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
...	...	...

Development Plan
SE 4G06

Team #6, Board Gamers
Ilao Michael, ilaom
Bedi Hargun, bedih
Dang Jeffery, dangj12
Ada Jonah, karaatan
Mai Tianzheng, mait6

[Put your blurb here. —SS]

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

4 Workflow Plan

Teammates are required to use the private GitHub repository named, 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 issues will be under 3 categories – Major, Minor, Bug.
Major issues will be when the main branch is not able to run or if the code keeps crashing on a specific action.
Minor issues will be when there are issues identified with the UI, inadequate error handling, uncaught errors in unit/integration testing, etc.
Bug issues will be when there are unintended results due to logical errors, inputs not being handled properly, outputs not being generated properly, etc.
Bugs and 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.

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 demonstartion, two parts of the system must be functional to a basic level. The first being the Simulation of the board game, it will not require all actions and rules to be implemented, only the core mechanics of the game. This core system should be implemented so all actions a player can make during their turn are available to be called, as if a real player is controlling them. This will allow a second system to make choices during a player's turn. This leads into the second part of a basic AI/ML Player that can control the first system. This basic player should be able to make intelligent choices based on the game state and past game states.

If these two systems can be implemented, the project will have a great chance of success as the architecture of the system will be able to support improvements to the AI/ML Player as well as implementing the rest of the game mechanics.

6 Technology

- Python will be used to develop the simulation engine and be used for simulation the AI players. The choice of this langauge 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.
- 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.
- Pytest will be used for integration and unit testing.
- Coverage.py will be used for code coverage as it integrates easily with pytest.
- There are no immediate plans for Continous Integration/Continous 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 Libararies 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 programing 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

[How will the project be scheduled? —SS]