EECS 581 Master Document:

Project Group 3 Minesweeper Game

Introduction:

Purpose & Scope:

This project showcases a beginner-friendly Minesweeper game, implemented in Python using Tkinter. Scope includes gameplay with a 10x10 grid: safe first-click and safe zone, reveal and flood reveal of zero tiles, adjacent-mine counts (0-8), right-click flagging with a cap at the mine count, and win/loss resolution.

Authors: EECS 581 Project Group 3

Genea Dinnall, Sam Kelemen, Megan Taggart, Jenny Tsotezo

Intended Audience:

Course evaluating requirements for code quality, other teams evaluating architecture, and extending features.

User Stories: with project requirements, tasks derived

User Story	Acceptance Criteria	Task Derived	Story Points	Person Assigned
As a player, I want a 10×10 board labeled A–J and 1–10 so that I can reference cells unambiguously.	Board renders 10×10 grid; cells addressable by label (e.g., “B7”); all cells covered initially, zero flags.	Implement board rendering; create cell addressing scheme; initialize covered state; verify no flags at start.	2	Genea Dinnall
As a player, I want to enter a mine count (10–20) at game start so that I can adjust difficulty.	Input validated (10–20 only);	Add input prompt; validate range.	2	Genea Dinnall
As a player, I want mines randomly placed so that each game is unique.	Random placement follows chosen count.	Implement random placement; ensure uniqueness; respect count; (opt.) seed mode.	3	Sam Kelemen, Genea Dinnall

As a player, I want the first click to be safe so that I never lose immediately.	First revealed cell is never a mine.	Defer mine placement until first click.	2	Sam Kelemen, Jenny Tsotezo, Megan Taggart
As a player, I want to have each cell I click reveal how many mines are touching it.	Show 0–8, representing the number of adjacent mines.	Implement count neighbors() function	2	Megan Taggart
As a player, I want to toggle flags on cells so I can mark suspected mines.	Covered cell toggles flagged/unflagged;	Implement toggleFlag(); update UI.	2	Jenny Tsotezo, Megan Taggart
As a player, I want victory/loss signal so I know when game is over.	Loss: reveal mine → all mines shown, input locked; Win: all safe uncovered; status indicator.	Implement win/loss detection; reveal all mines; show message.	2	Megan Taggart
As a player, I want simple controls (clicks/keys) so play is consistent.	Primary = uncover; Secondary = toggle flag; Keyboard = arrows + Enter/Space/F.	Map mouse clicks; implement keyboard nav/actions; block invalid reveals.	2	Genea Dinnall
As a dev, I want Board Manager to own cells so state is centralized/testable.	Methods: init, placeMines, get/set state, neighbors, counts; states encoded; unit tests cover adjacency.	Implement BoardManager class; provide API; write adjacency tests.	3	Sam Kelemen, Genea Dinnall
As a dev, I want Game Logic isolated so rules separate from UI.	API: start, reveal, toggleFlag, status; deterministic win/loss; flood correct; unit tests.	Implement GameLogic class; encapsulate rules; ensure status tracking; write tests.	1	Jenny Tsotezo
As a dev, I want Input Handler to map UI events so inputs decoupled from state.	Events map to Game Logic; reject illegal actions; no UI logic inside rules.	Build input handler; map events; add validation.	4	Genea Dinnall

As a dev, I want UI Renderer to update efficiently so view is clear.	Only changed cells re-render; numbers/flags/mines/status shown; handles end lockout.	Implement cell-based renderer; optimize for partial redraws; handle state changes.	5	Genea Dinall, Sam Kelemen

*See person-hours documentation below for implementation of each user story

Project Components Overview/Reference Point:

Explanation of .py files used, functions implemented in the file

cell.py — simple data object for each tile:

has_mine: bool, has_flag: bool, is_revealed: bool, neighbor_count: int.

board_manager.py — board construction & topology:

reset(mine_count); fresh grid, no mines placed.

neighbors(r,c); list[(r,c)]: valid adjacent coordinates.

count_adjacent_mines(r,c); int: computes 0–8 for a cell.

place_mines(safe_r, safe_c); random placement excluding first click + neighbors; then precomputes neighbor_count for all cells.

get_cell(r,c); Cell.

game_logic.py — rules & state transitions:

reveal_cell(r,c); list[(r,c)]: reveal and flood-reveal; loss if a mine.

toggle_flag(r,c); int: toggle flag with cap; supports win-by-flags.

Tracks is_first_click, is_game_over, did_win, revealed_safe_cells, flags_placed, total_safe_cells.

check_win() / check_loss() (or combined) set end state.

UI_renderer.py — Tkinter UI:

Takes user input, initializes board manager, and game logic

Builds button grid; binds left-click to reveal, right-click (and <Button-2> fallback) to flag.

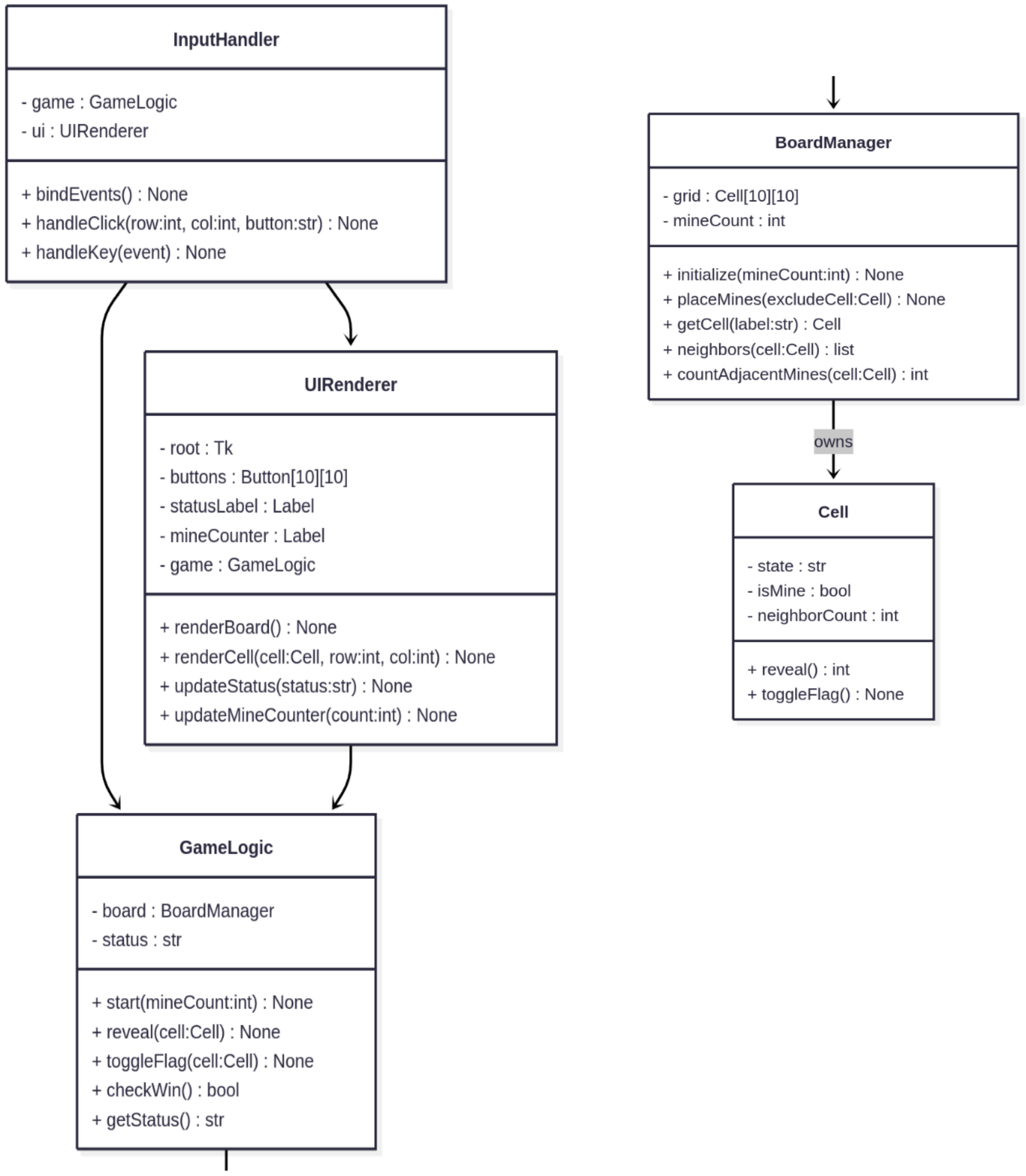
renderCell() updates numbers/flags/mines and disables revealed cells.

Shows You Win / Game Over dialog and calls endGame() to lock input.

main.py — entry point:

Initializes and runs GUI

System Architecture: UML Project Architecture


InputHandler
- game : GameLogic
- ui : UIRenderer
+ bindEvents() : None
+ handleClick(row:int, col:int, button:str) : None
+ handleKey(event) : None
UIRenderer
- root : Tk
- buttons : Button[10][10]
- statusLabel : Label
- mineCounter : Label
- game : GameLogic
+ renderBoard() : None
+ renderCell(cell:Cell, row:int, col:int) : None
+ updateStatus(status:str) : None
+ updateMineCounter(count:int) : None
GameLogic
- board : BoardManager
- status : str
+ start(mineCount:int) : None
+ reveal(cell:Cell) : None
+ toggleFlag(cell:Cell) : None
+ checkWin() : bool
+ getStatus() : str
BoardManager
- grid : Cell[10][10]
- mineCount : int
+ initialize(mineCount:int) : None
+ placeMines(excludeCell:Cell) : None
+ getCell(label:str) : Cell
+ neighbors(cell:Cell) : list
+ countAdjacentMines(cell:Cell) : int
owns
Cell
- state : str
- isMine : bool
- neighborCount : int
+ reveal() : int
+ toggleFlag() : None

*continues above to the right

Hours Logged: Task Tracking

User Story 1

User	Player
Requirement	Render covered a 10 x 10 board with addressing
Story Points	5

Name	Hours	Description
Genea Dinnall	.75	Created render board UI in tkinter

User Story 2

User	Player
Requirement	Add an input prompt with validation for mine quantity
Story Points	1

Name	Hours	Description
Genea Dinnall	.5	Created pop-up upon GUI initialization prompting user for count, and validation.

User Story 3

User	Player
Requirement	Create random mine placement without overlap
Story Points	2

Name	Hours	Description
Sam Kelemen	0.25	Created version 2 of mine placing to change return value for compatibility with architecture.
Genea Dinnall	.5	Created version 1 or mine placing with random sample of coordinates

User Story 4

User	Player
Requirement	Track the first click and prevent it from being a mine
Story Points	3

Name	Hours	Description
Sam Kelemen	0.25	Allowed devs to pass first click position to the bomb placement method.
Jenny Tsotezo	0.5	Ensure that the first click lands on a safe cell
Megan Taggart	.5	Ensured neighbors of the first-click cell are also not bombs, for playability

User Story 5

User	Player
Requirement	Create a function to count neighboring bombs
Story Points	2

Name	Hours	Description
Megan Taggart	.5	Created “compute_adj_mines” to work with previously implemented mine counting function, utilizes “cell.neighbor_count”

User Story 6

User	Player
Requirement	Implement a flag toggling function that links to the UI
Story Points	1

Name	Hours	Description
Jenny Tsotezo	0.25	Implemented the flag toggling function

Megan Taggart	.5	Connected flag-function in UI to flag-function in gamelogic.py, confirmed right-click toggling

User Story 7

User	Player
Requirement	Implement win/loss functionality that reveals mines and messaging
Story Points	2

Name	Hours	Description
Megan Taggart	.75	Implemented win checkers in various functions like reveal and flag-toggling, and initialized it in game logic

User Story 8

User	Player
Requirement	Create a user input system with either keyboard or clicks, and create a flag toggle input. Validate spaces
Story Points	3

Name	Hours	Description
Genea Dinnall	2.5	Worked on rendering in tkinter, added definitions. Determined generation of cell images on the game board.

User Story 9

User	Dev
Requirement	Create a board manager class, integrate the API, and write adjacency
Story Points	3

Name	Hours	Description

Sam Kelemen	0.1	Created the BoardManager class, and method definitions.
Genea Dinnall	1.5	Added V1 mine placing, get cells, neighbors, and count adjacent functions

User Story 10

User	Dev
Requirement	Create a Game logic class with rules, status tracking, and write tests
Story Points	5

Name	Hours	Description
Jenny Tsotezo	2	Create a game logic.py file with functions like reveal cell.

User Story 11

User	Dev
Requirement	Create an input handler, map events, and validate
Story Points	2

Name	Hours	Description
Genea Dinnall	1.25	Attached button click to reveal function, and added functions to work with reveal to maintain game logic

User Story 12

User	Dev
Requirement	Implement a cell-based renderer to handle state changes
Story Points	2

Name	Hours	Description

Sam Kelemen	.75	Added switching between render state based on flagging
Genea Dinnall	.25	Added revert state for flag being clicked again