University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 4

Date: August 5th, 2024

Group Number: 69

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Harold Yan	98683345	u0s9y	haroyn754@gmail.com
Marshall Xie	26692871	v8h5v	marshallxie16@gmail.com
Marcus Kam	70855218	i1h9h	marcuskam20@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

CPSC 304 Group Project

Project Description

Our project aims to help manage the data within a Dungeons and Dragons Game. We model the relationships between the entities (Players, NPC's, Locations, Items, etc.) and help track Events that occur within a Campaign. We code stuff. It was okay.

Repository Link:
https://github.students.cs.ubc.ca/CPSC304-2024S-T2/project_i1h9h_u0s9y_v8h5v

Changes to Schema

- Added ON DELETE CASCADE for certain tables, such as Weapons
- Added WeaponID, ArmourID, ConsumableIDs as foreign keys to Player to allow the player to 'equip' the items, and to limit equipped items to 1 of each.
- Added AUTO_INCREMENT to all tables whose Primary Key did not depend on a Foreign Key. This ensured that the user didn't need to specify the ID when creating a new instance, providing a better user experience.
- Updated PrimaryKey Names to match TableNames for more cohesion
- Added 'Notes' to Player to enable the DM to keep track of relationships, status ailments, and anything information that isn't tracked by the management system.

Copy of Schema: See commands.sql
After commands.sql (initialization script) is run, we get the following tables.

```
mysql> SHOW TABLES;
+---------------+
| Tables_in_dnd |
+---------------+
| acceptedquest |
| armour        |
| campaign      |
| consumable    |
| equipment     |
| equipmentstat |
| eventlog      |
| inventory     |
| item          |
| location      |
| material      |
| npc1          |
| npc2          |
| npcstat       |
| player        |
| player1       |
| playerstat    |
| quest         |
| weapon        |
+---------------+
19 rows in set (0.04 sec)

mysql>
```

Player Table

```
mysql> SELECT * FROM Player;
+----------+------------+----------+-------+------+--------+-----------+------+---------+------+------------+------------+-------+----------+----------+--------------+
| PlayerID | Username   | Class    | Level | Exp  | Health | MaxHealth | Mana | MaxMana | Gold | CampaignID | LocationID | Notes | WeaponID | ArmourID | ConsumableID |
+----------+------------+----------+-------+------+--------+-----------+------+---------+------+------------+------------+-------+----------+----------+--------------+
|        1 | PlayerOne  | Warrior  |    10 | 2500 |    100 |       100 |   50 |      50 |  500 |          1 |
1 | NULL  |     NULL |     NULL |         NULL |
|        2 | PlayerTwo  | Mage     |     8 | 2000 |     80 |        80 |  100 |     100 |  300 |          1 |
2 | NULL  |     NULL |     NULL |         NULL |
|        3 | PlayerThree | Assassin |    12 | 3000 |     90 |        90 |   60 |      60 |  400 |          2 |
3 | NULL  |     NULL |     NULL |         NULL |
|        4 | PlayerFour | Cleric   |    15 | 3500 |    120 |       120 |   80 |      80 |  600 |          2 |
4 | NULL  |     NULL |     NULL |         NULL |
|        5 | PlayerFive | Archer   |     7 | 1800 |     70 |        70 |   40 |      40 |  200 |          3 |
5 | NULL  |     NULL |     NULL |         NULL |
+----------+------------+----------+-------+------+--------+-----------+------+---------+------+------------+------------+-------+----------+----------+--------------+
5 rows in set (0.00 sec)

mysql>
```

```
mysql> SELECT * FROM PlayerStat;
+----------+----------+------------+
| PlayerID | StatName | StatValue  |
+----------+----------+------------+
|        1 | STR      |         15 |
|        2 | INT      |         20 |
|        3 | AGI      |         18 |
|        4 | VIT      |         22 |
|        5 | DEX      |         12 |
+----------+----------+------------+
5 rows in set (0.04 sec)
```

NPC Table(s)

```
mysql> SELECT * FROM NPC1;
+--------+-------+-----------+
| Race   | Level | MaxHealth |
+--------+-------+-----------+
| Dragon |    20 |       200 |
| Dwarf  |    12 |       120 |
| Elf    |     8 |        80 |
| Goblin |     5 |        50 |
| Orc    |    10 |       100 |
+--------+-------+-----------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM NPC2;
+--------+---------+-----------------------+--------+-------+--------+-------------+------------+
| NPC2ID | NPCName | Description           | Race   | Level | Health | AIBehaviour | LocationID |
+--------+---------+-----------------------+--------+-------+--------+-------------+------------+
|      1 | Pitnik  | A small green goblin. | Goblin |     5 |     50 | Hostile     |          2 |
|      2 | Bolg    | A large gray orc.     | Orc    |    10 |    100 | Hostile     |          5 |
|      3 | Sylphie | A slender elf.        | Elf    |     8 |     80 | Neutral     |          1 |
|      4 | Dain    | A stout dwarf.        | Dwarf  |    12 |    120 | Friendly    |          4 |
|      5 | Olwen   | A fearsome dragon.    | Dragon |    20 |    200 | Hostile     |          3 |
+--------+---------+-----------------------+--------+-------+--------+-------------+------------+
5 rows in set (0.00 sec)
```

Campaign Table

```
mysql> SELECT * FROM Campaign;
+-------------+----------------------+------------+------------+
| CampaignID | Title                | StartDate  | EndDate    |
+-------------+----------------------+------------+------------+
|          1 | The Great Adventure | 2024-01-01 | 2024-12-31 |
|          2 | The Lost Kingdom    | 2023-01-01 | 2023-12-31 |
|          3 | The Dark Forest     | 2022-01-01 | 2022-12-31 |
|          4 | The Rising Sun      | 2021-01-01 | 2021-12-31 |
|          5 | The Hidden Treasure | 2020-01-01 | 2020-12-31 |
+-------------+----------------------+------------+------------+
5 rows in set (0.00 sec)
```

Location Table

```
mysql> SELECT * FROM Location;
+------------+------------------+---------------------------------+------------------+
| LocationID | LocationName     | Description                     | ParentLocationID |
+------------+------------------+---------------------------------+------------------+
|          1 | Elven Forest     | A dense and dark forest.        |             NULL |
|          2 | Goblin Village   | A small and dirty village.      |             NULL |
|          3 | Dragon's Mountain | A tall and treacherous mountain. |             NULL |
|          4 | Dwarven Castle   | An old yet magnificent castle.  |             NULL |
|          5 | Orc Cave         | A deep and smelly cave.         |             NULL |
+------------+------------------+---------------------------------+------------------+
5 rows in set (0.00 sec)
```

Quest and AcceptedQuest Table

```
mysql> SELECT * FROM Quest;
+---------+--------------------------+-------------------------------------------------+-------------+----------+
| QuestID | QuestName                | Description                                     | Status      | Progress |
+---------+--------------------------+-------------------------------------------------+-------------+----------+
|       1 | Defeat the Dragon        | Defeat the dragon in the mountain.              | In Progress |    50.00 |
|       2 | Find the Lost Treasure   | Locate and retrieve the lost treasure.          | Completed   |   100.00 |
|       3 | Save the Village         | Protect the village from the invading forces.   | In Progress |    73.45 |
|       4 | Rescue the Royal Family  | Rescue the kidnapped royal family.              | Not Started |     0.00 |
|       5 | Destroy the Evil Artifact | Destroy the artifact causing chaos.            | Completed   |   100.00 |
+---------+--------------------------+-------------------------------------------------+-------------+----------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM AcceptedQuest;
+---------+----------+
| QuestID | PlayerID |
+---------+----------+
|       1 |        1 |
|       2 |        2 |
|       3 |        3 |
|       4 |        4 |
|       5 |        5 |
+---------+----------+
5 rows in set (0.00 sec)
```

Item Table

```
mysql> SELECT * FROM Item;
+--------+---------------+------------+-----------+-----------------------------------------------+
| ItemID | ItemName      | ItemType   | Rarity    | Description                                   |
+--------+---------------+------------+-----------+-----------------------------------------------+
|      1 | Wooden Sword  | Weapon     | Common    | A blunt wooden sword.                         |
|      2 | Shield        | Armor      | Uncommon  | A sturdy shield of steel.                     |
|      3 | Healing Potion | Consumable | Rare      | A healing potion.                             |
|      4 | Elven Bow     | Weapon     | Epic      | A longbow crafted from elven wood.            |
|      5 | Dwarven Helmet | Armor      | Legendary | A helmet forged from dwarven mithril.         |
|      6 | Iron Sword    | Weapon     | Common    | A basic iron sword.                           |
|      7 | Steel Axe     | Weapon     | Uncommon  | A sturdy steel axe.                           |
|     10 | Leather Armor | Armor      | Common    | Basic leather armor.                          |
|     11 | Chainmail     | Armor      | Uncommon  | Sturdy chainmail armor.                       |
|     12 | Plate Armor   | Armor      | Rare      | Heavy plate armor.                            |
|     13 | Iron Ingot    | Material   | Common    | A bar of iron.                                |
|     14 | Silver Ore    | Material   | Uncommon  | Raw silver ore.                               |
|     15 | Mana Elixir   | Consumable | Uncommon  | Restores mana.                                |
|     16 | Magic Scroll  | Consumable | Rare      | A scroll with magical properties.             |
|     17 | Smoke Bomb    | Consumable | Uncommon  | Creates a cloud of smoke.                     |
|     18 | Roasted Meat  | Consumable | Common    | Restores health.                              |
|     19 | Wood Log      | Material   | Common    | A sturdy log of wood.                         |
|     21 | Mithril Ore   | Material   | Epic      | Raw mithril ore, extremely rare and valuable. |
|     22 | Leather Hide  | Material   | Common    | Tanned leather hide from an animal.           |
+--------+---------------+------------+-----------+-----------------------------------------------+
19 rows in set (0.00 sec)
```

Equipment and EquipmentStat Table

```
mysql> SELECT * FROM Equipment;
+--------+------------+
| ItemID | Durability |
+--------+------------+
|      1 |         80 |
|      2 |         60 |
|      4 |        100 |
|      5 |         50 |
|      6 |         70 |
|      7 |         75 |
|     10 |         55 |
|     11 |         65 |
|     12 |         95 |
+--------+------------+
9 rows in set (0.00 sec)

mysql> SELECT * FROM EquipmentStat;
+--------+-----------+------------+
| ItemID | StatName  | StatValue  |
+--------+-----------+------------+
|      1 | STR       |         10 |
|      2 | VIT       |         20 |
|      4 | AGI       |         40 |
|      5 | INT       |         15 |
|      6 | STR       |          8 |
|      7 | STR       |         12 |
|     10 | DEF       |          5 |
|     11 | DEF       |         10 |
|     12 | DEF       |         20 |
+--------+-----------+------------+
9 rows in set (0.00 sec)
```

Weapon, Armour, Material, Consumable Table

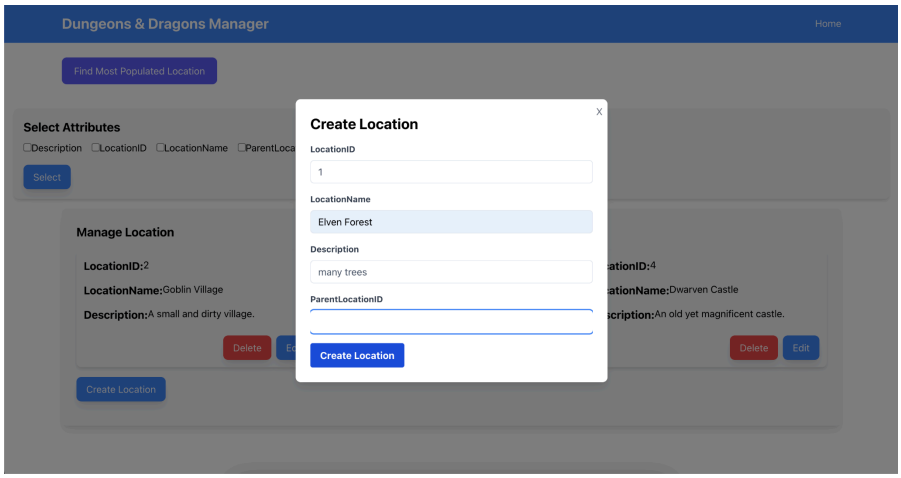
```
mysql> SELECT * FROM Weapon;
+--------+--------+
| ItemID | Attack |
+--------+--------+
|      1 |     15 |
|      4 |     25 |
|      6 |     20 |
|      7 |     30 |
+--------+--------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM Armour;
+--------+---------+
| ItemID | Defense |
+--------+---------+
|      2 |      10 |
|      5 |      30 |
|     10 |       8 |
|     11 |      15 |
|     12 |      25 |
+--------+---------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM Material
    -> ;
+--------+--------------+
| ItemID | MaterialType |
+--------+--------------+
|     13 | Iron         |
|     14 | Silver       |
|     19 | Wood         |
|     21 | Mithril      |
|     22 | Leather      |
+--------+--------------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM Consumable;
+--------+----------------+-------------+----------+
| ItemID | EffectType     | EffectValue | Duration |
+--------+----------------+-------------+----------+
|      3 | Healing        |          50 |        0 |
|     15 | Mana Restore   |          40 |        0 |
|     16 | Temporary Buff |          20 |      300 |
|     17 | Area Effect    |           0 |       10 |
|     18 | Healing        |          25 |        0 |
+--------+----------------+-------------+----------+
5 rows in set (0.00 sec)
```

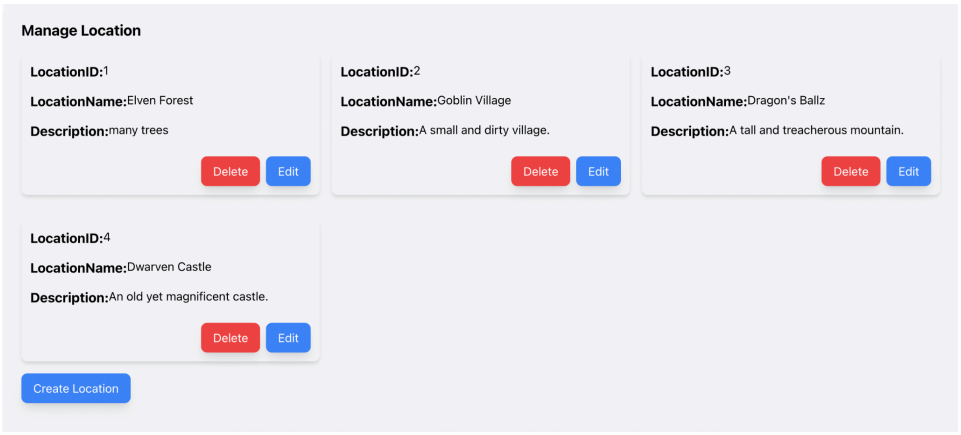
EventLog Table

```
mysql> SELECT * FROM EventLog;
+-------------+-----------------------------------------+--------------+-------------+
| EventLogID | Description                              | PlayerAction | CampaignID |
+-------------+-----------------------------------------+--------------+-------------+
|          1 | PlayerOne defeated Goblin                | Attack       |          1 |
|          2 | PlayerTwo found a treasure chest         | Search       |          1 |
|          3 | PlayerThree completed a quest            | Claim        |          2 |
|          4 | PlayerFour healed PlayerFive             | Heal         |          2 |
|          5 | PlayerFive discovered a secret passage   | Explore      |          3 |
+-------------+-----------------------------------------+--------------+-------------+
5 rows in set (0.00 sec)

mysql>
```

INSERT Query:
INSERT INTO ${table} (${columns})
VALUES (${placeholders});

```
JS index.js
JS index.js > ...
709   app.get('/api/project/:tableName', async (req, res) => {
727       const [rows] = await pool.query(query);
728       res.json(rows);
729     } catch (error) {
730       console.error('Error executing query:', error);
731       res.status(500).json({ error: 'Internal Server Error' });
732     }
733   });
734
735   // Create an Entity
736   const createEntity = async (tableName, req, res) => {
737     const { ...data } = req.body;
738     const columns = Object.keys(data).join(', ');
739     const values = Object.values(data);
740     const placeholders = values.map(() => '?').join(', ');
741
742     const table = capitalizeFirstLetter(tableName);
743     console.log("Inserting into table: ", table);
744
745     const query = `INSERT INTO ${table} (${columns}) VALUES (${placeholders})`;
746     try {
747       const [result] = await pool.query(query, values);
748       res.json({ ...data, ID: result.insertId });
749     } catch (err) {
750       const errorMessage = err.sqlMessage || `An error occurred while trying to insert into ${table}`;
751       res.status(500).json({ error: errorMessage });
752     }
753   };      MarshallXie16, yesterday • fixed some stuff
```

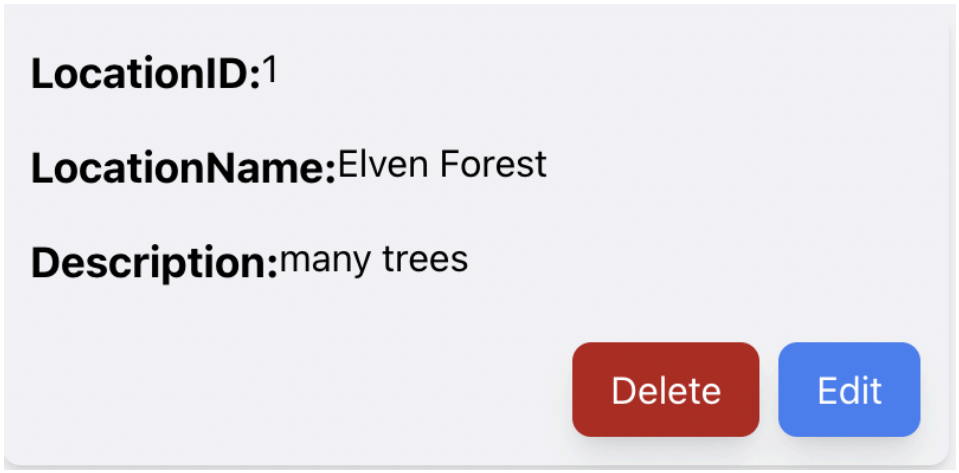


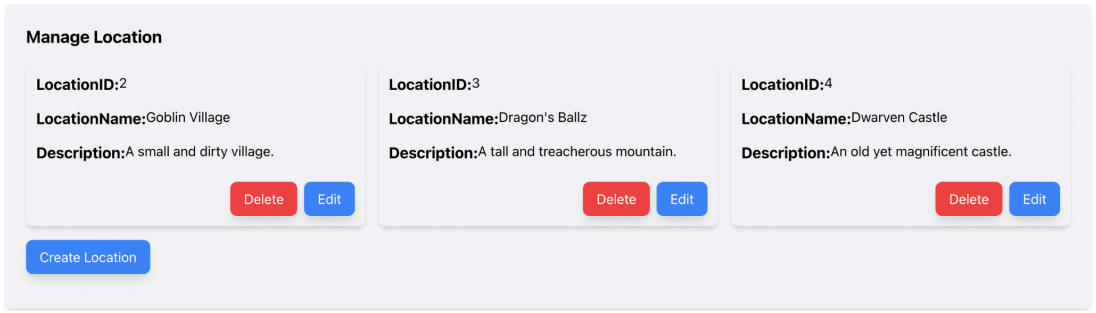


DELETE Query:

DELETE FROM ${table}
WHERE ${columnName} = ?;

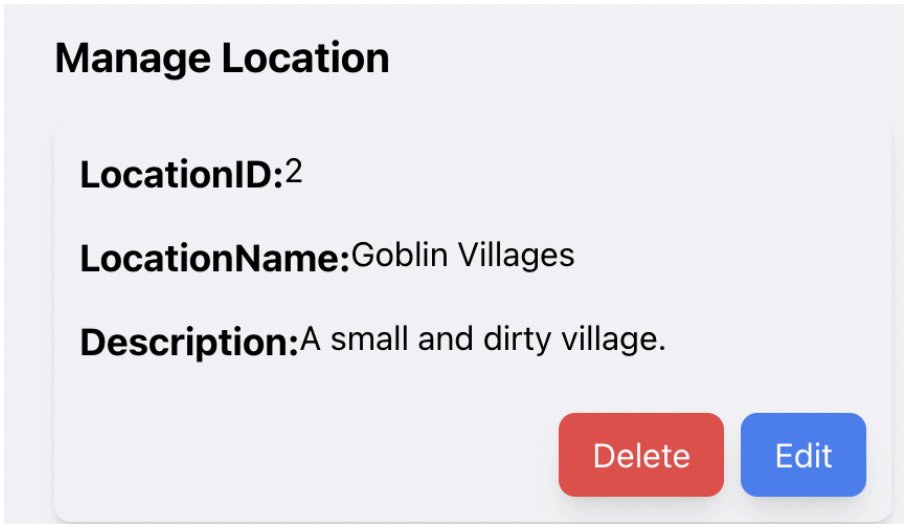
```
JS index.js
JS index.js > ...
760  const updateEntity = async (tableName, req, res) => {
778    console.error(`Error updating ${table}:`, err);
779    res.status(500).send(`Error updating ${table}`);
780  }
781  };
782
783  // Delete specific Entity from Table
784  const deleteEntity = async (tableName, req, res) => {
785    const id = req.params.id;
786    console.log(id);
787    const table = capitalizeFirstLetter(tableName);
788    const columnName = `${table}ID`;
789
790    const query = `DELETE FROM ${table} WHERE ${columnName} = ?`;
791    try {
792      const [result] = await pool.query(query, [id]);
793      if (result.affectedRows === 0) {
794        return res.status(404).send(`${table} not found`);
795      }
796      res.sendStatus(204);
797      console.log("Successful Deletion");
798    } catch (err) {
799      console.error(`Error deleting from ${table}:`, err);
800      res.status(500).send(`Error deleting from ${table}`);
801    }
802  };
803
```

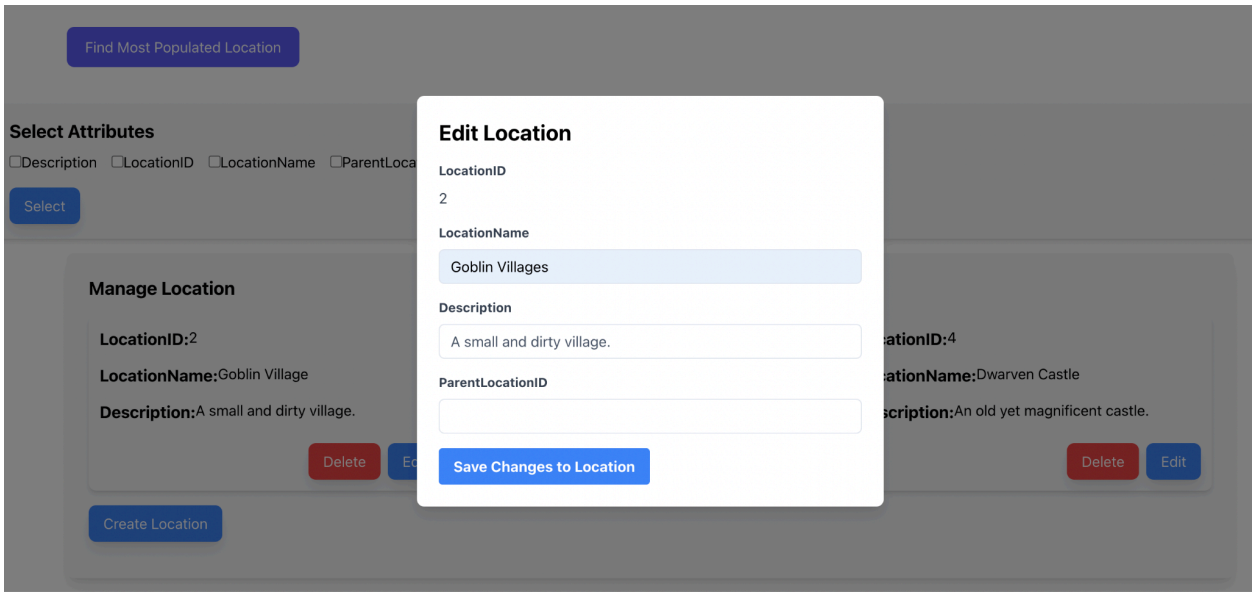




UPDATE Query:
UPDATE ${table} SET ${columns}
WHERE ${columnName} = ?;

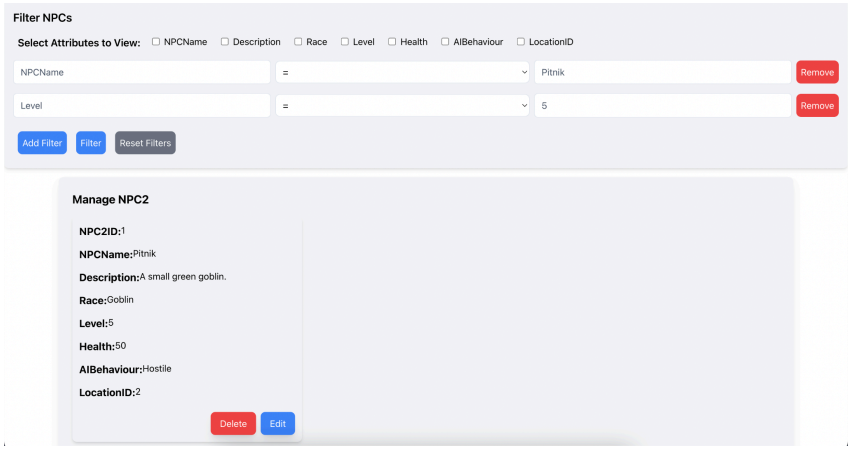
```
// Edit specific Entity from table
const updateEntity = async (tableName, req, res) => {
  const id = req.params.id;
  const { ...data } = req.body;
  const columns = Object.keys(data).map(key => `${key} = ?`).join(', ');
  const [firstValue] = Object.values(data);
  const values = [...Object.values(data), firstValue];

  const table = capitalizeFirstLetter(tableName);
  const columnName = `${table}ID`;

  const query = `UPDATE ${table} SET ${columns} WHERE ${columnName} = ?`;
  try {
    const [result] = await pool.query(query, values);
    if (result.affectedRows === 0) {
      return res.status(404).send(`${table} not found`);
    }
    res.json(data);
  } catch (err) {
    console.error(`Error updating ${table}:`, err);
    res.status(500).send(`Error updating ${table}`);
  }
};
```





SELECTION Query: ** show filtering thing
SELECT ${fields}
FROM NPC2
WHERE ${filter}; (varies depending on the state of fields and filter)

```
JS index.js
JS index.js > app.get('/api/npc') callback
113
114  // Route to handle NPC filtering
115  app.get('/api/npc', async (req, res) => {
116    try {
117      const { filter, fields } = req.query;
118
119      // Basic validation to ensure filter is a string
120      if (typeof filter !== 'string') {
121        return res.status(400).json({ error: 'Invalid filter parameter' });
122      }
123
124      // Construct the query
125      let query = `SELECT ${fields} FROM NPC2 WHERE ${filter}`;
126
127      if (fields.length == 0 && filter.length == 0) {
128        query = `SELECT * FROM NPC2`;
129      } else if (fields.length == 0) {
130        query = `SELECT * FROM NPC2 WHERE ${filter}`;      You, 7 hours ago • updated projection functionality to dynamically…
131      } else if (filter.length == 0) {
132        query = `SELECT ${fields} FROM NPC2`;
133      }
134
135      console.log(query);
136
137      // Execute the query
138      const [rows] = await pool.query(query);
139
140      res.json(rows);
141    } catch (err) {
142      console.error('Error fetching NPCs:', err);
143      res.status(500).json({ error: 'An error occurred while trying to fetch NPCs.' });
144    }
145  });
```



PROJECTION Query: ** show select table

SELECT table_name
FROM information_schema.tables
WHERE table_type='BASE TABLE'
AND table_schema = 'dnd';

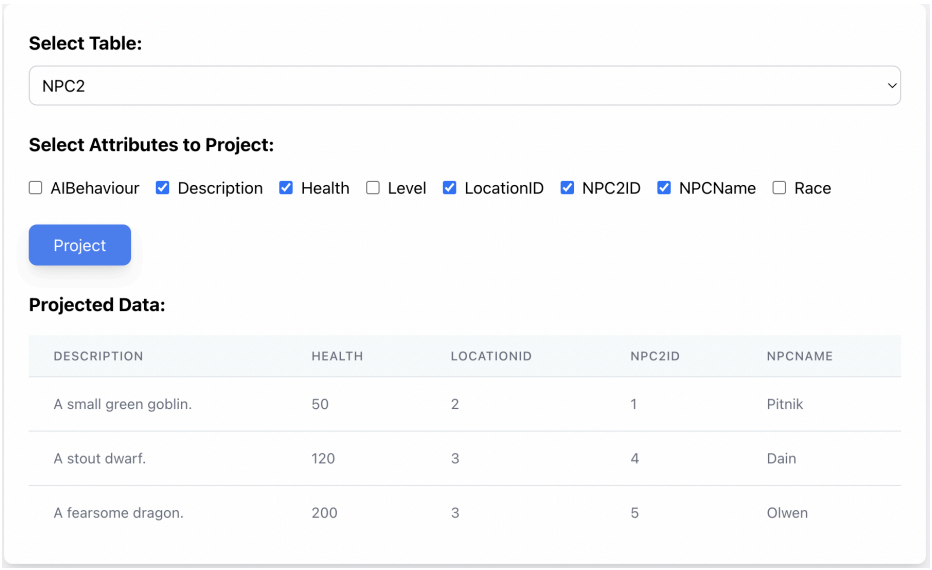
```
JS index.js
JS index.js > ...
634  // Route to get the list of tables
635  app.get('/api/tables', async (req, res) => {      You, 2 hours ago • added selection/projection requirement
636    try {
637      // query fetches all table names from the db called 'dnd'
638      const result = await pool.query(`
639        SELECT table_name
640        FROM information_schema.tables
641        WHERE table_type='BASE TABLE'
642        AND table_schema = 'dnd'
643      `);
644      // returns a list of lists of jsons where the first list contains all the json's of table names
645      const tables = result[0].map(item => item.TABLE_NAME);
646      res.json(tables);
647    } catch (error) {
648      console.error('Failed to fetch tables:', error);
649      res.status(500).json({ error: 'Failed to fetch tables' });
650    }
651  });
```

SELECT COLUMN_NAME
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_SCHEMA = ‘dnd’ and TABLE NAME = ${tableName};

```
JS index.js
JS index.js > app.get('/api/project/:tableName') callback
688  // Get attributes
689  app.get('/api/columns/:tableName', async (req, res) => {
690    const { tableName } = req.params;
691
692    try {
693      const [rows] = await pool.query(`
694        SELECT COLUMN_NAME
695        FROM INFORMATION_SCHEMA.COLUMNS
696        WHERE TABLE_SCHEMA = 'dnd' AND TABLE_NAME = ?
697      `, [tableName]);
698
699      // Extract column names from the result
700      const columnNames = rows?.map(row => row.COLUMN_NAME);
701      res.json(columnNames);
702    } catch (err) {
703      console.error('Error fetching column names:', err);
704      res.status(500).json({ error: 'An error occurred while trying to fetch column names.' });
705    }
706  });
```

SELECT ${validAttributes.join(‘, ‘)}
FROM ${tableName};

```
JS index.js
JS index.js > app.get('/api/project/:tableName') callback
707
708  // Projection function
709  app.get('/api/project/:tableName', async (req, res) => {
710    const tableName = req.params.tableName;
711    const attributes = req.query.attributes;
712
713    if (!tableName || !attributes) {
714      return res.status(400).json({ error: 'Table name and attributes are required' });
715    }
716
717    const attributeArray = attributes.split(',');
718    const validAttributes = attributeArray.map(attr => attr.trim()).filter(attr => attr); // Remove any empty or invalid attributes
719
720    if (validAttributes.length === 0) {
721      return res.status(400).json({ error: 'No valid attributes provided' });
722    }
723
724    const query = `SELECT ${validAttributes.join(', ')} FROM ${tableName}`;
725
726    try {      You, 8 hours ago • updated projection functionality to dynamically…
727      const [rows] = await pool.query(query);
728      res.json(rows);
729    } catch (error) {
730      console.error('Error executing query:', error);
731      res.status(500).json({ error: 'Internal Server Error' });
732    }
733  });
```



JOIN Query:

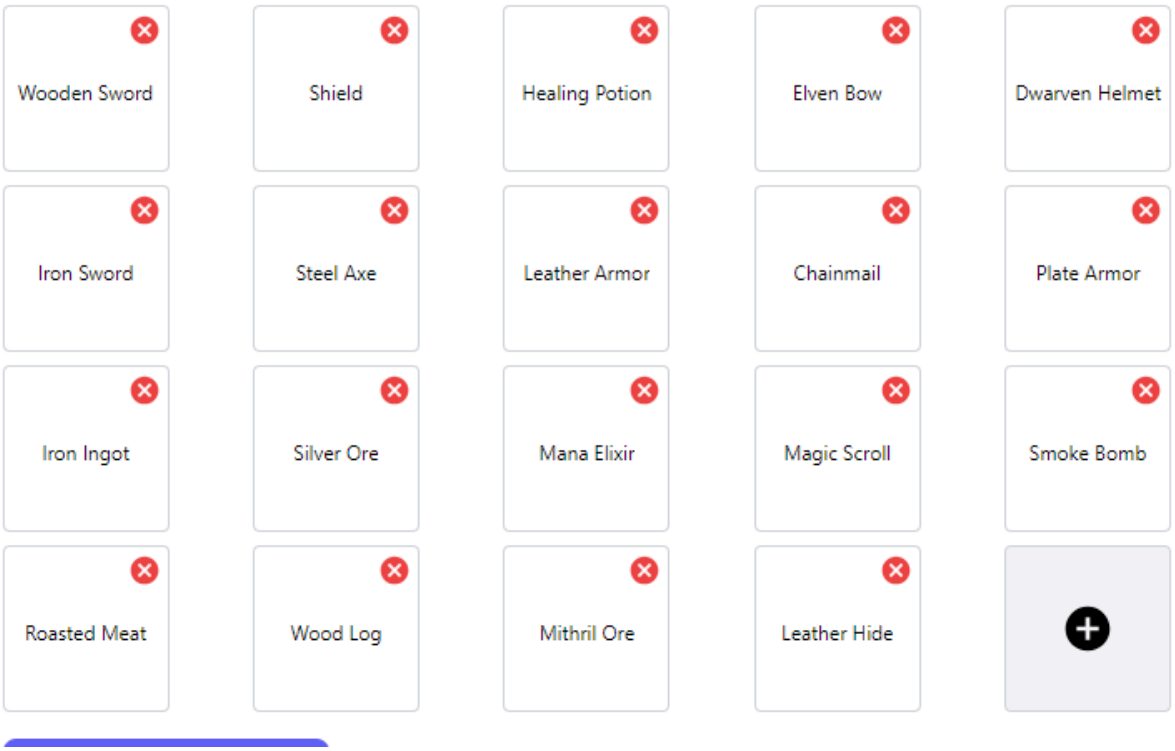
SELECT i.*, e.Durability, w.Attack, a.Defense, m.MaterialType, c.EffectType, c.EffectValue, c.Duration
FROM Inventory inv
JOIN Item i ON inv.ItemID = i.ItemID
LEFT JOIN Equipment e ON i.ItemID = e.ItemID
LEFT JOIN Weapon w ON i.ItemID = w.ItemID
LEFT JOIN Armour a ON i.ItemID = a.ItemID
LEFT JOIN Material m ON i.ItemID = m.ItemID
LEFT JOIN Consumable c ON i.ItemID = c.ItemID
WHERE inv.PlayerID = ?

```
163  // Get Player Inventory
164  app.get('/api/players/:id/inventory', async (req, res) => {
165    try {
166      const { id } = req.params;
167      const [rows] = await pool.query(`
168        SELECT i.*, e.Durability, w.Attack, a.Defense, m.MaterialType, c.EffectType, c.EffectValue, c.Duration
169        FROM Inventory inv
170        JOIN Item i ON inv.ItemID = i.ItemID
171        LEFT JOIN Equipment e ON i.ItemID = e.ItemID
172        LEFT JOIN Weapon w ON i.ItemID = w.ItemID
173        LEFT JOIN Armour a ON i.ItemID = a.ItemID
174        LEFT JOIN Material m ON i.ItemID = m.ItemID
175        LEFT JOIN Consumable c ON i.ItemID = c.ItemID
176        WHERE inv.PlayerID = ?
177      `, [id]);
178      res.json(rows);
179    } catch (err) {
180      console.error('Error:', err);
181      res.status(500).json({ error: 'An error occurred while trying to fetch player inventory.' });
182    }
183  });
```

PlayerOne

Equipment

Inventory



PlayerOne

Equipment

Elven Bow

Armour

Consumable

Inventory

Wooden Sword

Healing Potion

Healing Potion

Shield

Shield

Healing Potion

Dwarven Helmet

Iron Sword

Steel Axe

Leather Armor

Chainmail

Plate Armor

Iron Ingot

Silver Ore

Mana Elixir

Magic Scroll

Smoke Bomb

Roasted Meat

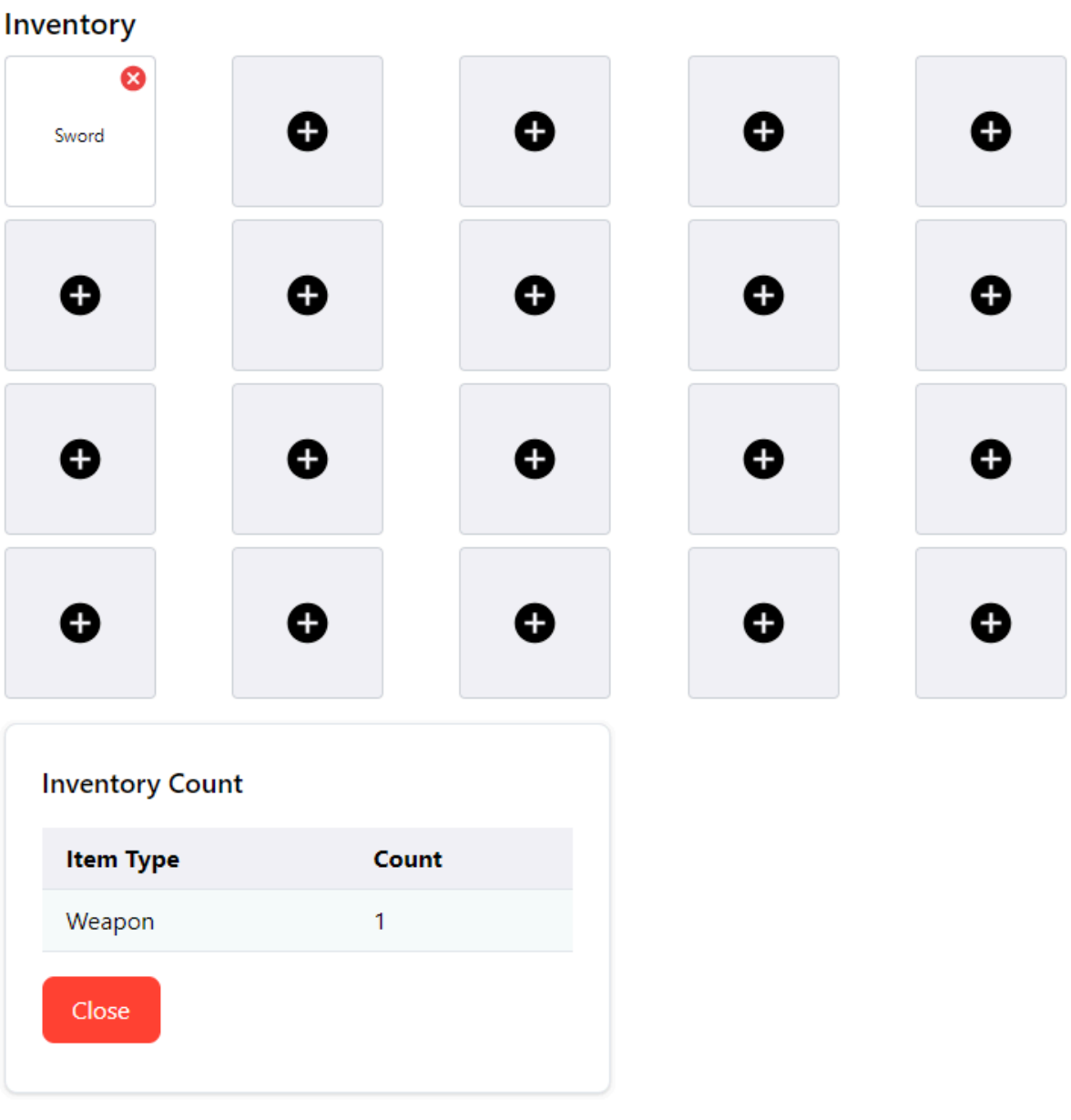
Wood Log

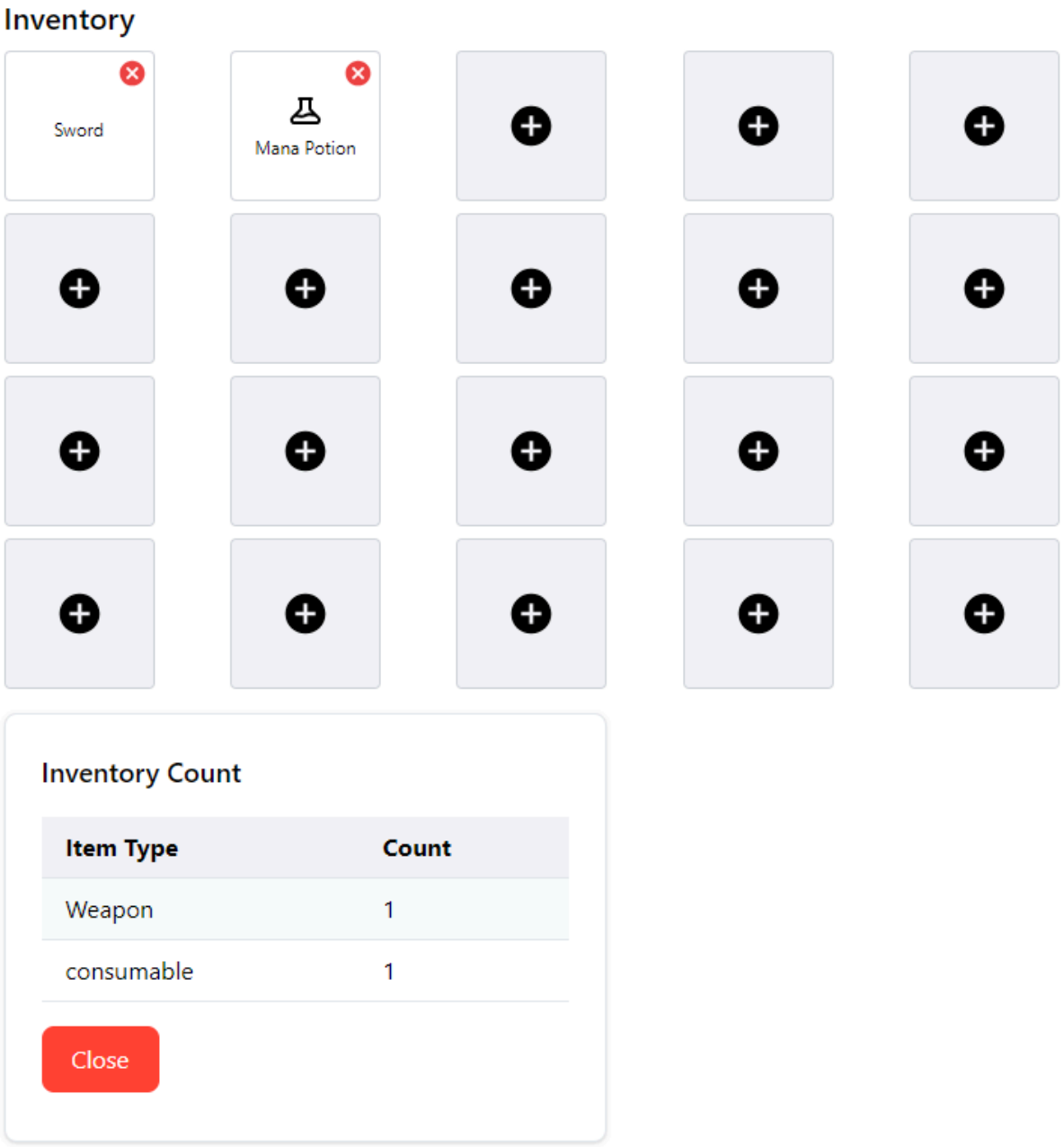
Mithril Ore

Leather Hide

Aggregation with Group By Query:
SELECT I.ItemType, COUNT(*) as Count FROM Inventory Inv, Item I WHERE Inv.ItemID = I.ItemID AND Inv.PlayerID = ? GROUP BY I.ItemType;

```
// Get number of items in player's inventory for each item type
app.get('/api/players/:id/inventory/count', async (req, res) => {
  try {
    const { id } = req.params;
    const [rows] = await pool.query('SELECT I.ItemType, COUNT(*) as Count FROM Inventory Inv, Item I WHERE Inv.ItemID = I.ItemID AND Inv.PlayerID = ? GROUP BY I.ItemType;', [id]);
    if (rows.length === 0) {
      res.json({ message: 'Inventory is empty' });
    } else {
      res.json({ result: rows });
    }
  } catch (err) {
    console.error('Error:', err);
    res.status(500).json({ error: 'An error occurred while trying to count the inventory items.' });
  }
});

```





Aggregation with Having Query:

```
SELECT LocationName
    FROM Location L, NPC2 N
    WHERE L.LocationID = N.LocationID
    GROUP BY L.LocationID
    HAVING COUNT(*) =
     (SELECT MAX(maxPopulation)
     FROM (SELECT COUNT(*) as maxPopulation
```

FROM Location L, NPC2 N
WHERE L.LocationID = N.LocationID
GROUP BY L.LocationID) as MaxPop);

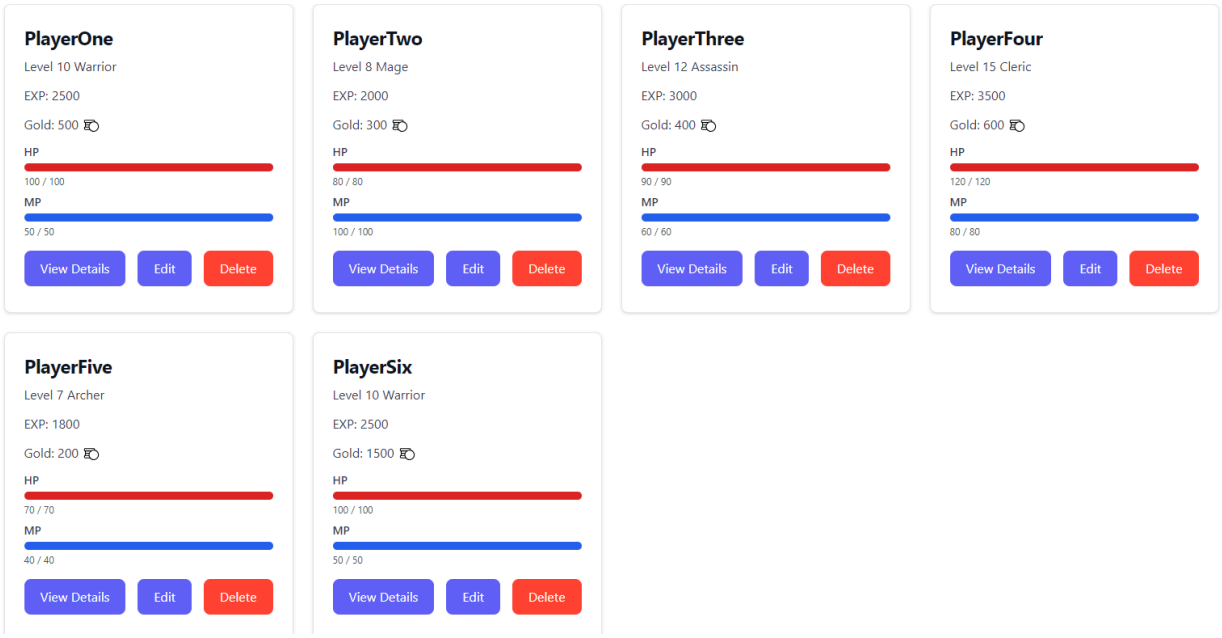
```
641   // Get most-populated locations
642   app.get('/api/location/most-populated', async (req, res) => {
643     try {
644       // most populated is defined by location with the most npcs
645       // find the location where the count of npcs is = the max count
646       const [rows] = await pool.query(`
647         SELECT LocationName
648         FROM Location L, NPC2 N
649         WHERE L.LocationID = N.LocationID
650         GROUP BY L.LocationID
651         HAVING COUNT(*) =
652           (SELECT MAX(maxPopulation)
653           FROM (SELECT COUNT(*) as maxPopulation
654                 FROM Location L, NPC2 N
655                 WHERE L.LocationID = N.LocationID
656                 GROUP BY L.LocationID) as MaxPop);
657       `);
```

Elven Forest, Goblin Village, Dragon's Mountain, Dwarven Castle, Orc Cave -> these are all the most populated locations!

Nested Aggregation with Group By Query:

SELECT Class, AVG(Gold) as averageGold
FROM (
SELECT Class, Gold
FROM Player
) AS PlayerGold
GROUP BY Class;

```
388   // Get average gold for each class of players
389   app.get('/api/players/average-gold-by-class', async (req, res) => {
390     try {
391       const [rows] = await pool.query(`
392         SELECT Class, AVG(Gold) as averageGold
393         FROM (
394           SELECT Class, Gold
395           FROM Player
396         ) AS PlayerGold
397         GROUP BY Class
398       `);
399       res.json(rows);
400     } catch (err) {
401       console.error('Error:', err);
402       res.status(500).json({ error: 'An error occurred while trying to fetch the average gold by class.' });
403     }
404   });
```

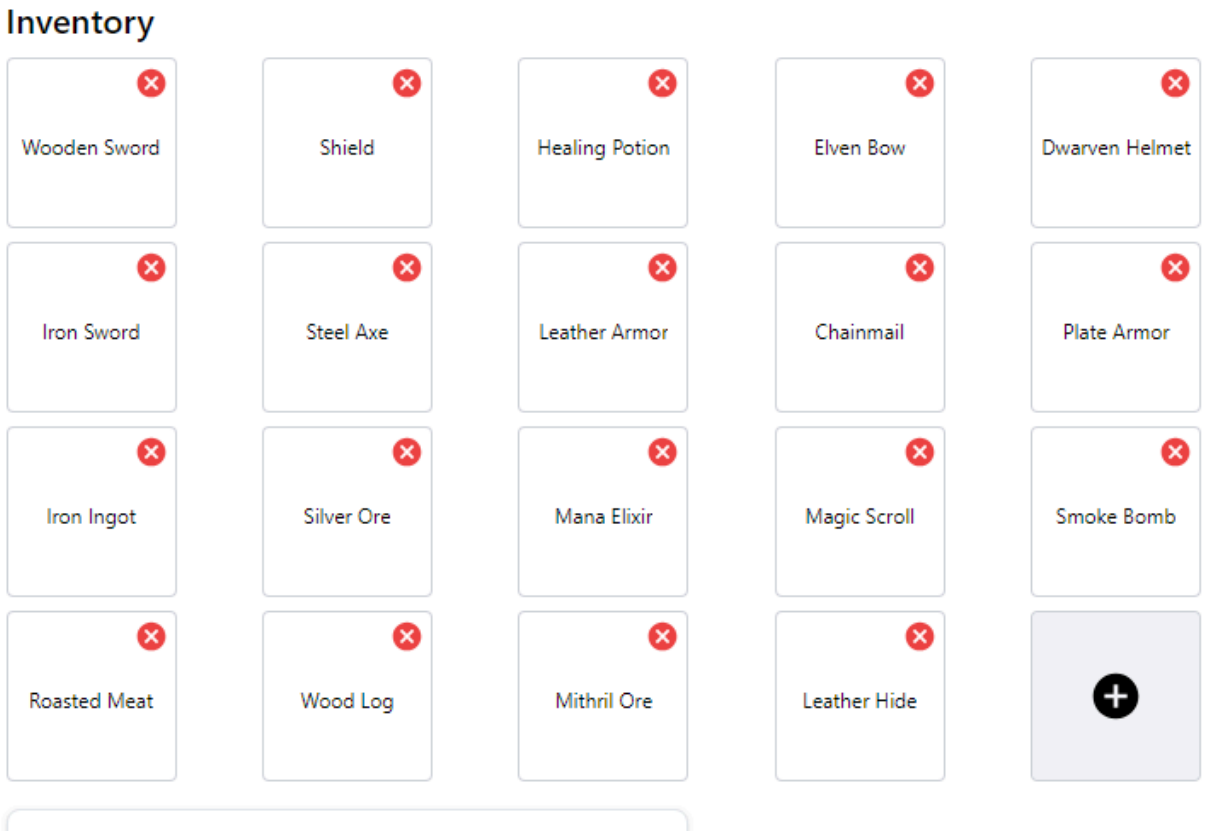


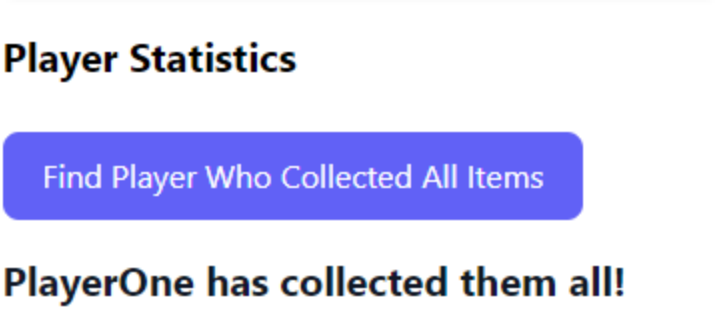
Division Query:

```
SELECT p.Username
FROM Player p
WHERE NOT EXISTS (
        SELECT i.ItemID
        FROM Item i
        WHERE NOT EXISTS (
                SELECT pi.ItemID
                FROM Inventory pi
```

```
            WHERE pi.PlayerID = p.PlayerID AND pi.ItemID = i.ItemID
        )
);
```

```
// Get players who have collected all items
app.get('/api/players/collected-all-items', async (req, res) => {
  try {
    const [rows] = await pool.query(`
      SELECT p.Username
      FROM Player p
      WHERE NOT EXISTS (
        SELECT i.ItemID
        FROM Item i
        WHERE NOT EXISTS (
          SELECT pi.ItemID
          FROM Inventory pi
          WHERE pi.PlayerID = p.PlayerID AND pi.ItemID = i.ItemID
        )
      )
    `);
```





README: (screenshot just in case)

README.md

Project Documentation

Calculated Stat Formulas

Core Stats

- **Attack:** `STR * 2 + DEX * 1 + AGI * 1 + weapon attack`
- **Defense:** `VIT * 2 + WIS * 1 + armor defense`
- **Accuracy:** `DEX * 2 + INT * 1`
- **Evasion:** `AGI * 2 + DEX * 1`

Derived Stats

- **HP:** `VIT * 5`
- **MP:** `INT * 3`

Damage Calculation

- **Damage:** `Attack - Defense`

Max Values

- **Max stat:** `100`
- **Max weapon stat:** `200`
- **Max accuracy/evasion:** `100%`