g., Dijkstra's Algorithm, A* Search) for optimizing taxi routes.

Graph Traversal: For navigating city maps.

Greedy Algorithms: For decision-making (e.g., which customer to pick up next based on proximity/profit).

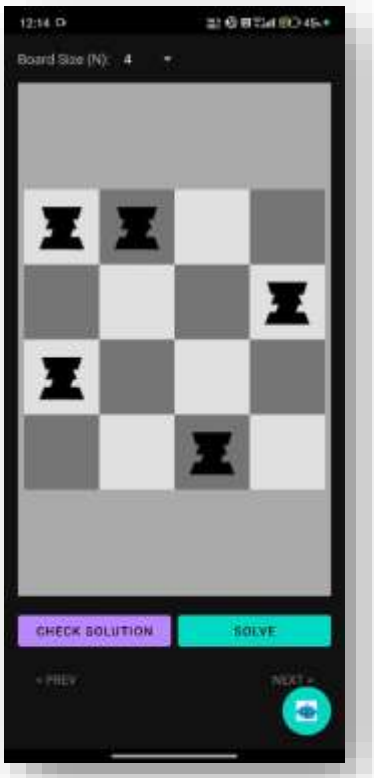
Gameplay: Users make strategic decisions regarding taxi assignments, route planning, and city expansion, observing the impact of underlying algorithms on their success.

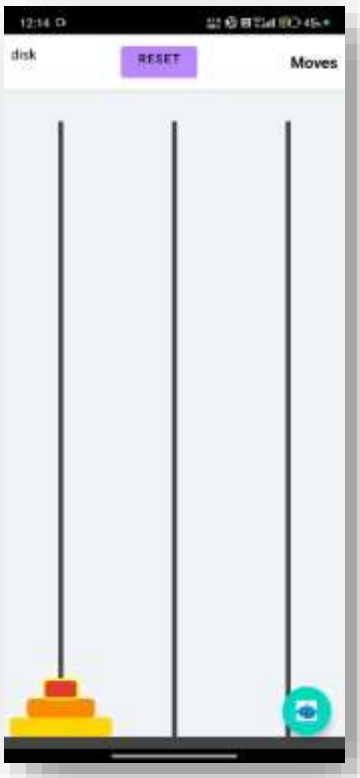
RESULTS

The development of the "**Algorithm Arcade**" application successfully met all outlined objectives, resulting in a stable, multi-functional Android application built on the Java language and Material Design framework.

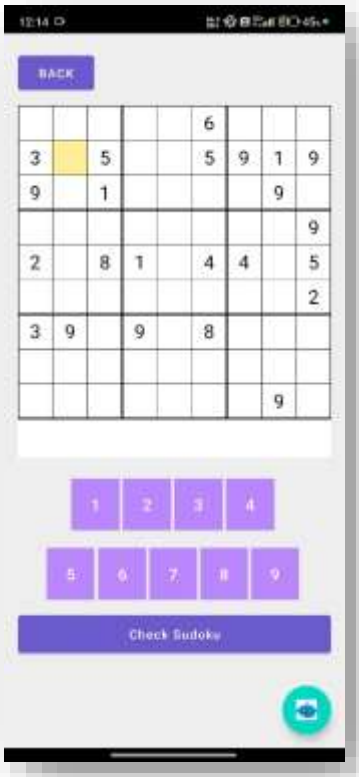
Testing and verification focused on two main areas: the successful deployment of the core application structure and the functional accuracy and performance of the integrated algorithms.

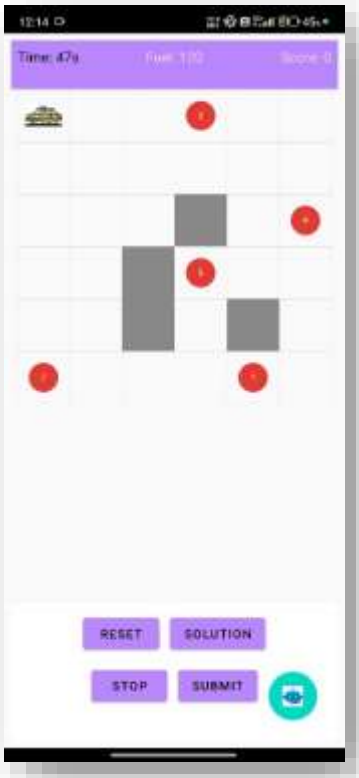
In summary, the implementation verified that the application successfully translates complex computer science algorithms into interactive, educational, and performance-validated mini-games. The consistent, efficient execution of recursion, backtracking, and shortest path algorithms forms the robust core of the "**Algorithm Arcade**"

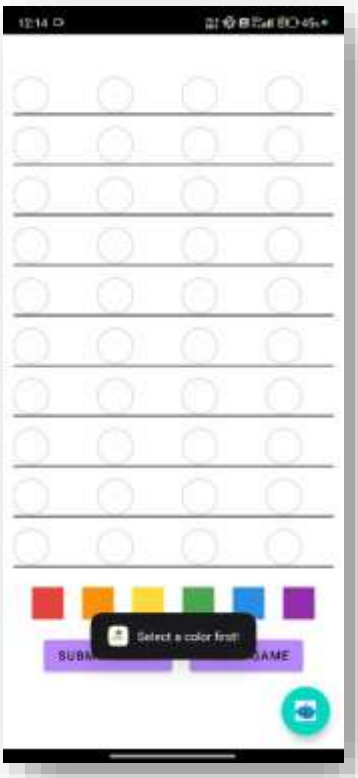

Board Size (N): 4
CHECK SOLUTION
SOLVE
< PREV
NEXT >


disk
RESET
Moves


NEW GAME
SOLVE


BACK
Check Sudoku


RESET
SOLUTION
STOP
SUBMIT


Select a color first!

CONCLUSION

"**Algorithm Arcade**" successfully achieved its primary objective by delivering an innovative and comprehensive Android platform that transforms complex computer science algorithms—such as Recursion (Tower of Hanoi) and Backtracking (N-Queens, Sudoku)—into six distinct, highly engaging, and accessible game experiences.

The project demonstrated robust technical implementation using Java and the Android SDK, underpinned by a premium, intuitive user experience achieved through strict adherence to Material Design principles, including interactive cards and gradient backgrounds.

Furthermore, the integration of a responsive AI chatbot successfully layered intelligent, on-demand assistance over the gameplay, positioning "Algorithm Arcade" as a powerful self-learning tool and a testament to the effective synthesis of robust software engineering and pedagogical design.

Looking ahead, to maximize its educational and commercial viability, future development will focus on four key areas: enhancing the transparency of learning through visual debugging (implementing instruction tracing and variable state inspection); boosting user engagement by introducing multiplayer and competitive modes (such as leaderboards); broadening the application's pedagogical scope by expanding the algorithmic library (to include visualizers for sorting algorithms and data structures); and evolving the chatbot into a true adaptive tutor with difficulty scaling and integrated conceptual assessments.

REFERENCES

1. Google Developers, “Material Design Guidelines.” Available: https://material.io/design
2. Android Developers, “Material Components for Android.” Available: https://developer.android.com/guide/topics/ui
3. Android Developers, “Android Studio User Guide.” Available: https://developer.android.com/studio/intro
4. Android Developers, “Android Jetpack Architecture Components.” Available: https://developer.android.com/jetpack
5. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C., Introduction to Algorithms, MIT Press.
6. Aho, A. V., Hopcroft, J. E., & Ullman, J. D., Data Structures and Algorithms, Addison-Wesley.
7. Dasgupta, S., Papadimitriou, C., & Vazirani, U., Algorithms, McGraw-Hill.
8. Knuth, D. E., The Art of Computer Programming, Addison-Wesley.
9. Nesbitt, G. J., “The Tower of Hanoi—Mathematical Properties,” The Mathematical Gazette, 1966.
10. Yato, T., & Seta, T., “Complexity and Completeness of Sudoku,” International Conference on Computers and Games, 2002.
11. Dijkstra, E. W., “A Note on Two Problems in Connection with Graphs,” Numerische Mathematik, 1959.
12. Hart, P. E., Nilsson, N. J., & Raphael, B., “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” IEEE Transactions on Systems Science and Cybernetics, 1968.
13. Google Cloud, “Dialogflow Documentation—Building Chatbots.” Available: https://cloud.google.com/dialogflow/docs
14. OpenAI, “Developer Documentation—Chatbot Integration.” Available: https://platform.openai.com/docs
15. Clifton, I. G., Android User Interface Design: Implementing Material Design for Developers, Addison-Wesley.
16. Android Developers, “Graphics and Custom Views in Android.” Available: https://developer.android.com/guide/topics/graphics
17. Pressman, R. S., & Maxim, B. R., Software Engineering: A Practitioner’s Approach, McGraw-Hill.