Minesweeper Architecture Overview

This Minesweeper implementation follows a modular, event-driven architecture with a separation of concerns. The project uses Python and Tkinter to create a GUI game with abstractions between game logic, user interface, and input handling.

ARCHITECTURE PRINCIPLES

1. Separation of Concerns

Each module has a single, well-defined responsibility:
- Game Logic: Pure game rules and state management
- UI Layer: Visual presentation and user interaction
- Input Handler: Event coordination between UI and game logic
- Board Manager: Data structure and mine placement algorithms

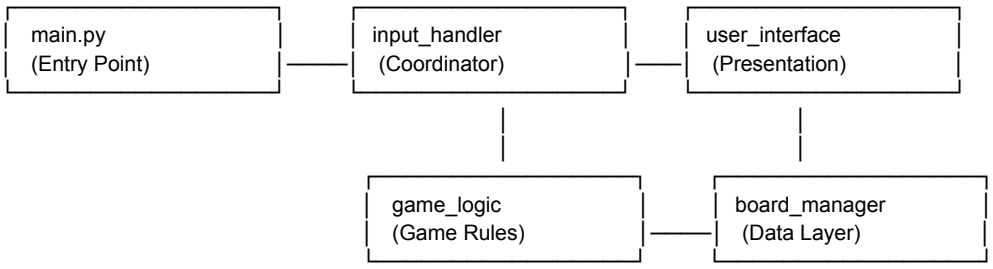
2. Event-Driven Design

The application responds to user events (clicks, keyboard input) through Tkinter's event system, ensuring responsive user interaction without blocking the UI thread.

3. Dependency Injection

Components receive their dependencies through constructor injection, making the system testable and loosely coupled.

System Architecture



CORE COMPONENTS

1. **main.py** - Application Entry Point
Responsibility: Initialize all components and start the application
Key Functions:
- Creates the Tkinter root window
- Instantiates all core components
- Establishes component relationships
- Starts the main event loop
Dependencies: All other modules

2. **board_manager.py** - Data Layer
Responsibility: Manage game board data structure and mine placement
Key Classes:
- Cell: Represents individual board cells with state (mine, covered, flagged, adjacent count)
- BoardManager: Manages the 2D board array and mine placement algorithms
Key Methods:
- initialize_board()
- place_mines()
- calculate_adjacent_counts()
- get_cell()

3. **game_logic.py** - Business Logic Layer
Responsibility: Implement game rules, state transitions, and victory conditions
Key Methods:
- reveal_cell()
- toggle_flag()
- check_victory()
- start_game()

4. **input_handler.py** - Event Coordination Layer
Responsibility: Process user input events and coordinate between UI and game logic
Key Methods:
- handle_left_click()
- handle_right_click()

5. **user_interface.py** - Presentation Layer
Responsibility: Create and manage the Tkinter GUI, handle visual feedback
Key Components:
- Game Board
- Status Display
- Control Panel
- Dialog System

DATA FLOW

Initialization: **main.py** → **BoardManager** → **GameLogic** → **UI**
User Interaction: **User** → **InputHandler** → **GameLogic** → **BoardManager** → **UI**
Game Over: **Mine Click** → **GameLogic** → **UI**

STATE MANAGEMENT

Game States: Initialization, Playing, Game Over, Victory
Cell States: Covered, Revealed, Flagged, Mine

ERROR HANDLING

- Input validation
- Exception handling

TESTING

- Unit testing
- Integration testing
- Manual testing

DEPLOYMENT

Dependencies: Python 3.x, Tkinter
Distribution: Standalone, Executable, Cross-Platform