University of Science and Technology of Hanoi

Basic Database

ICT 2.5

Final Report

Author: Ngô Xuân Minh — BI9 - 167
Trịnh Thảo Phương — BI9 - 191
Phạm Minh Long — BI9 - 158

July 2020

Contents

1 Introduction **2**

1.1 Onmyoji - a Japanese style game made in China 2

1.2 The tool for Onmyoji 2

1.3 Some in-game terminologies 2

2 User requirements **3**

2.1 Shikigami guide 3

2.2 Wanted quest tool 3

2.3 Souls information 5

3 Entity Relationship Diagram **5**

4 Database Schema **6**

5 SQL queries **7**

5.1 Creating the database 7

5.1.1 Characters table 8

5.1.2 Soul table 9

5.1.3 Wanted quest table 10

5.1.4 Guide table 10

5.2 SQL queries for user requirements 11

5.2.1 Shikigami guide 11

5.2.2 Wanted quest tool 13

5.2.3 Soul information 15

6 Fetching data **16**

1 Introduction

1.1 Onmyoji - a Japanese style game made in China

Onmyoji is a turn-based fantasy strategy game with PVP or PVE battles, where you can strengthen your beloved shikigami to build your dream tactical team and defeat various demons to become the ultimate onmyoji. Cross-platform play available for both Steam and English mobile versions.[1]

The game received the score of 7/10 on Steam. It also got rated 4.1/5 on Google Play and 4.6/5 on App Store

1.2 The tool for Onmyoji

The game is included with lots of characters (*shikigamis* or in short, *shikis*) with various of skills and effects. Not only there are numerous number of shikis, the number of *Souls* that the shikis equip is also immense. Moreover, the game play consists of complicated actions such as: *Wanted Quest, Assembly Boss, Exploration*, etc... Therefore, it is such difficult for starters who have little to no knowledge to the game.

The tool that we design include

- Shikigami guide
- Wanted Quest tool
- Souls information

Visit the link to our GitHub repo for more detail.

1.3 Some in-game terminologies

Here are some terminologies that is used during this report

In-game terminology	**Normal meaning**
Shikigami	Character
Soul	Equipment
Yokai	Monster

2 User requirements

2.1 Shikigami guide

Until now, there are over 150 shikigamis with various skills and effects. Thus leading to the fact that players, not even starters, might not be able to remember all of those shikis. It is recommended to memorize all of those information in order to achieve high ranking in the game. However, for starter, there are a few shikis that needed to focus on for ease experience.

In this guide, the database must include

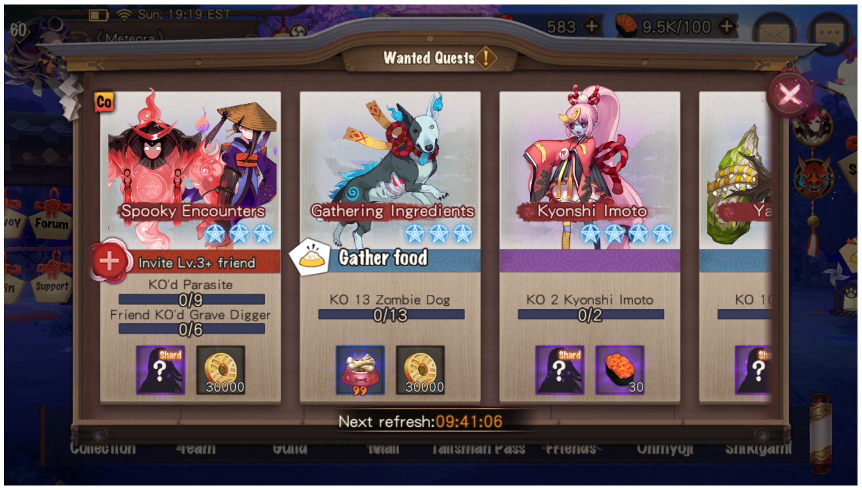
- Recommended shikis
- Recommended role
- Recommended soul set

The user might want to:

- Search for shikis that is recommended for a specific role
- Search for recommended roles for a specific shiki
- Search for recommended soul set for a specific shiki
- See the list of all shikis with recommended role, soul, effect and bonus effect

2.2 Wanted quest tool

Since wanted quest gives an significant amount of resources for starter, many player aims to complete all wanted quest for each day.

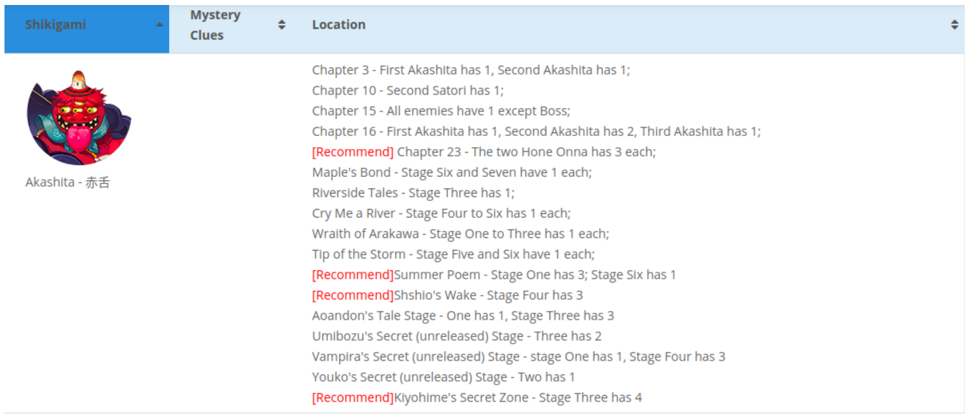


However, since they just have started playing the game, they might not be able to remember where that specific monster can be found. The wanted quest tool comes as a solution for the problem.

The tool must be able to:

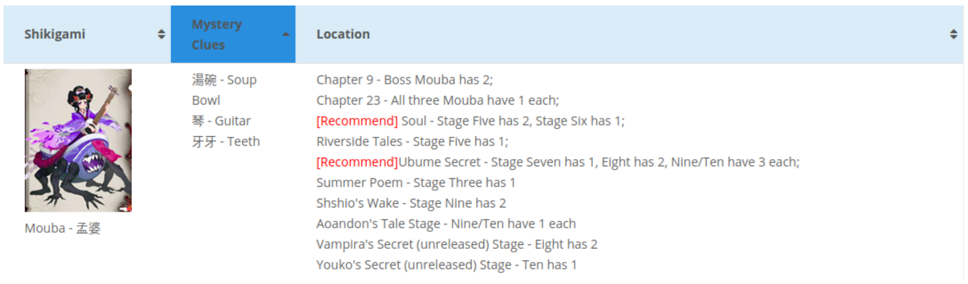
- From specific monster, the tool must be able to show places to find it.
- From specific hints, the tool must be able to identify the *yokais* and show places to find it.

Here is some examples of the result:



Shikigami	Mystery Clues	Location
Akashita - 赤舌		Chapter 3 - First Akashita has 1, Second Akashita has 1; Chapter 10 - Second Satori has 1; Chapter 15 - All enemies have 1 except Boss; Chapter 16 - First Akashita has 1, Second Akashita has 2, Third Akashita has 1; [Recommend] Chapter 23 - The two Hone Onna has 3 each; Maple's Bond - Stage Six and Seven have 1 each; Riverside Tales - Stage Three has 1; Cry Me a River - Stage Four to Six has 1 each; Wraith of Arakawa - Stage One to Three has 1 each; Tip of the Storm - Stage Five and Six have 1 each; [Recommend]Summer Poem - Stage One has 3; Stage Six has 1 [Recommend]Shshio's Wake - Stage Four has 3 Aoandon's Tale Stage - One has 1, Stage Three has 3 Umibozu's Secret (unreleased) Stage - Three has 2 Vampira's Secret (unreleased) Stage - stage One has 1, Stage Four has 3 Youko's Secret (unreleased) Stage - Two has 1 [Recommend]Kiyohime's Secret Zone - Stage Three has 4

Show places from specific monster



Shikigami	Mystery Clues	Location

Mouba - 孟婆	湯碗 - Soup Bowl
	琴 - Guitar
	牙牙 - Teeth
		Chapter 9 - Boss Mouba has 2;
		Chapter 23 - All three Mouba have 1 each;
		[Recommend] Soul - Stage Five has 2, Stage Six has 1;
		Riverside Tales - Stage Five has 1;
		[Recommend]Ubume Secret - Stage Seven has 1, Eight has 2, Nine/Ten have 3 each;
		Summer Poem - Stage Three has 1
		Shshio's Wake - Stage Nine has 2
		Aoandon's Tale Stage - Nine/Ten have 1 each
		Vampira's Secret (unreleased) Stage - Eight has 2
		Youko's Secret (unreleased) Stage - Ten has 1

Show places and monster's name from hint

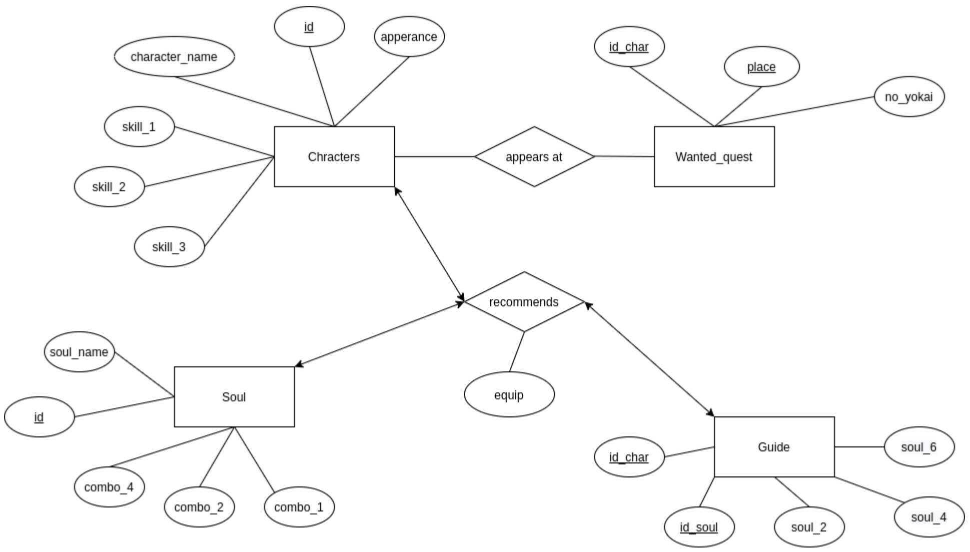
2.3 Souls information

The number of souls is not as high as the number on shikis. However, it is still a problem for starter to remember which souls has which effect.

Therefore, the database must be able to:

- From soul's name, show effect of soul set 2 or 4
- From soul's name, show the type of that soul
- From type of soul, show list of souls' name

3 Entity Relationship Diagram



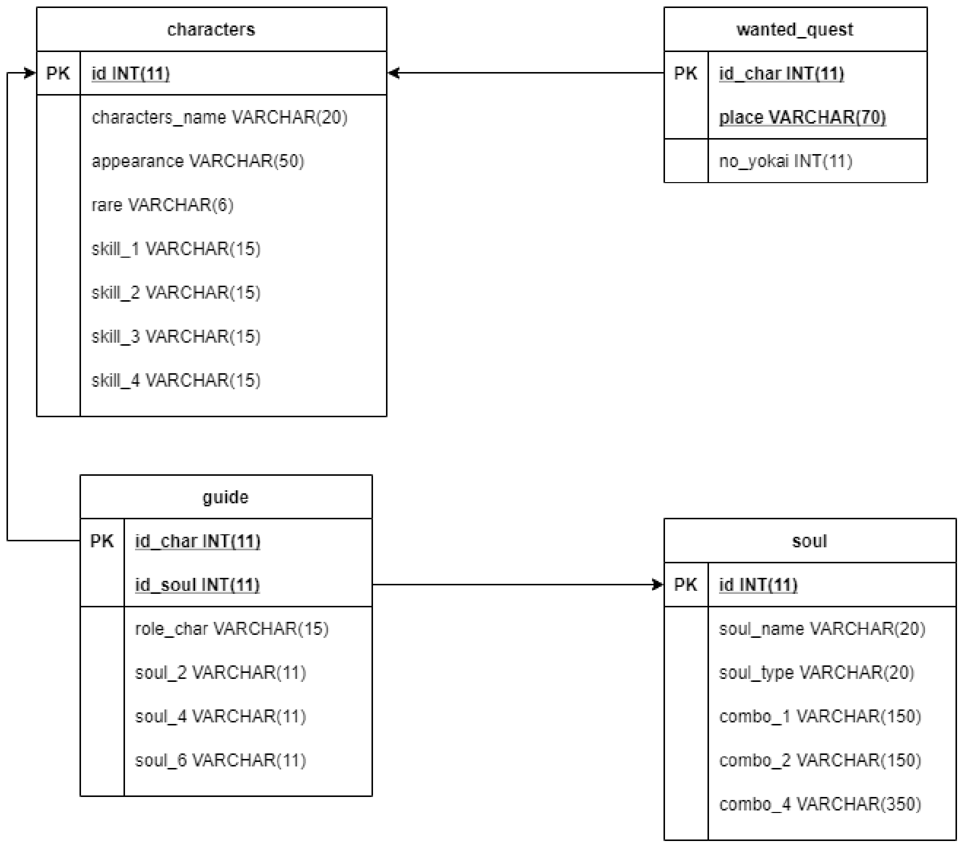
Entity Relationship Diagram

There are four entities in the ERD, they are Characters, Wanted_quest, Soul and Guide.

In the relation “appears at” between Characters and Wanted_quest, a character can appears at many places and a place can have many characters, so their relation must be in the form Many-to-many.

In the relation “recommends” between three entities Characters, Soul and Guide, each character is recommended to equipped a list in Guide and a set of souls in Soul, each guide in Guide is recommended to equipped a set of souls in Soul, so that the relations “recommends” of Characters to Guide, Characters to Soul and Guide to Soul are all One-to-One.

4 Database Schema



Database schema

There are 4 tables which are character (or shikigami), soul, wanted_quest and guide.

In the table ’characters", each character will have an unique id to create links with other tables so we make it the PRIMARY KEY and make it AUTO INCREMENT so it’s easier to calculate and manage the number of characters.

In the table "wanted_quest", the id of characters in the characters table is represented and “linked” by id_char. Since a monster can appear at many place so id and place are set as a primary key in order to be unique and not duplicated with other tuples.

In the table "soul", id is the primary key so that the table can communicate with other tables. Because of that, id cannot be NULL and have to be unique so we choose that column to have the extra “AUTO_INCREMENT”.

In the table "guide", id_char is linked to id in the table "characters"and id_soul is linked to id in the table "soul". A set of souls can be equipped by many monsters so id_char and id_soul are set as a primary key to be convenient in distinguishing souls to be equipped between monsters.

5 SQL queries

5.1 Creating the database

First, we delete the database if ’Onmyoji’ has already existed so that the new information and the old one cannot be mixed together. And then, create the new database with the name ’onmyoji’.

```
DROP DATABASE IF EXISTS onmyoji;
CREATE DATABASE IF NOT EXISTS onmyoji;
USE onmyoji
```

Based on the user requirements, we create 4 tables which are character (or shikigami), soul, wanted_quest and guide.

5.1.1 Characters table

The table will consist of id, name, appearance, rarity and the skills.

```
+-----------------+-------------+------+-----+---------+----------------+
| Field           | Type        | Null | Key | Default | Extra          |
+-----------------+-------------+------+-----+---------+----------------+
| id              | int(11)     | NO   | PRI | NULL    | auto_increment |
| characters_name | varchar(20) | YES  |     | NULL    |                |
| appearance      | varchar(50) | YES  |     | NULL    |                |
| rare            | varchar(6)  | YES  |     | NULL    |                |
| skill_1         | varchar(15) | YES  |     | NULL    |                |
| skill_2         | varchar(15) | YES  |     | NULL    |                |
| skill_3         | varchar(15) | YES  |     | NULL    |                |
| skill_4         | varchar(15) | YES  |     | NULL    |                |
+-----------------+-------------+------+-----+---------+----------------+
```

Each character will have an unique id to communicate with other tables so we make it the PRIMARY KEY and make it AUTO INCREMENT so it's easier to calculate and manage the number of characters.' Because the character table communicates with another table by the ID so the character's name doesn't have to be the key.

The appearance will contain the link of the image which shows the appearance of the character so we make it as a varchar type.

There is a maximum of 4 skills so there are 4 columns for each skill because we don't think it's a good idea to combine all the 4 skills together which make it a little bit messy and hard to see and follow.

The SQL queries for the Soul table is shown below:

```
CREATE TABLE IF NOT EXISTS characters(
    id INT(11) NOT NULL AUTO_INCREMENT,
    characters_name VARCHAR(20),
    appearance VARCHAR(50),
    rare VARCHAR(6),
    skill_1 VARCHAR(15),
    skill_2 VARCHAR(15),
    skill_3 VARCHAR(15),
    skill_4 VARCHAR(15),
    PRIMARY KEY(id)
);
```

5.1.2 Soul table

Similar with characters table, the soul table consist the id, name, skills (in this, i use the word combo)

```
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| id        | int(11)      | NO   | PRI | NULL    | auto_increment |
| soul_name | varchar(20)  | YES  |     | NULL    |                |
| soul_type | varchar(20)  | YES  |     | NULL    |                |
| combo_1   | varchar(150) | YES  |     | NULL    |                |
| combo_2   | varchar(150) | YES  |     | NULL    |                |
| combo_4   | varchar(350) | YES  |     | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
```

This table will communicate to other tables by the id of each soul so it's become the PRIMARY KEY. Because the id is the PRIMARY KEY, it cannot be NULL and have to be unique so we choose that column to have the extra "AUTO_INCREMENT".

The soul name and other columns will be in varchar type. In the combo_4, the text will be so long based on there being an explanation on that so it's too long to be in char type.
The SQL queries for the Soul table is shown below:

```
CREATE TABLE IF NOT EXISTS soul(
    id INT NOT NULL AUTO_INCREMENT,
    soul_name VARCHAR(20),
    soul_type VARCHAR(20),
    combo_1 VARCHAR (150),
    combo_2 VARCHAR(150),
    combo_4 VARCHAR(350),
    PRIMARY KEY(id)
);
```

5.1.3 Wanted quest table

```
+----------+------------+------+-----+---------+-------+
| Field    | Type       | Null | Key | Default | Extra |
+----------+------------+------+-----+---------+-------+
| id_char  | int(11)    | NO   | PRI | NULL    |       |
| place    | varchar(70)| NO   | PRI | NULL    |       |
| no_yokai | int(11)    | YES  |     | NULL    |       |
| hint     | varchar(50)| YES  |     | NULL    |       |
+----------+------------+------+-----+---------+-------+
```

The id of characters in the characters table is represented and “linked” by id_char. Each character can have many place and the place also can have many different characters so the table can’t have only one key for id_char or place. But the id_char with place can be one PRIMARY KEY because it is unique and not duplicated with other tuples. The number of yokai which is the number of characters in one place can have the save value so it cannot be a key.

```
CREATE TABLE IF NOT EXISTS wanted_quest(
    id_char INT NOT NULL,
    hint VARCHAR(50),
    place VARCHAR(70),
    no_yokai INT,
    PRIMARY KEY(id_char, place),
    FOREIGN KEY(id_char) REFERENCES characters(id)
);
```

5.1.4 Guide table

```
+-----------+------------+------+-----+---------+-------+
| Field     | Type       | Null | Key | Default | Extra |
+-----------+------------+------+-----+---------+-------+
| id_char   | int(11)    | NO   | PRI | NULL    |       |
| role_char | varchar(15)| YES  |     | NULL    |       |
| id_soul   | int(11)    | NO   | PRI | NULL    |       |
| soul_2    | varchar(11)| YES  |     | NULL    |       |
| soul_4    | varchar(11)| YES  |     | NULL    |       |
| soul_6    | varchar(11)| YES  |     | NULL    |       |
+-----------+------------+------+-----+---------+-------+
```

The guide table is linked with the characters table and soul table through id_char and id_soul. So these 2 have to be PRIMARY KEY.

Each character will have a different effect with a different type of soul and with a different number of that soul. One id_char can have more than one id_soul and one id_soul can go with different id_char so the table must have more than one key. That is the reason why we give both id_char and id_soul the PRIMARY KEY.

```
CREATE TABLE IF NOT EXISTS guide(
    id_char INT,
    role_char VARCHAR(15),
    id_soul INT,
    soul_2 VARCHAR(11),
    soul_4 VARCHAR(11),
    soul_6 VARCHAR(11),
    PRIMARY KEY(id_char, id_soul),
    FOREIGN KEY(id_char) REFERENCES characters(id),
    FOREIGN KEY(id_soul) REFERENCES soul(id)
);
```

5.2 SQL queries for user requirements

5.2.1 Shikigami guide

Search for shikis that is recommended for a specific role
To show the name of Shiki with the specific role, like 'Healer', write an INNER JOIN of *guide* and *characters* SQL query.

```
SELECT characters.characters_name FROM characters
INNER JOIN guide ON guide.id_char = characters.id
WHERE guide.role_char = 'Healer';
```

After that, the MySQL Server show:

```
+-----------------+
| characters_name |
+-----------------+
| Sakura          |
| Kusa            |
| hiyoribo        |
+-----------------+
```

Search for recommended roles for a specific shiki

If the user wants to know what role should a shikigami be in a battle, the tool can show what is the best role that shikigami should take. For example, with the shiki name 'Sakura'.

```
SELECT role_char
FROM guide
WHERE id_char =
    (SELECT id FROM characters
    WHERE characters_name = 'Sakura')
;
```

The tool will give the output which is the role of Sakura.

```
+-----------+
| role_char |
+-----------+
| Healer    |
+-----------+
```

Search for recommended soul set for a specific shiki

With this tool, the user miht know what soul should be suitable the best for each shiki and also the effect of each set of that soul so that the shiki will have the better contribute in the battle.

```
SELECT soul_name, soul_2, soul_4, soul_6
FROM soul, guide
WHERE soul.id = guide.id_soul AND guide.id_char =
    (SELECT id FROM characters WHERE characters_name = 'Sakura');
```

And the result will be shown.

```
+-----------+--------+--------+--------+
| soul_name | soul_2 | soul_4 | soul_6 |
+-----------+--------+--------+--------+
| Pearl     | HP%    | HP%    | Crit   |
+-----------+--------+--------+--------+
```

5.2.2 Wanted quest tool

From specific monster, show places to find it

For example, if the user wants to find where the Shikigami name 'Momo', just write these sql:

```
SELECT place
FROM wanted_quest, characters
WHERE wanted_quest.id_char = characters.id
AND characters.characters_name = 'Momo'
```

After that, I got the result:

```
+-----------------------------------+
| place                             |
+-----------------------------------+
| Aoandon's Tale Stage              |
| Chapter 8                         |
| Maple's Bond                      |
| Ootengu's Secret                  |
| Riverside Tales4                  |
| Secret of the Evil Blade          |
| Shshio's Wake                     |
| Soul - Stage Three                |
| Umbrella's Guard                  |
| Wraith of Arakawa                 |
| Youko's Secret (unreleased) Stage |
+-----------------------------------+
```

From specific hints, identify the monster and show places to find

For example, if the user wants to find the monster and places with the keyword 'Blue Skin'.

```
SELECT characters_name, place
FROM wanted_quest, characters
WHERE wanted_quest.id_char = characters.id AND hint like '%Blue skin%';
```

After the code, the result is shown here which show the shiki(or the character) with the hint 'Blue Skin' is Blue Imp. This shiki can be found at chapter 2, 5, 6, 10, 11, etc.

```
+----------------+---------------------------------------+
| characters_name | place                                 |
+----------------+---------------------------------------+
| Blue Imp        | Chapter 10                            |
| Blue Imp        | Chapter 11                            |
| Blue Imp        | Chapter 2                             |
| Blue Imp        | Chapter 5                             |
| Blue Imp        | Chapter 6                             |
| Blue Imp        | Chapter 8                             |
| Blue Imp        | Encounter                             |
| Blue Imp        | Extra Chapter "Hell Agent in training" |
| Blue Imp        | Riverside Tales10                     |
| Blue Imp        | Soul - Stage One                      |
+----------------+---------------------------------------+
```

See the list of all shikis with recommended role, soul, effect and bonus effect
This is the complete table show all the shikis with its role and what soul should be equipped. This tool also show all the effects and bonus effect of the soul.

```
    SELECT characters_name as shiki, role_char, soul_name,
           soul_2, soul_4, soul_6, combo_4 as effect
    FROM characters, guide, soul
    WHERE characters.id = guide.id_char
    AND soul.id = guide.id_soul;
```

The result list is very long so this is just a small of them in order to show the form of what the tool will be.

```
+----------+-----------+-------------+--------+--------+--------+---------------
| shiki    | role_char | soul_name   | soul_2 | soul_4 | soul_6 | effect
+----------+-----------+-------------+--------+--------+--------+---------------
| hiyoribo | Healer    | Tree Spirit | HP%    | HP%    | Crit   | + 20% healing
| Kusa     | Healer    | Tree Spirit | ATK%   | ATK%   | Crit   | + 20% healing
+----------+-----------+-------------+--------+--------+--------+---------------
```

5.2.3 Soul information

From soul's name, show effect of soul set 2 or 4

If the user want to find the set of 2 or 4 or both 2 and 4, run the below SQL code:

```
SELECT combo_2, combo_4
FROM soul
WHERE soul_name = 'Harpy';
```

The result will be printed to the screen:

```
+-------------+-------------------------------------+
| combo_2     | combo_4                             |
+-------------+-------------------------------------+
| Attack +15% | Gain 3 orbs upon defeating an enemy |
+-------------+-------------------------------------+
```

From soul's name, show the type of that soul

Similar to the code above, run the SQL query to show which type of that soul: 'normal' or ' boss'

```
SELECT soul_name, soul_type
FROM soul
WHERE soul_name = 'Harpy';
```

After that, the result is:

```
+-----------+-----------+
| soul_name | soul_type |
+-----------+-----------+
| Harpy     | normal    |
+-----------+-----------+
```

From type of soul, show list of souls' name

From the type, the tool must show a list of soul which have the same type.

```
SELECT soul_name
FROM soul
WHERE soul_type = 'Boss'
```

The below figure shows the name of the soul which have the same type is 'Boss'.

```
+---------------------+
| soul_name           |
+---------------------+
| Odokuro             |
| Namazu              |
| Tsuchigumo          |
| Oboroguruma         |
| Shinkirou           |
| Ghostly Songstress  |
+---------------------+
```

6 Fetching data

In this project, we use python to srapting the data from Onmyoji Guide by `BeautifulSoup` library. After that, we connect MySQL through Python with `mysql.connector` library and insert to it.

Each table will have a file scraping. The idea is to scraping data, each row will be store in an array. After that, use loop to insert into the MySQL.

For instance

```
...
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="onmyoji"
)
if(len(name) == len(appearance) == len(rare)):
    print('Finish fetching name, appearance, rare')
    count = 0
    mycursor = mydb.cursor()
    for i in range(len(name)):
        mycursor = mydb.cursor()
        sql = "INSERT INTO characters (characters_name, appearance, rare)
        VALUES (%s, %s, %s)"
```

```
            val = (str(name[i]), str(appearance[i]), str(rare[i]))
            mycursor.execute(sql, val)
            mydb.commit()
            count += mycursor.rowcount
        print(count, "record inserted.")
    ...
```

This is a quick look of a scraping file.

References

[1] Introduction of Onmyoji on Steam
https://store.steampowered.com/app/551170/Onmyoji/

[2] Python MySQL Tutorials
https://www.w3schools.com/python/python_mysql_getstarted.asp
https://www.w3schools.com/python/python_mysql_insert.asp