## High-Level Description

The goal of our project is to use Minecraft Education Edition (MEE) to introduce students to data analysis and Jupyter notebooks. Moreover, our project can be broken down into three major components: (1) websocket communication with MEE to get and store logged data; (2) creating a Jupyter notebook user interface for students to visualize their data and alter pre-written python code; (3) introductory methods of using this data in machine learning.

## Primary Users

*High School Students*

The primary users of this software will be high school aged students that have a novice level coding experience; moreover, most of their coding experiences are expected to have come from the coding learning arcs built into MEE. Furthermore, students will use the Jupyter notebook interface we implement to explore data that is collected during their Minecraft play time. Students will be able to alter pre-written python code to explore the data in their own ways.

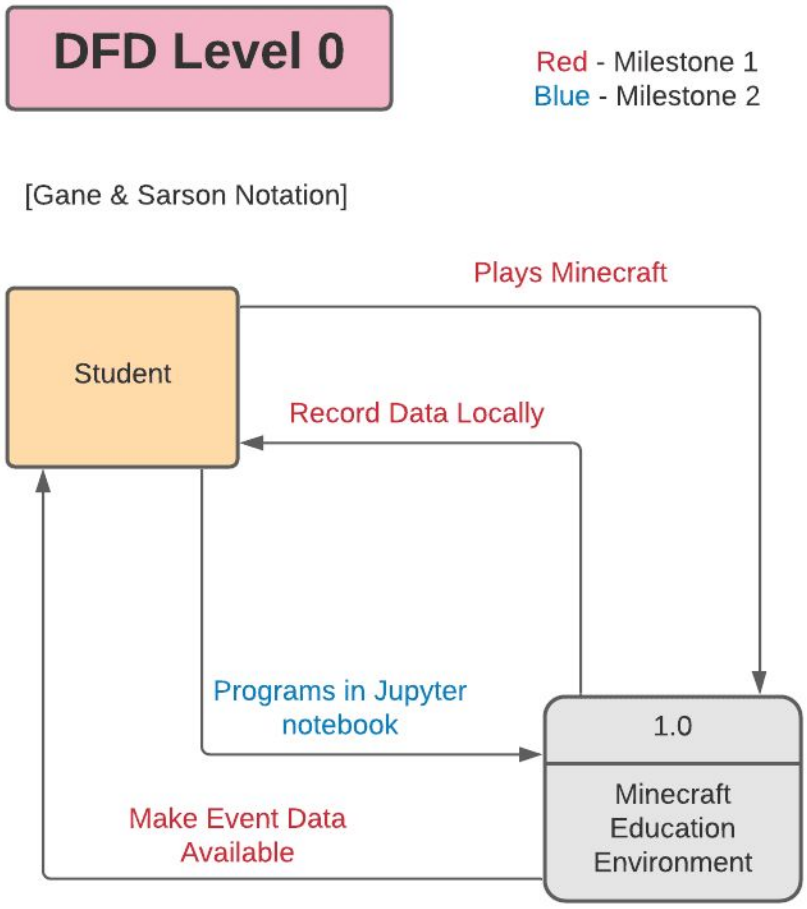
*User Story*

A Student will open up Minecraft Education Edition in their classroom. Once they have loaded a Minecraft World, they can play the game as normal with their player agent following them to collect data. Whenever the player wishes to, they can open up the CodeBuilder to view a Jupyter notebook. This notebook contains pre-written python code which they can run to illustrate their data. The player modifies this code to change the way the data is being visualized/analyzed.

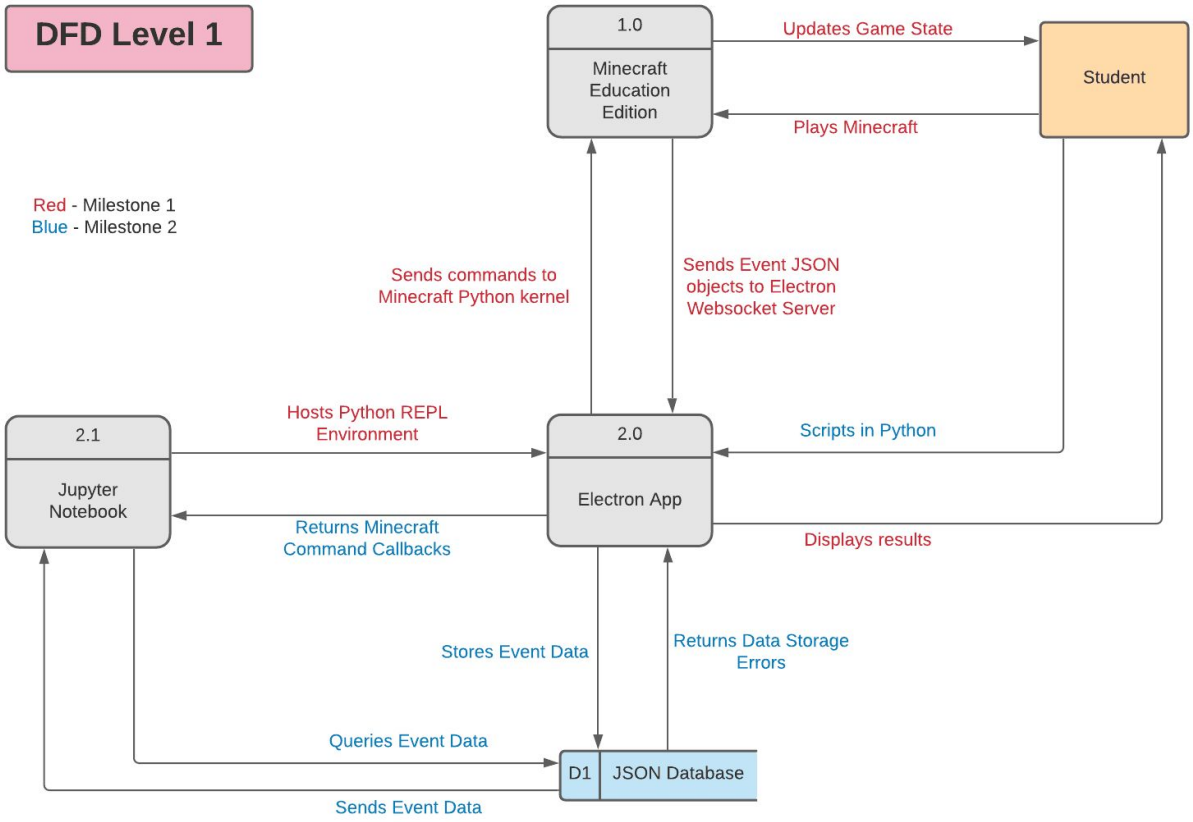
## Secondary Users

Students are expected to use Minecraft as a mode of education during both schooltime and in their freetime. This means they will have access to a teacher's help during school, or the help of their guardian(s) at home. Helpers will not be directly interacting with the game, rather providing feedback to the student verbally, without actually sitting down at their computer. Thus, we did not include them in our DFD.

**DFD Illustration and Explanation**




Our DFD Level 0, or context diagram, illustrates how our users are going to be interacting with our software at a high level. To expand on the arrows pointing into and out of the entities, the “Plays Minecraft” arrow refers to the user giving the inputs that they normally would to play Minecraft. This type of play is what our software will be using to record event data, such as what types of blocks they collect. Furthermore, our software will allow students to open up a Jupyter Notebook which can access said event data through their python programs/scripts.




For our Level 1 DFD, we go into more detail about how data will flow in what we call the ‘Minecraft Education Environment’. The environment will consist of 2 main components: Minecraft Education Edition, and an external desktop app. The external application will host a Python kernel, a Jupyter Notebook, and a database of some kind (this database is yet to be decided as the size and complexity of the data to be stored cannot currently be determined). If the student has opened up the desktop application, they will be able to query the event data collected from their play time in their Jupyter Notebook.

For Milestone One, we are planning to complete the functional requirements of recording and sending data. In order to do this, we will need to connect the minecraft player to a websocket server, thus, allowing us to fetch event data from their play. Our current plan is to use an Electron App to host a websocket server and Python kernel separate from the Python kernel within Minecraft. For Milestone Two, we are looking to store the data in a local database and have a Jupyter Notebook implemented in the Electron App such that students could write programs or scripts that use the data that was collected from their play. Additionally, we will provide sample data for them to analyze. For our Final Milestone, we will be using our sample data to create a model to train the player’s ingame agent. This is done to give students a completed example of how they can potentially use the data that our program collects.

**Peer Testing One Functional Requirements**

- Connect an external application to Minecraft Education Edition
- Store event data locally (rudimentary storage)
- Use Jupyter Notebook within external app to communicate with Minecraft Python kernel

**Peer Testing Two Functional Requirements**

- Make popular data science libraries compatible with Minecraft Python kernel externally
- Define all events necessary to store
- Implement a database that holds stored event data & can be queried
- Make event data accessible through external application

**Final Milestone Functional Requirements**

- Use event data to train the player agent, an in-game robot like helper within MEE.

**Non-functional Requirements and Environmental Constraints**

- Student can run all of this easily within MEE
- Simple enough that teachers could manage students using it in a classroom setting
- Requires basic knowledge of the game Minecraft for a good user experience

**Tech Stack**

*Game Environment*

Minecraft Education Edition is a platform based on the standard Minecraft Bedrock Edition, with additional features such as an integrated coding environment to foster a fun learning environment for people of all ages. The client decided this would be the game platform that we work on, since our goal is to introduce students to data science.

*External Application*

We are developing a desktop application which will connect with Minecraft Education Edition to allow for data collection and alteration. This external application will be built on the Electron framework. The client provided us with an application that is already integrated into the Minecraft Education Edition experience, and the app was written on the Electron framework. We will be modifying this app to achieve the same user experience while increasing functionality.

The application will be hosting a Python REPL environment to allow users to inject code into their Minecraft instance as well as store event data received from the game. There were few options for this feature, the most feasible being creating our own from a bare Python shell or integrating Jupyter Notebook into the application. Jupyter Notebook is used frequently within the data science field, and was chosen by the client for this reason.

*Data Storage*

While the user interacts with Minecraft, we will listen for various events and store said event data for later use in various data science example applications. This data will be stored locally since web-hosting a database would introduce needless complexity. The event data will be received as a JSON object, which makes any storage method that natively supports the JSON datatype very well-suited to our project. We will begin storing data in a JSON formatted local file until we have enough data to decide if a more robust storage system is necessary. We want the user to be able to import some sample data for modelling purposes. As a result, the scale of this data may require that we host a local database system to avoid potential file corruption or user error during file import. Below are 3 viable storage solutions which will be taken into consideration as the project progresses.

| | Local MongoDB | Local SQLite | Local JSON File Store |
|---|---|---|---|
| Pros | - Maintains JSON-Object structure of event data<br>- Works well with JS and Python calls | - Good JS and Python API<br>- Easy to query | - Simple, no setup<br>- Easy to load into Python |
| Cons | - May be difficult to host using an Electron app interface | - All event data will need to be restructured | - No security (from user error)<br>- Potentially inefficient |

**Testing**

*Unit Testing*

Pytest is the testing framework we will be using to test both frontend and backend python code. We are using this over Python's builtin unit test library because the syntax is simplified and, furthermore, members of our group have experience using it. Ultimately, Pytest will allow us to test individual units of code which will be integrated with our other units forming our system.

Jest will be the testing library used to test our JavaScript code, which is what our Electron App is based in. Jest has a lot of benefits including ease of set up and simplicity in its syntax. This will allow us to test individual units of our JavaScript code that we will integrate into our Electron App.

*Integration Testing*

We will be using an incremental approach to integrating and testing our software. This means that, in the case that one sub component is not yet complete, we will create a stub function to simulate the transfer of data that will be implemented. Moreover, we will have a top down approach in our incremental testing meaning we will be creating our higher level modules, like our data logging, before our lower level modules, like data manipulation. This is done to keep with our Agile development principles as it will allow us to stay flexible in the case that our requirements need to be changed.

*Regression Testing*

Part of our testing procedure will involve regression testing, which, simply put, is re-running our previous tests when new units are added. This is done to ensure that the new code does not conflict with existing working code. Moreover, our regression testing will be automated such that, when a new feature is added to the code base, unit tests will be automatically re-run to ensure that the additional features did not break features that were previously working. Running all previous tests from our test suite may include needless checks, so only critical feature tests will be run. This will reduce the time taken and system stress produced by regression testing.

**Question and Answer**

- Is there a potential for automated testing at any level of development?
    - We will be automating regression testing, since previous tests will have already been defined, running them upon a feature addition should be simple.
- Is it a mod for Minecraft?
    - No, our project is to create an app which interacts with the users Minecraft instance. The app will allow users to write scripts to analyze the data collected during their play. It will, thus, not be modifying the game, rather giving the player another means of interacting with it.
- What languages will you be programming in?
    - Python and JavaScript are the two programming languages we will be using. The reason for using JavaScript is because the base of our Electron App is built in JavaScript and, therefore, any JavaScript code could be easily added onto it. We are also using Python because the code which identifies events is written in Python.
- Will this be part of a student's coursework?
    - Our app is less intended to be part of a students coursework and more a tool for students to learn about data science individually. A major goal of our project is to make the app intrinsically valuable to students so they will pursue data science further.

- How will you be doing testing during development? Do you plan on getting younger children to test the program?
    - Testing will be an incremental process where we are writing tests for our code as we develop our software. Furthermore, we will be doing regression testing to ensure that parts do not break as we add units.
    - No, we do not plan on getting high school students to test our software.
- Also, are there any accommodations for disabled kids?
    - Minecraft is made for multiple platforms, so accessibility is handled by the users device, as well as Minecraft itself. And, thus, our job is not to implement any accessibility features to the game itself.
- How do you record this video in the game?
    - We used OBS Studio to record our screens for the video.
- Will this be free to use, or will the basic Minecraft version be required first? Will schools be able to use this program to teach classes or is it for a single user?
    - A student's institution must have access to Minecraft Education Edition for a student to have access.
    - Teachers can emulate a classroom setting through the hosting of a Minecraft instance, as well as have students play within their own instances. Our project's focus, however, is on a student using this software on their own.
- How is Minecraft going to make the process of learning data science easier for the learners?
    - The reason we believe Minecraft Education Edition will make it easier for students to learn data science is that it gives students a platform to use their data science knowledge to make things that are intrinsically meaningful to them.
    - An engaging setting allows for more effective learning, and Minecraft provides that setting for many students.
- Why choose Python over C++?
    - Since none of the code we are working with is written in C++, it made more sense for us to use JavaScript and Python because they make up the bulk of the code base we are working with. Moreover, our client recommended we use Python as students using Minecraft Education Edition will have some familiarity with it.
- What accessibility features are you planning on implementing?
    - Since our app is an external program to Minecraft Education Edition, it will not be accessible on mobile devices. Furthermore, our program will not add any accessibility features to Minecraft Education Edition.
- What database are you going to use?
    - Although we have not decided on our database yet (as it is too early to know how much data we will be storing and if a local database is truly necessary), an ideal database would be one that could handle JSON objects as that is how our event data is formatted. See *Data Storage* in the tech stack section for more information.