

Mine Seeker

Software Requirements Document

CMPT 276 Assignment #3

Contents

- 1. Revision History Table..... 3
- 2. Game Play 3
 - 2.1 Example Game 3
- 3. Software requirements specification..... 7
 - 3.1 Functional requirements..... 7
 - 3.2 Non-functional requirements. 9
- 4. Use Cases 12
- 5. Scenarios 13
 - 5.1 Scenario: Play game 13
 - 5.1.1 Initial assumptions 13
 - 5.1.2 Normal workflow 13
 - 5.1.3 Variation: Change game size 13
 - 5.1.4 Concurrent activities 13
 - 5.1.5 State on completion..... 13
- 6. Requirements Validation 14
 - 6.1 UI Mockup..... 14

1. Revision History Table

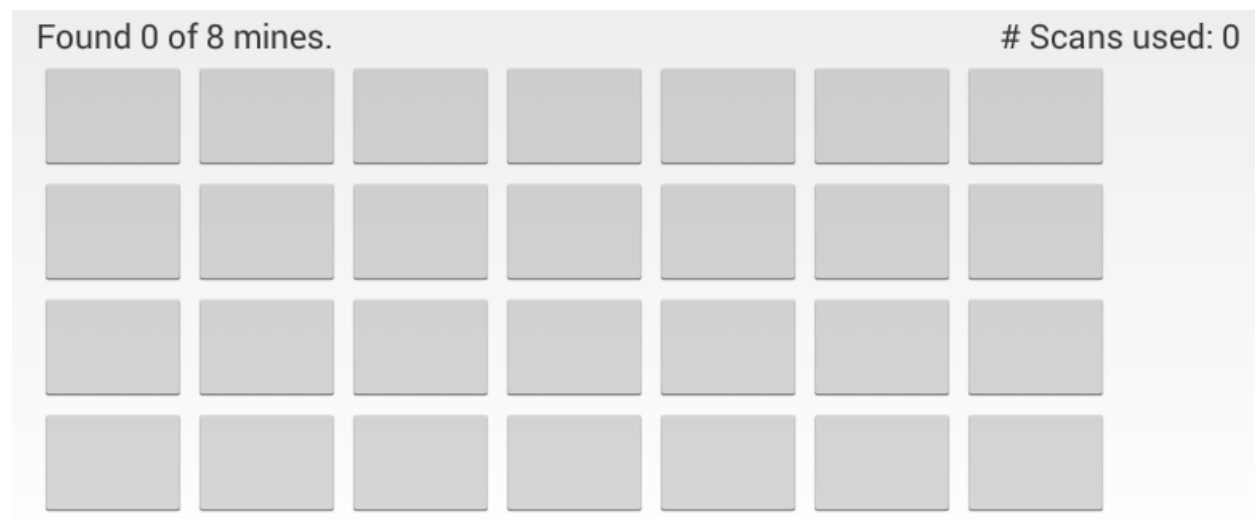
| Revision | Date | Summary | Author |
|----------|---------------------|--------------------|--------------|
| 1 | October 8th 2020 | Initial release | J. Thomas |

2. Game Play

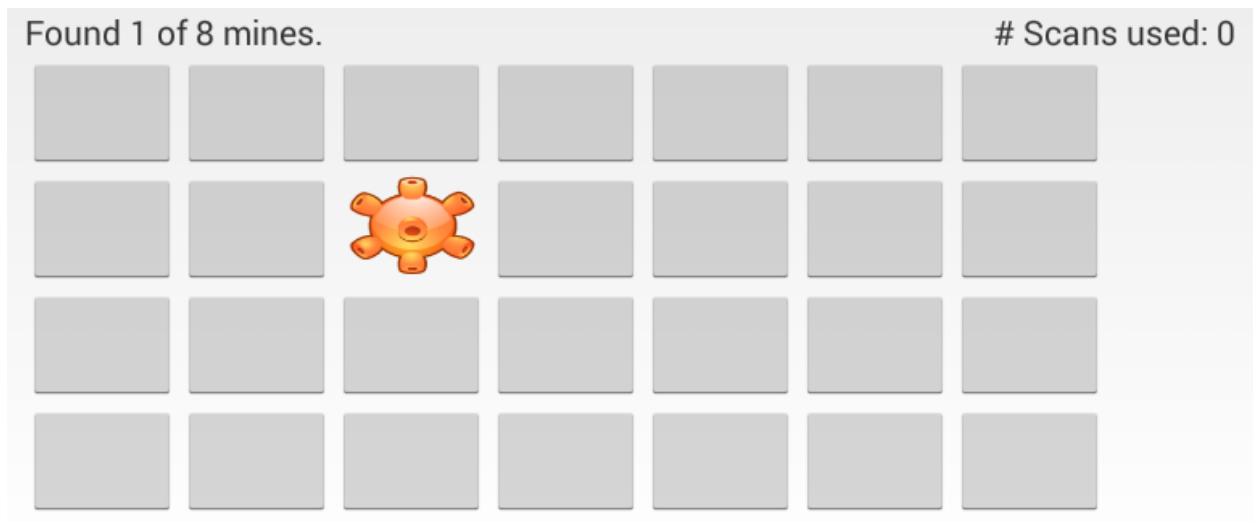
Mine Seeker is a game where the player tries to find a certain number of mines which are randomly placed in cells on the game board. The player taps on a cell to inspect it. If the cell contains a mine, then the mine is revealed. If there is no mine, then inspecting the cell triggers a scan which shows the counts of hidden mines in the same row and column as the selected cell. This information allows the player to make smart choices about which cells to inspect. The goal is to find all of the mines using the minimum number of scans.

2.1 Example Game

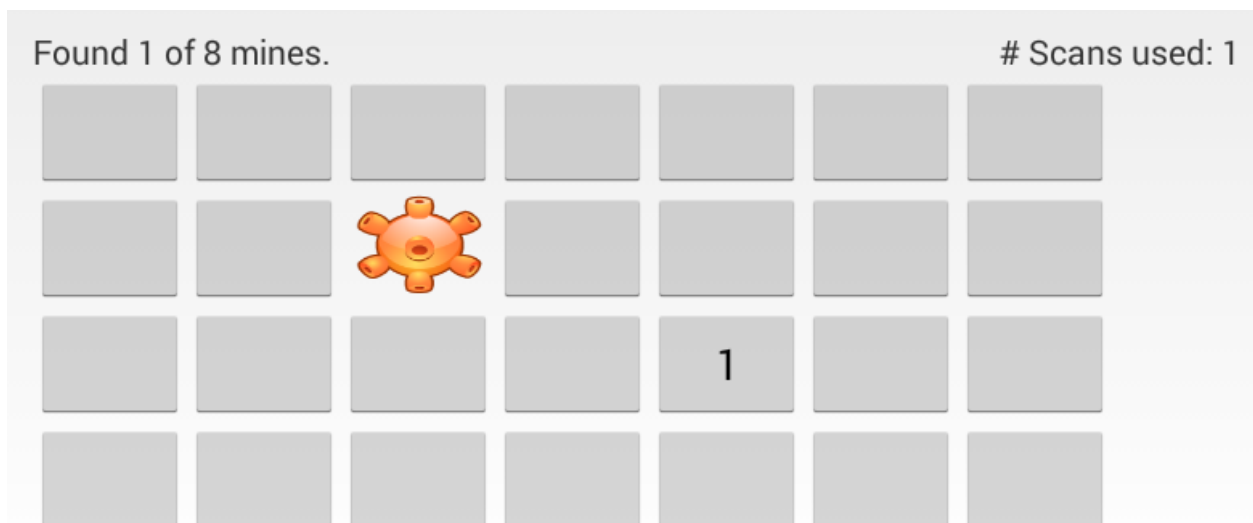
1. User initially sees the game board with all cells hidden. (There are additional UI requirements than those shown here, see the requirements section for more information).



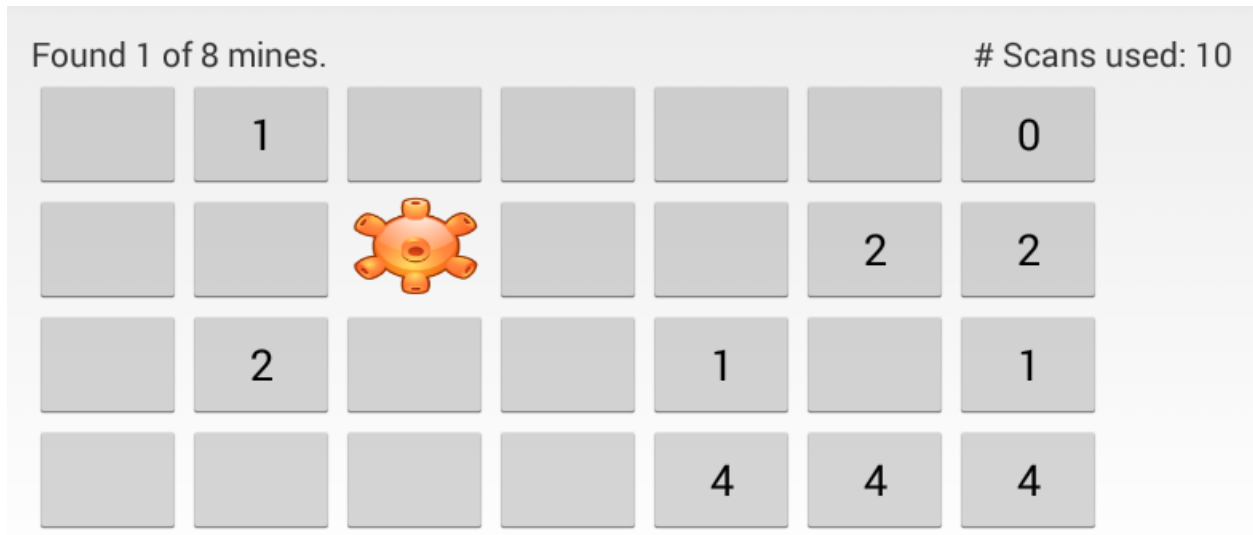
2. User selects a cell. If it has a mine, then the mine is revealed.



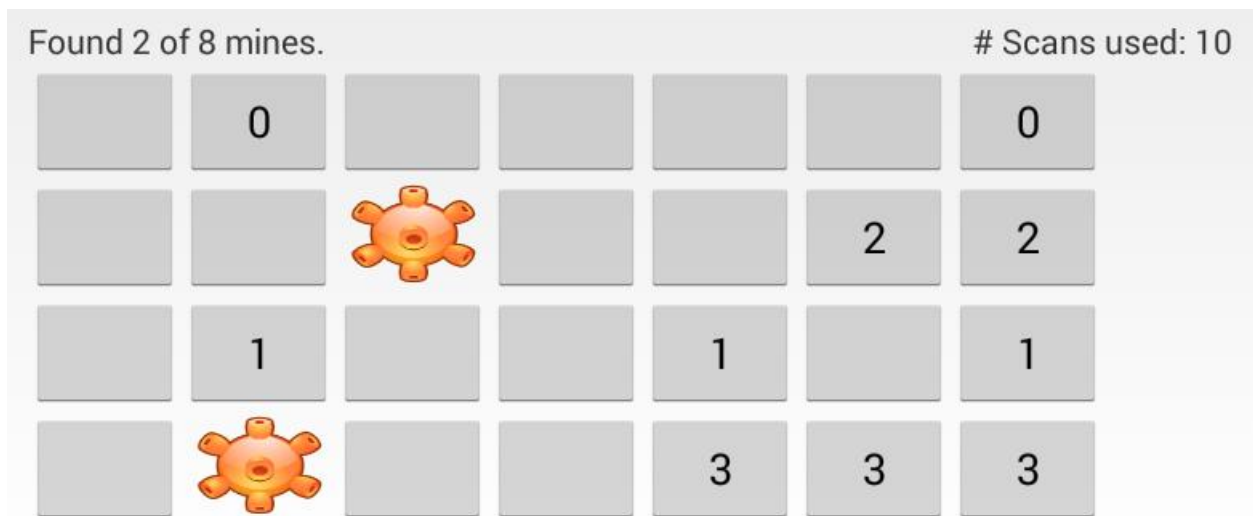
3. When the user taps a cell without a hidden mine (i.e. taps a cell with a known mine, or an unrevealed cell which has no mine), it triggers a scan of that cell's row and column. The cell then displays the number of hidden mines found in its row and column.



4. User continues selecting cells.





5. Using the information from the scans, the user can make informed choices about which cells to investigate.



When the user taps a mine, it is revealed. Hence, the “hidden mine” counts all decrease for the cells in the mine’s row and column (because that mine is no longer hidden).









6. The user may trigger a scan in a cell with a mine. This is done by tapping the cell showing a revealed mine. This scan is identical to scans in other cells.

Found 2 of 8 mines. # Scans used: 11

| | | | | | | |
|--|---|---|--|---|---|---|
| | 0 | | | | | 0 |
| | |  | | | 2 | 2 |
| | 1 | | | 1 | | 1 |
| |  | | | 3 | 3 | 3 |

7. Once the user has found all the mines, they win the game. The challenge is winning with the fewest scans. Note that revealing a mine does not count as a scan, however, scanning on a revealed mine (by tapping on an already revealed mine) does trigger a scan and counts as a scan. Once the user has found all the mines, all scans should show 0, because there are no more hidden mines in the game.

Found 8 of 8 mines. # Scans used: 12

| | | | | | | |
|---|---|---|---|---|---|---|
| | 0 | | | | | 0 |
|  | |  |  | 0 | 0 | 0 |
| | 0 |  | | 0 | | 0 |
|  |  |  |  | 0 | 0 | 0 |

3. Software requirements specification.

3.1 Functional requirements

1. The application's first screen must be a nice-looking welcome screen.

1.1 The program must start up showing the welcome screen.

1.2 Welcome screen must include at least the following elements:

- Name of application
- Name of application's author(s)
- One or more images. Could include a picture or cartoon of the authors, an icon for the application, or related images.

1.3 Welcome screen may include two or more different animations (such as fade, spin, or move). It may have complicated animations such as rotating and moving a block of elements at once.

1.4 Welcome screen must have a button (or similar interface) which allows the user to skip animations (if any) and go to the Main Menu.

1.5 The Welcome screen may automatically advance to the Main Menu after all animations (if any) have finished, plus at least 4 extra seconds.

2. The Main Menu must allow the user to navigate to the game, options, and help screens.

2.1 Display a button to navigate to the Game screen.

2.2 Navigating to the Game screen creates a new game with the correct configuration specified on the Options screen

2.3 Display a button to navigate to the Options screen.

2.4 Display a button to navigate to the Help screen.

2.5 Buttons displayed may be fancy and visually appealing featuring icons.

3. The Game screen must allow the user to play the Mine Seeker game.

3.1 Display text stating how many mines total there are on the game board (hidden or revealed)

3.2 Display text stating how many mines the player has revealed.

3.3 Display text stating how many scans it has taken the user so far this game.

3.4 Display a grid of buttons (or UI elements which have button-like functionality). The grid size is set by the options screen.

3.5 The number of mines on the game board is set by the options screen.

3.6 Tapping a cell investigates the cell, which either:

1) Reveals a mine if one is present.

2) Performs a scan if either no mine is present, or the mine has already been revealed.

Tapping on an already scanned cell has no effect and does not count as an additional scan.

3.7 When a mine is revealed, the button must indicate that it contains a mine. The button must display an icon or image on it showing it is a mine.

3.8 When a scan is performed, the count of hidden mines in the row and column is displayed in that button.

3.9 When a mine is revealed, any of the buttons in its row and column which show a count of hidden mines must be updated with the new count of hidden mines (count decreases by 1).

3.10 The scanning may be animated to show a scan happening (like a ship's radar searching), or a pulse wave going out across the row and column.

3.11 App may play a sound when it scans and when the user finds a mine.

3.12 App may vibrate when it scans and when the user finds a mine. Different vibration feel for each would be best.

3.13 May display text stating the total number of games started (saved between application launches).

3.14 May display text stating the best score so far of any completed game of this specific configuration (board size and number of mines); must save best score for each possible configuration.

4. When the player wins, congratulate the player and return to the Main Menu.

4.1 When the player finds the last mine, redraw the game board showing the mine and updated hidden-mine counts.

4.2 When the player finds all mines on the board, display a congratulations dialog.

4.3 The congratulations dialog must have at least one image, and some text congratulating the player.

4.4 When the player dismisses the dialog (taps OK, or the like), return to the Main Menu.

4.5 From the Main Menu, pressing the Android back button must then quit the application.

5. The Options screen must allow the user select board size and number of mines.

5.1 User can select the board size, from options including at least:

- 4 rows by 6 columns
- 5 rows by 10 columns
- 6 rows by 15 columns

5.2 User can select number of mines, from options including at least:

- 6 mines
- 10 mines
- 15 mines
- 20 mines

5.3 The game size and number of mines are saved between application runs.

5.4 May allow user to reset number of times game has been played and best scores for each game configuration (if supported).

6. The Help screen displays some information about who wrote the application and some text explaining the game.

6.1 The about-the-author text must include a hyperlink to the CMPT 276 home-page.

6.2 The game information text must explain some of the basics about the game. You must use your own wording, not copying the text from the assignment document. Your text should reflect the theme of your game.

6.3 The Help screen must provide the correct citation for any images, icons, or other resources (such as music) used in the game (for copyright purposes). Include a hyperlink if applicable.

6.4 Pressing the Android back button on the Help screen returns to the Main Menu.

3.2 Non-functional requirements.

1. The application must have the game play described in this document, but must have a theme of your choosing, such as:

- Searching for rebel bases in space.
- Finding virus infected cells to fight an infection.
- Identifying bugs in the Linux kernel.
- Finding gophers in a field.
- Finding bad-apples in a case of apples.
- Finding zombies in a graveyard.
- Anything else you can imagine!

1.1 Images chosen for backgrounds and buttons must be consistent with this theme.

1.2 Game help text must be consistent with this theme.

1.3 The theme may affect the name given to your application; it need not be Mine Seeker.

1.4 If you want to update game play slightly to be in line with your game's theme, you must consult the customer (It's me. I'm the customer).

2. Application source code must be maintainable.

2.1 Code must be well organized into methods and classes.

2.2 Game logic must be in a separate class from game UI class.

2.3 Game logic must be in a separate Java package from UI code.

2.4 Code must use good naming conventions and have good indentation and formatting.

3. The application runs on Android smartphones and tablets.

3.1 The application must run under at least Android OS version 10.0 (API 29, Q) or newer.

3.2 The application must display well on at least the "Pixel 2" configuration.

3.3 The application must support at least horizontal (landscape) orientation.

3.4 The application must not be able to be rotated to an unsupported orientation.

4. Pressing the "back" button on the Android phone must always take the user to the previous screen in a reasonable way.

4.1 From the Welcome screen, back should exit the program.

4.2 From the Main Menu, the game must exit without returning to the Welcome screen.

4.3 From other screens, back must return to the previous activity on the activity stack.

5. The application should appear complete and well built.

5.1 The application must have an appropriate (non-default) icon.

5.2 Each screen must have a background image. Each screen may use the same background image.

5.3 All text must be clearly readable over the background.

5.4 All text that appears on the UI must be read from the strings.xml file to support internationalization.

5.5 App may save game state if app is closed while playing.

6. Quick to learn for a new user.

6.1 An average grade 10 student must take no more than two minutes to learn to play the game after a one minute demonstration on how to use the application. This is a rhetorical example, you don't need to actually get a grade 10 student to play your game.

7. Must be responsive to the user.

7.1 Each user interaction (such as a button press) must start to generate its response within 0.5s when run on a real device. (Looser performance criteria applied for running in the emulator). This won't be strictly enforced, so don't worry too much about the exact timing unless your code is doing something which takes significantly more time than average.

4. Use Cases

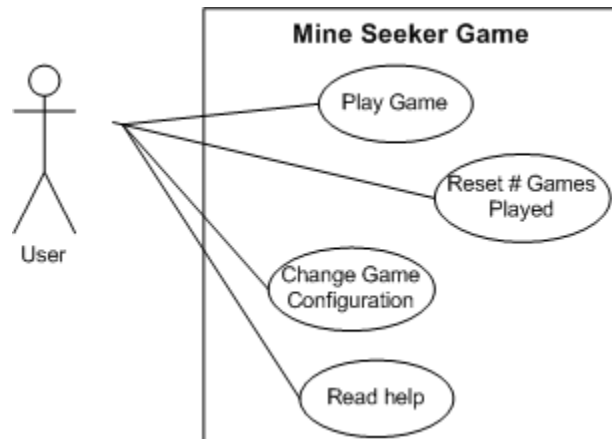


Figure 1: Use case diagram for Mine Seeker game

5. Scenarios

5.1 Scenario: Play game

5.1.1 Initial assumptions

User is logged into the phone and has just launched the application.

5.1.2 Normal workflow

User sees the welcome screen. Presses button to advance to the Main Menu screen.

User selects button to play game.

User taps on game cells to initiate scans and detect mines.

Game displays information about number of hidden mines detected in the row and column of each scanned cell.

Once all mines have been found, user sees congratulations message and is returned to the Main Menu.

5.1.3 Variation: Change game size

Before starting the game from the Main Menu, user selects to go to Options screen.

User changes game board size and number of mines in game.

User presses the Android back button to return to Main Menu.

Resumes normal workflow to begin playing game.

5.1.4 Concurrent activities

User may navigate away from application and return to the application without losing its state so long as the application does not close. If the application is closed, the game state may be lost.

5.1.5 State on completion

User is viewing the Main Menu.

6. Requirements Validation

6.1 UI Mockup

(Note about Figures: Your app's screens need not match the ones pictured here exactly. Your apps must also meet the requirements listed earlier even if the mock-ups shown here miss some feature or other).

1. User launches game; welcome shown; then advances to main menu.

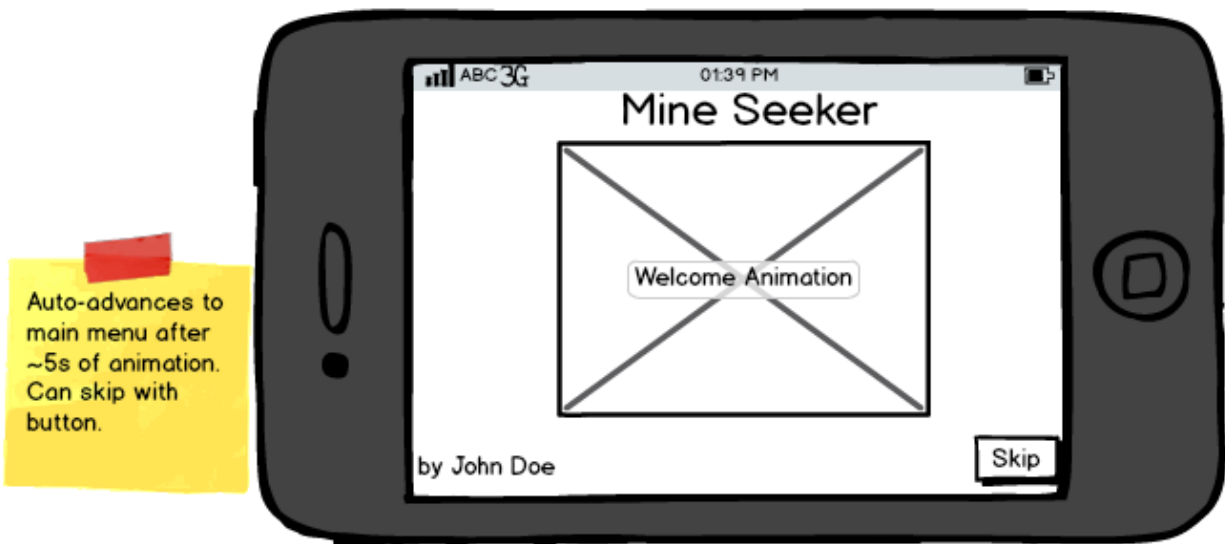


Figure 2: Welcome splash screen

2. User taps the button to play a game and changes to the game screen.

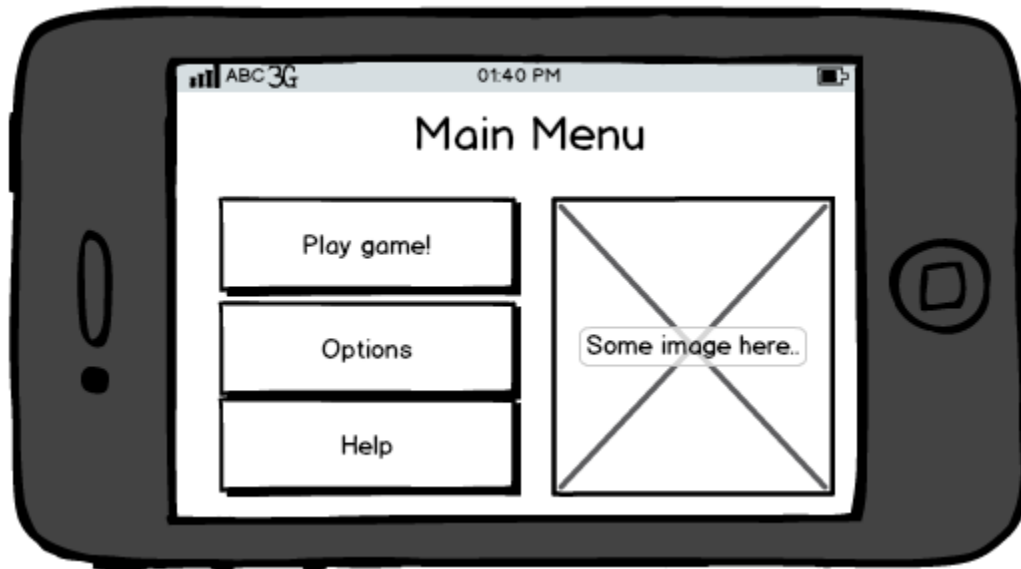


Figure 3: Menu allowing access to game, options, and help.

3. User shown game screen with grid of buttons. User taps buttons to scan, and to find mines.

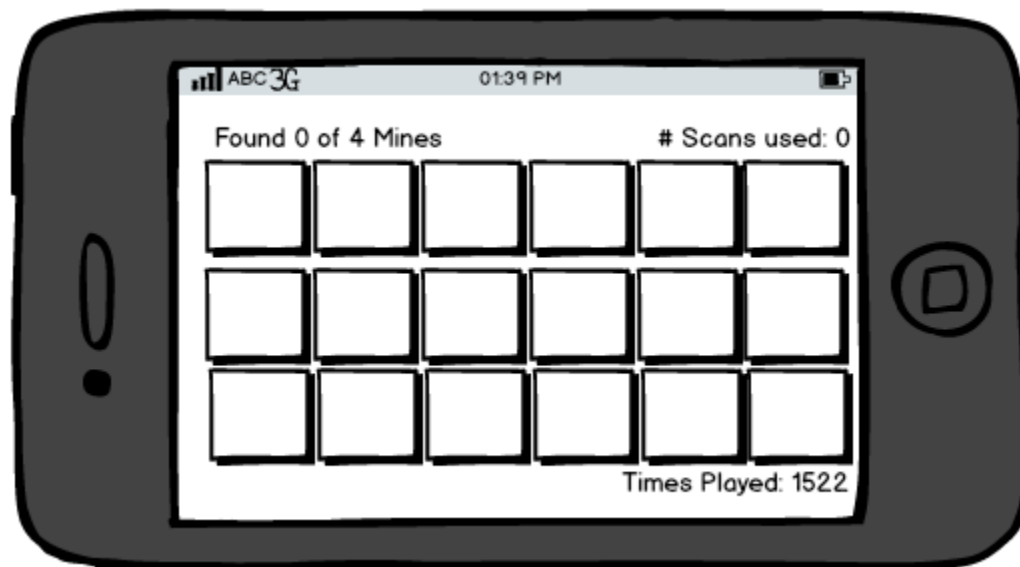


Figure 4: Game board before any moves.

4. As user finds mines, the scan numbers update.

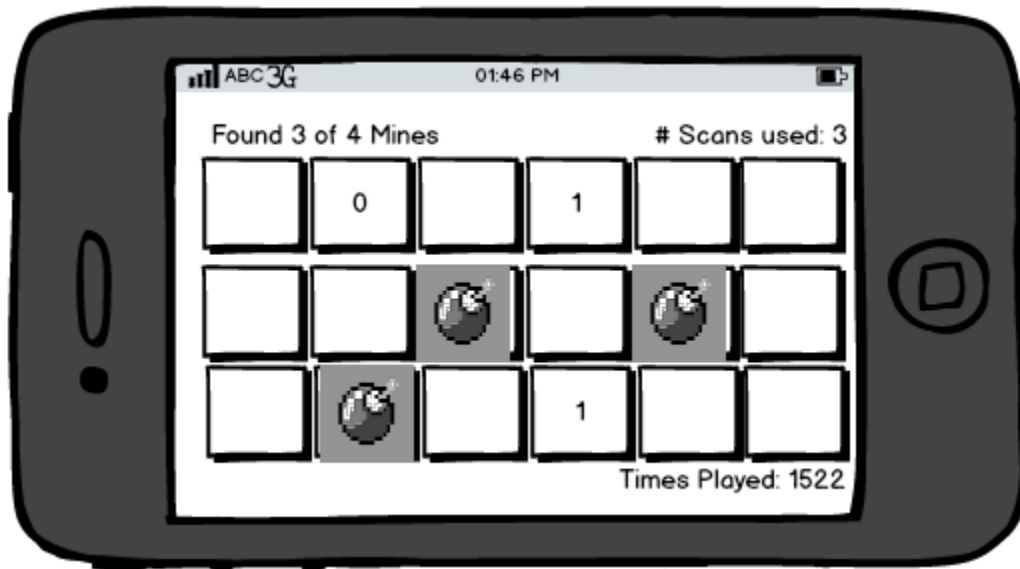


Figure 5: Game board after some user moves.

5. When user finds all mines, sees congratulations message and return to Main Menu.



Figure 6: Winning congratulations message

6. From Main Menu, user selects Options screen.

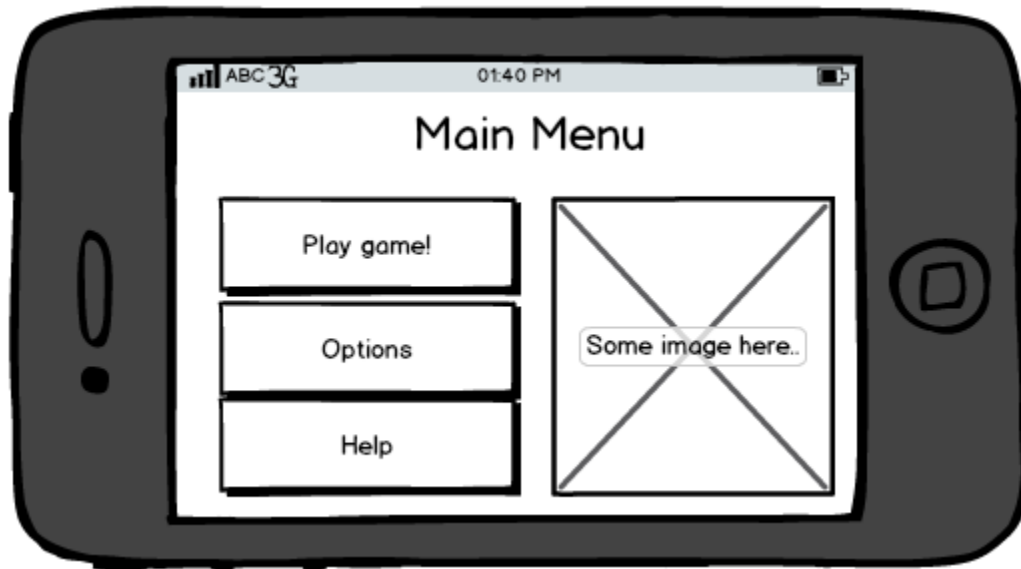


Figure 7: Main Menu

7. On Options screen, user can change game settings.

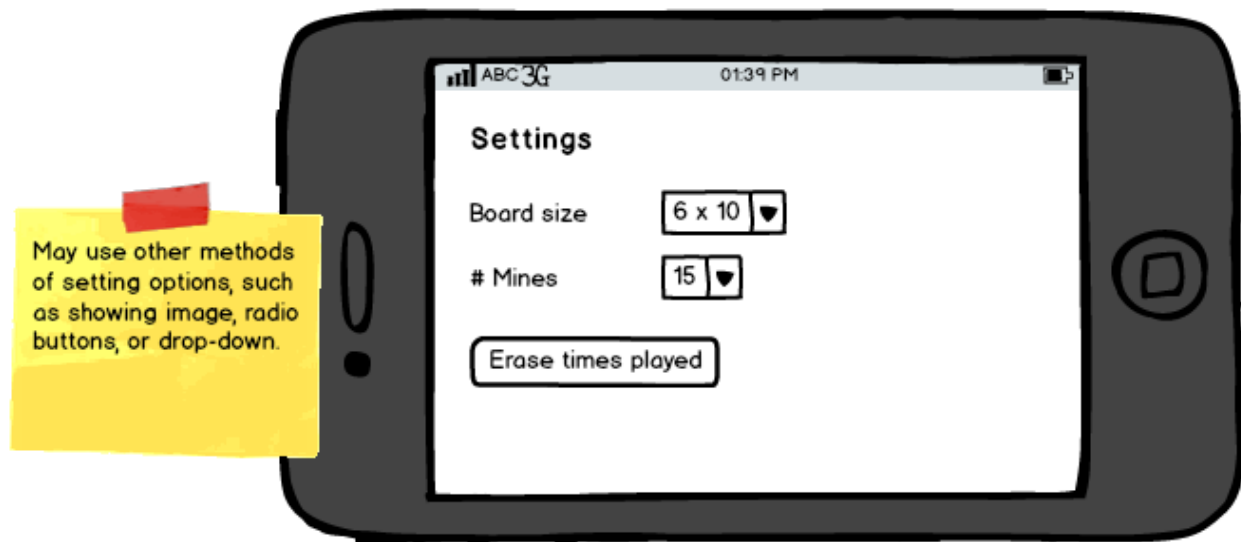


Figure 8: Options screen to change game settings

8. User returns to Main Menu, and then navigates to help screen.

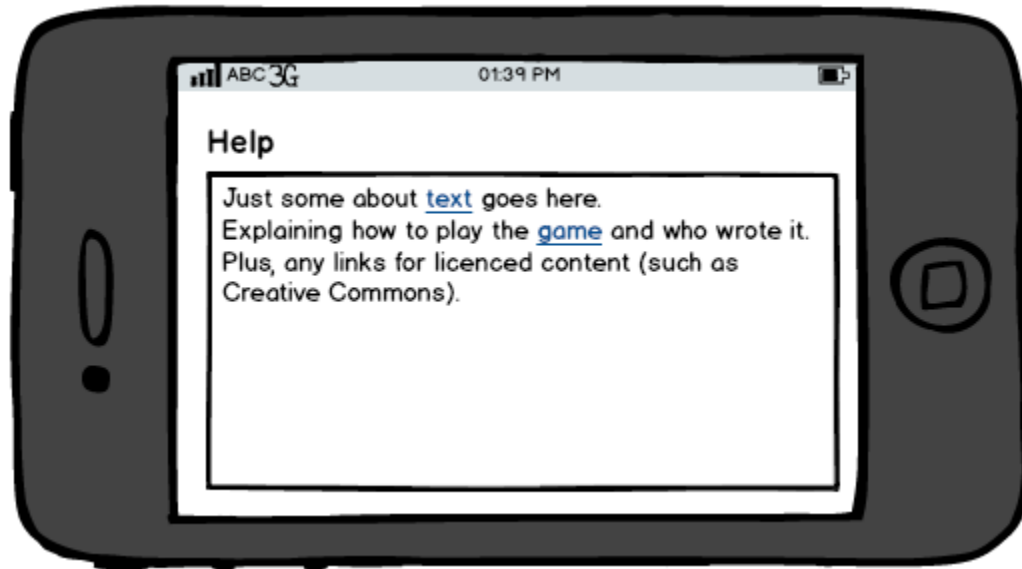


Figure 9: Help screen with game directions and about information.