Table 1: Revision History

Date	**Developer(s)**	**Change**
Sept 18^{th}	Michael Ilao	Tech Stack, POC, Coding Standard
Sept 22^{nd}	Hargun Bedi	Team Meeting Plan, Team Communication Plan, Workflow Plan
Sept 24^{th}	Michael Ilao	Project Schedule and Member Roles
Sept 25^{th}	Jeffrey Dang	Revisions on Development Plan
Apr 3^{rd}	Tianzheng Mai	Closed issues and final revised

Development Plan

This document will outline the development and documentation plan for An AI-based Approach to Designing Board Games project.

1 Team Meeting Plan

The team has decided to have weekly meetings on Fridays, from 11:30am to 12:30pm on our Microsoft Teams group. Additionally, the team can have ad-hoc meetings if the team unanimously deems it necessary.
The team has also scheduled weekly checkpoint meetings with our supervisor, Dr. Sebastien Mosser, on Tuesdays, from 5:00pm – 6:00pm.

1.1 Meeting Rules

- Team leader will chair the meeting.
- All team members will give an update on their assigned tasks.
- All agenda items will be discussed sequentially.
- All team members will have an opportunity to ask questions and voice any concerns they have.
- If a new task is being assigned to any member, the team should decide the appropriate timeline unanimously.
- If a member requests a revision to the meeting time, agenda, or deadline, they must provide a 48-hour notice and the changes will be agreed upon unanimously.
- At the end of each meeting, an agenda will be made for the next meeting.

2 Team Communication Plan

The team's communication will be done virtually, primarily on our Microsoft Team's group chat. Any questions, concerns, requests to change deadlines, or general discussions will be done on the Team's group chat. If necessary, we will also be meeting in-person, at the team's decided location prior to the meeting. Additionally, all emails sent to our supervisors, professor, or TA's must have all team members CC'd on it.

3 Team Member Roles

As the architecture will require ~~two~~ three main parts of development, our developers~~' effort~~' efforts will be focused so they can understand their specific system on a deeper level. The ~~two~~ three roles will be AI Game Agent developers, who will work on the Artificial Intelligence (AI) Game Agent~~and~~, the Simulation Structure developers who will focus on the architecture of the game engine and how it connects to the AI Game Agent~~.~~, and the Data Visualization Developer who will work on the data visualizer user interface to communicate complex data in different charts and identify key insights to the users.

- Jonah: AI Game Agent - Developer
- Jeffrey: AI Game Agent - Developer
- Hargun: ~~Simulation Agent~~ Data Visualization - Developer
- Michael: Simulation Structure - Developer
- Tianzheng: ~~Simulation Structure~~ Data Visualization - Developer

All the developers will be required to understand the system at a high-level on both parts. They will also be required to work on documentation, testing, requirements and actively attend all meetings with the Stakeholders.

4 Workflow Plan

Teammates are required to use the private GitHub repository, AIBoardGame. In order to facilitate simultaneous development of modules, we will be using the "feature-branch" workflow. First, each team member will pull the latest changes from the "main" branch into their local repository. Next, changes will be made by creating a new "feature" branch. It is expected that feature branches will be given descriptive names, like "ai-implementation" or "issue #31". The idea is to give a clear purpose to each branch. Once the changes are made, unit testing will be performed on the feature branch to ensure the code is working as intended. When the feature branch is merged with the main branch, 1 or 2 other team members may do integration testing to verify the if the merge did not cause any other issues.

4.1 Issue Tracking

All tasks will be completed and tracked using 4 label categories on GitHub issue tracker – *Major, Minor, Bug,* and *New Feature*. Each issue description should include all necessary details to completely understand the issue, and will be using one or more of the labels mentioned below. Issues can be assigned to the team member that can best solve the issue.

Major label will be used when the main branch is not able to run or if the code keeps crashing on a specific action.

Minor label will be used when there are issues identified with the UI, inadequate error handling, uncaught errors in unit/integration testing, etc.

Bug label will be used when there are unintended results due to logical errors, inputs not being handled properly, outputs not being generated properly, etc.

New Feature label will be used when a new feature or improvement is being added to the project.

All issues will be required to be fixed in a separate branch with an appropriate issue name. All issues will be reported to the team either in the team meeting or in the team group chat. Issues will be closed when the appropriate pull request is merged to the main branch by including a closing branch name/description (close issue #31).

4.2 Tags

Tags will be used to mark milestones in the development of the project. Major milestones will be incrementing the major version number (v2.0 to v3.0). Minor milestones will be incrementing the minor version number (v2.1 to v2.2). The team will decide on an ad-hoc basis on what will be considered a major/minor milestone as the project is in its development stage.

5 Proof of Concept Demonstration Plan

For a proof of concept demonstration, the base architecture must be working and implemented properly. This means that the simulation and AI Game Agent are fully modular and can be swapped with different AI Agents or even a different simulation engine. The simulation engine to be implemented will be a simple game of Tic-Tac-Toe. If the integration of AI Game Agents can play on the simulation of Tic-Tac-Toe, the project has a great chance of success as we will know the architecture is able to support more complex game agents and a more complex simulation engine. ~~These~~ After the implementation of Tic-Tac-Toe, the frontend data visualizer should utilize the AI Game Agents' data for different chart modules. The users can customize different data points, and view and compare the data and information in various charts. These three systems will be developed in parallel to maximize our time constraints. These two systems will be developed in parallel to maximize our time constraints. The major risk of this project is how to create a modular architecture and able to integrate a simulation engine that can be played by real players as well as AI players. Testing this will not be difficult for the POC as the game chosen Tic-Tac-Toe is a simple game and we can test the success of our AI's quickly as many simulations can be ran quickly.

In our demonstration we will be able to show that are architecture is modular enough to support different kinds of AI Game Agents and that they are seamlessly able to integrate into the simulation (by swapping different AI Game Agents in/out).

6 Technology

- Python will be used to develop the simulation engine and be used for simulation the AI players. The choice of this language is due to Python's Machine Learning/Artificial Intelligence libraries and the support for Object Oriented Programming.
- Object Oriented Programming will be used as the design methodology as to allow multiple developer to work seamlessly on the same project and for extensibility to other possible board games/simulations.
- To ensure common programming standards, developers will use pylint to maintain the same coding style across files. VSCode and prettier will be used for automatic formatting and linting.
- JavaScript, HTML, CSS, and ReactJS will be used to develop a data visualization interface that can fetch the output log file in the game engine backend to the frontend. React is a component-based frontend framework that breaks down complex UIs into smaller and reusable components and has a large ecosystem of libraries, tools, and resources that support the implementation of different charts.
- Pytest will be used for integration and unit testing for the Simulation Engine. For testing the AI Game engine, developers will need to build custom test to verify they are working to an acceptable degree.
- Coverage.py will be used for code coverage as it integrates easily with pytest.
- There are no immediate plans for Continuous Integration/Continuous Deployment as the project will be used for testing by the Stakeholders, which does not need to be hosted on any cloud environment.
- timeit Python library will be used for measuring performance and time of individual modules.
- ML/AI Libraries to be used will be PyTorch, Tensor Flow, NumPy and Pandas.
- The main tools used will be VSCode and any available Python extensions.

7 Coding Standard

The programming paradigm that will be used in this project is Object Oriented Programming or OOP.This will allow to developers to structure code for re-use and extensibility. This abstract way of programming will allow for the system to be integrate into different board games. Another standard that will be used is PascalCase for Class naming and camelCase for method and variable naming. PEP8 will also be used to enforce readable code and good python practices.

8 Project Scheduling

The project schedule will require developers to work on documentation and planning in parallel with the code. This will benefit developers as they will be able to adjust their methodologies and documentation faster than if it was done sequentially.

The project will be managed with GitHub Projects by assigning tasks to team members.

The Major milestones are the Requirements Doc Rev 0, POC Demo, Rev 0 Demo and Final Demo. These tasks will be broken down and split up by different sections that each team member can individually contribute to. Documentation will be assigned to team members based on what their dev-role is, AI Game Agent developers will work on documentation related to this part of development and Simulation developers will work on documentation relating to that. This will mean team members may be working on the same section together. At a high level all team members must still understand each part of the project and system.

Communication between the team will be a major factor to fully utilize each team members time. As there are multiple people working on the same part, they must work collaboratively to ensure there is no code or documentation overlap.