Terraria equipment databAse

IT&C 350 database design project

Winter 2024

Sam Swindler

Ethan Beere

Matthew Gregg

Spencer Baird

Table of Contents

Project Overview 3

Project Objective Statement 3

Project Stakeholders 3

App Requirements 4

Functional Requirements 4

Non-Functional requirements 4

Database Requirements 5

ER Diagram Images 5

Schema Diagram 5

Business Rules 6

Relational Schema Types 6

Database Documentation 7

GitHub Access 7

Accessing the Database 7

Data needed 7

# Project Overview

## Project objective Statement

Create an equipment preparation system for Terraria players that enables users to discover and create the best combination of equipment for their current in-game situation.

## Project Stakeholders

Who are the Stakeholders?

* Terraria players.
* Our creation team.

Who is going to be maintaining the site?

* We, the aforementioned four horsemen of this project, are the maintainers of the site.

A shortened interview with Mr. Sam, a Terraria expert.

Q: Why do you like Terraria?

A: There’s lots of bosses and lot of fun equipment. You can kill things with yo-yo's!

Q: What is the hardest thing about Terraria?

A: Acquiring stuff and getting ready for fights. Need to understand how to get ready.

Q: What would you envision an app of this type could do for you?

A: It could help me understand my options at where I am in the game.

Q: Limitations of current systems?

A: Terraria wiki exists, but you have to browse the wiki. Stats aren’t shown in a concise form.

Q: What features would you like to see?

A: Understand boss difficulties, and crafting trees.

# App Requirements

## Functional Requirements

Home Page Functionality

* After login, users can see a list of all Terraria items available for them to acquire and use.
* The list of available items changes based on filters specified by the user. With these filters:
  + Users can record what bosses their character has defeated so far.
  + Users can record whether they have entered hard-mode or not.
* Users can view individual items with statistics about each item.
* Users can select an item to add to their list of equipped items.
* Users can view items by the categories they belong to.

Character Page Functionality

* Users can see the current character statistics with the items they have equipped.
* Users can unequip items that were equipped from the Home Page.

User page Functionality

* Users can create, select, and delete Terraria characters.
* Users can change their password.
* Users can logout.

## non-Functional Requirements

Security

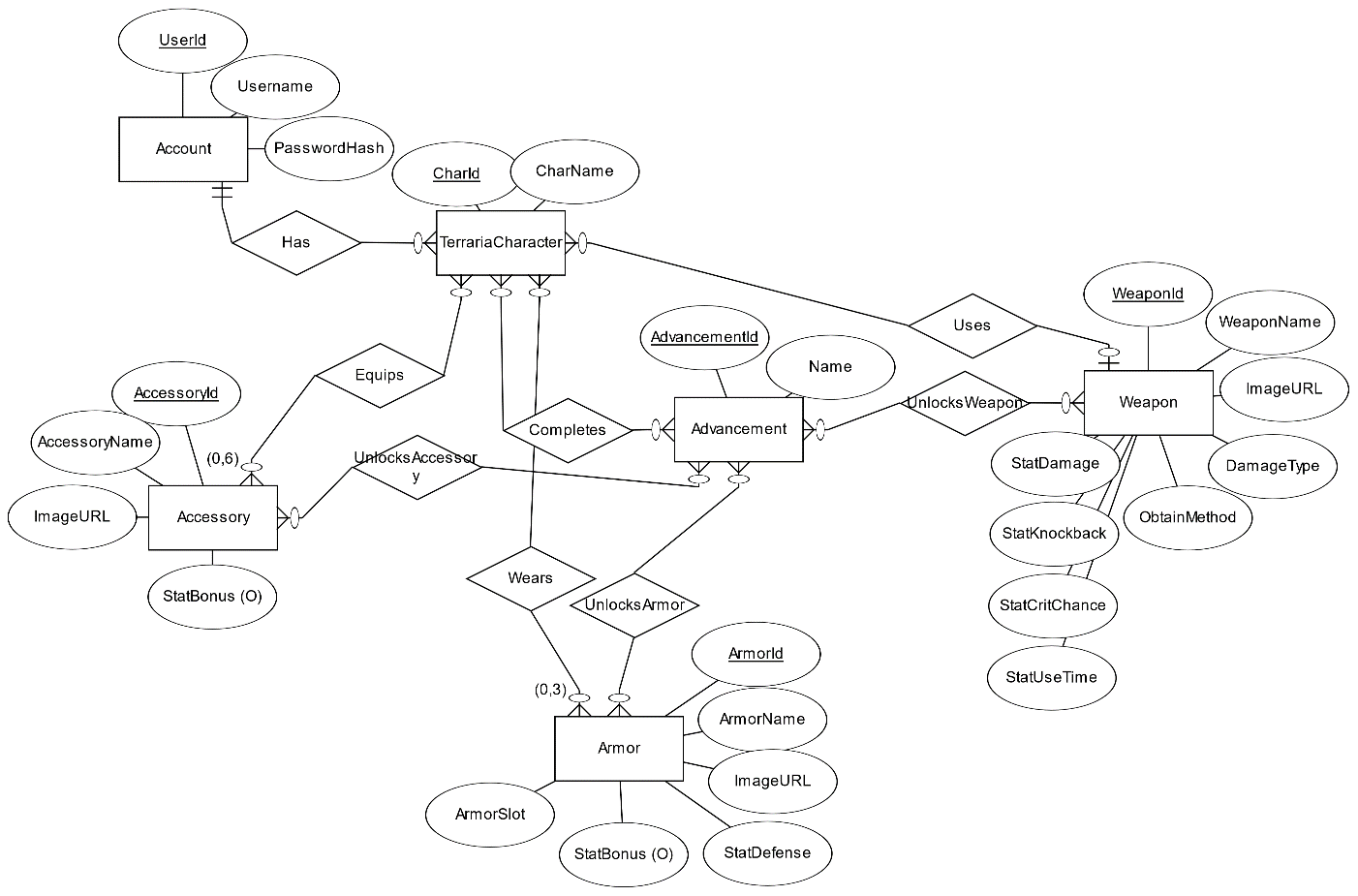
* Website protects against XSS.
* Website inputs will protect against SQL injections.

Availability

* 95% uptime and availability.

# Database Requirements

## ER Diagram Images



## Schema Diagram

## Business Rules

* Users can create up to 25 characters.
* Newly created characters have no items assigned to them.
* Characters can equip, at maximum, 1 weapon, 3 armor pieces (head, chest plate, boots), and 6 pieces of equipment.
* Users can have zero or one characters selected for viewing relevant and equipped items.
* The list of items cannot be changed from the front end and will be managed directly in the database by site administrators when needed.
* Usernames can only contain letters and numbers (no special characters).
* Attempts to equip more items than is permitted will not replace prior equipped items (operation will produce an error instead).
* If a user has no character selected, they are unable to equip any items.
* Users can have zero or more advancements selected when filtering data output.

## Relational Schema Types

**Account:**

Username (varchar)

PasswordHash (varchar)

UserID (serial)

**TerrariaCharacter:**

CharID (serial)

CharName (varchar)

UserId (bigint)

WeaponId (bigint)

**Weapon:**

WeaponID (serial)

WeaponName (varchar)

ImageURL (varchar)

StatDamage (varchar)

DamageType (varchar)

StatKnockback (varchar)

StatCritChance (varchar)

StatUseTime(varchar)

**Armor:**

ArmorID (serial)

ArmorName (varchar)

ImageURL (varchar)

StatDefense (int)

StatBonus (varchar)

ArmorSlot (int)

**Accessory:**

AccessoryId (serial)

AccesoryName (varchar)

StatBonus (varchar)

ImageURL (varchar)

**Advancement:**

Name (varchar)

AdvancementID (serial)

**UnlocksWeapon**

AdvancementId (bigint)

AccessoryId (bigint)

**UnlocksAccessory**

AdvancementId (bigint)

AccessoryId (bigint)

**UnlocksArmor**

AdvancementId (bigint)

ArmorId (bigint)

**Completes**

CharId (bigint)

AdvancementId (bigint)

**Equips**

CharId (bigint)

AccessoryId (bigint)

**Wears**

Charid (bigint)

ArmorId (bigint)

# Database Documentation

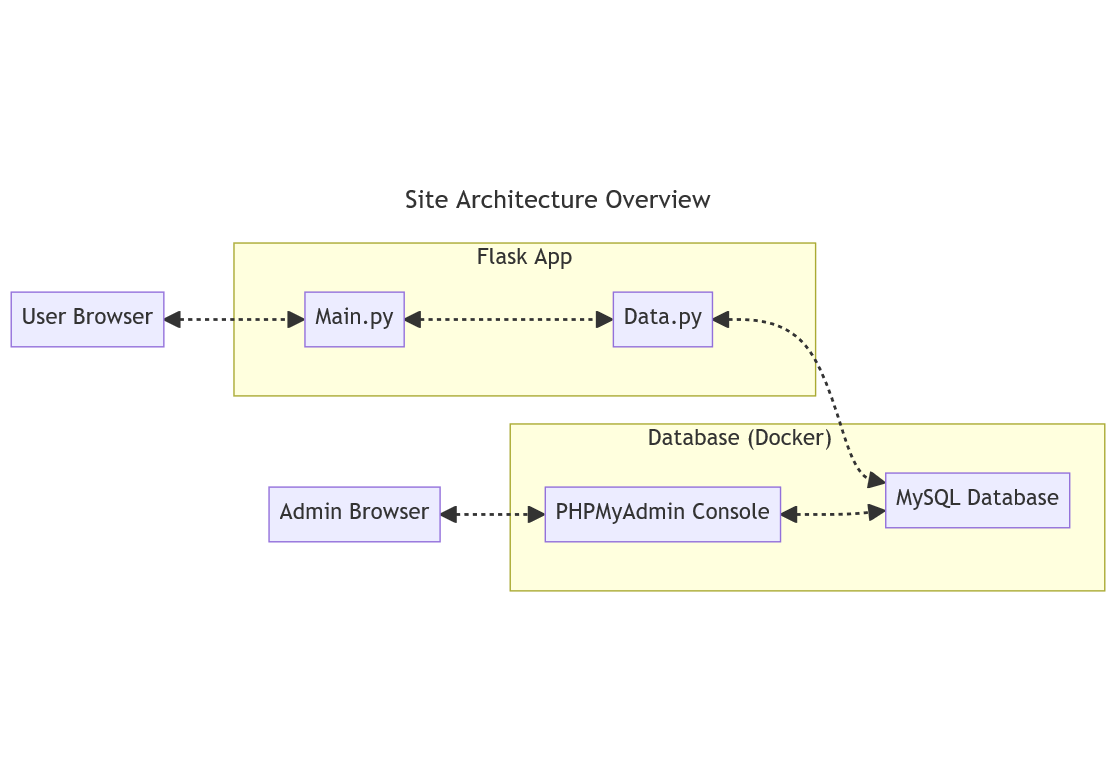


Figure 1 – Site Architecture Diagram

## Github Access

Scripts for creating, populating, and clearing the database are found on the GitHub repository under the Milestone 3 folder. The GitHub repository is accessible at <https://github.com/suspenceb/TerrariaCraftingRepo>.

## Accessing the Database

1. Connect to the IT&C VPN
2. Open a command prompt and SSH into the server with:
3. ssh username@172.16.32.12
4. Navigate to our Docker folder in /docker.
5. Turn on the Database and PHPMyAdmin with docker compose up -d (It can be turned off with docker compose down) (Information about the current images can be seen with docker ps)
6. Connect to PHPMyAdmin with http://172.16.32.12:8000 in your browser.

Note: For convenience, the domain db.beerefamily.org has been registered to point to 172.16.32.12 and can be used in place of the IP address.

## “Data for pages or something” (Jordan)

The following screenshots will display what data is needed in each screen taken from a mermaid diagram. All view statements are included on the github.

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A blue sign with white text

Description automatically generated

# API Documentation

As depicted in Figure 1 – Site Architecture Diagram, our design’s API uses a file titled “data.py” to provide a programming interface for communicating with the database. This interface is primarily utilized by “main.py” to satisfy the requests of the user.

Here is a list of application endpoints with brief descriptions for each endpoint. Additional details can be found by examining the comments provided in “data.py” itself.

|  |  |
| --- | --- |
| **App Endpoint** | **Description** |
| post\_login | Given a username and password, logs in a user, returning a session token or ‘None’ if no user is found. |
| delete\_login | Given a session token, removes user from active sessions. |
| get\_loggedin\_user | Given a session token, gets the corresponding UserId and Username. |
| update\_password | Hashes the provided password and stores it for the given UserId. |
| get\_user\_characters | Given a UserId, gets a list of characters associated with that user. |
| add\_character | Creates a new TerrariaCharacter with provided UserId and charName |
| delete\_character | Deletes the TerrariaCharacter corresponding to the provided charId |
| get\_items | Gets a list of items based on a provided list of advancements. |
| get\_advancements | Gets a list of all the advancements in the database |
| post\_equipment | Equips a given item to a given character |
| get\_character | Given a charId, gets the corresponding character |
| get\_characters | Same as “get\_user\_characters” |
| get\_armor | Given an armorId, gets the corresponding armor item |
| get\_character\_armor | Gets all armorId’s associated with a characterId |
| get\_equips | Gets all the accessoryId’s associated with a characterId |
| get\_accessories | Given an accessoryId, gets the corresponding accessory item |
| remove\_armor | Unequips an armor piece from a given character |
| remove\_accessory | Unequips an accessory from a given character |
| get\_weapon | Given a weaponId, gets the corresponding weapon item |
| get\_character\_weapon | Gets the weaponId of the weapon currently equipped to a given character |
| remove\_weapon | Uniquips a weapon from a given character |
| post\_register | Registers a new user in the database, storing username and password |

# Front-End Documentation

As depicted in Figure 1 – Site Architecture Diagram, our design’s Front-end primarily uses a file titled “main.py” to prepare the site pages requested by a user. This file receives requests from the user and satisfies them by using internal logic and making calls to endpoints in “data.py”.

Here is a list of user-facing endpoints with brief descriptions for each endpoint:

|  |  |
| --- | --- |
| User Endpoint | Description |
| “/favicon.ico” | Provides the site icon. |
| “/login” | Renders login screen and handles login attempts. |
| “/” | Renders Home screen, showing items based on user-selected filters |
| “/equipItem” | Handles requests from the Home screen to equip items |
| “/account” | Renders Account screen and handles requests to modify characters and the user’s account |
| “/logout” | Logs the user out of the site |
| “/character” | Renders the Character screen, computes statistics, and handles requests to remove items |
| “/register” | Renders the user registration screen and handles registration requests |

# Appendix 1: Low-fidelity paper prototypes

|  |
| --- |
| Login Screen |
| Account controls screen |
| Character Information Screen |
| Item Details Screen |
| Home page showing available items |

# Appendix 2: High-Fidelity Paper Prototypes

[images]