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) web 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 can modify this code to change the way the data is being visualized/analyzed.

Secondary Users

DFD
Level Zero
Description of how to interpret
[level0-dfd.png]
Explanation of what part we are working on by milestone

Level One
Description of how to interpret
[level1-dfd.png]
Explanation of what part we are working on by milestone

Peer Testing One Functional Requirements
- Data logging and storage locally
    - Hold data for future agent AI implementation
    - Allow users to share data to augment their agent AI
- Jupyter Notebook

- REPL environment for running minecraft commands
- Place where the students will easily be able to access data in game

Peer Testing Two Functional Requirements

- How are we handling storage?
  - Is it local? Web based?
- Agent feedback (juvenile AI model)
  - Implement agent programming / updating within REPL environment
- Bayesian Network to illustrate how the data could be connected

Final Milestone Functional Requirements

-

Non-functional Requirements and Environmental Constraints

-

Tech Stack

Already Chosen by our client

- Minecraft Education Edition (MEE), for reasons which are obvious
- Use Electron to host a jupyter notebook which will communicate with MEE
  - Introduces students to data science standards
  - Easy to understand REPL environment

Not Chosen Yet

- External AI Connection app
  - Electron-based
  - Flutter ???
- Database
  - Must be NoSQL (JSON format interacts well with both JS and Python)
  - Data is initially received as JSON

	Azure Cosmos DB	Amazon DynamoDB	Google Cloud Bigtable
Pros	- costless - high uptime (99.999% according to their website) - future integration of Azure notebooks easier - high redundancy	- widely used (robust) - PiTR very unique feature - guarantees ACIDity	- easy access to Google cloud services - high replication (similar to Azure)
Cons	- query monitoring information is sparse - does not strictly guarantee ACIDity	- does not seem to be very similar to MongoDB (which we are most familiar with) - not as cost effective as Azure	- not as cost effective as Azure - Google ecosystem can be overwhelming to interact with - not ACID

Testing

Unit Testing

Holistic Evaluations - For our Final Milestone

- Once our project is running we will need to ensure that interface is simple to navigate and easy to understand its functionality
- Will ask peers to review the program using the techniques learned from
-