Advancements in Advancements - Final Project Report

Group Name:

- Advancements in Advancements

Group Members:

- Matthew Bilderback
- James Miles

Problem Statement/Proposal:

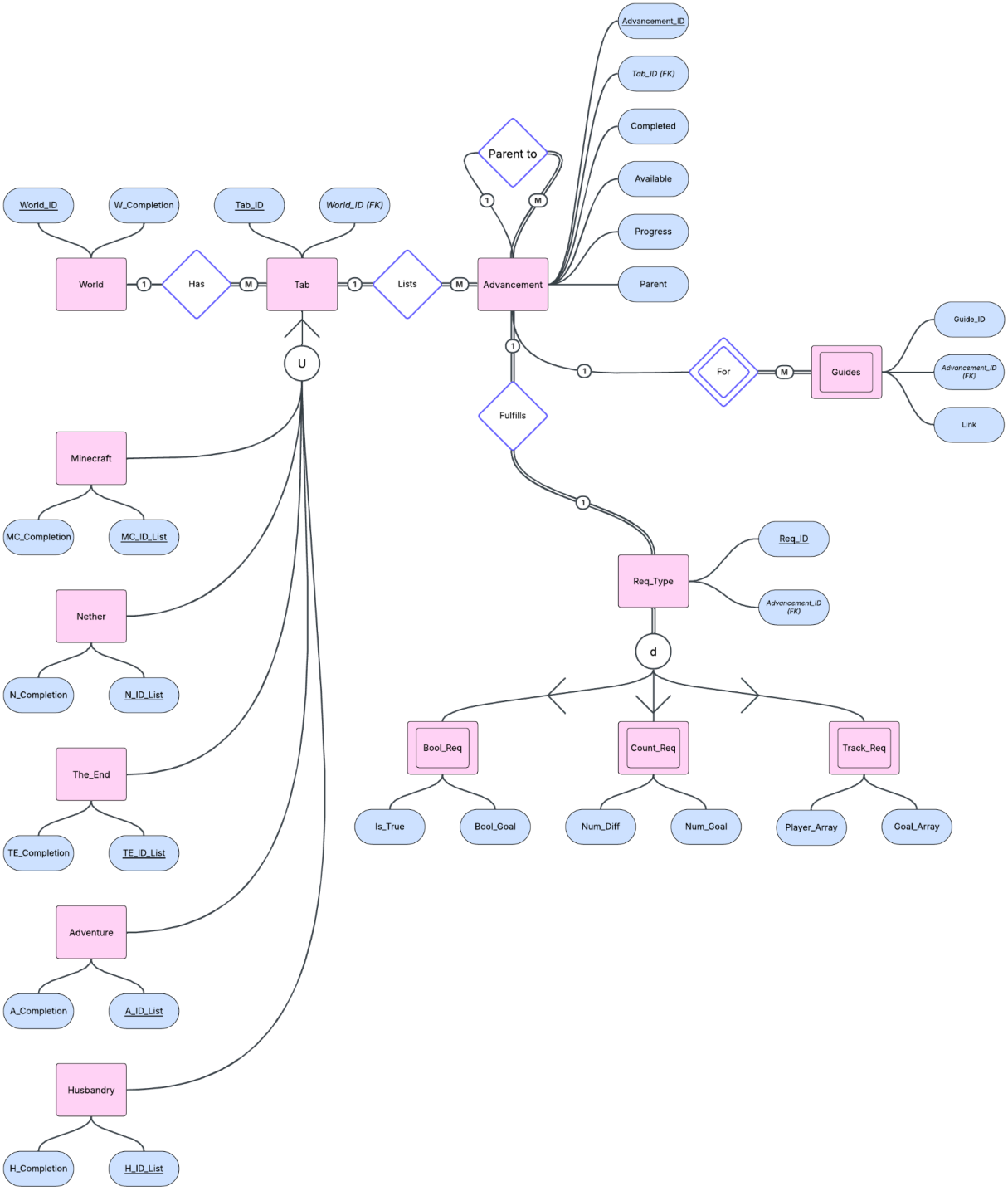
- Our project idea is to create a tool to view and track one's progression of Minecraft Java Edition advancements. This would require us to sort the necessary information that is handled by Minecraft's advancement system. The primary entities of this database system would be the worlds, the tabs, the advancements, the requirements, and the guides (main weak entity). Important attributes of these entities would include unique IDs for each strong entity, completion values (percentages for the world, each tab, and for non-boolean advancements), advancement goals (boolean, integer/float, or array), completion confirmation (boolean), guide links (string), availability (boolean), and parent advancements (string).

Conceptual Database Design:

Assumptions:

- A world can have at least one or many tabs, but a given tab has to belong to only one specific world (1:M).
- A tab has to contain at least one or many advancements, but a given advancement has to belong to only one specific tab (1:M).
- An advancement can have many guides, but a given guide has to belong to only one specific advancement (1:M).
- An advancement has to have only one type of requirement, and a specific requirement can only belong to one specific advancement (1:1).
- There will only be five tabs (Minecraft, Nether, The End, Adventure, and Husbandry)
- Not every advancement will have a parent.

EER Diagram:

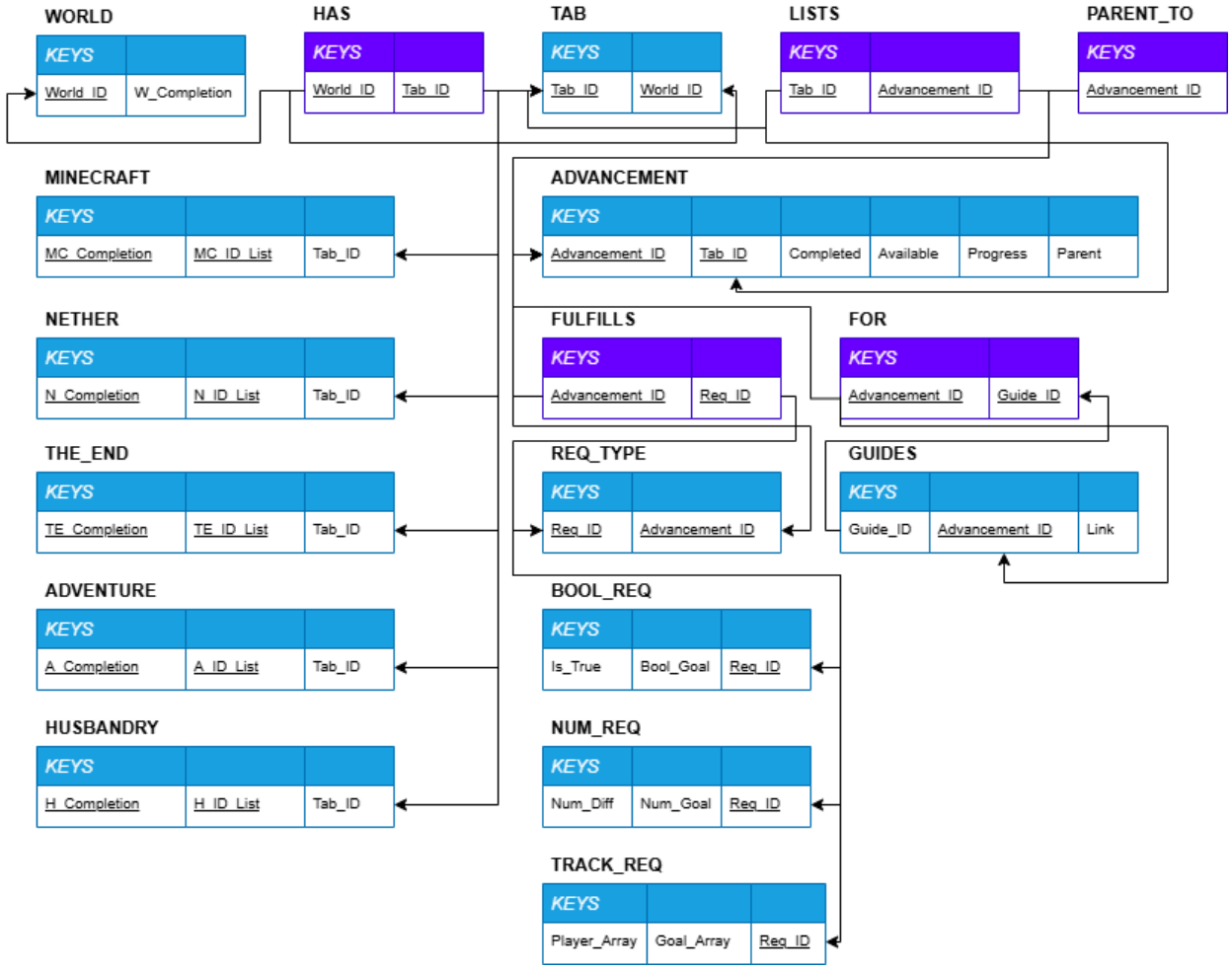

Advancement_ID
Tab_ID (FK)
Completed
Parent to
Available
Progress
Parent
World_ID
W_Completion
Tab_ID
World_ID (FK)
World
Has
Tab
Lists
Advancement
Guide_ID
Advancement_ID (FK)
Link
For
Guides
U
Fulfills
Minecraft
MC_Completion
MC_ID_List
Req_ID
Req_Type
Advancement_ID (FK)
Nether
d
N_Completion
N_ID_List
Bool_Req
Count_Req
Track_Req
The_End
Is_True
Bool_Goal
Num_Diff
Num_Goal
Player_Array
Goal_Array
TE_Completion
TE_ID_List
Adventure
A_Completion
A_ID_List
Husbandry
H_Completion
H_ID_List

Functional Requirements:

- Our database should have the following requirements:
    a. Sorting: Users can sort their advancements by tab (Minecraft, Nether, The End, Adventure, Husbandry).
    b. Filtering: Users can filter advancements by whether or not they give rewards. Potentially allow filtering by completion status (Available, Locked behind other Advancements, or Completed)
    c. Searching: Users can search for any advancement, regardless of whether or not the advancement is unlocked or hidden.
    d. Editing: Users can check and uncheck advancements based on what they currently have in the game. Potentially, we could also give users the ability to import their advancements .json file and automatically check off items.
    e. Information: Users can view detailed information about obtaining each advancement, such as the in-game description, the actual requirement, any possible rewards, and the resource path.
    f. Additional Resources: Users can access additional resources, such as links to YouTube tutorials, the Minecraft Wiki, or anything else pertinent to completing a given advancement.

Logical Database Design:

Relational Model Diagram:



Summary Table of Data Types:

Table	Attribute	Type	Constraint
World	World_ID	INTEGER	Primary Key
World	W_Completion	FLOAT	NOT NULL
Tab	Tab_ID	INTEGER	Primary Key & Must be in domain (0,4)
Tab	World_ID	INTEGER	Foreign Key
Minecraft	MC_Completion	FLOAT	NOT NULL
Minecraft	MC_ID_List	BLOB	Primary Key

Nether	N_Completion	FLOAT	NOT NULL
Nether	N_ID_List	BLOB	Primary Key
The_End	TE_Completion	FLOAT	NOT NULL
The_End	TE_ID_List	BLOB	Primary Key
Adventure	A_Completion	FLOAT	NOT NULL
Adventure	A_ID_List	BLOB	Primary Key
Husbandry	H_Completion	FLOAT	NOT NULL
Husbandry	H_ID_List	BLOB	Primary Key
Advancement	Advancement_ID	INTEGER	Primary Key
Advancement	Tab_ID	INTEGER	Foreign Key
Advancement	Completed	BOOLEAN	NOT NULL
Advancement	Available	BOOLEAN	NOT NULL
Advancement	Progress	FLOAT	NOT NULL
Advancement	Parent	INTEGER	
Guides	Guide_ID	INTEGER	Unique
Guides	Advancement_ID	INTEGER	Foreign Key
Guides	Link	VARCHAR(n)	
Req_Type	Req_ID	INTEGER	Primary Key
Req_Type	Advancement_ID	INTEGER	Foreign Key
Bool_Req	Is_True	BOOLEAN	
Bool_Req	Bool_Goal	BOOLEAN	NOT NULL
Count_Req	Num_Diff	INTEGER	
Count_Req	Num_Goal	INTEGER	NOT NULL
Track_Req	Player_Array	BLOB	
Track_Req	Goal_Array	BLOB	NOT NULL

Has	World_ID	INTEGER	Foreign Key
Has	Tab_ID	INTEGER	Foreign Key
Lists	Tab_ID	INTEGER	Foreign Key
Lists	Advancement_ID	INTEGER	Foreign Key
Fulfills	Advancement_ID	INTEGER	Foreign Key
Fulfills	Req_ID	INTEGER	Foreign Key

Application Program Design:

Function 1: Add Advancement

Input: World, Tab, Requirements, Parents, Advancement_name

Insert Advancement_name into Tab of world with requirements: Requirements and parents: Parents

Assign New Advancement ID

Function 2: Delete Advancement

Input: Advancement_id

Check If Advancement is parent to other advancements

If yes, Delete Advancement(Child_advancement_id)

If No, Delete Advancement(Advancement_id)

Function 3: Modify

Input: Advancement_id, Attribute, Value

Change The attribute of certain Advancement_id to Value

Function 4: Search

Input: Query

For each advancement_attribute compare Query with the data

If equal return attribute

Else compare with advancement

If equal return advancement

Else compare with tab

If equal return tab

Else compare with world

If equal return world

Else return Nothing

Function 5: Info grab

Input: Advancement_ID

Obtain Requirements for Advancement_ID

Return Requirements

Obtain the Guides for Advancement_ID

If there are none: Return Not found

Else Return a list of The Guides

Graphical User Interface Design:

- Here is a mockup of what our application will look like. Visually, it will attempt to replicate the style of Minecraft.

Installation Instructions:

- **Intended Operating System(s):**
  - Windows 10
  - Windows 11
- **List of Dependencies:**
  - Project Repository
  - Latest Release of Python
  - Flask + MySQL Extension
  - MySQL, MySQL Shell, MySQL Workbench (v8.0.42, Windows, mysql-installer-web-community-8.0.42.0.msi)
  - IDE (Visual Studio Code)
- **Installation Steps for our Database System:**
  - Clone the database repository
  - Download/install Python
  - Open the command line and download/install Flask and the MySQL extension with the following commands:
    - pip install flask
    - pip install flask-mysql
  - Download/install MySQL, MySQL Shell, and MySQL Workbench. When you are installing these, be sure to input the correct settings for your server:
    - Config Type: Development Computer
    - TCP/IP
    - Port: 3306
  - In addition, set the user settings as follows:
    - Username: root
    - Password: advancement
  - Open MySQL Workbench, select “Local instance MySQL80,” and input the password.
  - Go to “File” and select “Open SQL Script”
  - Open the SQL script included with the repository and run it as a query to create the tables and populate the database.
  - Download/install the IDE of your choice and open the cloned repository.
  - Run the following command in the terminal and open the locally hosted website in the default web browser:
    - python app.py

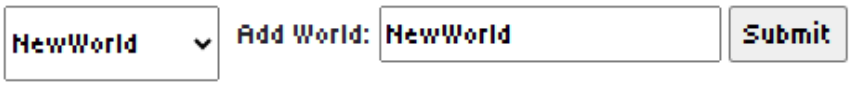
User Manual:

This is for end-users who may not have database knowledge.

Describe precisely how to use your system step by step with screenshots of your system interface and sample outputs.

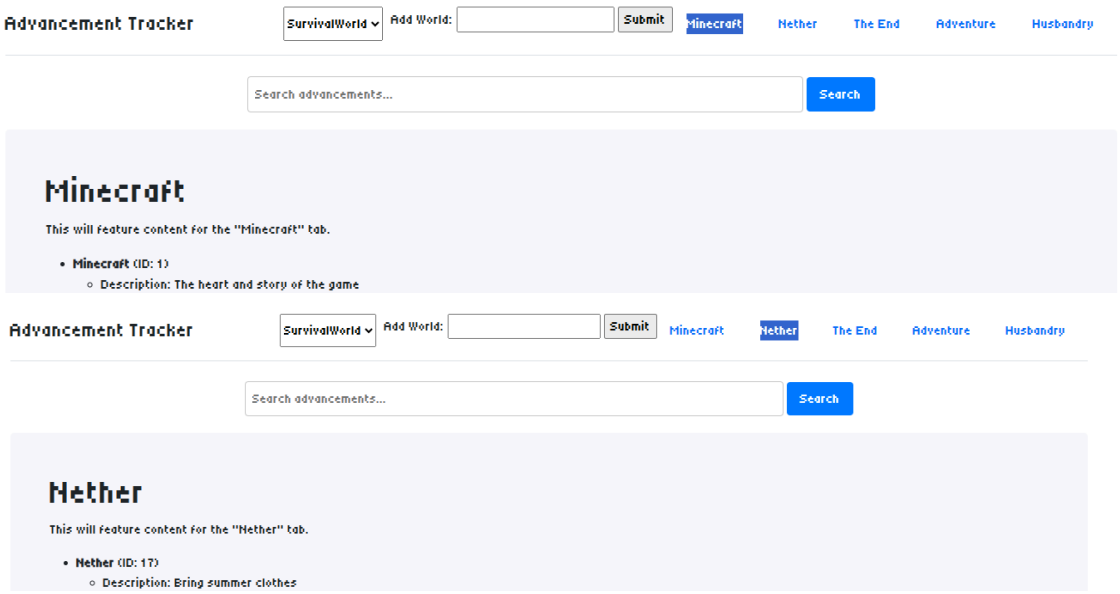
- **How to use the Advancement Tracker:**
  - **Add Worlds:** Worlds can be added and selected with the text box and dropdown on the page's header:

  - Add Output:



  - **Navigation:** Advancements are naturally divided by individual tabs, Clicking on the Blue Text will display all advancements in a certain tab:

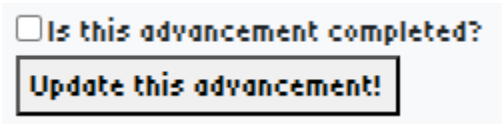
  - Navigation Output:



  - **Searching:** Using the search bar, one can search for certain advancements by name.

- Searching Output:

- **Editing:** Advancement progress can be updated through checkboxes, and submitting changes to the database.



- Editing Output: