JayHawk Dream Team

Project 2: Upgrades To Team 1's Game Software Architecture Document

Version 1.0

Revision History

Date	Version	Description	Author
3/10/2025	1.0	Compilation of Notes	Carlos Mbendera

Table of Contents

Software Architecture Document

1. Introduction

This Software Architecture Document (SAD) outlines the architectural framework and high level design for Project 2, an update of a pygame version of Minesweeper developed by another team (Team 1) as part of our EECS 581 Coursework at the University of Kansas. This blueprint helps ensure that the development aligns with the Software Requirements Specifications (SRS), guiding the team towards successful implementation.

1.1 Purpose

This document delivers a comprehensive architectural overview of our updates capturing key decisions and presenting them via various architectural views. It is designed for the project team, to guide implementation, and for stakeholders seeking detailed insights into the software's capabilities.

1.2 Scope

The scope of this document encompasses the complete architecture of the updates we wrote and our general understanding of the preexisting code. In particular this document covers interactive and automatic AI Solver, adding missing features that were omitted by Team 1 alongside our choice to add Sound Effects to the game.

1.3 Definitions, Acronyms, and Abbreviations

- **SRS**: Software Requirements Specification
- **API**: Application Programming Interface
- **UI**: User Interface
- **pygame**: A python framework that offers function calls for features often used in game development

1.4 References

- EECS 348 Software Engineering Course Syllabus, Professor Hossein Saiedian, University of Kansas, Spring Semester 2024.
- EECS 581Project 1: Minesweeper System Development Specifications, Professor Hossein Saiedian, University of Kansas, Fall Semester 2025.
- EECS 581Project 2: Minesweeper Project Update Specifications, Professor Hossein Saiedian, University of Kansas, Fall Semester 2025.
- ChatGPT

1.5 Overview

This document covers our software design architecture. It will touch on various items such as size, interface, and structure. It covers:

- **Architectural Representation:** Outlines views and models representing the system.
- **Architectural Goals and Constraints:** Details architectural requirements and objectives.
- **Logical View:** Describes system decomposition into packages and classes.
- **Interface Description:** Provides high-level details of key interfaces.
- **Development Description:** Overview of the hours we worked and our development process

2. Architectural Representation

The software architecture for the system encompasses multiple views that collectively provide a comprehensive understanding of the system's structure and behavior. Each view focuses on specific aspects of the architecture guiding the development and evolution of the system.

1. Overview:

Description: Provides a high level overview of the system architecture, highlighting its key components and their interactions.

Model Elements:

- Core Runtime: Single event loop (pygame) managing input → state mutation → render → audio.
- State Collections: Grid matrix, bomb set, revealed set, flagged set, AI action queue, timers, game outcome flags.
- Functional Blocks:
    - Initialization (pygame, mixer, asset load, bomb count dialog)
    - Board Generation (bomb placement, number derivation, safe-first adjustment)
    - Interaction Handling (mouse routing, button activation, AI mode cycling)
    - Reveal & Propagation (flood fill expansion)
    - AI Turn Orchestration (queue stepping with timed animation)
    - Rendering (UI banner, labels, cells, highlight, overlays, popups)
    - Audio Feedback (looped background + contextual SFX)

2. Logical View:

Description: Describes the logical organization of the system, focusing on the decomposition of functionality into modules or layers.

Model Elements:

- Present Logical Segments (all inside minesweeper.py):
    - Board Logic: generate_bombs, ensure_safe_start, generate_numbers, flood_fill
    - Game Control: main() loop, win/loss evaluation, first click safeguard
    - UI & Rendering: title/timer counters, coordinate labels, cell painting, popup, AI highlight
    - Input Layer: mouse event parsing (buttons → actions; board clicks → reveal/flag)
    - AI Mediation: action queue management, timed step execution, visual highlighting
    - Utility: auto_solve, bomb count dialog get_bomb_count
    - Media Handling: mixer init, sound asset loading, volume assignment
- AI and Pygame Interface Points:
    - AI Strategy Hook: take_turn(...) returns iterable of ('flag' | 'reveal', (r,c))
    - Pygame Services: event queue, surfaces, timing, audio playback

3. Process View

Description: Shows the dynamic behavior of the system, illustrating how processes and threads interact to fulfill user input on the Minesweeper board..

Model Elements:

- Primary Cycle (per frame):
    - Poll events (quit, mouse down, keyboard for dialog)
    - Dispatch button interactions (Auto Solve, AI mode cycle, popup actions)
    - Translate board clicks → flag or reveal
    - Execute AI animation step if queue pending and step interval elapsed
    - Update win/lose conditions (safe cell count vs revealed)
    - Compose frame (UI banners → labels → grid → overlays)

    - Play context sounds (bomb, flag, reveal, win/lose)
- **Data Flow:**
    - Input (mouse) → coordinate mapping → state mutation (revealed/flagged)
    - AI request → queued actions → timed dequeue → state updates → optional termination (loss/win)
    - State → rendering functions → screen surface → display flip

4. **Development View**
Description: Focuses on the organization of the development effort, including the structure of source code, version control, and development environments.

Model Elements:
- **Code Layout:** Single script minesweeper.py plus ai.py and asset directory for audio.
- **Version Control**: Git / GitHub.
- **Tooling & Environment**: Visual Studio Code (Python + pygame extensions), Pycharm local audio assets, standard Python interpreter.
- **Process & Workflow**: Incremental feature layering (sounds → AI queue animation → UI polish).

3. Architectural Constraints

- **Developmental tools**: We will utilize Visual Studio Code for its versatility, Git and GitHub for version control and PyCharm / VSCode with the Python Extension, ensuring a robust development environment.
- **Design:** The inherited project is a single monolithic script
- **Usability**: We hope to bring our experience from Project 1 into this project by maintaining an intuitive user interface and solid technical backend to enhance gameplay experience.
- **Design and implementation strategy:** We plan to follow a waterfall development methodology, to ensure phases and requirements are met in a structured progression.

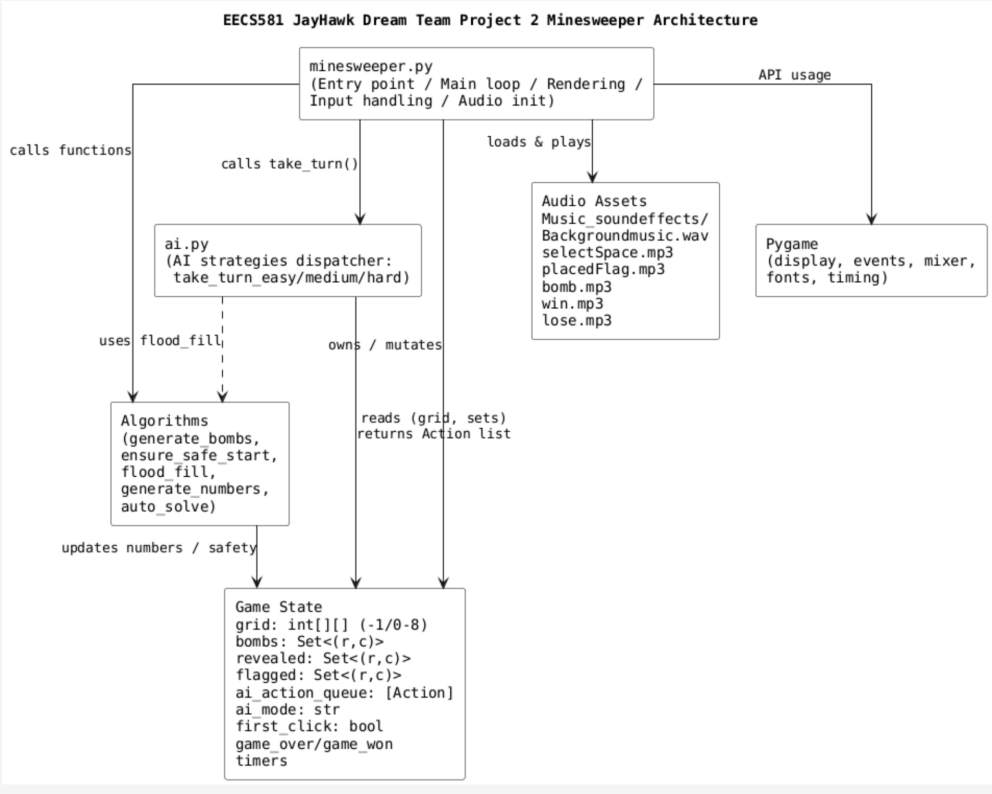
4. Logical View

4.1 Overview

The overall design of the Minesweeper program is divided into various major subsystems that interact to deliver the game's functionality. These are the

- **Game Loop Controller:** Central event loop sequencing
- **Board State & Data:** 2D grid, sets for bombs, revealed, flagged, plus AI action queue and timing markers
- **Bomb & Number Generation:** generate bombs, numbers , and safe start
- **Reveal Logic:** flood fill
- **Input & Interaction:** Mouse dispatch, UI buttons, popup buttons, bomb count dialog
- **AI Integration:** Auto Solver and Interactive Mode
- **Rendering & UI:** Full frame redraw
- **Audio Feedback:** Background loop plus contextual SFX

4.2 Architecturally Significant Design Modules or Packages



5. Interface Description

The controls for the game were not modified whilst we worked on this project and are as follows:

Action	**Input**
Reveal cell	Left click
Place/remove flag	Right click
Cycle AI mode	Click "AI: " button
Auto Solve instantly	Click "Auto Solve" button
Confirm bomb count	Enter / Confirm button after typing (10–20)
Play Again (after end)	Click green "Play Again"
Quit (after end)	Click red "Quit"

6. Development Description

6.1 Overview

The updates were built following an Agile system with scrums lasting a week. Every Monday, we met with our scrum master and discussed upcoming and completed work. The expected and actual times for the work are as follows:

6.2 Time Taken

Name	Expected Time	Actual Time
Carlos Mbendera	2–3 hours	2 Hours
Yoseph Ephrem	1 hour 30 min	2 hours 30 min
Cole DuBois	1 hour	2 hours 30 min
Blake Ebner	45 min	45 min
Jonathan Johnston	1 hour 30 min	1 hour 30 min
Mahdi Essawi	1 hour 30 min	1 hour

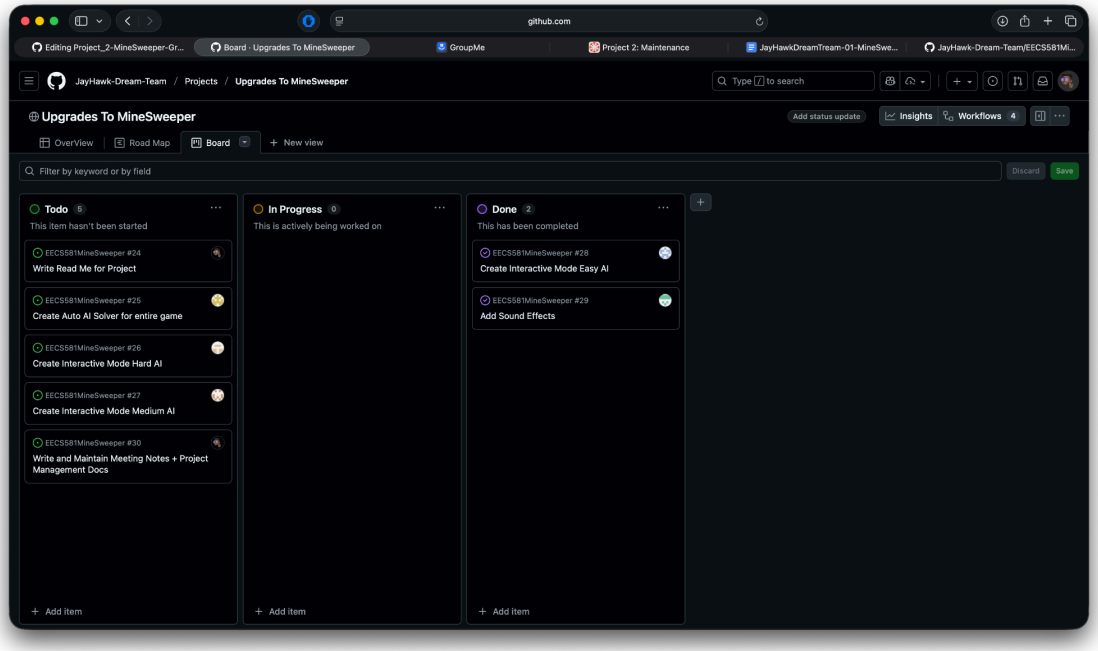
6.3 Time Tracking Tools

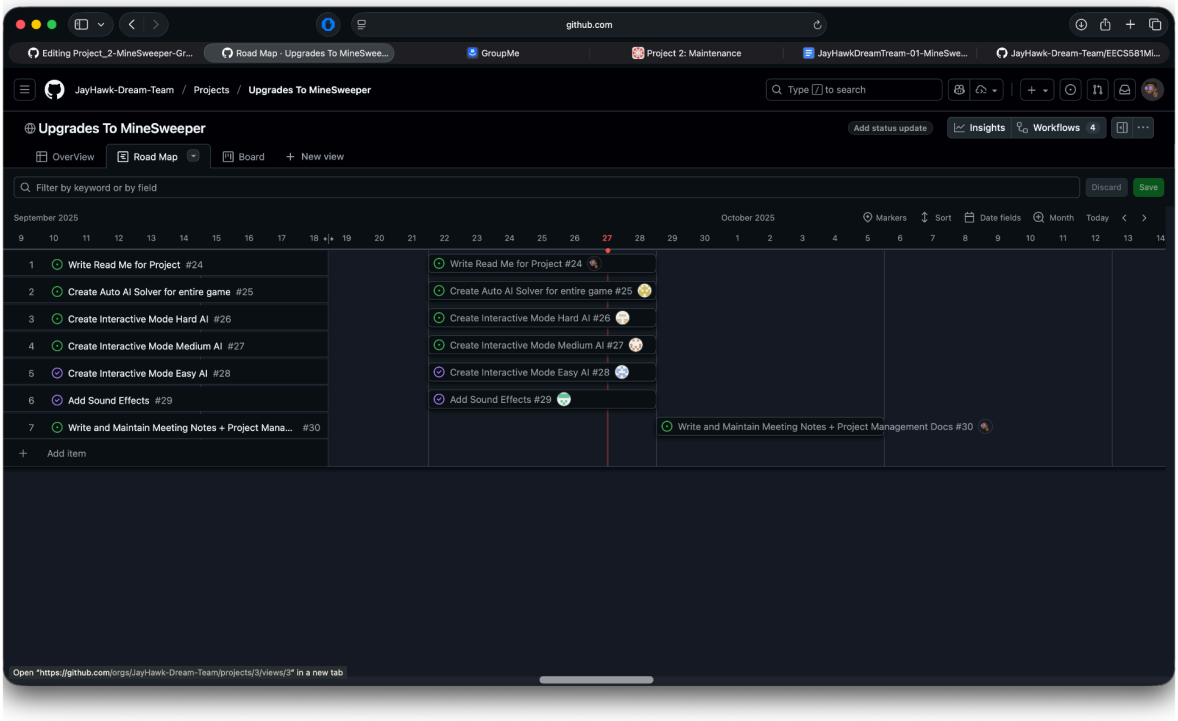
Similar to Project 1, we used GitHub extensively and GitHub Projects to keep track of our roles, tasks and timelines of each other's work.

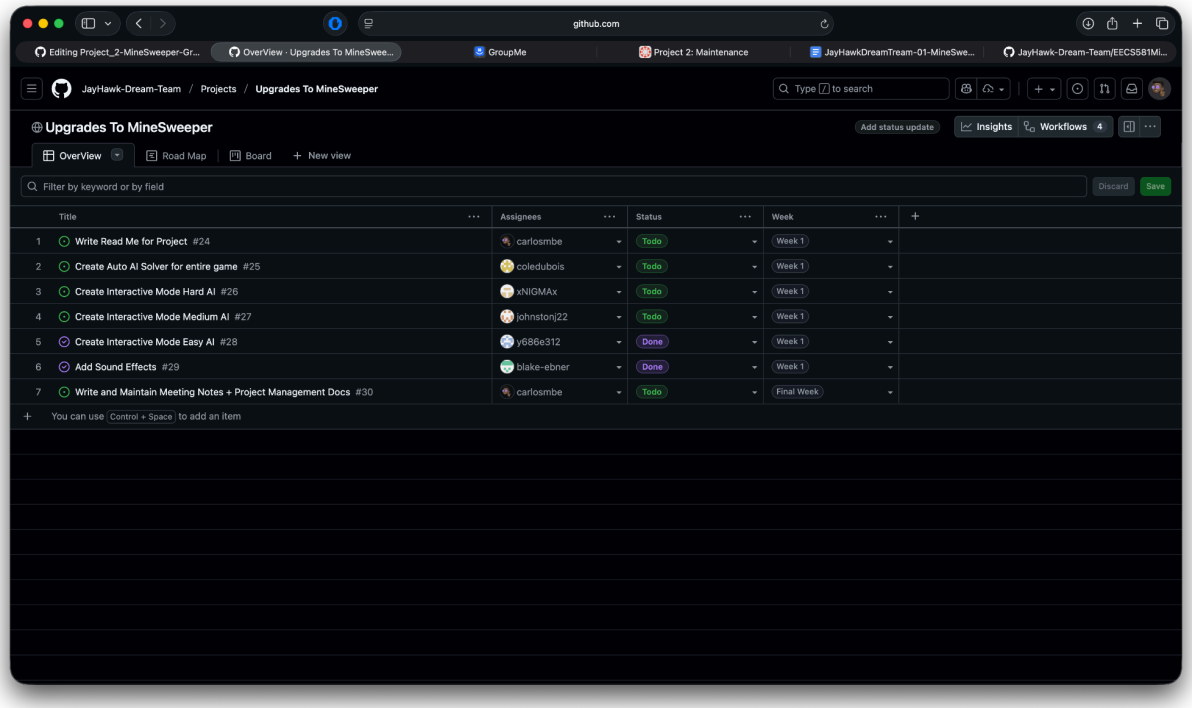
By observing our GitHub Project, you shall notice that we had a planning board to separate tasks into 3 sections, Todo, In Progress, Done. This made it possible for everyone to be synchronized on the status of components they may be dependent on.

Additionally, we also had used the Roadmap feature to get a rough timeline of when everything was needed on an iteration by iteration basis. In this case, each iteration was a week.

A new addition for this project is that we also used a simple list view. This was because the tasks we had were a lot smaller than project 1.


github.com
JayHawk-Dream-Team / Projects / Upgrades To MineSweeper
Upgrades To MineSweeper
Add status update
Insights
Workflows 4
OverView
Road Map
Board
New view
Filter by keyword or by field
Discard
Save
Todo 5
This item hasn't been started
EECS581MineSweeper #24
Write Read Me for Project
EECS581MineSweeper #25
Create Auto AI Solver for entire game
EECS581MineSweeper #26
Create Interactive Mode Hard AI
EECS581MineSweeper #27
Create Interactive Mode Medium AI
EECS581MineSweeper #30
Write and Maintain Meeting Notes + Project Management Docs
Add item
In Progress 0
This is actively being worked on
Add item
Done 2
This has been completed
EECS581MineSweeper #28
Create Interactive Mode Easy AI
EECS581MineSweeper #29
Add Sound Effects
Add item


github.com
JayHawk-Dream-Team / Projects / Upgrades To MineSweeper
Upgrades To MineSweeper
Add status update
Insights
Workflows 4
OverView
Road Map
Board
New view
Filter by keyword or by field
Discard
Save
September 2025
October 2025
Markers
Sort
Date fields
Month
Today
Write Read Me for Project #24
Create Auto AI Solver for entire game #25
Create Interactive Mode Hard AI #26
Create Interactive Mode Medium AI #27
Create Interactive Mode Easy AI #28
Add Sound Effects #29
Write and Maintain Meeting Notes + Project Mana... #30
Add item
Open "https://github.com/orgs/JayHawk-Dream-Team/projects/3/views/3" in a new tab



6.4 Meeting Notes

For this project, we had 2 meetings on September 21, 2025 and September 28, 2025 after meeting with our TA and understanding the scope of the project. Further communication occurred via a group chat in Group Me. By September 29, 2025, prior to our TA meeting, most of the work was already complete.

Meeting Notes – September 22, 2025

During our weekly scrum meeting, we demoed project 1 and began discussing Project 2 upgrades. After our session with the TA, we continued the conversation by ourselves as a team and agreed to the following.

Key Decisions

- Custom Feature: We agreed to add a suite of audio features and sound effects. Blake will lead this work.
- Visual Features: After reviewing the inherited project's source code, we noticed several missing visual elements. Mahdi volunteered to handle this.
- AI Solver & Project Management: We outlined responsibilities for the AI Solver and general project organization.

Task Assignments:

- Carlos – Documentation, project management, and GitHub repository setup (tasks include assigning work, taking meeting notes, and setting up a fork of the repo).
- Blake – Sound effects and audio features.
- Yoseph – Easy Mode AI.
- Jonathan – Medium Mode AI
- Cole – Auto Solver
- Mahdi – Hard Mode AI + Visual features.

Meeting Notes – September 29, 2025

Following our meeting with our TA we had realized that we had misunderstood the auto solver's

requirements. In particular, we created an AI that would complete the game of minesweeper in one swoop. However, after our demo, the TA mentioned that it should also have 3 difficulty levels similar to the interactive AI. Since Jonathan had experience working on Medium mode and the AI Solver was already complete to some extent, he volunteered to handle these implementations, whilst the rest of the team would offer help in any way that was possible if space was available. Mahdi also planned to get the visuals polished up this week.