“MAZE SOLVER”

TRIBHUVAN UNIVERSITY

INSTITUTE OF SCIENCE AND TECHNOLOGY

PRIME COLLEGE

NAYABAZAR, KHUSIBUN

BACHELORS IN COMPUTER APPLICATION

PROJECT PROPOSAL

SUBMITTED BY:

SAGAN KRISHNA TAMRAKAR (20810404)

ABSTRACT

The Maze Solver is an Android application whose primary goal is to solve the mazes provided by the user or the existing mazes. It includes features like creating custom mazes and the ability to save them for future usage. This app helps user to understand how the maze solving process actually works. However small problems such as storage issues might occur. To make is more helpful, the system should be easy to use and accessible to more people. Understanding its benefits and limitations can help people improve their maze solving skills.

Written By: Sagan Krishna Tamrakar

Table of Contents

1. INTRODUCTION

1.1 Introduction

Maze-Solving Applications has long been a captivating challenge in the field of computer science, offering valuable insights into pathfinding algorithms and optimization techniques. As one of the most common problems encountered in both academic and real-world applications, mazes provide an excellent environment for testing different algorithms. Different algorithms have their unique approach and benefits, are often applied to a variety of practical problems, from navigation systems to video games and puzzle-solving applications.

The goal of the Maze-Solver project is to create an intuitive, Android-based mobile application that empowers users to generate, solve, customize, and save mazes for future use. By utilizing Java as the core programming language, the application will deliver a smooth and responsive experience that allows users to easily navigate through various maze levels. With the addition of customizable features, users can adjust mazes providing endless possibilities for creating unique challenges. Furthermore, by incorporating popular maze-solving algorithms such as A* and Dijkstra's Algorithm, users will be able to witness the efficiency and effectiveness of these algorithms in real-time, making the maze-solving process both educational and entertaining.

Incorporating these algorithms into a mobile application serves to enhance the user experience by offering practical solutions and interactive elements. While A* and Dijkstra's algorithms are designed to find the shortest path in a maze, their implementation in the application will also highlight differences in performance, efficiency, and usability. Whether users are learning about the underlying principles of these algorithms or simply looking for a fun and challenging puzzle experience, this platform will provide an engaging way to explore the fascinating world of maze-solving.

1.2 Problem Statement

Traditional maze-solving applications often lack customization features, limiting user interaction and engagement. Many applications provide only pre-defined mazes with fixed structures, restricting creativity and adaptability for users. Furthermore, the absence of pathfinding algorithms in many existing apps results in inefficient solutions and limited learning opportunities for users interested in algorithmic problem-solving.

Additionally, users may find it frustrating when they cannot save their progress or revisit previously solved mazes. This project aims to bridge these gaps by introducing an application that not only generates and solves mazes efficiently but also allows users to create, customize, and store mazes for future use. By integrating A* and Dijkstra's algorithms, this application ensures optimal pathfinding solutions, making maze-solving both an educational and engaging experience. Many applications do not allow users to save their progress or modify maze structures. This project addresses these gaps by offering a dynamic and interactive maze generation, customization, and saving functionality within a mobile application.

1.3 Objectives

- To develop an Android-based maze-solving application using Java.
- To implement A* and Dijkstra's algorithms for automatic maze-solving.
- To allow users to save and reload customized mazes.

1.4 Scope and Limitation

The following are the scopes for Maze-Solver Application:

- Users can generate mazes with adjustable sizes and complexities.
- Users can navigate manually or use A* and Dijkstra's algorithms for automatic solving.
- Provides an intuitive and engaging user interface.
- Allows users to store and reload mazes for future use.
- Helps users understand pathfinding algorithms and their applications.
- Does not require an internet connection for core functionalities.

The following are the limitations of the application:

- Supports only A* and Dijkstra's algorithms.
- May lag on lower-end devices when handling large mazes.
- No multiplayer or competitive maze-solving features.
- Not designed for navigation, robotics, or AI-based real-world solutions.
- Users cannot create unique maze shapes or multi-floor mazes.
- The number of saved mazes depends on device storage capacity.

2. IMPLEMENTATION

2.1 Development Model

For this project, we have used Agile method. Agile Methodology is an iterative and flexible development approach that focuses continuous improvements, customer collaboration, and rapid delivery. Unlike traditional linear models, Agile focuses on small, incremental updates and adapts to changing requirements based on user feedback. Since the Maze-Solving Application involves features like maze generation, pathfinding algorithms (A and Dijkstra's), UI interactions, and storage*, Agile allows for gradual implementation and testing rather than building everything at once.

3. Literature Review

"Pathfinding Algorithms for Maze Solving: A Comparative Study" (Lee et al., 2019): This study examines various pathfinding algorithms, comparing their efficiency in solving mazes. The research highlights the advantages of A* and Dijkstra's algorithms, emphasizing their accuracy, computational efficiency, and applicability in real-time problem-solving.

"Mobile Applications for Interactive Learning: Enhancing User Engagement through Gamification" (Chen et al., 2020): Chen et al. explore the role of interactive learning through mobile applications. The study emphasizes how gamification and user-friendly interfaces contribute to better user engagement and knowledge retention, supporting the application's goal of making maze-solving an enjoyable and educational experience.

"Customizable Digital Mazes: Enhancing User Creativity in Algorithmic Problem Solving" (Williams et al., 2018): This case study analyzes the impact of customizable maze applications on user creativity and engagement. The findings suggest that allowing users to create and modify mazes enhances their problem-solving skills and overall learning experience.

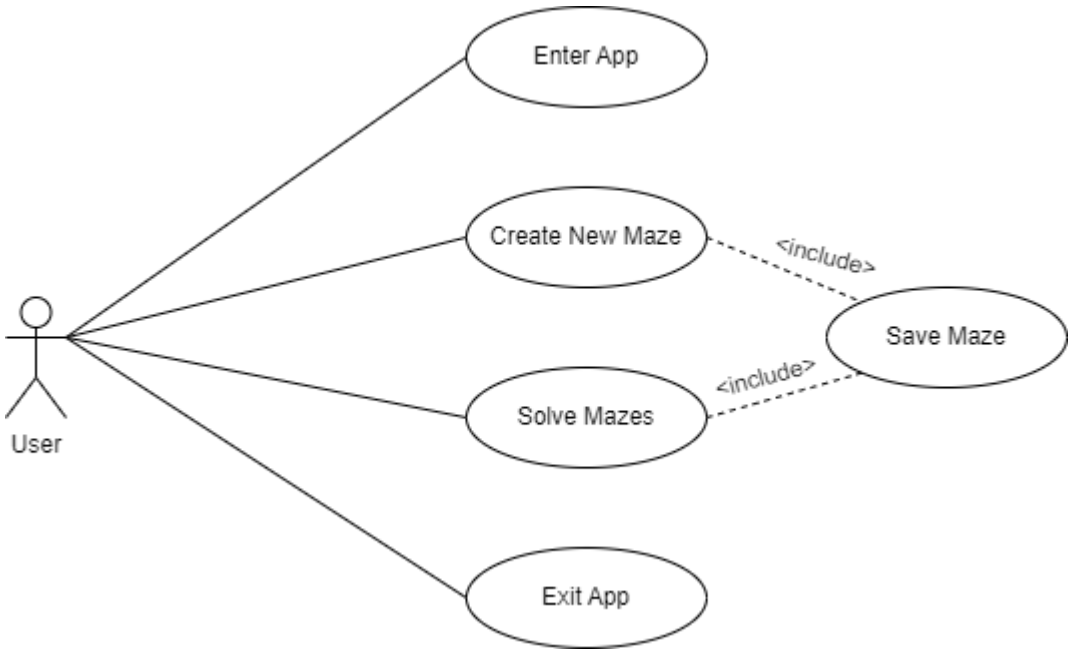
"Artificial Intelligence in Puzzle Games: The Role of Algorithmic Complexity" (Smith and Patel, 2017): Smith and Patel investigate how artificial intelligence and advanced algorithms contribute to the effectiveness of puzzle-solving applications. The research supports the implementation of A* and Dijkstra's algorithms, as they provide optimal pathfinding solutions while maintaining computational efficiency. "Algorithmic Approaches to Maze Solving" (Smith et al., 2020): Discusses various pathfinding algorithms such as Depth-First Search (DFS), Breadth-First Search (BFS), and A*.

4. Requirement Analysis

4.1 Functional Requirement

The Maze Solver is designed to provide an efficient and intelligent solution for navigating complex mazes. The functional requirements ensure the system's capability to analyze and solve mazes optimally using the A* search algorithm. The application should accurately recognize and interpret maze structures based on user inputs, allowing users to visualize the shortest path in real time. Interactive visualization enhances user experience by dynamically displaying the solving process. Additionally, performance optimization ensures efficient pathfinding, even in large and complex mazes.

4.1.1 Use Case Diagram



4.2 Non-Functional Requirements:

- **Performance:** The app is expected to solve complex mazes efficiently. It should be able to handle rendering without performance degradation.
- **Usability:** The interface should be user-friendly ensuring easy navigation and smooth experience.
- **Security:** There should be secure local storage for saving user mazes.
- **Reliability:** The application is required to maintain 24/7 availability, with minimal downtime allocated for maintenance and updates. It should also be able to handle and recover from unexpected failures or errors.

4.2.2 Feasibility Study

i. Technical Feasibility

The technical feasibility study assesses the viability of implementing the Maze Solver Using A* Algorithm from a technological standpoint. This evaluation aims to ensure that the proposed system can be successfully developed, implemented, and maintained, considering the available technology infrastructure and resources.

- **Hardware Requirements:**

  The Maze Solver Using A* Algorithm requires minimal hardware resources, making it accessible on a wide range of devices. A standard smartphone or tablet with sufficient processing power is adequate for running the application smoothly. The device should have some free storage space to save and retrieve maze data efficiently. Additionally, a stable internet connection is recommended for accessing cloud-based features, if implemented.

- **Software Requirements:**

  The application is developed using Android Studio, with Java as the primary backend programming language. The app is designed to run seamlessly on Android devices, ensuring compatibility across various screen sizes and resolutions. The user interface is built using XML, while the pathfinding logic is implemented with the A* algorithm for efficient maze solving. These software components ensure a responsive and interactive user experience, making maze-solving both intuitive and efficient.

ii. Operational Feasibility

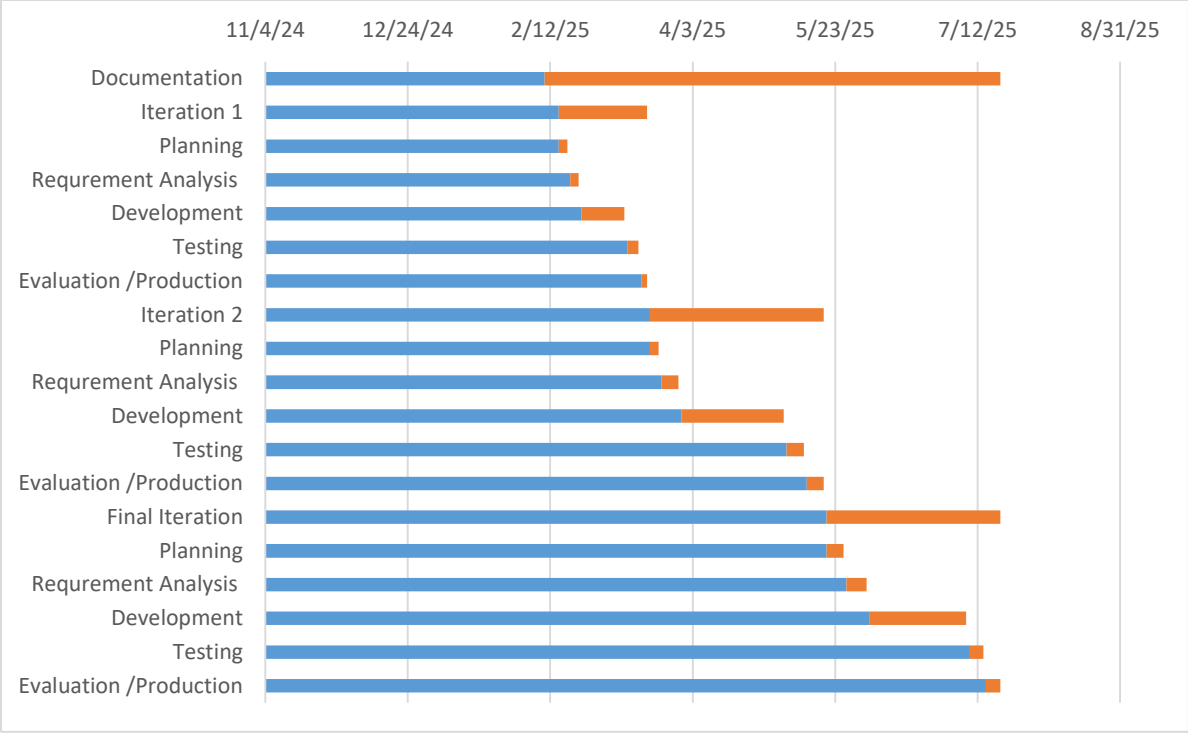
This checks if the application is easy to use and does not require advanced technical knowledge for its usage. It should support offline mode for solving mazes, meaning it should not require any network connection for its operation.

iii. Economic Feasibility

The system must be cost-effective and provide good value. Since it's being developed as a project, costs are low, and most resources are already available. This makes the Maze Solver an affordable solution for enhancing trekking.

iv. Schedule Feasibility

Task	Start Date	End Date	Duration	Progress
Documentation	2/10/2025	7/20/2025	160	100%
Iteration 1	**2/15/2025**	**3/18/2025**	**28**	**90%**
Planning	2/15/2025	2/18/2025	3	100%
Requirement Analysis	2/19/2025	2/22/2025	3	50%
Development	2/23/2025	3/10/2025	16	70%
Testing	3/11/2025	3/15/2025	4	80%
Evaluation /Production	3/16/2025	3/18/2025	2	100%
Iteration 2	**3/19/2025**	**5/19/2025**	**61**	**91%**
Planning	3/19/2025	3/22/2025	3	100%
Requirement Analysis	3/23/2025	3/29/2025	7	72%
Development	3/30/2025	5/5/2025	37	85%
Testing	5/6/2025	5/12/2025	7	100%
Evaluation /Production	5/13/2025	5/19/2025	7	100%
Final Iteration	**5/20/2025**	**7/20/2025**	**61**	**90%**
Planning	5/20/2025	5/26/2025	7	100%
Requirement Analysis	5/27/2025	6/3/2025	8	70%
Development	6/4/2025	7/8/2025	35	90%
Testing	7/9/2025	7/14/2025	6	100
Evaluation /Production	7/15/2025	7/20/2025	5	100


11/4/24
12/24/24
2/12/25
4/3/25
5/23/25
7/12/25
8/31/25
Documentation
Iteration 1
Planning
Requrement Analysis
Development
Testing
Evaluation /Production
Iteration 2
Planning
Requrement Analysis
Development
Testing
Evaluation /Production
Final Iteration
Planning
Requrement Analysis
Development
Testing
Evaluation /Production

5.Algorithm

5.1 A*(A-star) Algorithm

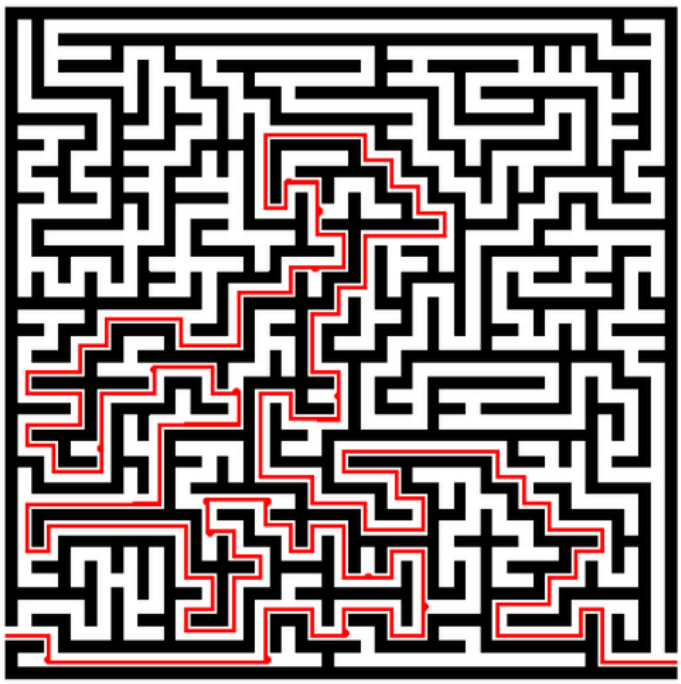
The A* algorithm is a smart way to find the shortest path between two points. It uses both the actual distance traveled so far and an estimate of how far it is to the goal to decide which path to take next. By focusing on the most promising paths, it quickly finds the best route. If the estimate is accurate, A* ensures the shortest path is found. It’s commonly used in video games, robots, and GPS systems because it’s both fast and reliable.

5.2 Dijkstra's Algorithm

Dijkstra's Algorithm is a graph-based shortest path algorithm used to find the shortest distance between a starting node and all other nodes in a weighted graph. It works by selecting the node with the smallest known distance, updating its neighbors with the shortest possible paths, and repeating this process until all nodes have been visited. The algorithm is widely used in network routing and mapping applications, such as GPS navigation. It guarantees the shortest path in graphs with non-negative weights and operates efficiently using a priority queue.

6. Expected Outcome

The main goal of this project is to create a fully functional Android application for generating, solving, and customizing mazes. Having the ability for users to save and reload their maze configurations. The final product should be something similar to the picture below.



7. References

1. Pathfinding Algorithms for Maze Solving: A Comparative Study (Lee et al., 2019)
2. Mobile Applications for Interactive Learning: Enhancing User Engagement through Gamification (Chen et al., 2020)
3. Algorithmic Approaches to Maze Solving (Smith et al., 2020)
4. Customizable Digital Mazes: Enhancing User Creativity in Algorithmic Problem Solving (Williams et al., 2018)