



Java Programming (19ECSP301)

Project Report

On

Maze Runner (Beat the maze within time)

Submitted by

Praveen Naik

01FE18BCS153

Prasad V Patil

01FE18BCS148

Team No:- 14

Faculty Incharge:- Madhura Shettar

SCHOOL OF COMPUTER SCIENCE & ENGINEERING

**HUBLI – 580 031
(India)**

Academic year 2020-21

INTRODUCTION

The goal of this project is to gain some experience in GUI programming, while simultaneously implementing and using the design practices and tools learned throughout the course.

We are going to implement a maze game using Java Eclipse IDE with GUI interface. The maze game is a game in which the purpose of the game is for the player to make their way from the start point to the end point. The layout of the maze is such that the player must follow a series of narrow winding corridors with many false turns and dead-ends until they eventually reach the finish point.

PROBLEM STATEMENT

The task is to design and implement a maze game using object oriented programming concepts and GUI interface (i.e. Java Swing). The game must have n number of levels which can be added by the admin. The user should be provided with a set of instructions about the game and there must be time bound within which he/she has to clear the level, failing to do so would block his/her entry to the next level. The user should also be provided with an option to view the solution to clear the level, which must be implemented using a shortest path search algorithm.

OBJECTIVES

1. Admin should be able to add or delete the levels.
 2. Users must register before entering the game.
 3. Users must login to access the game if he/she is an existing player.
 4. Users must clear initial levels to go to next levels.
 5. There will be time constraint for every level which distinguishes the level of difficulty.
 6. Users can exit the game at any point of time.
 7. In case the user fails to solve the maze, he can view the solution.
 8. After solving the maze the most optimal path will be displayed to the user if it exists.
 9. The user profile will be saved and will be available for next execution.
 10. There should be instructions available for both admin and users.
-

LIST OF CLASSES

1. MainMenu
2. UserInterface
3. MazeMapMaker
4. Maze
5. MapMakerTile
6. Instructions
7. Person
8. Player
9. User
10. Admin
11. CountdownTimer
12. Tile

DESIGN PATTERNS

1. **Singleton pattern :**

It is a design pattern that restricts the instantiation of a class to one "single" instance. This is useful when exactly one object is needed to coordinate actions across the system. Here admin class, Instructions class can be created through a singleton pattern.

2. **Observer Design pattern:**

An observer Pattern says that “just define a one-to-one dependency so that when one object changes state, all its dependents are notified and updated automatically”. For timer class and maze class we can use observer design patterns.

When the maze class starts the timer should start counting downwards and when the timer reaches zero that means when the timer stops, the game must stop automatically and the user must be unable to continue the game.

CLASS DIAGRAM

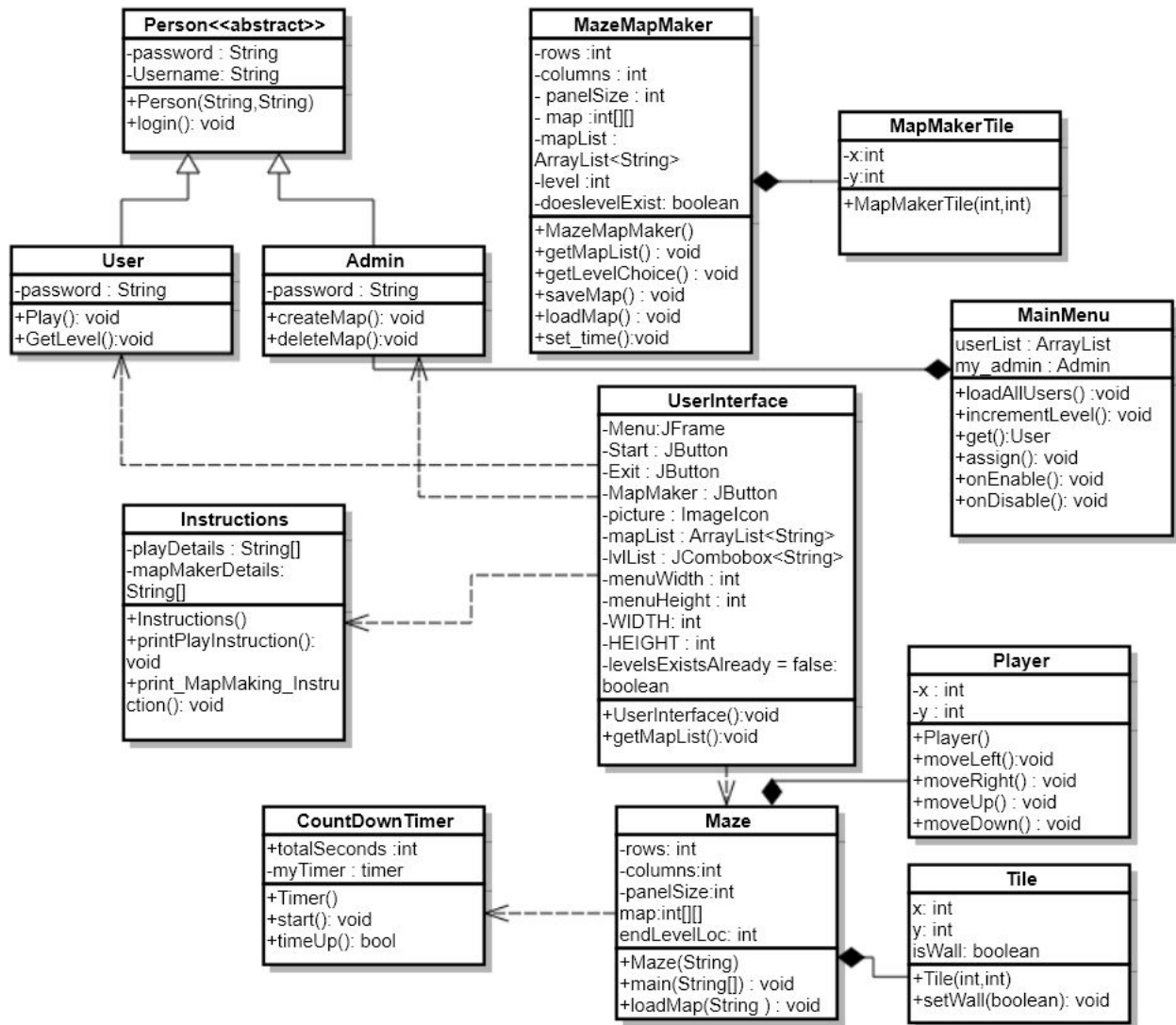


Fig.1: Class Diagram

DATA STRUCTURES AND ALGORITHMS USED

Data Structures:-

1. Array list data structure:-

ArrayList is a part of collection framework and is present in java.util package. It provides us with dynamic arrays in Java. Though, it may be slower than standard arrays but can be helpful in programs where lots of manipulation in the array is needed. This class is found in java.util package. It is used for storing:

- All the levels in Game
- Profile of User
- Time for each level

2. HashMap data structure:-

HashMap is a part of the Java collection framework. It uses a technique called Hashing. It implements the map interface. It stores the data in the pair of Key and Value. HashMap contains an array of the nodes, and the node is represented as a class. It uses an array and LinkedList data structure internally for storing Key and Value. Here it is used for mapping between levels and the respective time bound.

3. 2-D array data structure:-

2D array can be defined as an array of arrays. The 2D array is organized as matrices which can be represented as the collection of rows and columns. However, 2D arrays are created to implement a relational database look alike data structure. It provides ease of holding bulk of data at once which can be passed to any number of functions wherever required. We use it for storing the Maze Map.

4. List and stack data structure:-

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO (Last In First Out) or FILO (First In Last Out). The List interface provides a way to store the ordered collection. It is a child interface of Collection. It is an ordered collection of objects in which duplicate values can be stored. Since List preserves the insertion order, it allows positional access and insertion of elements. Both are used for Dijkstra's algorithm.

Algorithms:-

Dijkstra's Algorithm:-

Dijkstra's algorithm finds the least expensive path in a weighted graph between our starting node and a destination node, if such a path exists. At the end of the algorithm, when we have arrived at the destination node, we can print the lowest cost path by backtracking from the destination node to the starting node. We have used this algorithm for getting the shortest path between start point and the end point in the maze.

RESULTS

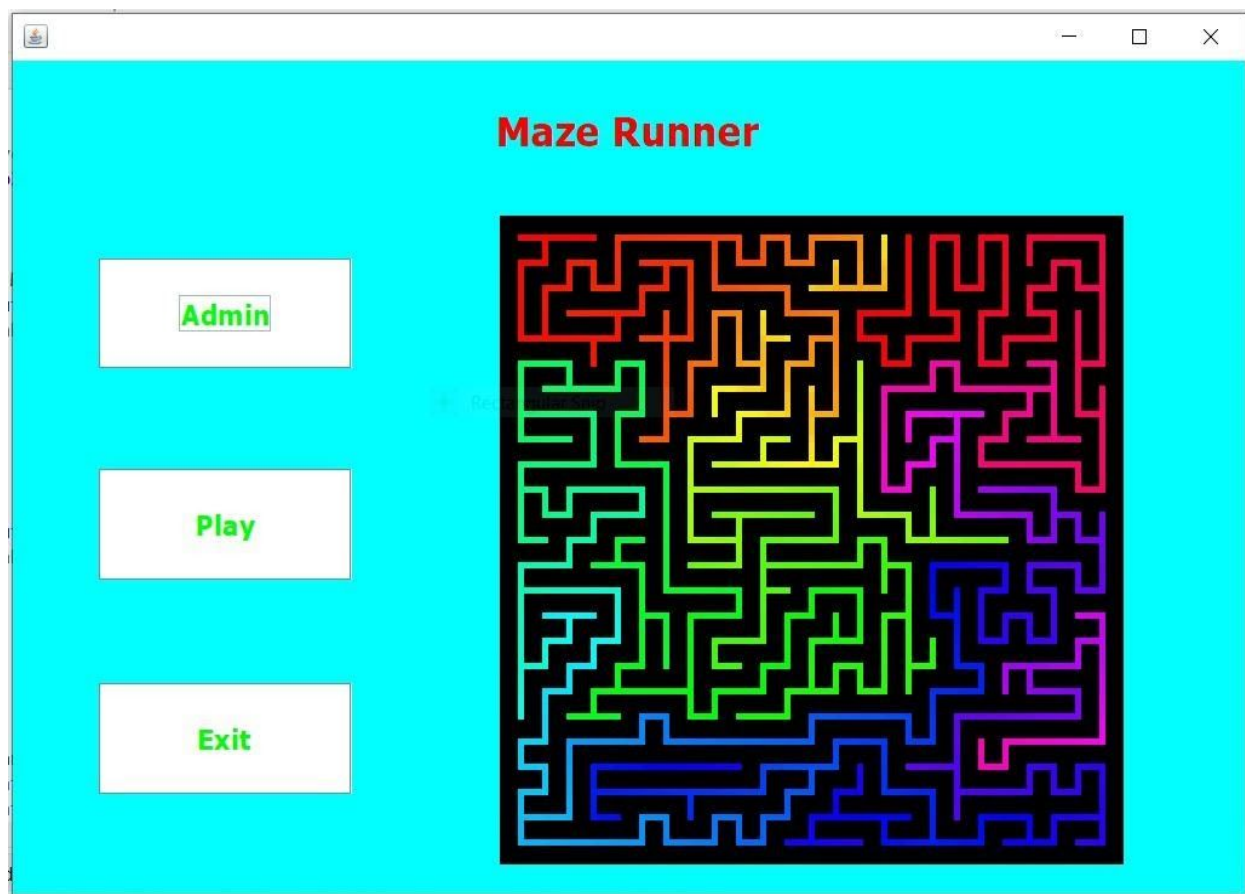
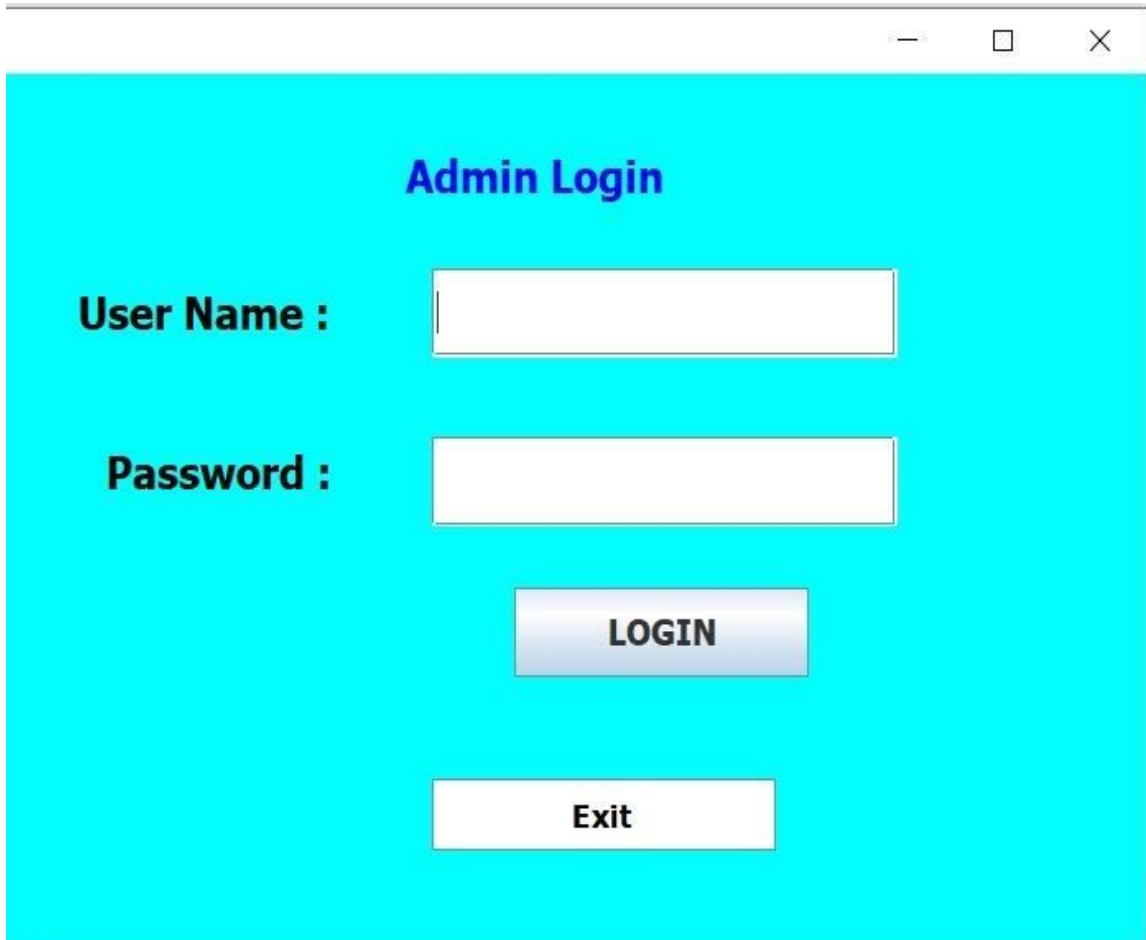


Fig.2: Main Menu of the Maze Game



The image shows a screenshot of a web application window titled "Admin Login". The window has a light blue background and a white border. At the top right, there are standard window control buttons (minimize, maximize, close). The main content area is light blue and contains the following elements:

- Admin Login**: A title in bold blue text.
- User Name :**: A label in bold black text next to a white text input field.
- Password :**: A label in bold black text next to a white password input field.
- LOGIN**: A blue button with white text, centered below the password field.
- Exit**: A white button with black text, centered below the LOGIN button.

Fig.3: Login Page for Admin

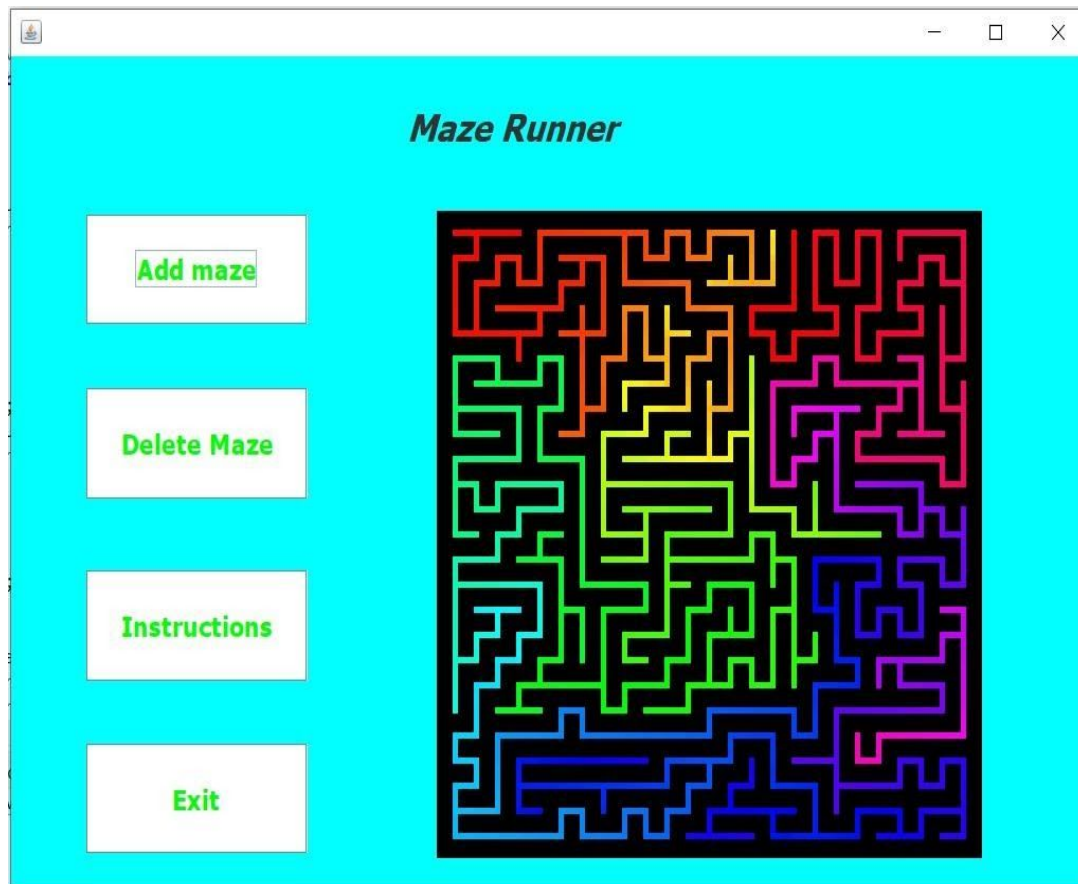


Fig.4: Main Menu of Admin

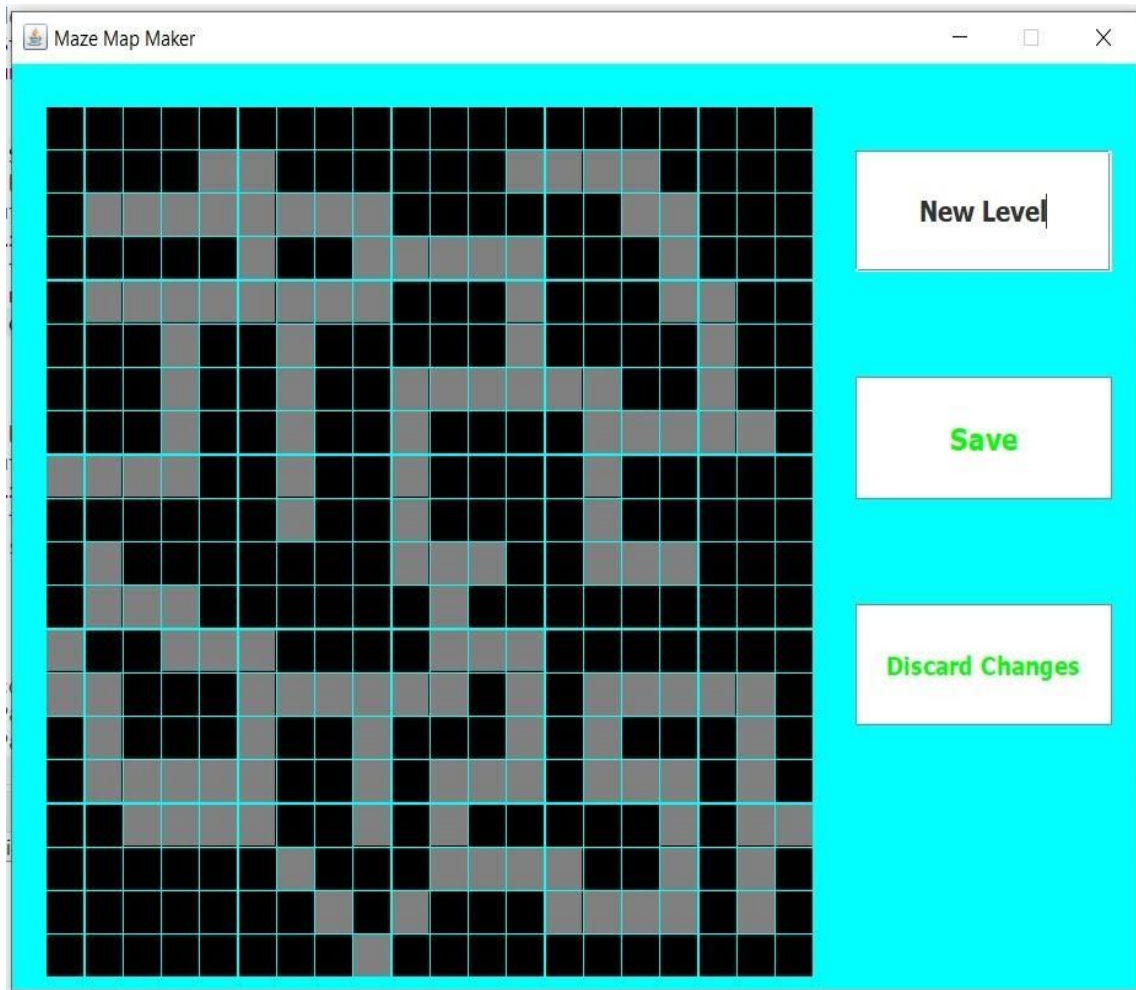
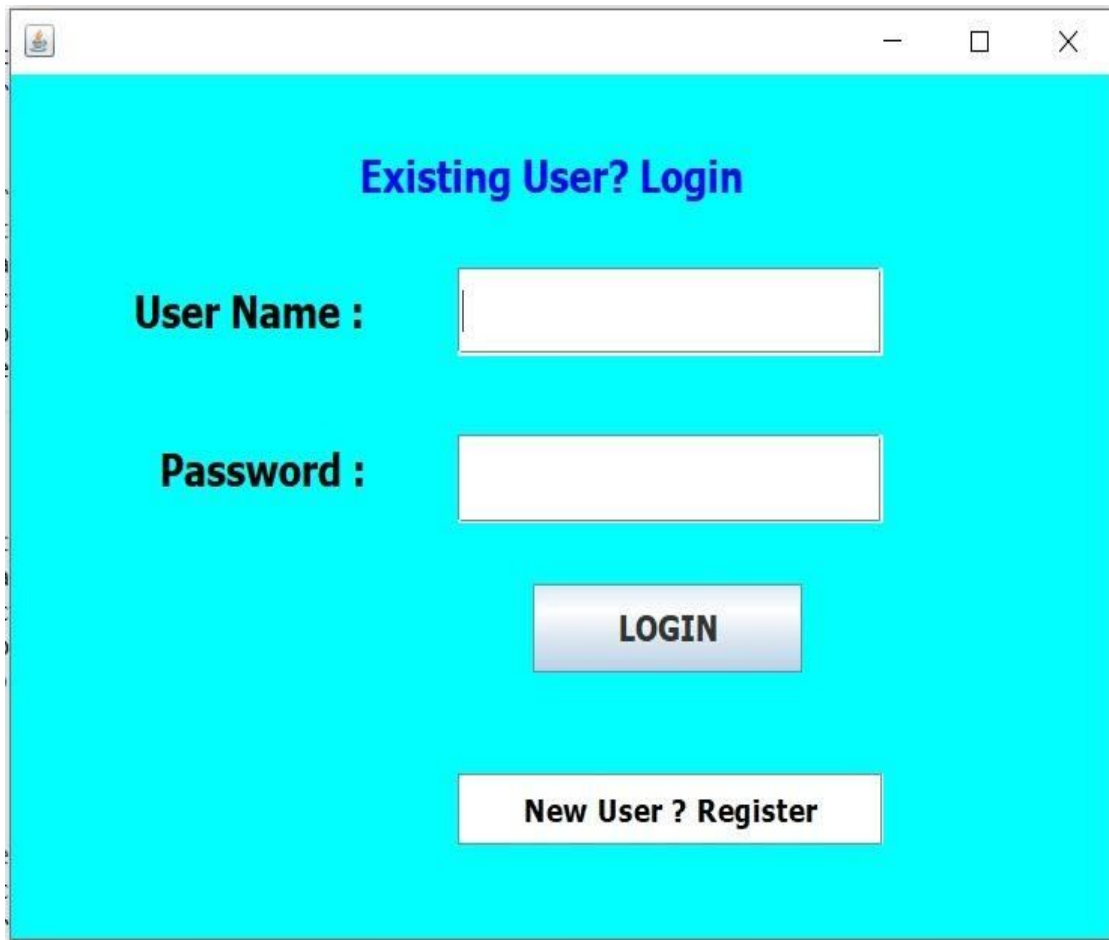


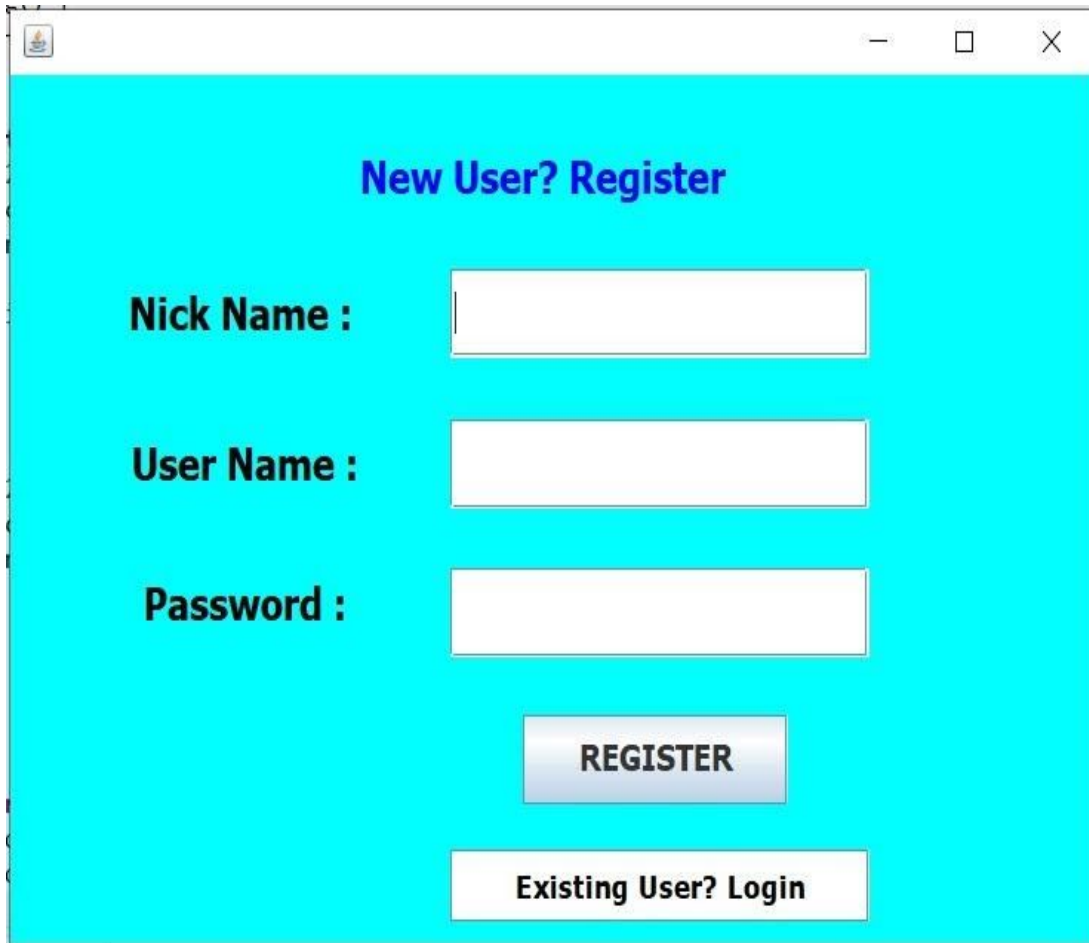
Fig.5: Option for admin to create Maze.



A screenshot of a web application window titled "Existing User? Login". The window has a cyan background and a standard Windows-style title bar with a minimize button, a maximize button, and a close button. The main content area contains the following elements:

- The title "Existing User? Login" in bold blue text at the top center.
- A label "User Name :" in bold black text to the left of a white text input field.
- A label "Password :" in bold black text to the left of a white password input field.
- A blue "LOGIN" button with white text, centered below the password field.
- A white button with black text "New User ? Register" centered at the bottom of the form area.

Fig.6: User Login



A screenshot of a web browser window titled "New User? Register". The window has a cyan background. It contains three input fields for "Nick Name", "User Name", and "Password". Below these fields is a blue "REGISTER" button. At the bottom, there is a white button with the text "Existing User? Login".

New User? Register

Nick Name :

User Name :

Password :

REGISTER

Existing User? Login

Fig.7: New User Registration

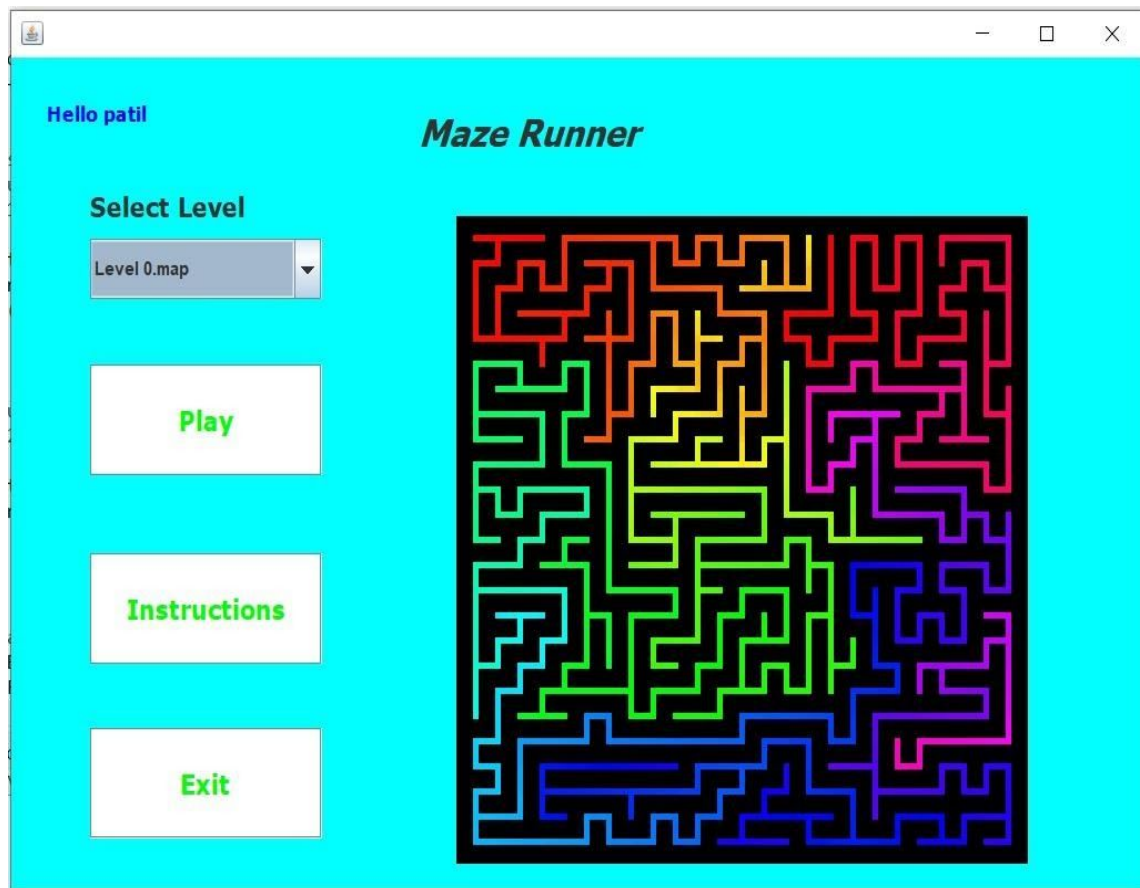


Fig.8: Menu for User after login

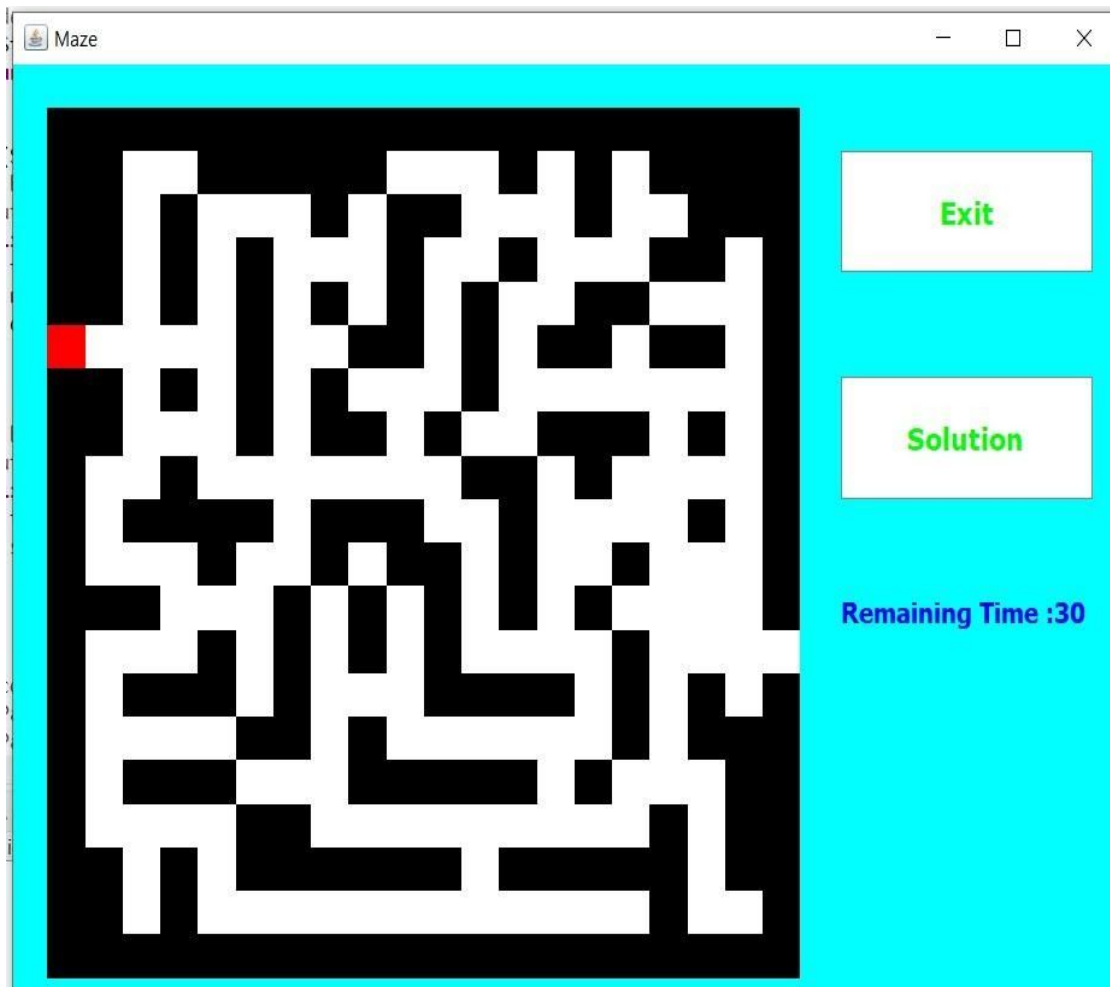


Fig.9: Maze Game During Play

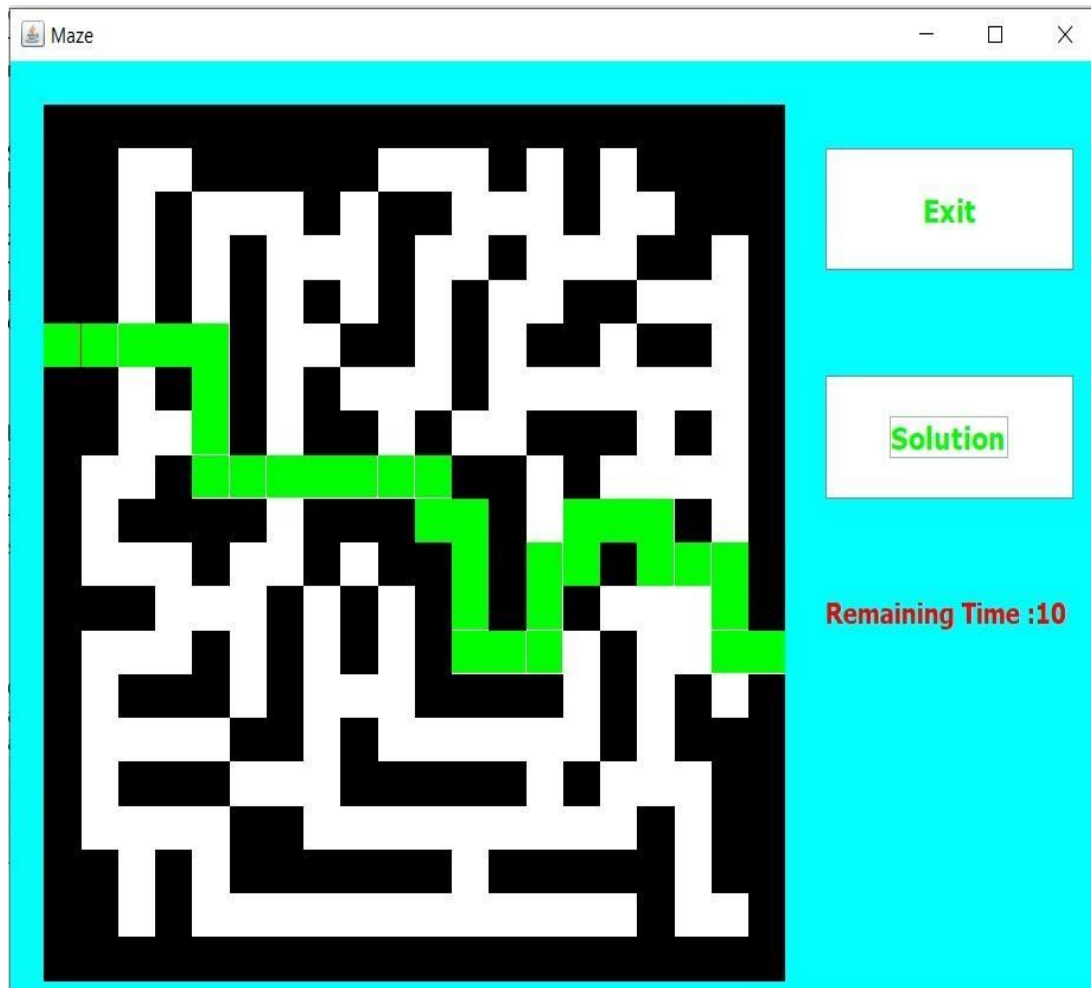


Fig.10: Solution of the maze using Dijkstra's algorithm
