Trivia Maze

Create a maze that a player must navigate through from entrance to exit. The maze is composed of rooms. Each room has 1 or more doors (the design is up to you). In order for the user to pass through a door, s/he must correctly answer a question.

The type of questions asked are up to you, but you should have multiple choice, true/false, and short answer (one or two words/ one or two numbers). The questions (and their corresponding answers) must be stored in a SQLite database. The format of the database is up to you, but you might want to categorize your questions into the different formats you must display them in.

If a the user is unable to answer a question, that door is then locked permanently. If the user is unable to make it from the entrance to the exit (due to locked doors), the game is lost.

You may display one room at a time, the entire maze, or the current room and the entire maze.

You are welcome to implement variations on this theme, but run them by your instructor first. You might place items in the room that can help the user (magic key that gets you through one door, a hint (ala "Who Wants to be a Millionaire") that reduces the multiple choice options or gives you the first letter/digit of an answer that must be typed.

Other important details

- The size of the maze is up to you, but must be at least four by four rooms. You can randomly place the entrance and exit, or you can fix them at opposite ends.
- Provide the means for the user to save the game and of course load a saved game. Serialization can help greatly with this.
- Provide an admin tool that allows for entry of questions and answers into the database. It is expected this admin tool will be part of the software itself (perhaps a menu option)
- Make sure user input is scrubbed as necessary to prevent undesired behavior from the program. More specifically user input should not crash the program or make it behave unexpectedly.
- Include cheats as necessary to allow for easy testing of the program. Be sure and document those cheats at the top of your source code (and anywhere else you deem necessary/helpful).
- Extra points will be given if you successfully incorporate audio and/or video into the questions. Be sure and document this in your final submission -- state you did extra credit and what comprises that extra credit.
- Here is a zip file that is a sample of what you MIGHT do for your game. The program was built by a single individual many years ago as part of my .NET Programming in C# class (CSCD 371). The project does not contain everything you need to do, it is here to give you a feel for what can/might be done. Note that the .NET Framework needs to be installed on your machine to run it. It is a Windows based program, so won't run on Mac or Linux.
    - Trivia Maze by Russ Utt (15Mb): TriviaMazeRussUtt.zip

Tools

You MUST use **git** to version your project. You *MUST* also use Pivotal Tracker to plan and manage your project (if your team knows of another project management tool that does what Pivotal Tracker does, let me know and I might let you use that if you prefer). You MUST incorporate unit tests into your code base. On the Java side you can use JUnit, on the C# side there is NUnit and also the unit testing suite provided by Visual Studio. You are welcome to use Eclipse or IntelliJ for Java projects and Visual Studio or VS Code for C# projects.

Deliverables

1. Each team will give project updates regarding progress, functionality, OO concepts employed, and patterns used. The update days will be posted as assignments here on Canvas. The update *MUST* include UML class diagrams that represent your current project state. At least once this quarter each team will report status to the rest of the class (via a Zoom session). A demo of your project is welcome. Your presentation should be from 5 to 7 minutes. Each person on the team MUST discuss what s/he did. This presentation requirement will be posted separately to Canvas.
2. On the day scheduled for our final each team will give a 10 minute (max) recorded presentation on your project. Include visual aids (presentation slides, etc.) as well as a demonstration of the application you built. Details on recording and submitting will be posted separately on Canvas.
3. SRS Document
4. UML class diagrams, one use case diagram, one use case description
5. Database schema
6. Source code with unit tests - submit your entire project folder
    1. Source code must conform to coding standards chosen by your team.
    2. Submit your coding standards (updated as necessary) along with code and unit tests
7. Installer for program
8. Presentation material
9. Something that shows history and/or usage of git.
10. The above items from 3 through 9 should be submitted in a zip file to Canvas
11. Each team member will perform a peer review (separate assignment will be posted for this on Canvas). Your points for the project will be based on the peer review.

Note that class project grade shown on Canvas is the grade/score for team. Individual grades will be adjusted based on peer review score when final grades are calculated.