Purpose: This describes the structure of the project to make it easier to make additions for future groups

Components:

- **Board Manager** (constants.go): Defines global constants like BoardSize, MinMines, MaxMines, GridSpacing, etc.
- **Game Logic** (game-handler.go):
    - Creates a new game with mines placed randomly (NewGameHandler)
    - Finds adjacent mine counts (AddNumbers)
    - Ensures that the first click is not a bomb (moveBombFrom)
    - Handles cell uncovering (Click, RevealZero), flagging (ToggleFlag), and end game conditions
- **User Interface** (ui-handler.go):
    - Used Fyne framework
    - Used to create the grid and overlays for covered/uncovered/flagged cells.
    - Text on each cell (number, flag, or bomb)
    - Creates the end game message
- **Input Handler** (ui-handler.go):
    - Processes user clicks
    - On left click (Tapped): Calls “Click” in Gamehandler
    - On right click (TappedSecondary): Calls “ToggleFlag” in Gamehandler
- **Setup Module** (setup.go):
    - Creates the initial screen where the user chooses the number of mines in the game.
    - This validates/invalidates the user input.

Data Flow

- User Input → Input Handler (clickableRect) → Backend Game Logic → Update UI

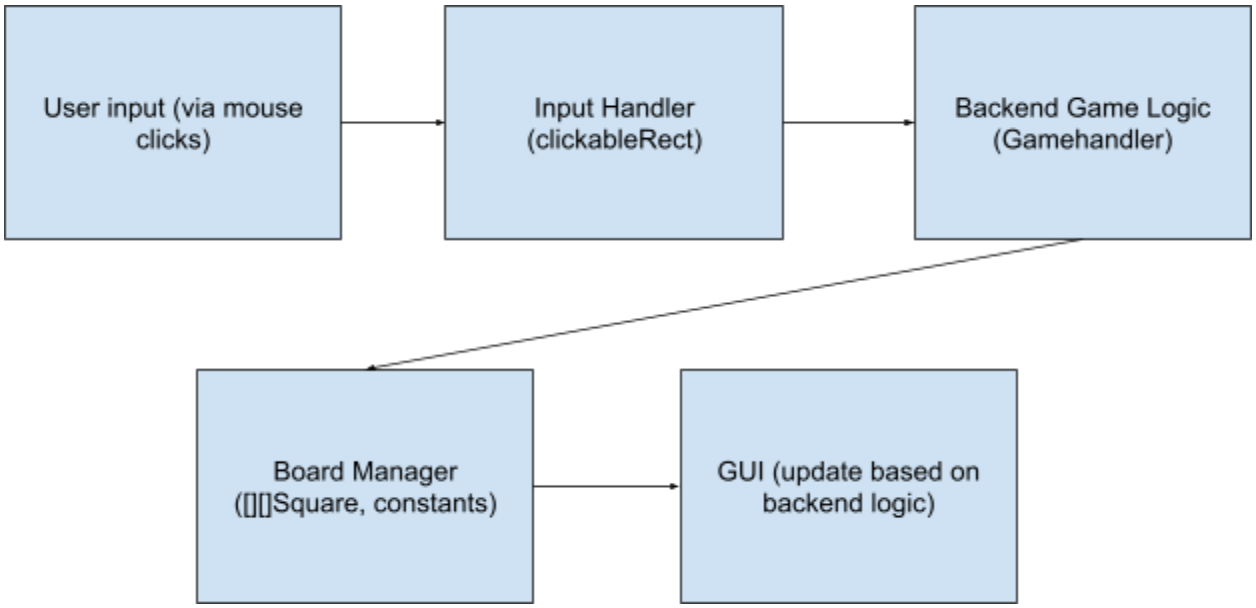
Key Data Structures

- 2D Grid of Squares ([][]Square) in game-handler.go
- Game State Object: Tracks the board, number of mines, first-click, etc.
- Global Constants in constants.go: Has global constants such as board size, range for mine numbers, window size, etc.

Assumptions

- Fixed 10x10 grid
- Mine count that is user-specified (10-20 mines)
- First click is safe

Diagram:


User input (via mouse clicks)
Input Handler (clickableRect)
Backend Game Logic (Gamehandler)
Board Manager ([][]Square, constants)
GUI (update based on backend logic)