



**CREATED BY**

Adam Greene

Chris Kirkwood

Rachel Amponsah

Steven Kelsey

Weeldy Benjamin

**FOR**

Kylie Nowokunski

14 November 2024

### 1. Table of Contents

**1 Table of Contents**

**2 Game Overview**

**3 Gameplay and Mechanics**

**3.1 Gameplay**

**3.2 Mechanics**

**3.3 Rules**

**4 Game Flow**

**4.1 Pre-Game Setup and Game Start**

**4.2 Game Progression**

**4.3 End of Game**

**5 Multiplayer Features**

**5.1 Team-Based Mechanics**

**6 World/Map Plans**

**6.1 Pre-Game Lobby**

**6.2 World Setup**

**6.3 Map Verification**

**7 User Interface**

**7.1 Command-Based Interface**

**7.2 Inventory-Based Interface**

**7.3 Book-Based Interface**

**8 Scoring and Progress Tracking**

**8.1 Targets, Objectives, and Scoring**

**8.2 Objective Completion Tracking**

**8.3 HUD (Heads-Up Display)**

**9 Customization and Configurations**

**9.1 Config.yml Settings**

**9.2 Adding Objectives From File**

**9.3 Future-Proofing**

**10 Libraries, Frameworks, and Deployment**

**10.1 Core Dependencies**

**10.2 Command Framework**

**10.3 Event Framework**

**11 Development**

**12 Suggested Additional Features**

**12.1 Custom World Generation**

**12.2 Custom Music**

**12.3 Automatic Map Verification**

**13 Appendix**

**2. Game Overview**

Administrators on the KSU Minecraft Esports server frequently run events that many students participate in. A popular event that is held on this server are “speedrun” events - where players or teams race to complete an objective or set of objectives within a set time period. The team achieves these objectives first - or the team that has achieved the most objectives before the time runs out - wins. Currently, these events are held and executed manually: administrators have to spectate the teams and manually credit them points for completing objectives. Furthermore, objectives cannot always be completed on a given seed within the specified world border. Among many other difficulties, hosting these events is not an easy task. This speedrun game mode plugin seeks to solve this problem by streamlining this process.

The Kennesaw State University Minecraft Speedrun plugin will be a competitive, objective and team-based speedrun game mode that incorporates timed-challenges where multiple teams compete to complete various tasks as soon as possible. Main focuses on this game mode include team cooperation, efficiency, and strategy to complete the objectives as soon as possible. The number of objectives that can be set is limitless and covers all possible event-based actions in Minecraft. This includes killing entities, breaking blocks, obtaining items, or entering any specified location.

The game mode will support two different speedrun modes: the first is a default speedrun, where one or multiple objectives are set, and the first team to complete all objectives wins the game. The second is a weighted speedrun, where each objective can be specified with a point weight. Naturally, more difficult objectives should be assigned a higher weight, however this is to the discretion of the server administrators. The total amount of points to win a weighted speedrun can be set in this situation, which can be less than the sum of the weight of all objectives.

**3. Gameplay and Mechanics**

**3.1 Gameplay**

There is only one objective in this game mode - and that is to complete the objectives! The list of objectives can be as short or extensive as desired. This could be a single objective, such as killing the Ender Dragon, or a longer list of objectives which are easier to achieve, such as collecting a certain number of blocks. Because objectives and speedruns can be weighted, there is a strategy component to this game. In a weighted speedrun scenario, players will have to decide which objectives they should attempt to reach the point goal in the quickest time possible. They may choose to split up, each trying to accomplish a lower-weighted objective to contribute towards their overall points, or they may team up, all opting to work together to achieve a more difficult - but higher in weight - objective, such as killing the Warden. There are a variety of different strategies that can be used, so players will have to think quickly about how they plan to complete these objectives before time runs out.

**3.2 Mechanics**

This core of what defines this game mode can be implemented in vanilla Minecraft - its main purpose is to streamline its implementation for whoever is setting up and hosting the speedrun on the back-end. The mechanics of the speedrun game mode will mostly be based on the core game of Minecraft with minimal changes for the player. This includes game physics, combat, blocks, items, and inventories. Players will control their character either using a mouse and keyboard or game console controller. Players will be able to enter combat with other players or environmental entities. They may also interact with the world by mining or building. Players will have the standard Minecraft inventory, which consists of 36 slots, each of which can hold an item stack of 64 of the same item (for most items), 4 armor slots, consisting of a corresponding slot for a helmet, chest plate, leggings, and boots, as well as an off-hand item slot.

Introduced mechanics that are non-administrative are minimal, but are outlined below:

* **Enhanced Combat Tracking:** While combat itself will be left unchanged from the vanilla game, it will be logged differently. To ensure that players always get rewarded for kills that make, combat will be logged in a time-based manner. If an entity dies, even from an event that is unrelated to a player (e.g., fall damage, burning, etc.), the player that engaged in combat with that entity most recently (to a set cooldown time) will be attributed to the kill. Likewise, certain indirect events that kill entities instantly will be attributed to players that cause the events, even if no combat has been logged previously between the player and the entity. This includes explosions from beds and explosions from end crystals.
* **Action Tracking:** Just like in the vanilla game, players will interact with the world by mining, fighting, building, and moving. These actions will be tracked by the plugin, and if they align with an objective requirement, the player’s team will be rewarded with points that are equal to the weight of the objective. All players on the server will also be notified when an objective is completed.
* **Point System:** While there is no economy introduced that contributes to in-game rewards, teams are rewarded points for completion of an objective, which contributes to the ultimate goal of winning the speedrun event.
* **Time Limit:** Gameplay will be limited to a specified time limit. Players will only be able to achieve their objectives while the timer is active.
* **Spawning:** Unlike the vanilla Minecraft spawn point which is located somewhere around (0, 0), players will be initially spawned in groups with their teammates at a specified distance from that spawn point. Teams themselves will also be spawned equidistantly from each other. If a player dies they will respawn at their team spawn point rather than the vanilla spawn point, unless they change their spawn point by other means (i.e. bed or respawn anchor).
* **Screen Flow:** The plugin will leverage Minecraft’s built-in HUD (e.g., scoreboard, boss bar), to display information such as remaining time and team progress. Additionally, the plugin will leverage Minecraft’s inventory system to provide for a GUI allowing users to select their team prior to the start of the speedrun. Finally, the plugin will leverage Minecraft’s book UI to provide a GUI for viewing objectives. More information on the GUI can be found in section 7.2 of this document.
* **Other Commands:** Players can view the full list of objectives they need to achieve as well as game rules using the command line.

**Key Mechanics Introduced**:

* **Enhanced Combat Tracking**: Ensures players are credited for kills within a cooldown period.
* **Action Tracking**: Tracks interactions such as mining and fighting to award points for objectives.
* **Point System**: Points are awarded based on objective completion, with an optional weighted system.
* **Spawn Management**: Teams spawn equidistantly from each other and respawn at team-specific spawn points.
* **HUD Display**: Uses Minecraft’s scoreboard system to display timers and team progress.

All other mechanics are purely administrative and accessed through the command line. These include ways to set or return various speedrun attributes, set objectives or game rules, and start or stop the speedrun. More information on administrative commands can be found in section 7.1 of this document.

**3.3 Rules**

Rules that affect the mechanics of vanilla Minecraft gameplay can be adjusted through the vanilla Minecraft command */gamerule*. Additionally, game rules can be set in a more permanent manner using the *config.yml.* They correspond directly to the game rules that already exist in the vanilla game. Please view Appendix (E) for a list of game rules. In addition to these, there is only one additional rule that is not administrative:

1. By default, players are not allowed to engage in PvP with their teammates. This can be changed in the server *config.yml*.

Additional back-end rules introduced by this plugin are listed below:

1. The number of teams available shall be up to 16 by default, but will change dynamically depending on the team size set by administrators and the player count of the server as follows:

***A. Number of Teams Available = ⌈Player Count / Team Size⌉***

***B. Team Size ≥ ⌈Player Count / 16⌉***

**Explanation A:** In the team GUI, the number of teams listed to be available to join will change dynamically based on the current player count (excluding administrators except if specifically included) and the team size limit that has been explicitly set. Specifically, this number must be equal to the player count divided by the team size, rounded up to the next whole number if there is any decimal point. For example, if there are 100 players on the server, and the team size is set at 10, the number of teams available in the GUI will be 10. If there are 104 players on the server, and the team size is set at 10, then the number of teams available in the GUI will be 11.

**Explanation B:** The team size attribute set by administrators must be greater than or equal to the player count divided by 16, always rounded up to the next whole number if it is a decimal. For example, if the player count is 100 and the team size is 6, then it does not satisfy this rule: 6 ≥⌈100/16⌉ → 6 ≥⌈6.25⌉ → 6 ≥7 is FALSE. In this situation, the team size will be automatically incremented to satisfy this rule → Team Size = 7.

1. Teams will be automatically balanced such that the number of players on a team does not differ more than one player from every other team.

**Explanation:** Players may be moved to a different team when the speedrun begins based on this rule. Team selection will be on a first-come-first-serve basis, meaning that the most recent players to select their team may be subject to having their team changed, while the first players to select their team will generally be safe from an adjustment. For example, if there are 100 players on the server, and the team size is 10, then there are 10 teams: each time is full. If another player joins before the game begins, then that player will be assigned to an 11th team but will be alone on the team. The plugin will automatically grab the last player that has joined teams 3-10 and add them to team 11, effectively making the size of each team either 9 or 10. Players that have not selected a team will be used to balance teams before players are pulled from their selected team.

1. Once a game is started, objectives **may** be added if necessary. The speedrun point limit will automatically increase by a number equal to the weight of the new objective.
2. Removing objectives will decrease the speedrun point limit by a number equal to the weight of the objective that is removed.
3. Once a game is started, the world border cannot be decreased; it can be increased.
4. Teams may not be disabled after a speedrun has started.
5. The seed may not be changed after a speedrun is started.
6. Admins, by default, are included in speedruns and are assumed to be participating. They are included in any team calculations. Admins can be explicitly excluded using the command /speedrun participate.
7. If attributes are not specified, they will, by default, be set to the following:

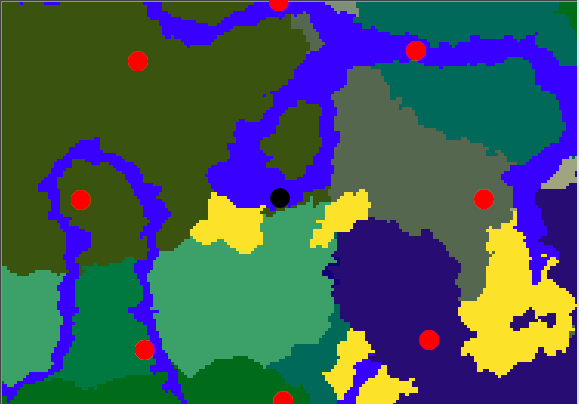
* Seed - Random
* World Border - 5,000
* Time Limit (minutes) - 60
* Team Size - 5
* Spawn Radius - 300
* Gamerules - All False
* Point Limit - Sum of Weight of all Objectives

**4. Game Flow**

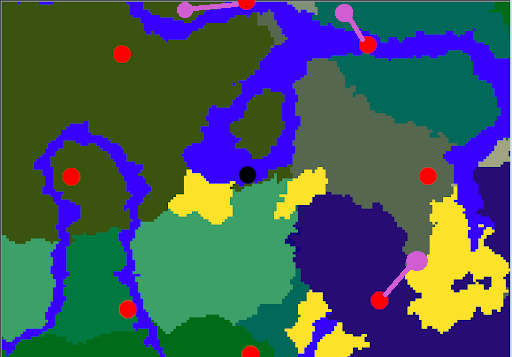
**4.1 Pre-Game Setup and Game Start**

After administrators set all objectives, customize gamerules, and modify speedrun attributes, the speedrun map can be generated using the command */speedrun generateWorld*. At this time, the plugin will generate the world and then automatically locate spawn points for all the teams. If spawn points are considered dangerous, the plugin will make multiple attempts at locating a safe area nearby. If it cannot do this after a certain number of attempts, admins will be notified, where they will have the option to manually update that team’s spawn point or pick a new seed. After the map is verified by an admin, the speedrun is ready to be started. Once ready, the administrator may execute the command */speedrun start*. At the point, teams will be reorganized according to *rule #2* in section (3.3). Each team will then be teleported equidistantly from each other. Their distance from (0, 0) is equal to the set spawn radius.

In the event that a spawn point would be set to an area with no solid blocks the plugin should move the team further around the spawn circle’s circumference.

****

The image above demonstrates a situation where the spawn radius is 300 and there are 8 teams participating. Team spawn points are demonstrated as the red points, and (0, 0) is demonstrated as the black point in the center. All teams are 300 blocks from the spawn point and equally distant from each other.



The image above demonstrates how these original calculated spawn positions may be moved to areas that have a safer spawn location as can be seen in purple. If the body of water or unsafe location is too large, an admin will be notified and will be recommended to set the spawn point manually.

**4.2 Game Progression**

After being teleported to their respective spawn points, players will officially begin the speedrun. Teams will race to complete as many (or all) objectives as possible within the set time limit. Teams may have to decide which objectives are worth attempting given their spawn point, skillset, and time limit. Using vanilla Minecraft’s scoreboard system, a scoreboard will be displayed so that players can view the time remaining, as well as how many points they have as well as how many they need to win the game. They can view information on their objectives using the command */objectives*.

**4.3 End of Game**

The game will end once a team reaches a point limit, or once the time limit expires. In an event where the time limit expires before a team reaches the point limit, the winning team will be considered the team that achieved the most points. This will be announced to all players on the server. A title screen will be displayed for all players that indicates the winner, as well as a sound that can be set in the *config.yml*.

**5. Multiplayer Features**

**5.1 Team-Based Mechanics**

The team system necessitates multiple mechanics to build the team based speedrun gameplay experience. These mechanics are the fundamentals of the team play system that the KSU Minecraft Esports administration put in place manually before this plugin was conceptualized. When a player completes an objective, the points associated with the objective are assigned to the team as a whole. Team members can work together in many different creative ways to score points, but there is nothing stopping teams from splitting up to complete scoring objectives. There is also nothing stopping multiple teams from creating alliances to fight against a common enemy.

Team members will have a color indicator on their player name that corresponds to their team color both in chat as well as in-game. This allows players who encounter other players during a speedrun to better differentiate if a player is on their team or not. Inter-team based PvP will be an option that is enabled by default, and adds an additional layer of competitive play to the speedrun game mode. Players may choose to take advantage of Minecraft’s item drop on death mechanic to assist in their own progression. Conversely, players may have to defend their teams during important objective progress that are worth more points, as other teams may swoop in and steal their achievement after the objective is complete. The team that gets the last hit on the Ender Dragon gets the points..

Intra-team based PvP will be an option that is set to false by default in the *config.yml*. Administrators can enable intra-team PvP to allow for teammates to engage in combat with each other. Enabling intra-team PvP could add a twist to the speedrun - almost acting as a sort of “hardcore” mode. Players who are feeling villainous may decide to turn on their teammates. We are not going to lie, this option is best left unmodified - but that is up to you.

**6. World/Map Plans**

**6.1 Pre-Game Lobby**

A pre-game lobby has been created; it is not necessary to be used for the plugin to work. As of the time that this document is being updated, this map is not complete. It is not nor never was part of the project’s requirements and is being made from courtesy. Images of the lobby can be viewed below.

**Lobby Map Images**



**6.2 World Setup**

The map will be procedurally generated using Minecraft’s default world generation engine. The seed for the map can be specified by an administrator or randomly generated by the plugin. Additionally, a world border may be specified - no team or player may leave the confines of the world border. The world will be generated prior to the game starting.

**6.3 Map Verification**

The world generated by a selected world seed will need to be checked to verify if certain objectives can be completed within the current set world border. Admins will need to manually go through a world before a tournament and find objective criteria. This is especially true with smaller world borders, as Minecraft’s base generation is optimized for very large game worlds. In this case random seeds are not recommended.

**7. User Interface**

**7.1 Command Interface**

Speedrun commands will be controlled by admins to help influence various factors of the speedrun such as setting objectives, setting game rules, and modifying speedrun attributes. These commands will be accessible to administrators and will be integrated into the UI. Some commands will be accessible by all users, such as the ability to list all objectives and game rules, and the ability to switch to and from team chat. The UI will be simplistic and intuitive to use and limited in scope. This design choice will allow the game to have a straightforward experience to game setup. The plugin will feature one main command that will include an extensive list of subcommands to be used for administrative-end functions. Subcommand suggestions will be implemented wherever necessary. For ease-of-use, user commands are implemented directly rather than as subcommands of /speedrun.

**Player Commands - ksu.speedrun.user**

Player commands are mainly used for general speedrun information that is necessary for both players and administrators. These commands are outlined below:

* ***/help - alias for admins: /speedrun help***
  + Function: Main help command; outlines all commands that the player has permission to execute.
* ***/objectives - alias for admins: /speedrun objectives***
  + Function: Displays a user-friendly interface that allows for players to see all complete and incomplete objectives, as well key information about the objectives.
* ***/team - alias for admins: /speedrun team***
  + Function: Allows players to select a team as an alternative to the UI.
* ***/spawn***
  + Allows players to teleport to their team’s spawnpoint. Players cannot use this command while in combat.
* ***/scoreboard*** 
  + Players can toggle the scoreboard display on and off during a game.

**Admin Commands - ksu.speedrun.admin**

***/speedrun [arg(s)]***

Admin commands are mainly used for pre-game configuration, and are outlined below.

* ***/speedrun addObjective [objectiveType] [target] <optional flags>***
  + Function: Adds an objective to the objective list for the speedrun.
  + ObjectiveType: There are four possible objective types that can be set:
    - KILL
      * Target: See Appendix (A)
    - ENTER
      * Target: See Appendix (B)
    - OBTAIN
      * Target: See Appendix (C)
    - MINE
      * Target: See Appendix (D)
    - FROMFILE
      * Target: File Name
  + Flags:
    - Weight flag: *-w <integer>*
      * Set the weight of an objective. Weights that are not set will default to a value of 1.
    - Number flag: *-n <integer>*
      * Set the number of events required to complete an objective. Does not apply to the ENTER objective.
* ***/speedrun remObjective [id]***
  + Function: Remove an objective from the objective list for the speedrun.
  + ID: Corresponds to the ID number next to an objective from the command */speedrun objectives*.
* ***/speedrun setSeed [seed]***
  + Function: Sets the seed for the world to be generated for the speedrun. A random seed will be used if this command is not executed.
  + Seed: Corresponds to the seed requested for the speedrun map.
* ***/speedrun getSeed***
  + Function: Returns the seed of the speedrun map.
* ***/speedrun setBorder [borderradius]***
  + Function: Sets the world border radius for the speedrun map. Players cannot go outside the radius of this border.
  + BorderRadius: Radius of the world border, in blocks.
* ***/speedrun getBorder***
  + Function: Returns the world border radius (in blocks) for the speedrun.
* ***/speedrun setTimeLimit [timeLimit]***
  + Function: Sets the time limit for the speedrun. Speedruns will automatically end when the time limit is reached. The team with the most points will be determined as the winner. A time limit of 60 minutes will be set by default.
  + TimeLimit: Time limit of the speedrun, in minutes.
* ***/speedrun getTimeLimit***
  + Function: Returns the time limit (in minutes) for the speedrun.
* ***/speedrun setSpawnRadius [spawnRadius]***
  + Function: Set the spawn radius from (0, 0) for the speedrun. Teams will spawn in groups at this distance from (0, 0). A spawn radius of 300 blocks will be set by default. The spawn radius must be less than the world border radius.
  + SpawnRadius: Spawn radius for the speedrun, in blocks.
* ***/speedrun getSpawnRadius***
  + Function: Returns the spawn radius (in blocks) for the speedrun.
* ***/speedrun setGamerule [gamerule] [boolean]***
  + Function: Set a gamerule for the speedrun. Gamerules will only be active while a speedrun is in progress. All gamerules are set to false by default.
  + GameRule: Includes all gamerules in the default Minecraft namespace, in the exact same corresponding text format.
  + Boolean: True or False
* ***/speedrun setTeamSize [teamsize]***
  + Function: Sets the preferred amount of members each team should have.
  + TeamSize: Size limit for each team, as an integer value.
* ***/speedrun getTeamSize***
  + Function: Returns the size limit for each team.
* ***/speedrun toggleTeams***
  + Function: Toggles teams as enabled or disabled. Speedruns with teams disabled will be every man for himself. Teams will be dismissed and team selectors will be removed from players’ inventory.
* ***/speedrun setPointLimit [pointLimit]***
  + Function: Changes the points required to win the speedrun to the number specified. By default, the point limit is equal to the number of objectives present.
  + PointLimit: Point limit for the speedrun to be completed, as an integer value.
* ***/speedrun getPointLimit***
  + Function: Returns the point limit.
* ***/speedrun resetAttributes***
  + Function: Reset all attributes (objectives, time limit, team size, etc.) back to default.
* ***/speedrun generateWorld***
  + Function: Generates a speedrun map according to the seed. Overrides a previously generated world (if the speedrun has not yet been started). Also locates safe spawn points for teams and notifies admins if no safe spawn locations can be found for specific teams alongside the corresponding team number.
* ***/speedrun getTeamSpawnLocations***
  + Function: Returns a list of all team spawn locations next to a corresponding team ID.
* ***/speedrun setTeamSpawnLocation [teamId]***
  + Function: Overrides the automatic spawn location set for a specific team to whatever location the admin who sends the command is at.
  + TeamId: The number that corresponds to the ID returned alongside a team when the command ***/speedrun getTeamSpawnLocations*** is run.
* ***/speedrun participate***
  + Function: Admins who run this command will be included in the speedrun gamemode.
* ***/speedrun start***
  + Function:
    - If a speedrun world has been generated, this command will start the game.
* ***/speedrun stop***
  + Function: Force a speedrun game to end.

**7.2 Inventory-Based Interface**

The graphical user interface for team selection will be integrated using the Bukkit *Inventory* interface. This GUI will allow players to select which team they would like to join, and is displayed to the user after they run the command */speedrun team* or activate the team selector object that will be placed in their inventory when they join the game. Players may freely switch teams at their discretion prior to a game beginning. The following images are examples of how the inventory-based GUI will be presented to players.

**Team Selector - Right or Left Click to Trigger**

****

**Team Selection GUI**

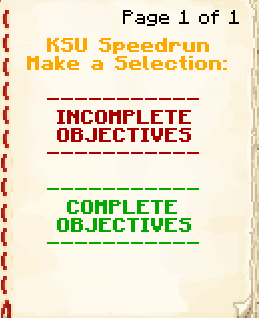
****

**7.3 Book-Based Interface**

The graphical user interface for viewing objectives will be integrated using the Bukkit *Book* interface. The Objective UI gives the player clear information on current objectives and their progress toward completing them.The following images are examples of how the book-based GUI will be presented to players.

***Please view the next page…***

**Main Objective UI Pg. 1 of Incomplete Objectives UI**



**8. Scoring and Progress Tracking**

**8.1 Targets, Objectives and Scoring**

The administrative command interface allows for the assignment of objectives of various types. There are only four main objective types, but counting all specific targets within each objective type, there are a total of 2,728 possible individual objectives that can be assigned to players - almost 100,000 if amount flags are considered. There are millions of ways in which even two objectives can be combined. Because there is no limit to the amount of objectives that can be assigned to a speedrun, the possibilities are limitless and cover the majority of possible achievements a player can accomplish in Minecraft. The command interface for objectives was previously outlined within section (7.1) of this document, however a more general description of objectives can be seen below:

* **Kill Objectives:** These objectives are achieved when a player or team kills a specified targeted entity in game. Only living entities can be targeted. Please view Appendix (A) for specific entity types. Kill objectives can be flagged with an amount, where the objective is not achieved until a certain number of specific entities are killed.
* **Enter Objectives:** These objectives are achieved upon entering a given location in the game. Locations themselves can be of three different types: Biome, Structure, or Portal. The portal type is more specific than dimension type because it describes the dimension TO as well as the dimension FROM, and can be used to detect portal events where dimensions do not change, such as the portal from the main End island to the outer islands. Please view Appendix (B) for specific enter objective types.
* **Obtain Objectives:** These objectives are achieved when a player obtains a specific item in the game. Obtain objectives can be flagged with an amount, where the objective is not achieved until a certain number of specific items are obtained. As long as the number of items each team member has totals to the amount required for the objective, it is completed. A single player does not have to hold all of the specific items. Please view Appendix (C) for more information.
* **Mine Objectives:** These objectives are achieved when a player or team mines a specific block in the game. This may seem redundant when presented after obtain objective, but it does have its own specific use. Firstly, this objective is triggered at a different time than the obtain objective: when a block is mined rather than when an item is picked up. Secondly, this event covers certain blocks that may not drop items when mined, such as monster spawners. Mine objectives can also be flagged with an amount, where the objective is not achieved until a certain number of specific blocks are obtained. This does not consider the fact that players can mine a block, place it down, and mine it again. Numbered mine objectives should only be used in a situation where mining a block does not drop the specific block (e.g., mine 5 monster spawners). Please see Appendix (D) for more information.

These objectives are directly linked to the scoring system of the game: one objective is equivalent to one point towards a team score by default. The weight (points awarded) of an objective will be adjustable by server administrators. There will be two different speedrun scoring scenarios: unweighted speedruns and weighted speedruns. Unweighted speedruns will not incorporate modified weight scores and will have all objectives weighted to the default value of one. Weighted speedruns have objective scores determined by admins, and may have a modified final points requirement. Below are two examples of possible speedrun situations:

**Normal (unweighted) speedrun:**

* Objective 1: Kill Ender Dragon (1 point - default)
* Objective 2: Kill Wither (1 point - default)
* Final required points: 2 - default

In the example above, a team wins when they complete all objectives.

**Weighted speedrun:**

* Objective 1: Get 32 logs (1 point - default)
* Objective 2: Get 128 cobblestone (1 point - default)
* Objective 3: Get 16 diamonds (2 points – set by admin)
* Objective 4: Obtain a mending book (4 points – set by admin)
* Final required points: 4 – set by admin

In the example above, a team may win by either completing objectives 1-3, or completing *just* objective 4.

**8.2 Objective Completion Tracking**

Event handlers and synchronous tasks will be used to track event completion. When an event is completed, all players in the game will be notified by a chat message, and the leaderboard will be updated. Players may check the list of objectives at any time using the command */objectives*.

**8.3 HUD (Heads-Up Display)**

The HUD provides real-time information to players during the speedrun. This helps them stay informed of their team’s status and the game’s progression.

* **Scoreboard**: The scoreboard displays key information, including:
  + **Remaining Time**: Shows the countdown timer for the speedrun, keeping players aware of how much time is left.
  + **Team Points**: Displays the points each team has accumulated, allowing players to track their standing compared to other teams.
  + **Objectives Progress**: Shows the points required to win and updates as teams complete objectives.
* **HUD Notifications**: Important game events, such as the start or end of the speedrun, are displayed as title messages on the screen. When a game ends, a title display and sound play to notify all players.
* **Team Display**: During the game, the HUD shows which team the player belongs to, along with the names of teammates and their status (e.g., alive, completed objectives).

**9. Configuration, Customization, and Content Addition**

**9.1 Config.yml Settings**

The plugin implements a *config.yml* file to allow for further modifications aside from the in-game command interface. Within the configuration file, administrators can customize all messages that non-administrative players may see using MiniMessage format. This includes announcement messages and chat formatting. Other attributes that can be modified within the config.yml include title configuration for game over messages, scoreboard configuration, which defines how the scoreboard is displayed, team definitions and UI configuration, game rule configuration (the same to that of vanilla Minecraft’s game rules), and structure configuration, where the average radius, height, and y-coordinate of certain structures is defined.

**9.2 Adding Objectives From File**

The plugin will allow admins to pull a list of objectives from a .txt file to be added to the objective list all at once. This is especially useful when a speedruns needs to be made from lots of smaller weight objectives and players are intended to complete as many as possible within the time limit.

This feature will also allow admins to keep lists they have previously created and later reload them for a similar experience in a later speedrun tournament.

**9.3 Future-Proofing**

The main importance in the *config.yml* lies in its implementation for future-proofing. With maintainability and scalability being prioritized in this project, there is a large emphasis in ensuring that this plugin is able to support future versions of Minecraft. All entities, structures, blocks, and items are dynamically implemented in the plugin as much as possible with the resources given. To support any future dimension or portal types in Minecraft, new *Portal* objects can be declared in the configuration file by providing the dimension TO and the dimension FROM.

The biggest issue presented to our team is resolving a way to verify maps after the addition of new dimensions, structures, and biomes and all corresponding blocks, entities, and items. The plugin will automatically support these new items: new entities will be supported in the kill event; new blocks will be supported in the mine event; new items will be supported in the obtain event; new structures and biomes will be supported in the enter event, and new portals can be added through the configuration. The plugin will not, however, know by default where these objects are located in the game and thus will not be able to verify maps by itself.

Another issue presented was in locating structures. Specifically, the Bukkit library provides for a very useful utility called *StructureSearchResult*. This utility allows for the nearest location of a structure to be returned by providing for a start location, a *Structure*, and a radius. The location returned from this interface does not, unfortunately, return the Y-coordinate for the structure located. The best viable workaround to this is setting the Y value for each structure to be based on where they are located on average. For structures that generate in the world surface, this is not a problem as the Y-coordinate can be set at the location of the highest block in the world (the ground). For underground structures, however, this is more of a problem as the Y-coordinate may vary more significantly for these structures. This is the only current problem our team is still facing at the time of writing this document, and the only proposed solution is looping through all nearby blocks from ground level to bedrock until a block related to the structure is located.

Nonetheless, these average Y-coordinate values cannot be hard-coded into the system, as that does not support the addition of future structures. Thus, the average Y-coordinate of all structures can be set in the *config.yml*, and this supports future structures as well. Some structures are much larger than others, therefore the detection radius (the point at which a player “enters” the structure) varies from structure to structure. This value can also be modified for every structure in the game through the configuration file.

**10. Libraries, Frameworks, and Deployment**

**10.1 Dependencies**

The speedrun game mode will not be a standalone game and will run through a Minecraft server, essentially acting as its “engine”. Servers using this plugin must be running on PaperMC version 1.21 or later. This game mode will not be supported by Bukkit or Spigot Minecraft servers, as it will implement utilities only available through the PaperMC API. The game mode will be playable through a Minecraft Java Edition client without any additional client-side modifications. Additionally, this game mode can be played using a Minecraft Bedrock Edition client without any additional client-side modifications, however it will be dependent on the server running GeyserMC. Minecraft Bedrock Edition players will not be able to join the server if the GeyserMC plugin is not installed and set up.

A permission management plugin is strongly recommended to be used alongside this plugin, however this is not a true dependency: because there are only two defined permission nodes in this game mode, they could be assigned using PaperMC’s default *permissions.yml* file, with operators having the permission *ksu.speedrun.admin* and regular users having the permission *ksu.speedrun.user*.

This game mode will not handle the pre-game lobby. Assuming the server administrators want to have players wait in a lobby prior to the start of the game mode, attributes such as login spawn point, build permissions, disabled PvP, etc., must be handled by other plugins or by functionalities included in vanilla Minecraft. Aside from this, this game mode will be able to function autonomously without dependencies on any other Minecraft plugins or external utilities.

**10.2 Command Framework**

Commands will be handled using PaperMC’s built-in command API. Specifically, they will be implemented using Paper API’s *BasicCommand* interface, which, aside from handling command execution, also handles command suggestions and permissions. The base */speedrun* command permission will be implemented using this interface. Subcommand permissions will be implemented using Bukkit’s *Permission* class, which is included within the Paper API. Commands will be registered when the plugin enables, allowing for an interface throughout the lifecycle of the plugin.

**10.3 Event Framework**

Similar to the command framework, most events will be handled using Bukkit’s built-in event API. Specifically, they will be implemented using Bukkit’s *Listener* interface. This interface is included within the Paper API, as the Paper API is a framework that optimizes the Spigot API, which itself is a framework that optimizes Bukkit. All events will be implemented in this way aside from the *PlayerMoveEvent*, which will be implemented using a synchronous task that will repeat every two seconds. The rationale behind this is that Bukkit’s *PlayerMoveEvent* is called every time a player makes a slight movement, which can be up to 20 times per second, for every player who is moving on the server. Because there is no immediate urgency for a player move event to trigger at the exact same tick that the player is moving, this will not have a noticeable negative effect on the response time of the server; it will be 80 times less intensive than Bukkit’s *PlayerMoveEvent*. This repeating task must be synchronous because it directly interacts with world and player data. Any logic within this task that does not interact with world and player data will be handled off of the main thread so long as it does not disrupt the task’s integrity.

**11. Development**

This project is planned to be able to support all future versions of Minecraft within its scope using the *config.yml* file. In the configuration file there will be examples set by default so that administrators will understand how to map links, set average Y-coordinates, and structure radii. While extensive configuration may be required, the software itself should not need to be updated.

In the event that the software itself needs an update, the Software Design Specification (SDS) of this project will provide the best assistance in understanding the structure of the plugin, the subcomponents, and interactions. The SDS can be found and downloaded from the main branch of this project’s GitHub. The GitHub itself will also prove likely the most valuable source in keeping this project alive. The project’s GitHub also contains a Wiki for the project, which may contain some additional information that is not present in this document. The link to the project’s GitHub can be found [here](https://github.com/KSU-Team-speedrun/KSU-MC-Speedrun).

**12. Suggested Additional Features**

**12.1 Custom world generation**

The customization of the world generation in Minecraft would allow for even more interesting gameplay experiences. Custom biomes can add an interesting twist to the game, and provide a sense of mystery and interest for veteran players. This addition may be a good way to rebalance the players' skills as players with more experience playing with one kind of world generation may find the new world generation to be more difficult than the one they are more familiar with, as well as refreshen the player experience.

**12.2 Custom Music**

To add to the excitement of playing in a speedrun competition or just to add to the ambiance of minecraft itself during the speedrun, the addition of customized music with specific start and stop cues, tracked by systems already in place within the plugin, would be impactful. If the KSU Esports administration chooses to do any sort of live videography of the Minecraft speedrun tournaments, this addition would undoubtedly be a huge improvement to the viewing experience with little to no input from the admins.

**12.3 Automatic Map Verification**

This would help eliminate much of the setup time needed to run tournaments. It would also allow the KSU Esports team to put a speedrun semi-permanently as a game mode that runs automatically without much intervention.

**13. Appendix**

**A. Living Entity Types (as of Minecraft 1.21):**

ELDER\_GUARDIAN, WITHER\_SKELETON, STRAY, HUSK, ZOMBIE\_VILLAGER, SKELETON\_HORSE, ZOMBIE\_HORSE, ARMOR\_STAND, DONKEY, MULE, EVOKER, VEX, VINDICATOR, ILLUSIONER, CREEPER, SKELETON, SPIDER, GIANT, ZOMBIE, SLIME, GHAST, ZOMBIFIED\_PIGLIN, ENDERMAN, CAVE\_SPIDER, SILVERFISH, BLAZE, MAGMA\_CUBE, ENDER\_DRAGON, WITHER, BAT, WITCH, ENDERMITE, GUARDIAN, SHULKER, PIG, SHEEP, COW, CHICKEN, SQUID, WOLF, MOOSHROOM, SNOW\_GOLEM, OCELOT, IRON\_GOLEM, HORSE, RABBIT, POLAR\_BEAR, LLAMA, PARROT, VILLAGER, TURTLE, PHANTOM, COD, SALMON, PUFFERFISH, TROPICAL\_FISH, DROWNED, DOLPHIN, CAT, PANDA, PILLAGER, RAVAGER, TRADER\_LLAMA, WANDERING\_TRADER, FOX, BEE, HOGLIN, PIGLIN, STRIDER, ZOGLIN, PIGLIN\_BRUTE, AXOLOTL, GLOW\_SQUID, GOAT, ALLAY, FROG, TADPOLE, WARDEN, CAMEL, SNIFFER, BREEZE, ARMADILLO, BOGGED, PLAYER

**B. Enter Objective Types (as of Minecraft 1.21):**

OCEAN, PLAINS, DESERT, WINDSWEPT\_HILLS, FOREST, TAIGA, SWAMP, MANGROVE\_SWAMP, RIVER, NETHER\_WASTES, THE\_END, FROZEN\_OCEAN, FROZEN\_RIVER, SNOWY\_PLAINS, MUSHROOM\_FIELDS, BEACH, JUNGLE, SPARSE\_JUNGLE, DEEP\_OCEAN, STONY\_SHORE, SNOWY\_BEACH, BIRCH\_FOREST, DARK\_FOREST, SNOWY\_TAIGA, OLD\_GROWTH\_PINE\_TAIGA, WINDSWEPT\_FOREST, SAVANNA, SAVANNA\_PLATEAU, BADLANDS, WOODED\_BADLANDS, SMALL\_END\_ISLANDS, END\_MIDLANDS, END\_HIGHLANDS, END\_BARRENS, WARM\_OCEAN, LUKEWARM\_OCEAN, COLD\_OCEAN, DEEP\_LUKEWARM\_OCEAN, DEEP\_COLD\_OCEAN, DEEP\_FROZEN\_OCEAN, THE\_VOID, SUNFLOWER\_PLAINS, WINDSWEPT\_GRAVELLY\_HILLS, FLOWER\_FOREST, ICE\_SPIKES, OLD\_GROWTH\_BIRCH\_FOREST, OLD\_GROWTH\_SPRUCE\_TAIGA, WINDSWEPT\_SAVANNA, ERODED\_BADLANDS, BAMBOO\_JUNGLE, SOUL\_SAND\_VALLEY, CRIMSON\_FOREST, WARPED\_FOREST, BASALT\_DELTAS, DRIPSTONE\_CAVES, LUSH\_CAVES, DEEP\_DARK, MEADOW, GROVE, SNOWY\_SLOPES, FROZEN\_PEAKS, JAGGED\_PEAKS, STONY\_PEAKS, CHERRY\_GROVE, CUSTOM, PILLAGER\_OUTPOST, MINESHAFT, MINESHAFT\_MESA, MANSION, JUNGLE\_PYRAMID, DESERT\_PYRAMID, IGLOO, SHIPWRECK, SHIPWRECK\_BEACHED, SWAMP\_HUT, STRONGHOLD, MONUMENT, OCEAN\_RUIN\_COLD, OCEAN\_RUIN\_WARM, FORTRESS, NETHER\_FOSSIL, END\_CITY, BURIED\_TREASURE, BASTION\_REMNANT, VILLAGE\_PLAINS, VILLAGE\_DESERT, VILLAGE\_SAVANNA, VILLAGE\_SNOWY, VILLAGE\_TAIGA, RUINED\_PORTAL, RUINED\_PORTAL\_DESERT, RUINED\_PORTAL\_JUNGLE, RUINED\_PORTAL\_SWAMP, RUINED\_PORTAL\_MOUNTAIN, RUINED\_PORTAL\_OCEAN, RUINED\_PORTAL\_NETHER, ANCIENT\_CITY, TRAIL\_RUINS, TRIAL\_CHAMBERS, WORLD\_TO\_NETHER, NETHER\_TO\_WORLD, WORLD\_TO\_END, END\_TO\_END, END\_TO\_WORLD

**C. Obtain Objective Types (as of Minecraft 1.21):**

Omitted for brevity. This list contains every possible obtainable item in Minecraft (1,480). Items are named according to their Paper API Material (org.bukkit.Material) name value.

**D.** **Mine Objective Types (as of Minecraft 1.21):**

Omitted for brevity. This list contains every possible obtainable item in Minecraft that is also a placeable block (1,060). Items are named according to their Paper API Material (org.bukkit.Material) name value.

**E. GameRules (as of Minecraft 1.21):**

announceAdvancements, blockExplosionDropDecay, commandBlocksEnabled, commandBlockOutput, commandModificationBlockLimit, disableElytraMovementCheck, disableRaids, doDaylightCycle, doEntityDrops, doFireTick, doInsomnia, doImmediateRespawn, doLimitedCrafting, doMobLoot, doMobSpawning, doPatrolSpawning, doTileDrops, doTraderSpawning, doVinesSpread, doWeatherCycle, doWardenSpawning, drowningDamage, enderPearlsVanishOnDeath, fallDamage, fireDamage, forgiveDeadPlayers, freezeDamage, globalSoundEvents, functionCommandLimit, keepInventory, lavaSourceConversion, logAdminCommands, maxCommandChainLength, maxCommandForkCount, maxEntityCramming, minecartMaxSpeed, mobExplosionDropDecay, mobGriefing, naturalRegeneration, playersNetherPortalCreativeDelay, playersNetherPortalDefaultDelay, playersSleepingPercentage, projectilesCanBreakBlocks, pvp, randomTickSpeed, recipesUnlock, reducedDebugInfo, respawnBlocksExplode, sendCommandFeedback, showBorderEffect, showCoordinates, showDeathMessages, showTags, snowAccumulationHeight, spawnChunkRadius, spawnRadius, spectatorsGenerateChunks, tntExplodes, tntExplosionDropDecay, universalAnger, waterSourceConversion