

03-T2 Minecraft

Project Development Document

CS 4850 - Section 01 – Spring 2026

March 22, 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. Implementation Detail Narrative.....	2
1.0 System Integration & Workflow Overview.....	2
1.1 Component Name.....	2
1.2 Component Name.....	2
2. Database and External Connections.....	3
3. Project Setup and Environment Configuration.....	3
3.1 Prerequisites.....	3
3.2 Installation Steps:.....	3
For Developers.....	3
For Users.....	3
3.3 Verification.....	3
4. Development Conclusion or Summary.....	4

1. Implementation Detail Narrative

1.0 System Integration & Workflow Overview

- The gamemode comes in the form of a Paper API plugin - a portable executable that is loaded into modified version of the Minecraft server software at runtime. Internally, various classes and containers organize the lifecycle of loading/unloading the game worlds, as well as connecting and redistributing players.
- As the plugin loads in at server startup, it calls the TurfWars class' onEnable() method, which runs startup operations, and then creates the ArenaManager and the GameManager. These are singleton objects that handle the loading of world files and the logical lifecycle of matches respectively. At present, the plugin creates two empty games at startup, and then waits for a player to use the /join [name] command to join one of them, at which point they are teleported into that respective world. When the game ends or the server shuts down, that game's copy of the world is unloaded, and all players are teleported back to the lobby world

1.1 Arena Manager

- **Conceptual Overview:** A singleton class that handles all I/O operations for the server. It is instantiated on plugin startup, and then is called upon whenever a new world needs to be created. To do so, it copies the template world into a new directory, and then loads that into a new World object (provided by the Paper API).
- **Development Process:** Began by implementing these functions in the "main" TurfWars class, but refactored to be its own object for separation of concerns.

1.2 Game Manager

- **Conceptual Overview:** Another singleton class that stores a HashMap of Game objects, and has methods to create and delete matches of the turf war gamemode. The Game class itself talks to ArenaManager to request file operations when needed.
- **Development Process:** Was originally part of the monolithic ArenaManager class, but split off again for more separation of concerns.

2. Database and External Connections

- The plugin does not connect to any external or database or web API, rather handling things entirely through local file storage and code. The Paper API provides methods for accessing the relevant directories assigned to the plugin's container, while files and folders themselves are moved using the Java File class and its built-in methods. Everything is contained within the plugin's data folder, which is located at the minecraft server's root/plugins/TurfWars

3. Project Setup and Environment Configuration

This section will include the steps for setting up the project in a developmental environment, and for end users how to set up the project to play the minigame.

3.1 Prerequisites

- Minecraft version 1.20.1-1.21.11
- Java Development Kit (JDK) version 17 or higher
- Java version 21
- Integrated Development Environment (IDE) capable of running Java
- Git
- Gradle

3.2 Installation Steps:

For Developers

1. Clone the repository (<https://github.com/minecraft-turf-wars/minecraft-turf-wars>)
2. Install dependencies
3. Configure environment variables
4. Insert the command `./gradlew build` into the command terminal

For Users

1. Install an instance of Minecraft version 1.20.1-1.21.11
2. Connect to the Kennesaw State University Esports Minecraft Server (mc.esports.kennesaw.edu)
3. Walk through corresponding portal to be connected to pre-game lobby
4. Have Fun!

3.3 Verification

Developers can confirm the project built successfully via the terminal response showing a successful gradle build, or confirming the .bat file was formed. **Users** can verify a successful build of the project if they are loaded into a pregame lobby queue and eventually placed into a populated game.

4. Development Conclusion or Summary

- The current build assumes that the minigame is running on its own independent server, using the server's main world as the lobby. It accepts commands to create games, teleport players to them, and shut them down. The business logic dictating the actual game mechanics is yet to be implemented.