Capstone Project Report

Minecraft Team 3

Kennesaw State University

Ryan Shah - Reece Clark - Alex Millerioux - Tristan Lance - Liliana Pantoja

IT 4983: IT Capstone Project

Website: https://sites.google.com/view/minecraftteam3/home

April 21, 2024

Executive Summary

With the rise of the popular video game, Minecraft, came the ability to expand and customize the game via plug-ins and mods to keep it fresh after 13 years of dominance in the video game industry. The game also gives you the ability to host your own server to play with friends, random users from around the world, etc. With familiarity playing and enjoying this widely acclaimed game over the years, our team has embarked on developing a new ticketing and reporting plug-in for the official Kennesaw State University (KSU) Minecraft server. Our project aims to enhance the overall gaming experience by introducing a new system for managing in-game issues. By leveraging modern development tools/technologies, we strive to create a seamless and intuitive solution catering to the players and moderation team within the server. Our development process utilizes IntelliJ IDEA, a renowned integrated development environment that allows our team to work efficiently, collaborate on code easily, and deliver high-quality code. To ensure effective version control and collaboration, we have chosen GitHub as our repository, leveraging its capabilities for code management/tracking changes. Furthermore, we have selected MariaDB, a highly scalable open-source database management system to handle the storage and retrieval of data related to player reports, tickets, and associated metadata. MariaDB's compatibility, good security features, and advanced replication functions make it an ideal choice for managing critical game data. Through this project, we aim to enhance the gaming experience for the KSU Minecraft community while demonstrating our proficiency in leveraging modern tools/technologies. By combining our expertise in software development/QA testing with an understanding of the gaming industry's needs, we strive to deliver a solution that sets the new standard for in-game support within the KSU Minecraft server.

Table of Contents

Background

KSU Esports has an official KSU Minecraft server with over 450 members. They mainly use free open-source software to run the server. Some examples of software they currently use are moderation and event plug-ins, Discord bots, etc. Currently, if they have an issue, bug, or suggestion for one of their pieces of software, they must contact the original developer to see if they are interested in working on the software, and if not, they must work on the application themselves. Their current software also isn't always a perfect fit for the server/community; it may be missing features that they need. They also use Minecraft version Paper for their server as well.

The primary goal of this project is to create an in-game ticketing and reporting system that Minecraft players can easily use to submit any moderation requests they may have. It will feature an easy-to-use user interface for initiating and tracking tickets. The system needs seamless integration with the official KSU Minecraft Discord server as another access mechanism and to support notifications. As this tooling involves operation at the server level, it should prioritize resource efficiency and minimize performance impact through thoughtful architectural decisions. Code organization and modularity are also important to build a maintainable, adaptable ticketing platform for preserving long-term utility and the ability to extend with new capabilities in the future. By facilitating player reporting, the ticketing system aims to foster accountability within the game community.

Minecraft, one of the most popular video games of all-time, allows players to build and explore virtual worlds (Li, 2023). One of the key features that contribute to Minecraft’s popularity is its extensibility through plugins. Plugins are to Minecraft what extensions are to browsers. They allow third-party developers to write additional code and plug it into the server, extending the game’s functionality (Whalen, 2022). Minecraft plugins can range from simple modifications like making milk restore hunger to adding entirely new and original game modes or items (Adding Plugins). They do not modify the game as in the case of Minecraft mods and are more limited in what they can do. However, this also means they only need to be installed on the server’s side.

The Paper plugin built on Spigot enhances server performance significantly and provides advanced features/API. It incorporates many enhancements for better efficiency (PaperMC). Paper plugin also incorporates asynchronous chunk loading and major enhancements to elements like light engines, hoppers, and entities. It augments Bukkit/Spigot APIs, offering developers expanded functionalities. It’s backed by a dynamic community of developers and administrators, with immediate support for any issues.

Project Outcomes and Achievement Summary

The project's main goal was to develop a functional ticketing plugin for the KSU esports Minecraft server that would be more functional than the plugin they were currently using. To that extent the plugin developed succeeds as it is more customizable, configurable, and a better fit for the server. For the specific project goals, the plugin achieves the following:

- Supports the latest version of Paper: The plugin was built and tested on the current Paper release and supports the latest version of Minecraft, as well as future versions as it does not use any raw Minecraft server API calls.
- Chat commands: The plugin has chat commands covering all ways it is meant to be used, allowing players to create, update, and close tickets, as well as commands for players with permission to view tickets, teleport to where the tickets were created, and edit permissions for users.

- Editable permissions for commands: The plugin config allows for the setting of permission groups for players to have access to different commands depending on what group they are in within the plugin.
- 100% of in-game/console messages are configurable through a config file and support MiniMessage format: The plugin has a config file that allows all messages to be altered as well as certain parts of their formatting using MiniMessage.
- Store all ticket information to SQL-like database: All tickets created are stored in a database that can be configured within a config file to choose between a few different database options.
- Handle most logic off the main thread: All commands run by players are executed Asynchronously for better efficiency.
- Stores player information upon ticket creation: When a ticket is created the ticket is given an ID number, the player's username and UUID is stored, the location the ticket is created is stored, the ticket creation time is stored, and a description of the ticket is stored in the database configured in the config file.
- Project can interface with a Discord bot's API: All tickets created by a player have their information sent and posted using the Discord API Webhooks into a configurable chat.
- Project uploaded to GitHub: The files are publicly available on a GitHub repository.

Our primary goal when tackling the project was to create an easy to use, configurable, and scalable plugin that the KSU esports staff could use as an improvement on their current system that didn't quite fit their needs. The keyword there is configurable. Throughout our work on the project, we attempted to use different database hosts that might have been good for the KSU esports Minecraft server. From MariaDB to MySQL to SQLite, various testing was done on all three databases at points of development. MariaDB seemed to be the best option due to it being more scalable and configurable as it has more storage engines, more supported connections, and generally faster speed (Ahmed, 2022). As our goal was making the plugin highly configurable, the end-product used our testing of all three databases to be able to implement the ability for the plugin to use any of the three with the change of a few values in a config file. Similarly, the config file can change text displayed when a user inputs a command and the colors of the text allowing sponsors to easily change important elements of the plugin as they see fit. We were able to implement these changes using features of MiniMessage, a framework allowing highly customizable text UI in Minecraft.

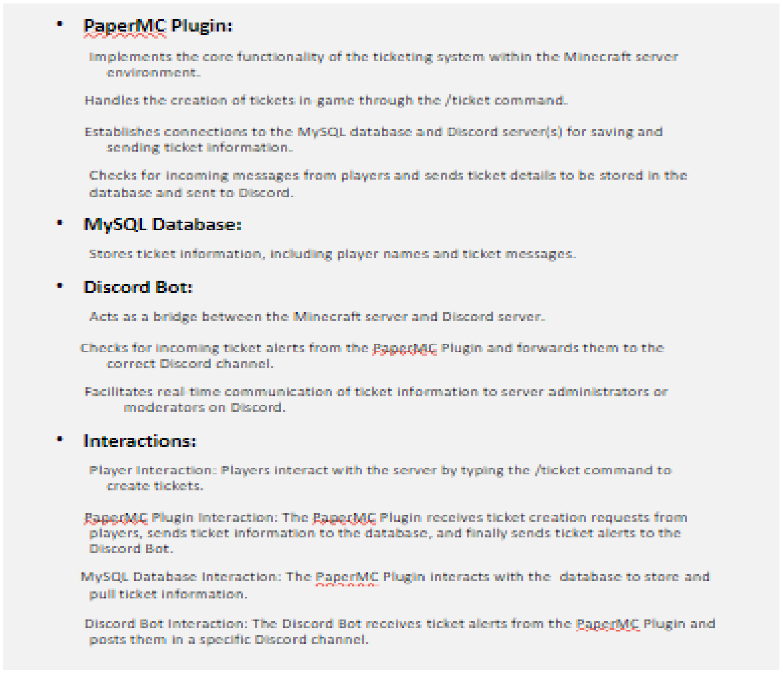
Project Planning and Management Summary

This project consists of three different milestones. The first deliverable was drafting the architecture for the Minecraft paper plug-in. Some of the things that we came up with initially were:

- Implements the core functionality of the ticketing system within the Minecraft server environment.
- Handles the creation of tickets in-game through the /ticket command.
- Establishes connections to the MySQL database and Discord server(s) for saving and sending ticket information.
- Checks for incoming messages from players and sends ticket details to be stored in the database and sent to Discord.

The second deliverable was setting up our testing server on our own local computers. The project sponsor gave us directions on this and pointed out resources to help us accomplish this deliverable in this milestone. The guide kind of explained that if you prefer to host a Minecraft server without frequently interacting with the command line, Windows is likely the ideal operating system choice, either on your personal computer or a spare machine at home. The initial step is to ensure that you have installed the appropriate Java JDK version compatible with your desired Minecraft version. The current Minecraft release (1.20.1) utilizes Java 17, which is the version we will be installing in this tutorial. With Java properly installed, the subsequent steps will guide you through downloading and configuring the local files, as well as ensuring that your Windows firewall won't pose any issues during the setup process (Self Host). This milestone we got hammered with feedback from the professor and our sponsor, and it was a wake-up call for the group to put more effort into the project. The feedback we received was "After reviewing all of the milestone presentations again for the four groups I sponsor, I wanted to reach out with some concerns about team 3. I am a little bit concerned with their level of progress. I do not want to see them fail." Also, some key takeaways we had in this milestone were that regular communication was key and due to differing schedules, getting all the members in one meeting was challenging at times.

Below are the completed deliverables in milestone 1. The first figure shows the full architecture draft we made for the add-on. The second figure shows that the test server was set up in Minecraft successfully.



- **PaperMC Plugin:**

  Implements the core functionality of the ticketing system within the Minecraft server environment.

  Handles the creation of tickets in-game through the /ticket command.

  Establishes connections to the MySQL database and Discord server(s) for saving and sending ticket information.

  Checks for incoming messages from players and sends ticket details to be stored in the database and sent to Discord.

- **MySQL Database:**

  Stores ticket information, including player names and ticket messages.

- **Discord Bot:**

  Acts as a bridge between the Minecraft server and Discord server.

  Checks for incoming ticket alerts from the PaperMC Plugin and forwards them to the correct Discord channel.

  Facilitates real-time communication of ticket information to server administrators or moderators on Discord.

- **Interactions:**

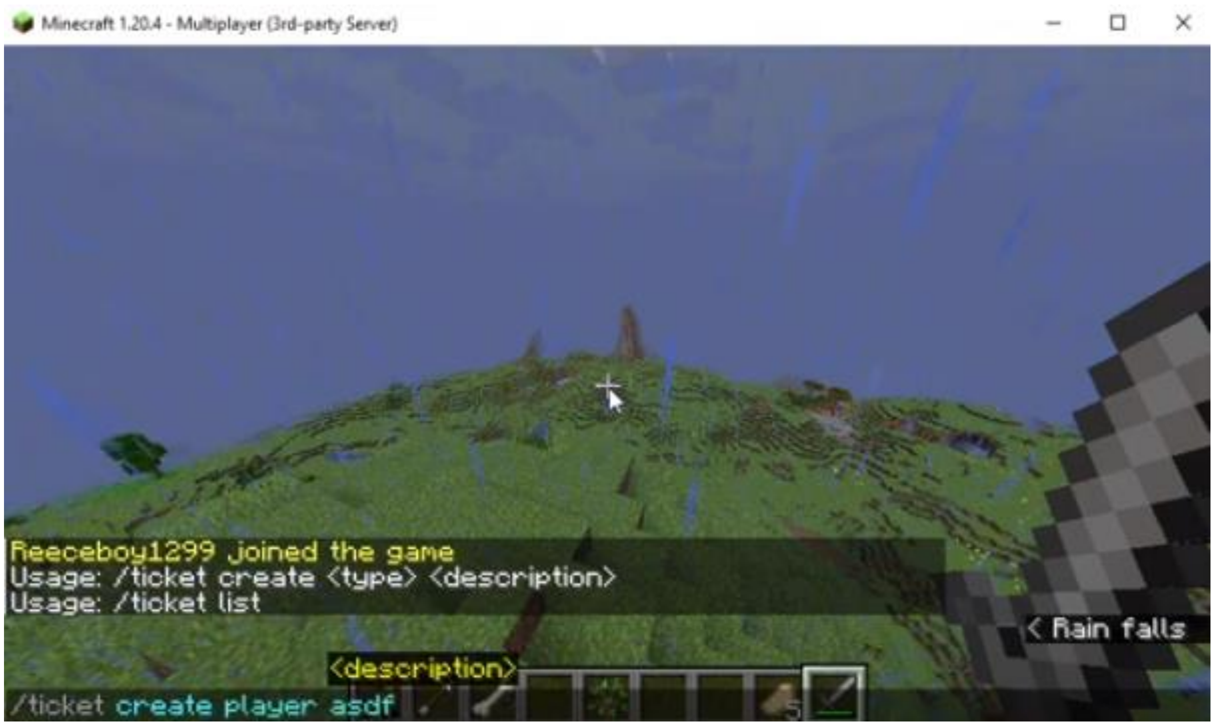
  Player Interaction: Players interact with the server by typing the /ticket command to create tickets.

  PaperMC Plugin Interaction: The PaperMC Plugin receives ticket creation requests from players, sends ticket information to the database, and finally sends ticket alerts to the Discord Bot.

  MySQL Database Interaction: The PaperMC Plugin interacts with the database to store and pull ticket information.

  Discord Bot Interaction: The Discord Bot receives ticket alerts from the PaperMC Plugin and posts them in a specific Discord channel.

Fig 1: Our Minecraft Paper Add-On Architecture Draft



```
Reeceboy1299 joined the game
Usage: /ticket create <type> <description>
Usage: /ticket list
/ticket create player asdf
```

Fig. 2 – Test Server Set-up

For milestone two the deliverables were coding and implementation of chat commands for the plug-in and setting up the plug-in to connect with a database. The first thing we had to do was establish what IDE (Integrated Development Environment) we were going to use for the plug-in. The sponsor requested us to use IntelliJ IDEA which is a powerful IDE tailored specifically for Java and Kotlin programming languages. Its primary goal is to boost developer productivity by automating mundane and repetitive tasks. With features like intelligent code completion, static code analysis, and refactoring capabilities, IntelliJ IDEA streamlines the development process, allowing developers to concentrate on the creative aspects of software development (IntelliJ IDEA). The next thing we had to decide on was what database management tool we were going to use. Playing around with different ones we asked some of the other groups what database management software they were using, and we ultimately decided on MariaDB. MariaDB is an open-source relational database management system (RDBMS) that is a fork of the widely popular MySQL database. It was created by the original developers of MySQL after concerns arose about MySQL's acquisition by Oracle Corporation. MariaDB provides a drop-in replacement for MySQL, offering full compatibility with its APIs, commands, and data formats (MariaDB Enterprise). Lastly, we had to figure out what repository we were going to use, and it was recommended to use GitHub per request from our sponsor. GitHub is a web-based platform designed for developers to collaborate on software projects effectively. It leverages Git, a distributed version control system, as its core technology. GitHub not only enables developers to create, store, and manage their codebase but also facilitates code sharing and collaboration. Additionally, it offers a suite of tools for efficient project management, including issue tracking, feature requests, task management, continuous integration, and documentation through integrated wikis (GitHub). After implementing the chat commands came testing the plug-in out, we were able to test the plug-in utilizing a break/fix cycle which describes breaking as “identifying errors/bugs” and fixing as “working backwards to implement fixes for bugs identified” (Break/Fix, 2024). With the use of all these tools and methods, we were able to successfully complete our milestone two deliverables. As a group, we took in the feedback from milestone one as we significantly advanced the project and showed progress each week during our weekly meetings with the sponsor. Some key takeaways from this milestone were the importance of asking questions and using the correct software to achieve success.

Milestone three involved taking what we did in the previous milestone and expanding on it integrating this such as MiniMessage UI to the plug-in and working on displaying tickets to a discord channel. Also, storing all the required information in a database. For this milestone we also utilized a break/fix cycle for testing as well.

The Minecraft add-on project's success stemmed from normal communication and collaboration among team members throughout our duration, with meetings planned and scheduled in advance to discuss aspects like the plugin features, and presentation requirements, while always reviewing the project requirements document provided for comprehensive understanding. To facilitate communication and file-sharing, tools like Discord and Microsoft Teams were utilized, with each team member assigned specific tasks for distributed work, leveraging external resources like documentation, video tutorials, and online websites/articles. While we were able to meet the sponsor's needs, some areas for improvement include enhancing knowledge of the Paper API, better time management, and proactive communication to prevent delays or misunderstandings.

Team contribution summary

Reece: Reece's role was to code the Minecraft paper plugin using IntelliJ IDEA, where he worked on developing the core functionality by handling player interactions, and managing the server-side logic, which required a deep understanding of the Paper API and Bukkit framework, as well as proficiency in Java programming. Alongside the backend development, he focused on crafting an intuitive and visually appealing user interface (UI) using the MiniMessage library, involving creating custom formatting codes, implementing hover texts, and designing clickable components. This process required a thorough understanding of various Java libraries/frameworks such as MiniMessage, Bukkit, and Paper API, while also demanding patience/attention to detail for debugging/testing to ensure the plugin's stability and reliability within a Minecraft server environment.

Alex: Alex's role was to manage the database and code the Minecraft paper plugin using IntelliJ IDEA, where he worked on developing the core functionality by handling player interactions, and managing the server-side logic, which required a deep understanding of the Paper API and Bukkit framework, as well as proficiency in Java programming. Alongside the backend development, he focused on crafting an intuitive and visually appealing user interface (UI) using the MiniMessage library, involving creating custom formatting codes, implementing hover texts, and designing clickable components. This process required a thorough understanding of various Java libraries/frameworks such as MiniMessage, Bukkit, and Paper API, while also demanding patience/attention to detail for debugging/testing to ensure the plugin's stability and reliability within a Minecraft server environment.

Tristan: Tristan's role was to thoroughly test the Minecraft paper plugin ensuring its functionality, stability, and reliability within a Minecraft server environment. This involved rigorously testing various aspects of the plugin, such as player interactions, and server-side logic, to identify and resolve any potential issues or bugs. Additionally, he focused on testing the user interface (UI) implemented using the MiniMessage library, verifying the correctness of custom formatting codes, hover texts, and clickable components. The testing process required attention to detail, patience, and a comprehensive understanding of the Paper API, Bukkit framework, and the plugin's codebase. Through extensive break/fix cycles, he aimed to deliver a polished yet seamless experience for players, ensuring the plugin's smooth integration and operation within a Minecraft server.

Liliana: Liliana's role was to thoroughly test the Minecraft paper plugin ensuring its functionality, stability, and reliability within a Minecraft server environment. This involved rigorously testing various aspects of the plugin, such as player interactions, and server-side logic, to identify and resolve any potential issues or bugs. Additionally, she focused on testing the user interface (UI) implemented using the MiniMessage library, verifying the correctness of custom formatting codes, hover texts, and clickable components. The testing process required attention to detail, patience, and a comprehensive understanding of the Paper API, Bukkit framework, and the plugin's codebase. Through extensive break/fix cycles, she aimed to deliver a polished yet seamless experience for players, ensuring the plugin's smooth integration and operation within a Minecraft server.

Ryan: Ryan's role was to lead the team. He was responsible for ensuring that everyone stayed on track and that the team met all objectives within the designated timeframes. Ryan oversaw team coordination and organization throughout the entire project. Moreover, he made sure that the team collaborated efficiently and resolved any potential conflicts that arose. Additionally, Ryan contributed to some coding/testing, preparing the presentation slides, and drafting the research paper. His leadership/involvement was pivotal in guiding the team toward successful project completion.

Workload summary

For Milestone one, the man-hour subtotal was 14 days. This involved research, drafting the architecture for the add-on, and setting up the test Minecraft server. The deliverables for this milestone are:

- Creating a Minecraft account (1 day)
- Joining the KSU Discord/Minecraft Server (1 day)
- Downloading IntelliJ IDEA (1 day)
- Researching Minecraft Paper/Fabric Development (3 days)
- Drafting the architecture for the add-on (4 days)
- Setting up test environment (2 days)

For Milestone two, the man-hour subtotal was 28 days. This involved coding the add-on, implementing chat commands, and setting up the database. The deliverables for this milestone are:

- Familiarize yourself with the current KSU Minecraft system (6 days)
- Start coding the add-on (4 days)
- Implementing chat commands (3 days)
- Testing chat commands (2 days)
- Fixing any bugs/issues (4 days)
- Setting up plug-in to connect with database (2 days)
- Catch up week (7 days)

For Milestone three, the man-hour subtotal was 22 days. This involved finishing the plug-in, storing all the information in the database, and integrating the plug-in with Discord. The deliverables for this milestone are:

- Touch up plug-in/polish UI (5 days)
- Storing all the required information in the database (4 days)
- Push in-game tickets to Discord channel (4 days)
- Testing/Fixing Week (7 days)
- Catch-up days (3 days)

Team Reflection on Project Experience

Our experience as a group working on an in-game ticketing and reporting system for the official Kennesaw State University Esports Minecraft Discord and official server has been incredibly challenging as well as extremely valuable in advancing our understanding of programming and the software development lifecycle, project management, the collaborative development process as well as bug testing and troubleshooting.

Initially, we spent a great deal of time trying to fully grasp the requirements and goals set forth by our professor, advisor, and sponsor. Our main goal was to ensure that we dutifully explored the needs of the Kennesaw State University Esports Minecraft Community when designing the ticketing system. Through multiple meetings between our stakeholders and ourselves, we created a strong outline of the requirements needed to achieve this goal. To facilitate the development of the plugin, we leveraged the IDE IntelliJ which is a “feature rich IDE enables rapid development and helps in improving code quality,” due largely in part to its popularity, intuitive interface and debugging tools as well as its ability to interact with our repository, GitHub, quickly and easily (Intellij idea). Furthermore, the use of IntelliJ allowed the collaborative process to remain central to the development of our plugin especially when paired with other tools such as Discord, Microsoft Teams, and Outlook.

Regarding testing the plugin, we embraced a rigorous testing approach throughout the software development lifecycle primarily through IntelliJ’s debugging tools and by engaging with each other. Continuous testing based on the feedback from both sources allowed us to resolve issues and polish our product efficiently.

In conclusion, our journey working on the IT Capstone project was both enriching and rewarding. As a team, we dedicated significant time and effort to help us understand the needs of both the stakeholders and members of the KSU Esports community. Leveraging IntelliJ as our primary IDE facilitated seamless development, while collaborative tools like Discord and Microsoft Teams enhanced teamwork. Throughout the development lifecycle, we prioritized rigorous testing and engaged in continuous feedback loops, resulting in efficient issue resolution and product refinement. This experience has not only evolved our technical skills in both programming, software development, bug testing and troubleshooting but has also heightened our understanding of project management and how it is combined with the collaborative process. Moving forward, we are incredibly proud of the solution we have created and excited to see its positive impact on the KSU Esports Minecraft community.

Appendix

MinecraftTeam3Plugin - GitHub - This website shows our GitHub repository for the project. It outlines all the source code used to build this plug-in.

Project Plan - OneDrive - This file explains the main goals and objectives for the Minecraft Add-ons project. It lists plans that were used throughout the project and the deliverables for each milestone.

Gantt Chart - OneDrive - This file shows the entire schedule for the project's duration. It shows specific time lengths and tasks associated with each one.

Weekly Log Reports - OneDrive - This file shows the weekly logs for the project. This shows what the team accomplished and discussed each week during meetings. It also shows the roles each team member did throughout the week.

Milestone 1 Presentation - OneDrive - This file is the Milestone 1 Presentation. It goes over the beginning phase of the project.

Milestone 2 Presentation - OneDrive - This file is the Milestone 2 Presentation. It goes over the plugin and database function.

Milestone 3 Presentation - OneDrive - This file is the Milestone 3 Presentation. It goes over the complete plug-in and discord integration.

Minecraft Add-ons Team 3 (google.com) - This is the team’s website for the project. It contains information about the project goals and objectives and showcases each milestone work product for the project.

Bibliography

Adding Plugins. PaperMC Documentation. (n.d.). https://docs.papermc.io/paper/adding-plugins

Ahmed, S. (2022, November 7). *Key differences – comparing mariadb with mysql*. The Official Cloudways Blog. https://www.cloudways.com/blog/mariadb-vs-mysql/

Break/fix Approach. Level Access. (2024, January 25). https://www.levelaccess.com/blog/why-the-break-fix-approach-to-accessibility-is-broken/#:~:text=The%20break%2Ffix%20cycle%20is,digital%20experience%20creation%20life%20cycle.

GitHub. (n.d.). https://github.com/

IntelliJ IDEA Overview. Jetbrains. (n.d.). https://www.jetbrains.com/help/idea/discover-intellij-idea.html

Intellij idea - introduction. Tutorialspoint. (n.d.). https://www.tutorialspoint.com/intellij_idea/intellij_idea_introduction.htm

Li, J. (2023, October 16). *"Minecraft" is the best selling video game of All time*. Hypebeast. https://hypebeast.com/2023/10/microsoft-minecraft-all-time-best-selling-video-game-over-300-million-copies-sold

MariaDB Enterprise Server. MariaDB. (n.d.). https://mariadb.com/

PaperMC. (n.d.). https://papermc.io/

Self Host - Standalone Java - Windows. setup.md. (n.d.). https://www.setup.md/guides/self-host/win#setting-up-the-server

Whalen, D. (2022, April 13). *What are Minecraft plugins?* . Apex Hosting. https://apexminecrafthosting.com/about-plugins/