**Project 1: Minesweeper System Development**

**Course:** EECS 581 Software Engineering II, Fall 2025

**Professor Hossein Saiedian**

Overview

Develop Minesweeper, a single-player puzzle game. Players interact with a 10x10 grid, uncovering cells to reveal numbers indicating adjacent mines while avoiding detonation. Players can flag suspected mine locations.

Requirements

**HINT: Game Setup**

* Board Configuration
* Size: 10x10 grid
* Columns labeled A–J; rows numbered 1–10
* Mine Configuration
* Number of mines: User-specified, 10 to 20
* Randomly placed at game start
* First clicked cell (and optionally adjacent cells) guaranteed mine-free
* Initial State: All cells start covered, with no flags

**Gameplay**

* Players uncover a cell by selecting it (e.g., clicking)
* Uncovering a mine ends the game (loss)
* Uncovering a mine-free cell reveals a number (0–8) indicating adjacent mines
* Cells with zero adjacent mines trigger recursive uncovering of adjacent cells
* Players can toggle flags on covered cells to mark suspected mines

**Mine Flagging**

* Players place/remove flags on covered cells to indicate potential mines
* Flagged cells cannot be uncovered until unflagged
* Display remaining flag count (total mines minus placed flags)

**Player Interface**

* Display a 10x10 grid showing cell states: covered, flagged, or uncovered (number or empty for zero adjacent mines)
* Show remaining mine count (total mines minus flags)
* Provide a status indicator (e.g., “Playing,” “Game Over: Loss,” “Victory”)

**Game Conclusion**

* Loss: Triggered by uncovering a mine, revealing all mines
* Win: Achieved by uncovering all non-mine cells without detonating any mines

Submission Requirements

* Code Freeze: Freeze code on the master branch of the team’s GitHub repository by the due date. Compliance is based on the final commit timestamp
* Demo: Demonstrate project progress during the weekly GTA/team meeting using the master branch as of the code freeze
* Artifacts: Store all code and documentation in the team’s GitHub repository on the master branch
* Peer Reviews: Submit individual Team Peer Evaluation forms via Canvas by the due date

HINT: System Architecture

* **Purpose:** Describes the high-level structure to facilitate feature extensions by the Project 2 team
* **Components:**
  + Board Manager: Manages the 10x10 grid as a 2D array, tracking cell states (covered, flagged, uncovered, mine)
  + Game Logic: Handles gameplay rules, including mine placement, cell uncovering, recursive revealing, and win/loss detection
  + User Interface: Renders the grid, status indicators (e.g., mine count, game state), and user inputs (clicks for uncovering/flagging)
  + Input Handler: Processes user inputs (e.g., clicks, key presses) and communicates with Game Logic to update the Board
* **Data Flow:**
  + User input (click) → Input Handler validates and sends to Game Logic
  + Game Logic updates Board state (e.g., uncover cell, place flag)
  + Board state changes trigger UI updates (e.g., render number, flag, or mine)
* **Key Data Structures:**
  + 2D array (10x10) for grid: stores cell states (0 = covered, 1 = flagged, 2 = uncovered number, 3 = mine)
  + Game state object: tracks mine count, flags remaining, and win/loss status
* **Assumptions:**
  + Fixed 10x10 grid size
  + Mine count user-specified (10–20) at game start

Language and Platform

Teams choose the development environment and the programming language (e.g., CLI, HTML/CSS/JavaScript, Python, C, C++, Go).

**Grading Criteria (100 Points)**

1. **Working Product Demonstration (40 Points)**
   * Platform: Conducted on a device of your choice during the weekly GTA/team meeting
   * Evaluation:
     + Presence of all specified features
     + Withstands stress testing (penalties for crashes or memory leaks)
     + Intuitive interface, requiring no manual
2. **System Documentation (40 Points)**
   * Person-Hours Estimate (10 Points): Detail your methodology for estimated hours
   * Actual Person-Hours (10 Points): Day-by-day accounting of each member’s hours (excluding EECS 581 lectures)
   * System Architecture Overview (20 Points): High-level description and diagram of system components, data flow, and key data structures
3. **Code Documentation and Comments (20 Points)**
   * Prologue Comments: Include for each file
     + Function, class, module name and brief description
     + Inputs and outputs
     + External sources (e.g., generative AI, StackOverflow) with attribution
     + Author’s full name and creation date
   * In-Code Comments:
     + Comment major code blocks and/or individual lines to explain functionality
     + Indicate whether code is original, sourced, or combined
     + Ensure clarity for GTA and Project 2 team comprehension
4. **Source Attribution**
   * Clearly identify external code sources and rephrase comments distinctly
   * Failure to attribute sources constitutes academic misconduct (see the course syllabus)

***Mandatory Peer Evaluation (-25 points if not completed)***

Each team member must complete the peer evaluation: Act as a manager and divide a $10,000 bonus among team members (submit on Canvas).

**Project Evaluation Rubric**

1. Working Product Demonstration (40 Points)

* **Exceeds Expectations (90–100%)**  
  All specified features are present (board configuration, gameplay, mine flagging, player interface); system is stable under stress testing; user interface is intuitive without requiring a manual; code is highly modular and extensible.
* **Meets Expectations (80–89%)**  
  Most specified features are present (at least three of: board configuration, gameplay, mine flagging, player interface); system is mostly stable but may have minor issues under stress testing; user interface is mostly intuitive but may require minimal guidance.
* **Unsatisfactory (0–79%)**  
  Two or fewer specified features are fully implemented; system crashes or has significant memory leaks; user interface is confusing or requires extensive guidance.

2. Estimate of Person-Hours (10 Points)

* **Exceeds Expectations (90–100%)**  
  Detailed methodology for estimating person-hours is complete, clear, and well-justified, enabling easy understanding by the GTA and Project 2 team.
* **Meets Expectations (80–89%)**  
  Methodology for estimating person-hours is provided but lacks some clarity or detail, making it slightly difficult to understand.
* **Unsatisfactory (0–79%)**  
  No estimate provided (0 points); or estimate provided without any methodology or explanation (60 points).

3. Actual Accounting of Person-Hours (10 Points)

* **Exceeds Expectations (90–100%)**  
  Complete day-by-day accounting from each team member, detailing hours spent on coding, testing, meetings, and documentation (excluding EECS 581 lectures), with clear and accurate records.
* **Meets Expectations (80–89%)**  
  Incomplete day-by-day accounting from team members, or includes non-project time (e.g., EECS 581 lectures), or minor inaccuracies in reporting.
* **Unsatisfactory (0–79%)**  
  No accounting provided, or accounting is fabricated or significantly incomplete.

4. System Documentation (20 Points)

* **Exceeds Expectations (90–100%)**  
  Comprehensive system architecture overview and documentation in the GitHub repository’s master branch, including detailed descriptions and diagrams of components, data flow, and key data structures, enabling the Project 2 team to easily extend the system.
* **Meets Expectations (80–89%)**  
  System documentation is mostly complete but missing minor details or lacks some clarity in describing components, data flow, or data structures, requiring slight effort from the Project 2 team.
* **Unsatisfactory (0–79%)**  
  System documentation is missing significant details, lacks diagrams, or is insufficient for the Project 2 team to understand and extend the system.

5. Code Documentation and Comments (20 Points)

* **Exceeds Expectations (90–100%)**  
  Prologue comments in each file include function/class/module name, description, inputs/outputs, external sources with attribution, author’s name, and creation date; major code blocks and individual lines are clearly commented to explain functionality, with clear attribution for original, sourced, or combined code.
* **Meets Expectations (80–89%)**  
  Prologue comments are present but missing some required elements (e.g., inputs/outputs or attribution); some major code blocks or individual lines lack comments, or attribution is incomplete.
* **Unsatisfactory (0–79%)**  
  Prologue comments are missing entirely, or major code blocks and lines have minimal or no comments, or external sources are not attributed, risking academic misconduct.