

03-T2 Minecraft

Software Requirement Specification

CS 4850 - Section 01 – Spring 2026

February 8, 2026

 Matthew Pflieger Team Leader Developer	 Andrew Burgos Developer	 John McDowell Developer	 Hezekiah Holloman Documentation	 Chase Holt Documentation
---	---	---	--	--

Team Members:

Name	Role
Matthew Pflieger (Team leader)	Developer
Chase Holt	Documentation
Andrew Burgos	Developer
Hezekiah Holloman	Documentation
John McDowell	Developer
Sharon Perry	Project Owner or Advisor

Table of Contents

1.0 Introduction.....	3
1.1 Overview.....	3
1.2 Project Goals.....	3
1.3 Definitions and Acronyms.....	3
1.4 Assumptions.....	3
2.0 Design Constraints.....	3
2.1 Environment.....	4
2.2 User Characteristics.....	4
3.0 Functional Requirements.....	4
3.1 Game State & Lobby Management.....	4
3.2 Turf Game Mode Mechanics.....	5
3.3 Combat Mechanics.....	5
3.4 Team Balancing.....	5
4.0 Non-Functional Requirements.....	5
4.1 Security.....	6
4.2 Capacity.....	6
4.3 Usability.....	6
4.4 Other.....	6

1.0 Introduction

The SRS document specifies the requirements for a functional Minecraft turf wars minigame which will be implemented as an PaperMC plugin on the KSU esports minecraft server. The plugin will provide match creation based on player count, team assignment, territory control, game phases, and clean up between matches.

1.1 Overview

The turf wars minigame will be a plugin that is designed for the KSU esports minecraft server. The plugin will allow for players to join matches, be assigned to a team, and switch between a build phase and capture phase to gain territory in an arena. The arena size will be based on the amount of players competing.

1.2 Project Goals

A functional turf wars minigame mode that can be played in varying team sizes, assign teams that are balanced, and keep track of turf claims.

1.3 Definitions and Acronyms

PaperMC: a Spigot server implementation.

Plugin: a server-side add-on to extend minecraft.

1.4 Assumptions

The server will run a supported PaperMC version. The players will have a stable network connection to play on the server.

2.0 Design Constraints

The functionality of the gamemode will be limited by the Paper API and the way in which it "sandboxes" plugins and contains their access. Our implementation will be entirely contained within a Paper API plugin, which manages our minigame's I/O access and connection to the rest of the KSU Minecraft server.

2.1 Environment

The end product will have to be a portable, sandboxed Java module that does not depend strongly on outside information or make external I/O requests.

2.2 User Characteristics

Users will connect to a lobby subserver, from which they will be routed into instances of the arena world to play the game.

3.0 Functional Requirements

3.1 Game State & Lobby Management

- **Arena:** The game will take place in a defined world(s) where any changes do not affect the main server lobby, or any of the other game modes already present on the server.
- **Queue System:** Players will be able to join a lobby via a portal system that will transport them to a "queue world." Inside of the queue world players will wait for enough players (4-8 players) to start the game. The queue world will also have a parkour mini game implementation to give players an activity while they wait.
- **Auto-Reset:** Upon completion of the game the arena(s) must automatically revert to its original state (removing all player-placed blocks and resetting the turf line). Also, it must return the players back to the server lobby.

3.2 Turf Game Mode Mechanics

- **“Turf” Tracking:** The system must track the divide line between the two teams.
- **Kill-Based Advancement:** When a player kills an opponent, the divide line will move X blocks into the enemy territory.
- **Build Restrictions:** Players must be restricted to placing blocks only in their teams “turf,” and not being able to place blocks on the enemy team’s “turf.”
- **Win Condition:** The game ends when one of the team’s territory reaches 100% of the arena area, or whichever team has the most area controlled when the timer runs out.

3.3 Combat Mechanics

- **One-Shot Kills:** The system should register any arrow hit as an instant kill and begin respawning the player.
- **Respawn Mechanics:** The respawn mechanics will include a dynamic timer that is less for smaller games (2v2, 4v4, etc.) and greater for larger games (8v8, 16v16, etc.).
- **Materials and Ammo:** Each player should spawn in with 16 blocks and 16 arrows. During a predetermined interval of time (e.g. every 15 seconds) each player will receive an additional block and arrow.
- **Arrow Retrieval:** The system should automatically clear arrows from the ground to prevent lag.

3.4 Team Balancing

- **Auto-Balancing:** The system must distribute players into two teams Gold Team and Black Team ensuring even numbers at the start of the match.
- **Friend Queuing:** The system should prioritize two or more players queuing together being placed on the same team.
- **Spawn Protection:** Players should be invincible for 2-3 seconds after respawning to prevent “spawn camping.”

4.0 Non-Functional Requirements

4.1 Security

- All players must not be able to use cheats or commands for gameplay advantages
- Players must not be able to glitch out of designated play area
- Only the Admin must have power to start, stop, or reset the game

4.2 Capacity

- The game must be able to hold a capacity of 32 players
- The game must support all entities without lag
- The game must support player action responsibly at an average speed of 100 ms

4.3 Usability

- Players must understand how to play game by start time
- Game must display time players have remaining in rounds
- Game must display win/lost result on screen
- Game must display the size of the lobby/teams

4.4 Other

- The game must get through playable rounds even when errors accrue
- The game must not lose player data when they disconnect from game
- The game must not crash when player disconnects in the middle of game round