**Database Project RPG**

**CSCI 6333/6315**

**Rene Reyes**

**Abel Rosalez**

**Jonathan Chavez**

**Introduction**

* 1. Purpose of this document

This document will explain the logic and tools used to develop Project\_RPG. A role-playing game in which players pick their character in a fictional setting. The first part is intended for explaining the interaction of each function, and they reveal the detailed input and output of each function used in the project. The last part is more detailed and specific, and intended mainly for developer’ use.

1.1 Overview

Project\_RPG was created to exhibit the necessities required to develop a Role Playing game (RPG), underlying infrastructure of a database, and implementation of advanced database techniques taught in CSCI6315/6333. Customers that stay loyal to any game, and who are more interested in role playing games look at the quality of the game’s database in deciding on how much time and money they will invest in the game. The reason for this kind of behavior is mainly due to the types of character attributes and improvements inputted in the game, which is a sign of the effort the developers put into the game. For these objective reasons, designing a well-balanced game is dependent on the data that is imparted into the game. The quality of data that is invested in a RPG, makes marketing so much easier because the database has become ever so popular when choosing a game to play. Games such a World of Warcraft, Dungeon and Dragons, and Ever quest all reveal what is available to players at each level of a character’s life online. For example, a human would learn new abilities, and become stronger through increasing their stats as they level and gain better equipment. Superiorly advanced characters are just as important as the choice of weapons a character is eligible to attain, which in hindsight relates to an overarching goal of being able to beat the game. The aforementioned explanation is the basis of most RPG games. Project\_RPG is a RPG database. There are over 40 different characters that were developed for testing purposes. This amount of data input into the database is visible through a graphical user interface where features such as adding and deleting data are applicable.

**Interface Description**

Module Interface

WindowsApacheMyPhpadmin is used for the maintenance of the Apache, Localhost and MyPHPAdmin provides the bridge between our database server and our MySQL database.

Web Browser is used as an interface by the Developer Interface, UserAccount Interface, GameDataSearch Interface. These interfaces depending on the role of the User, it will need to match a specific username and password to grant access to the Interface connection with the database.

**Interface Description**

Module Design

All codes and images of this project associated with the Project\_RPG design is stored in the \RPG\. In the case of the Project\_RPG, there is only two folders that are associated with the Project, which is where we store the images and the php file codes. Refer to Table 6.1 for Software Reservations

|  |  |  |
| --- | --- | --- |
| # | Content | Directory |
| 1 | Sub-function Code | \RPG\PHPFile\ |
| 2 | Image | \RPG\Image |
| 3 | Application | \RPG\ |

**Authentication Function Set**

Developer Authorization

Description: Determine if the Developer login name is valid

Input:

Developer User Name, password

Output:

Returns ‘true’ if the Developer Input is authorized.

Returns ‘false’ if the Developer Input is not authorized.

Begin: If not connect to database, then

DatabaseConnection.ConnectToDB;

Retrieve record of user password from Developer User table;

If there is no record of the Developer User name from User table then

Return false;

Else if password is not correct;

Return false;

Else

Return True;

End

UserLogin Authorization

Description: Determine if the UserAccount login name is valid

Input:

UserAccount User Name, password

Output:

Returns ‘true’ if the UserAccount Input is authorized.

Returns ‘false’ if the UserAccount Input is not authorized.

Begin: If not connect to database, then

DatabaseConnection.ConnectToDB;

Retrieve record of User password from UserAccount User table;

If there is no record of the UserAccount User name from User table then

Return false;

Else if password is not correct;

Return false;

Else

Return True;

End

DataSearch Authorization

Description: Determine if the DataSearch login name is valid

Input:

GameDataSearch User Name

Output:

Returns ‘true’ if the GameDataSearch Input is authorized.

Returns ‘true’ if the GameDataSearch Input is not authorized.

Begin: If not connect to database, then

DatabaseConnection.ConnectToDB;

Retrieve record of User password from GameDataSearch User table;

If there is no record of the GameDataSearch User name from User table then

Return true;

Else

Return True;

End

**Database Access Function Set**

ConnectToDB

Description: Establish a connection to the database through a MySQLi Extension

Input: none

Output:

Pseudocode:

Begin:

Create a server connect MySQLi extension;

Instantiate an Object Oriented MySQLi connection to DB;

End

Developer Login Type Status

Description: Get the type of Login User

Input:

User name

Output:

Status of an input user name

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve record of Login user status from User table;

Return user status = Developer

If Developer = true

Return true;

Else

Return false;

End

UserAccount Login Type Status

Description: Get the type of Login User

Input:

User name

Output:

Status of an input user name

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve record of Login user status from User table;

Return user status = UserAccount

If UserAccount = true

Return true;

else

Return false;

End

**Data Retrieval Function Set**

ABILITY Retrieval

Description: Get all the ABILITY values present in the Table to which the user login is assigned

Input:

User login name

Output:

Ability\_ID, Ability\_Name, Ability\_Type, Power\_Level, Effect\_Apply, Effect\_Remove, Element

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve Ability\_ID related to the user login from ABILITY table;

Retrieve Ability\_Name related to Ability\_ID from ABILITY table;

Retrieve Ability\_Type related to Ability\_ID from ABILITY table;

Retrieve Power\_Level related to Ability\_ID from ABILITY table;

Retrieve Effect\_Apply related to Ability\_ID from ABILITY table;

Retrieve Effect\_Remove related to Ability\_ID from ABILITY table;

Retrieve Element related to Ability\_ID from ABILITY table;

Return Ability\_ID, Ability\_Name, Ability\_Type, Power\_Level, Effect\_Apply,

Effect\_Remove, Element from ABILITY table;

End

ACCOUNT Retrieval

Description: Get all the ACCOUNT values present in the Table to which the user login is assigned

Input:

User login name

Output:

Account\_ID, Email, Password, First\_Name, Last\_Name

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve Account\_ID related to the user login from ACCOUNT table;

Retrieve Email related to Account\_ID from ACCOUNT table;

Retrieve Password related to Account\_ID from ACCOUNT table;

Retrieve First\_Name related to Account\_ID from ACCOUNT table;

Retrieve Last\_Name related to Account\_ID from ACCOUNT table;

Return Account\_ID, Email, Password, First\_Name, Last\_Name,

from ACCOUNT table;

End

ACCOUNT\_BILLING Retrieval

Description: Get all the ACCOUNT\_BILLING values present in the Table to which the user login is assigned

Input:

User login name

Output:

Account\_ID, Card\_Number, Security\_Code, Expiration\_Month, Expiration\_Year

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve Account\_ID related to the user login from ACCOUNT table;

Retrieve Card\_Number related to Account\_ID from ACCOUNT\_BILLING table;

Retrieve Security\_Code related to Account\_ID from ACCOUNT\_BILLING table;

Retrieve Expiration\_Month related to Account\_ID from ACCOUNT\_BILLING table;

Retrieve Expiration\_Year related to Account\_ID from ACCOUNT\_BILLING table;

Return Account\_ID, Card\_Number, Security\_Code, Expiration\_Month, Expiration\_Year from ACCOUNT\_BILLING table;

End

ARMOR Retrieval

Description: Get all the ARMOR values present in the Table to which the user login is assigned

Input:

User login name

Output:

Armor\_ID, Armor\_Name, Armor\_Type, Armor\_Material, Stats, Stats\_Effect, Element

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve Armor\_ID related to the user login from ARMOR table;

Retrieve Armor\_Name related to Armor\_ID from ARMOR table;

Retrieve Armor\_Type related to Armor\_ID from ARMOR table;

Retrieve Armor\_Material related to Armor\_ID from ARMOR table;

Retrieve Stats related to Armor\_ID from ARMOR table;

Retrieve Stats\_Effect related to Armor\_ID from ARMOR table;

Retrieve Element related to Armor\_ID from ELEMENT table;

Return Armor\_ID, Armor\_Name, Armor\_Type, Armor\_Material, Stats, Stats\_Effect, Element from ARMOR table;

End

CHARACTER CLASS DATA Retrieval

Description: Get all values from CHARACTER\_CLASS\_DATA present in the table to which the user login is assigned

Input:

User login name

Output: Class\_Type, Stats

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve Class\_Type related to the user login from CHARACTER\_CLASS\_DATA table;

Retrieve Stats related to the Class\_Type from CHARACTER\_CLASS\_DATA table;

Return Class\_Type, Stats from CHARACTER\_CLASS\_DATA table;

End

CHARACTER\_HAS\_ABILITY Retrieval

Description: Get all values from CHARACTER\_HAS\_ABILITY present in the table to which the user login is assigned.

Input:

User login name

Output: Character\_Name, Ability\_ID

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve Character\_Name related to the user login from PLAYER\_CHARACTER table;

Retrieve Ability\_ID related to the Character\_Name from ABILITY table;

Return Character\_Name, Ability\_ID From CHARACTER\_HAS\_ABILITY table;

End

CHARACTER\_RACE\_DATA Retrieval

Description: Get all values from CHARACTER\_RACE\_DATA present in the table to which the user login is assigned.

Input:

User login name

Output: Race\_Type, Stats, Faction

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve Race\_Type related to the user login from CHARACTER\_RACE\_DATA table;

Retrieve Stats related to the Race\_Type from CHARACTER\_RACE\_TYPE table;

Retrieve Faction related to the Race\_Type from CHARACTER\_RACE\_TYPE table;

Return Race\_Type, Stats, Faction from CHARACTER\_RACE\_DATA table;

End

ELEMENT Retrieval

Description: Get all values from CHARACTER\_RACE\_DATA present in the table to which the user login is assigned.

Input:

User login name

Output: Element

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve Element related to the user login from ELEMENT table;

Return Element From ELEMENT table;

End

ELEMENT\_WEAKNESS\_AND\_RESISTANCES Retrieval

Description: Get all values from ELEMENT\_WEAKNESS\_AND\_RESISTANCES present in the table to which the user login is assigned.

Input:

User login name

Output: Element, Weak\_Against, Resistant\_Against

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve Element related to the user login from ELEMENT table;

Retrieve Weak\_Against related to the Element from ELEMENT\_WEAKNESS\_AND\_RESISTANCES table;

Retrieve Resistant\_Against related to the Element from ELEMENT\_WEAKNESS\_AND\_RESISTANCES table;

Return Element, Weak\_Against, Resistant\_Against From ELEMENT\_WEAKNESS\_AND\_RESISTANCES table;

End

ENEMY Retrieval

Description: Get all values from ENEMY present in the table to which the user login is assigned.

Input:

User login name

Output: Enemy\_ID, Enemy\_Name, Stats, Element

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve Enemy\_ID related to the user login from ENEMY table;

Retrieve Enemy\_Name related to the Enemy\_ID from ENEMY table;

Retrieve Stats related to the Enemy\_ID from ENEMY table;

Retrieve Element related to the Enemy\_ID from ELEMENT table;

Return Enemy\_ID, Enemy\_Name, Stats, Element From ENEMY table;

End

Enemy\_Armor\_Drop Retrieval

Description: Get all values from ENEMY present in the table to which the user login is assigned.

Input:

User login name

Output: Enemy\_ID, Armor\_ID, Drop\_Rate

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve Enemy\_ID related to the user login from ENEMY table;

Retrieve Armor\_ID related to the Enemy\_ID from ENEMY\_ARMOR\_DROP table;

Retrieve Drop\_Rate related to the Enemy\_ID from ENEMY\_ARMOR\_DROP table;

Return Enemy\_ID, Armor\_ID, Drop\_Rate From ENEMY\_ARMOR\_DROP table;

End

ENEMY\_HAS\_ABILITY Retrieval

Description: Get all values from ENEMY\_HAS\_ABILITY present in the table to which the user login is assigned.

Input:

User login name

Output: Enemy\_ID, Armor\_ID

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve Enemy\_ID related to the user login from ENEMY table;

Retrieve Ability\_ID related to the Enemy\_ID from ABILITY table;

Return Enemy\_ID, Ability\_ID From ENEMY\_HAS\_ABILITY table;

End

ENEMY\_IN\_REGION Retrieval

Description: Get all values from ENEMY\_IN\_REGION present in the table to which the user login is assigned.

Input:

User login name

Output: Enemy\_ID, Region\_ID

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve Enemy\_ID related to the user login from ENEMY table;

Retrieve Region\_ID related to the Enemy\_ID from REGION table;

Return Enemy\_ID, Region\_ID From ENEMY\_IN\_REGION table;

End

ENEMY\_ITEM\_DROP Retrieval

Description: Get all values from ENEMY\_ITEM\_DROP present in the table to which the user login is assigned.

Input:

User login name

Output: Enemy\_ID, Item\_ID, Drop\_Date

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve Enemy\_ID related to the user login from ENEMY\_ITEM\_DROP table;

Retrieve Item\_ID related to the Enemy\_ID from ITEM table;

Retrieve Drop\_Rate related to the Enemy\_ID from ENEMY\_ITEM\_DROP table;

Return Enemy\_ID, Item\_ID, Drop\_Rate From ENEMY\_ITEM\_DROP table;

End

ENEMY\_WEAPON\_DROP Retrieval

Description: Get all values from ENEMY\_WEAPON\_DROP present in the table to which the user login is assigned.

Input:

User login name

Output: Enemy\_ID, Weapon\_ID, Drop\_Date

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve Enemy\_ID related to the user login from ENEMY\_WEAPON\_DROP table;

Retrieve Item\_ID related to the Enemy\_ID from WEAPON table;

Retrieve Drop\_Rate related to the Enemy\_ID from ENEMY\_WEAPON\_DROP table;

Return Enemy\_ID, Weapon\_ID, Drop\_Rate From ENEMY\_WEAPON\_DROP table;

End

EQUIPPED\_ARMOR Retrieval

Description: Get all values from Equipped\_Armor present in the table to which the user login is assigned.

Input:

User login name

Output: Armor\_ID, Character\_Name

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve Armor\_ID from ARMOR table;

Retrieve Character\_Name related to the Armor\_ID from PLAYER\_CHARACTER table;

Return Enemy\_ID, Item\_ID, Drop\_Rate From EQUIPPED\_ARMOR table;

End

EQUIPPED\_WEAPON Retrieval

Description: Get all values from Equipped\_Weapon present in the table to which the user login is assigned.

Input:

User login name

Output: Weapon\_ID, Character\_Name

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve Weapon\_ID from EQUIPPED\_ARMOR table;

Retrieve Character\_Name related to the Weapon\_ID from WEAPON table;

Return Weapon\_ID, Character\_Name From EQUIPPED\_WEAPON table;

End

ITEM Retrieval

Description: Get all values from ITEM present in the table to which the user login is assigned.

Input:

User login name

Output: Item\_ID, Item\_Name, HP\_MP, Effect\_Remove

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve Item\_ID from ITEM table;

Retrieve Item\_Name related to the Item\_ID from ITEM table;

Retrieve HP\_MP related to the Item\_ID from ITEM table;

Retrieve Effect\_Remove related to the Item\_ID from ITEM table;

Return Item\_ID, Item\_Name, HP\_MP, Effect\_Remove From ITEM table;

End

NPC Retrieval

Description: Get all values from NPC present in the table to which the user login is assigned.

Input:

User login name

Output: NPC\_ID, NPC\_Name, Stats, Element

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve NPC\_ID from ITEM table;

Retrieve NPC\_Name related to the NPC\_ID from NPC table;

Retrieve Stats related to the NPC\_ID from NPC table;

Retrieve Element related to the NPC\_ID from ELEMENT table;

Return NPC\_ID, NPC\_Name, Stats, Element From NPC table;

End

NPC\_HAS\_ABILITY Retrieval

Description: Get all values from NPC present in the table to which the user login is assigned.

Input:

User login name

Output: NPC\_ID, Ability\_ID

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve NPC\_ID from NPC table;

Retrieve Ability\_ID related to the NPC\_ID from ABILITY table;

Return NPC\_ID, Ability\_ID From NPC\_HAS\_ABILITY table;

End

NPC\_IN\_REGION Retrieval

Description: Get all values from NPC present in the table to which the user login is assigned.

Input:

User login name

Output: NPC\_ID, Region\_ID

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve NPC\_ID from NPC table;

Retrieve Region\_ID related to the NPC\_ID from REGION table;

Return NPC\_ID, Region\_ID From NPC\_IN\_REGION table;

End

NPC\_IN\_TOWN Retrieval

Description: Get all values from NPC present in the table to which the user login is assigned.

Input:

User login name

Output: NPC\_ID, Town\_ID

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve NPC\_ID from NPC table;

Retrieve Town\_ID related to the NPC\_ID from TOWN table;

Return NPC\_ID, Town\_ID From NPC\_IN\_TOWN table;

End

PLAYER\_CHARACTER Retrieval

Description: Get all the PLAYER\_CHARACTER values present in the table.

Input:

User login name

Output:

Character\_Name, Class\_Type, Race\_Type, Gender, Char\_Level, Account\_ID

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve Character\_Name From PLAYER\_CHARACTER table;

Retrieve Class\_Type related to Character\_Name from CHARACTER\_CLASS\_DATA table;

Retrieve Race\_Type related to Character\_Name from CHARACTER\_RACE\_DATA table;

Retrieve Gender related to Character\_Name from PLAYER\_CHARACTER table;

Retrieve Char\_Level related to Character\_Name from PLAYER\_CHARACTER table;

Retrieve Account\_ID related to Character\_Name from ACCOUNT table;

Return Character\_Name, Class\_Type, Race\_Type, Gender, Char\_Level, Account\_ID From PLAYER\_CHARACTER table;

End

REGION Retrieval

Description: Get all values from REGION present in the table to which the user login is assigned.

Input:

User login name

Output: Region\_ID, Region\_Name, Region\_Description

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve Region\_ID from REGION table;

Retrieve Region\_Name related to the REGION\_ID from REGION table;

Retrieve Region\_Description related to the REGION\_ID from REGION table;

Return NPC\_ID, Region\_ID, Region\_Description From REGION table;

End

TOWN Retrieval

Description: Get all values from TOWN present in the table

Input:

User login name

Output: Town\_ID, Town\_Name, Region\_ID

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve Town\_ID from TOWN table;

Retrieve Town\_Name related to the TOWN\_ID from TOWN table;

Retrieve Region\_ID related to the Town\_ID from REGION table;

Return Town\_ID, Town\_Name, Region\_ID From TOWN table;

End

WEAPON Retrieval

Description: Get all the WEAPON values present in the Table

Input:

User login name

Output:

Weapon\_ID, Weapon\_Name, Weapon\_Type, Handed, Stats, Stats\_Effect, Element

Begin

If not connect to Database, then

DatabaseConnection.ConnectToDB;

Retrieve Weapon\_ID related to the user login from WEAPON table;

Retrieve Weapon\_Name related to Weapon\_ID from WEAPON table;

Retrieve Weapon\_Type related to Weapon\_ID from WEAPON table;

Retrieve Handed related to Weapon\_ID from WEAPON table;

Retrieve Stats related to Weapon\_ID from WEAPON table;

Retrieve Stats\_Effect related to Weapon\_ID from WEAPON table;

Retrieve Element related to Weapon\_ID from ELEMENT table;

Return Weapon\_ID, Weapon\_Name, Weapon\_Type, Handed, Stats, Stats\_Effect, Element from WEAPON table;

End

**Developer Configure Function Set**

Developer Display Tables

Description: Get the Developer Configure tables to which the login user is assigned

Input: Developer User login name

Output: ABILITY, CHARACTER\_CLASS\_DATA, CHARACTER\_HAS\_ABILITY, CHARACTER\_RACE, DATA, ELEMENT, ELEMENT\_WEAKNESS\_AND\_RESISTANCES, ENEMY, ENEMY\_ARMOR\_DROP, ENEMY\_HAS\_ABILITY, ENEMY\_IN\_REGION, ENEMY\_ITEM\_DROP, ENEMY\_WEAPON\_DROP, NPC, NPC\_HAS\_ABILITY, NPC\_IN\_REGION, NPC\_IN\_TOWN, REGION, TOWN, WEAPON TABLES

Begin

If not connected to Database then,

DatabaseAccess.ConnectToDB;

Retrieve Authorized Developer Tables related to Developer User Login from Developer User Login Tables;

Display ABILITY, CHARACTER\_CLASS\_DATA, CHARACTER\_HAS\_ABILITY, CHARACTER\_RACE, DATA, ELEMENT, ELEMENT\_WEAKNESS\_AND\_RESISTANCES, ENEMY, ENEMY\_ARMOR\_DROP, ENEMY\_HAS\_ABILITY, ENEMY\_IN\_REGION, ENEMY\_ITEM\_DROP, ENEMY\_WEAPON\_DROP, NPC, NPC\_HAS\_ABILITY, NPC\_IN\_REGION, NPC\_IN\_TOWN, REGION, TOWN, WEAPON TABLES;

Return ABILITY, CHARACTER\_CLASS\_DATA, CHARACTER\_HAS\_ABILITY, CHARACTER\_RACE, DATA, ELEMENT, ELEMENT\_WEAKNESS\_AND\_RESISTANCES, ENEMY, ENEMY\_ARMOR\_DROP, ENEMY\_HAS\_ABILITY, ENEMY\_IN\_REGION, ENEMY\_ITEM\_DROP, ENEMY\_WEAPON\_DROP, NPC, NPC\_HAS\_ABILITY, NPC\_IN\_REGION, NPC\_IN\_TOWN, REGION, TOWN, WEAPON TABLES;

End

Developer Create Table

Description: Add Tables to Project\_RPG Database

Input: Developer User login name

Output: Developer Create Table Module

Begin

If not connected to Database then,

DatabaseAccess.ConnectToDB;

Retrieve Authorized Developer Tables related to Developer User Login from Developer User Login Tables;

Display Developer Create Table Module;

Return Developer Create Table Module;

End;

Developer Insert Data

Description: Allow Developer insertion of Data into Existing Tables

Input: New Data into Tables

Output: Existing tables with new Data

Begin

If not connected to Database then,

DatabaseAccess.ConnectToDB;

Retrieve Authorized Developer Tables related to Developer User Login from Developer User Login Tables;

Display Developer Insert Data Module;

Return Developer Insert Data Module;

End;

**UserAccount Configure Function Set**

UserAccount Display Tables

Description: Get the UserAccount Configure tables to which the login user is assigned

Input: UserAccount Login Name

Out: UserAccount Login Name

Begin

If not connected to Database then,

DatabaseAccess.ConnectToDB;

Retrieve Authorized UserAccount Tables related to UserAccount from UserAccount Login Tables;

Display UserAccount Table Info;

UserAccount Edit Tables

Description: Get the User Account Configure tables to which the login user is assigned

Input: Developer User login name

Output: Player\_Character, Equipped\_Weapon, Equipped\_Armor

Begin

If not connected to Database then,

DatabaseAccess.ConnectToDB;

Retrieve Authorized User Account Tables related to User Account Login from User Account Tables;

Display Authorized Player\_Character, Equipped\_Weapon and Equipped\_Armor

Return Authorized User Login

**GameSearch Configure Function Set**

GameSearch Display Tables

Description: Get the Developer Configure tables to which the login user is assigned

Input: Developer User login name

Output: ABILITY, CHARACTER\_CLASS\_DATA, CHARACTER\_HAS\_ABILITY, CHARACTER\_RACE, DATA, ELEMENT\_WEAKNESS\_AND\_RESISTANCES, ENEMY, ENEMY\_ARMOR\_DROP, ENEMY\_HAS\_ABILITY, ENEMY\_IN\_REGION, ENEMY\_ITEM\_DROP, ENEMY\_WEAPON\_DROP, NPC, NPC\_HAS\_ABILITY, NPC\_IN\_REGION, NPC\_IN\_TOWN, REGION, TOWN, WEAPON TABLES;

Begin

If not connected to Database then,

DatabaseAccess.ConnectToDB;

Retrieve Authorized Developer Tables related to Developer User Login from Developer User Login Tables;

Display ABILITY, CHARACTER\_CLASS\_DATA, CHARACTER\_HAS\_ABILITY, CHARACTER\_RACE, DATA, ELEMENT\_WEAKNESS\_AND\_RESISTANCES, ENEMY, ENEMY\_ARMOR\_DROP, ENEMY\_HAS\_ABILITY, ENEMY\_IN\_REGION, ENEMY\_ITEM\_DROP, ENEMY\_WEAPON\_DROP, NPC, NPC\_HAS\_ABILITY, NPC\_IN\_REGION, NPC\_IN\_TOWN, REGION, TOWN, WEAPON TABLES;

Return ABILITY, CHARACTER\_CLASS\_DATA, CHARACTER\_HAS\_ABILITY, CHARACTER\_RACE, DATA, ELEMENT\_WEAKNESS\_AND\_RESISTANCES, ENEMY, ENEMY\_ARMOR\_DROP, ENEMY\_HAS\_ABILITY, ENEMY\_IN\_REGION, ENEMY\_ITEM\_DROP, ENEMY\_WEAPON\_DROP, NPC, NPC\_HAS\_ABILITY, NPC\_IN\_REGION, NPC\_IN\_TOWN, REGION, TOWN, WEAPON TABLES;

End

**Report Generating Function Set**

Game Search Data Interface Module Detail

**Home Page**

Name: Game Data Search.html

Description: Game Data Search

Input Variable:

None

Output Variable:

None

Link to:

|  |  |
| --- | --- |
|  | Page name |
| 1 | Items |
| 2 | Weapons |
| 3 | Armor |
| 4 | Abilities |
| 5 | Enemy |
| 6 | NPC |
| 7 | Towns |
| 8 | Regions |

**Items Page**

Name: Items Data Page.html

Description: Items Data Page

Input Variable:

None

Output Variable:

|  |  |  |
| --- | --- | --- |
|  | Description | Type |
| 1 | Text List of Items | String |

**Detailed Items Page Info**

Name: Detailed Item Page.html

Description: Detailed Items Data Page

Input Variable:

None

Output Variable:

|  |  |  |
| --- | --- | --- |
|  | Description | Type |
| 1 | Item Name | String |
| 2 | HP\_MP | Binary to Decimal |
| 3 | Effect\_Remove | Binary to Decimal |

**Weapon Page**

Name: Weapon Data Page.html

Description: Weapon Data Page

Input Variable:

None

Output Variable:

|  |  |  |
| --- | --- | --- |
|  | Description | Type |
| 1 | Text List of Weapons | String |

**Detailed Weapon Page Info**

Name: Detailed Weapon Page.html

Description: Detailed Weapon Data Page

Input Variable:

None

Output Variable:

|  |  |  |
| --- | --- | --- |
|  | Description | Type |
| 1 | Weapon Name | String |
| 2 | Weapon Type | String |
| 3 | Weapon Handed | String |
| 4 | Stats | Binary to Decimal |
| 5 | Element | String |

**Armor Page**

Name: Armor Data Page.html

Description: Armor Data Search

Input Variable:

None

Output Variable:

|  |  |  |
| --- | --- | --- |
|  | Description | Type |
| 1 | Text List of Armor | String |

**Detailed Armor Page**

Name: Detailed Armor Data Page.html

Description: Detailed Armor Data

Input Variable:

None

Output Variable:

|  |  |  |
| --- | --- | --- |
|  | Description | Type |
| 1 | Armor Name | String |
| 2 | Armor Type | String |
| 3 | Armor Material | String |
| 4 | Stats | Binary to Decimal |
| 5 | Element | String |

**Abilities Page**

Name: Abilities Data Page.html

Description: Abilities Data Search

Input Variable:

None

Output Variable:

|  |  |  |
| --- | --- | --- |
|  | Description | Type |
| 1 | Text List of Abilities | String |

**Detailed Abilities Page**

Name: Detailed Abilities Data Page.html

Description: Detailed Abilities Data

Input Variable:

None

Output Variable:

|  |  |  |
| --- | --- | --- |
|  | Description | Type |
| 1 | Ability Name | String |
| 2 | Ability Type | String |
| 3 | Ability Power | Integer |
| 4 | Effect\_Apply | Binary to Decimal |
| 5 | Effect\_Remove | Binary to Decimal |
| 6 | Element | String |

**Enemy Page**

Name: Enemy Data Page.html

Description: Enemy Data Search

Input Variable:

None

Output Variable:

|  |  |  |
| --- | --- | --- |
|  | Description | Type |
| 1 | Text List of Enemies | String |

**Detailed Enemy Page**

Name: Detailed Enemy Data Page.html

Description: Detailed Enemy Data

Input Variable:

None

Output Variable:

|  |  |  |
| --- | --- | --- |
|  | Description | Type |
| 1 | Enemy Name | String |
| 2 | Stats | Binary to Decimal |
| 3 | Element | String |

**NPC Page**

Name: NPC Data Page.html

Description: NPC Data Search

Input Variable:

None

Output Variable:

|  |  |  |
| --- | --- | --- |
|  | Description | Type |
| 1 | Text List of NPC | String |

**Detailed NPC Page**

Name: Detailed NPC Data Page.html

Description: Detailed NPC Data

Input Variable:

None

Output Variable:

|  |  |  |
| --- | --- | --- |
|  | Description | Type |
| 1 | NPC Name | String |
| 2 | Stats | Binary to Decimal |
| 3 | Element | String |

**Towns Page**

Name: Town Data Page.html

Description: Town Data Search

Input Variable:

None

Output Variable:

|  |  |  |
| --- | --- | --- |
|  | Description | Type |
| 1 | Text List of Towns | String |

**Detailed Towns Page**

Name: Town Data Page.html

Description: Town Data Search

Input Variable:

None

Output Variable:

|  |  |  |
| --- | --- | --- |
|  | Description | Type |
| 1 | Detailed Town Info | String |

**Regions Page**

Name: Regions Data Page.html

Description: Regions Data Search

Input Variable:

None

Output Variable:

|  |  |  |
| --- | --- | --- |
|  | Description | Type |
| 1 | Detailed Town Info | String |

**Detailed Regions Page**

Name: Regions Data Page.html

Description: Detailed Regions Data Search

Input Variable:

None

Output Variable:

|  |  |  |
| --- | --- | --- |
|  | Description | Type |
| 1 | Region Name | String |
| 2 | Region Description | String |

**MAIN STATS** (IN ORDER):

* STA – Stamina
* STR – Strength
* AGI – Agility
* DEF – Defense
* MIN – Mind
* INT – Intellect
* EVA – Evasion
* ACC – Accuracy

Character\_Race\_Data & Character\_Class\_Data:

* The stats here is a 8 bytes, each byte represents a stat. Therefore, the max amount for each stat is 255,

Example,

0011 0101 0111 0111 0111 0101 0101 0101 0011 0101 0111 0111 0111 0101 0101 0101

STA = 53 STR = 119 AGI = 117 DEF = 85 MIN = 53 INT = 119 EVA = 117 ACC = 85

* Depending on the Race and Class chosen that will determine the base stats of a character. The stats corresponding to your characters race and class will be added to create the base stats of a player’s character

Weapon and Armor:

* Stats works the same as Character\_Race\_Data & Character\_Class\_Data.
* Stats\_Effect is 1 byte, each bit correspond to each of the stat, so if 1 then you add the amount else if 0 then you subtract the amount. This will give the effect of sacrificing certain stats for others.

NPC and Enemies:

* They don’t have weapons nor armor (other than graphically), but they do have abilities. Therefore, the stats on here will be their complete stats. (All their stats will be determined just from this.

**Calculating A Character’s Stats**

1. Read the stats for the character’s race, read byte by byte. Each byte corresponds to a stat, in the order given above. Calculate the value of the byte, and store it in an array that holds each stat separately.
2. Do the same for the stats of the character’s class, expect you add them up to each corresponding stat in the array.
3. After Collecting all the base stats, you multiply each stat by (1 + (character’s level / max level)).
4. Read the stats of each weapon and armor equipped on the character. Then you check the Stats\_Effect, which is 1 byte. Each bit in the Stats\_Effect bit corresponds to a stat, if the corresponding bit is 0 then subtract the amount of the stat given by the armor or weapon. Else if the bit is 1 then you add.

**Calculating an Enemy and NPC Stats**

1. Read the stats for the character’s race, read byte by byte. Each byte corresponds to a stat, in the order given above. Calculate the value of the byte, and store it in an array that holds each stat separately.
2. Do the same for the stats of the character’s class, expect you add them up to each corresponding stat in the array.

**Effect\_Remove or Effect\_Apply**, 1 Byte, each bit represents an effect it heals, in the order:

* + Poisoned
  + Slow
  + Blind
  + Paralyzed
  + Sapped
  + Frozen
  + Sleep
  + Confused

Item:

* On item table, we have HP\_MP, 2 Bytes. First byte is for amount of HP healed and Second byte is for amount of MP restored.

Ability:

* Effect\_Apply and Effect\_Remove works the same as above, plus the ability has a type, and a power level of one digit between 1-9;

**Values Calculated by Stats**

* Physical Attack – This is a regular attack using a weapon. This is calculated by using the STR stat. ACC is also used to determine if the attack is critical.
* Range Attack- This is an attack using a range weapon. This is calculated by using the AGI stat. ACC is also used to determine if the attack is critical.
* Physical Block – This is the amount the defending character blocks from an enemies Physical Attack. This is calculated by using the DEF.
* Range Block – This is the amount the defending character blocks from an enemies Range Attack. This is calculated by using the DEF and EVA.
* Miss Probability – This is the probability of a character dodging any attack and avoiding any damage. This is calculated using EVA.
* Hit Probability – This is the probability of a character landing an attack. This is calculated using ACC.
* Magical Attack – This is the amount of damage caused by a magical type ability. It uses MIN, ability power level, and ability element.
* Magical Heal - This is the amount of healing caused by a magical type ability. It uses INT, ability power level, and ability element.
* Magical Defense – This is the amount of damage blocked. This is calculated by using both INT and MIN.
* Health Points – The amount of health points. It is calculated by using the STA stat.
* Magic Points – The amount of magic points. It is calculated by using the both MIN and INT.

**Normalization of Schemas:**

* **Account** (Account ID, Email, Password, First Name, Last Name, Card Number, Security Code, Expiration Date, Card Owner Name)
  + Account was decomposed into 2 different tables.
  + **ACCOUNT**: Holds account info, email is unique,
    - **Account** (Account ID, Email, Password, First Name, Last Name)
    - Functional dependencies: BCNF
      * Account\_ID -> Email, Email -> Account\_ID
      * Email -> Password, First\_Name, Last\_Name
  + **ACCOUNT\_BILLING**: Holds account billing info, multiple cards could be registered for each account,
    - **Account\_Billing** (Account ID, Card Number, Security Code, Expiration Date, Card Owner Name)
    - Functional dependencies: BCNF
      * Only one trivial dependencies, ALL\_ATTRIBUTES -> ALL\_ATTRIBUTES
* **Character** (Character Name, Class, Race, Gender, Faction, Base Stats, Level, Account\_ID)
  + The Character table was decomposed into 3 different tables, each holds specific types of details about character types.
  + **PLAYER\_CHARACTER**: This tables hold basic information of a player's character. Since each Character\_Name is unique, Character\_Name is associated with the player's Account\_ID. However, the Account\_ID is associated with multiple characters owned by the player.
    - **Player\_Character** (Character Name, Class\_Type, Race\_Type, Gender, Char\_Level, Account\_ID)
    - Functional dependencies: BCNF
      * Character\_Name -> Class\_Type, Race\_Type, Gender, Char\_Level, Account\_ID
  + **CHARACTER\_CLASS\_DATA**: This tables holds the base stats your character has for its class.
    - **Character\_Class\_Data** (Class\_Type, Stats)
    - Functional dependencies: BCNF
      * Class\_Type -> Stats
  + **CHARACTER\_RACE\_DATA**: This table holds the base stats your character has for its race. Each race is in either faction, Alliance or Swarm.
    - **Character\_Race\_Data** (Race\_Type, Stats, Faction)
    - Functional dependencies: BCNF
      * Race\_Type -> Stats, Faction
* **Player Inventory** (Inventory ID, Size, Character\_Name)
  + Kept the same; no changes (YET).
  + Functional dependencies: BCNF
    - Inventory\_ID -> Character\_Name, Size
* **Item** (Item ID, Item Name, Stats)
  + Attributes in this table have been changed.
  + **ITEM**: Holds the data for the existing items in the game.
    - **Item** (Item\_ID, Item\_Name, HP\_MP, Effect\_Remove)
    - Functional dependencies: BCNF
      * Item\_ID -> Item\_Name
      * Item\_Name -> HP\_MP, Effect\_Remove
* **Weapon** (Weapon ID, Weapon Name, W\_Type, Stats)
  + Attributes in this table have been changed.
  + **WEAPON**: Holds the data for the existing weapons in the game.
    - **Weapon**(Weapon ID, Weapon\_Name, W\_Type, Handed, Stats, Stats\_Effect, Elements)
    - Functional dependencies: BCNF
      * Weapon\_ID -> Weapon\_Name
      * Weapon\_Name -> Weapon\_Type, Handed, Stats, Stats\_Effect, Element
* **Armor** (Armor ID, Armor Name, A\_Type, A\_Material, Stats)
  + Attributes in this table have been changed.
  + **ARMOR**: Holds the data for the existing armor in the game.
    - **Armor**(Armor ID, Armor\_Name, A\_Type, A\_Material, Stats, Stats\_Effect, Elements)
    - Functional dependencies: BCNF
      * Armor \_ID -> Armor \_Name
      * Armor \_Name -> Armor \_Type, Armor\_Material, Stats, Stats\_Effect, Element
* **Ability** (Ability ID, Ability Name, Element, Damage, Heal, Stat Effect, Status Effect)
  + Attributes in this table have been changed.
  + **ABILITY**: Holds the data for the existing abilities in the game.
    - **Ability**(Ability ID, Ability\_Name, Ability\_Type, Power\_Level, Effect\_Apply, Effect\_Remove, Element)
    - Functional dependencies: BCNF
      * Ability\_ID -> Ability\_Name
      * Ability\_Name -> Ability\_Type, Power\_Level, Effect\_Apply, Effect\_Remove, Element
* **Enemy** (Enemy ID, Enemy Name, Element, Stats)
  + This table has been kept the same; no changes.
  + Functional dependencies: BCNF
    - * Enemy\_ID -> Enemy\_Name
      * Enemy\_Name -> Stats, Element
* **NPC** (NPC ID, NPC Name, Element, Stats)
  + This table has been kept the same; no changes.
  + Functional dependencies: BCNF
    - * NPC\_ID -> NPC\_Name
      * NPC\_Name -> Stats, Element
* **Town** (Town ID, Town Name, Region\_ID)
  + This table has been kept the same; no changes.
  + Functional dependencies: BCNF
    - * Town\_ID -> Town\_Name, Region\_ID
* **Region** (Region ID, Region Name, Region Description)
  + This table has been kept the same; no changes.
  + Functional dependencies: BCNF
    - * Region\_ID -> Region\_Name
      * Region\_Name -> Region\_Name, Region\_Description
* **Equipped\_Weapon** (Character Name, Weapon\_ID, Hand)
  + This table has been kept the same; no changes.
  + Functional dependencies: BCNF
    - * Character\_Name -> Weapon\_ID, Hand
      * Hand might be removed.
* **Equipped\_Armor** (Armor ID, Character Name)
  + This table has been kept the same; no changes.
  + Functional dependencies: BCNF
    - * Characters could be associated with multiple armor, but can only wear one of each type. As well as, each armor could be associated with many characters. If Armor type is not added as an attribute, then no dependency in this table, only trivial dependency. Else:
        + Character\_Name -> Armor\_ID, Armor\_Type
        + Armor\_Type might be added.
* **Item\_in\_Inventory** (Inventory ID, Item ID, Amount)
  + Kept the same; no changes (YET).
  + Inventory\_ID could be associated with multiple items, as well as, Item\_ID could be associated with multiple Invenotry\_ID. No functional dependency except the trivial, is found here.
* **Weapon\_in\_Inventory** (Inventory ID, Weapon\_ID, Amount)
  + Kept the same; no changes (YET).
  + Inventory\_ID could be associated with multiple items, as well as, Weapon\_ID could be associated with multiple Invenotry\_ID. No functional dependency except the trivial, is found here.
* **Armor\_in\_Inventory** (Inventory ID, Armor ID, Amount)
  + Kept the same; no changes (YET).
  + Inventory\_ID could be associated with multiple items, as well as, Armor\_ID could be associated with multiple Invenotry\_ID. No functional dependency except the trivial, is found here.
* **Enemy\_Item\_Drop** (Enemy ID, Item ID, Drop\_Rate)
  + This table has been kept the same; no changes.
  + Functional dependencies: BCNF
    - * Enemy\_ID, Item\_ID -> Drop\_Rate
* **Enemy\_Weapon\_Drop** (Enemy ID, Weapon ID, Rate)
  + This table has been kept the same; no changes.
  + Functional dependencies: BCNF
    - * Enemy\_ID, Weapon\_ID -> Drop\_Rate
* **Enemy\_Armor\_Drop** (Enemy ID, Armor ID, Rate)
  + This table has been kept the same; no changes.
  + Functional dependencies: BCNF
    - * Enemy\_ID, Armor\_ID -> Drop\_Rate
* **Character\_Has\_Ability** (Ability Id, Character Name)
  + This table has been kept the same; no changes. Since this table only contains two attributes, it is know to already be in BCNF.
* **Enemy\_Has\_Ability** (Ability Id, Enemy ID)
  + This table has been kept the same; no changes. Since this table only contains two attributes, it is know to already be in BCNF.

**s**

* **NPC\_Has\_Ability** (Ability Id, NPC ID)
  + This table has been kept the same; no changes. Since this table only contains two attributes, it is know to already be in BCNF.
* **Enemy\_In\_Region** (Enemy ID, Region ID, Rarity)
  + This table has been kept the same; no changes.
  + Functional dependencies: BCNF
    - * Enemy\_ID, Region\_ID -> Rarity
      * Rarity might be removed.
* **NPC\_In\_Region** (NPC ID, Region ID)
  + This table has been kept the same; no changes. Since this table only contains two attributes, it is know to already be in BCNF.
* **NPC\_In\_Town** (NPC ID, Town ID)
  + This table has been kept the same; no changes. Since this table only contains two attributes, it is know to already be in BCNF.
* **Town\_In\_Region** (Town, Region)
  + This table has been kept the same; no changes. Since this table only contains two attributes, it is know to already be in BCNF.