Austin Thompson

Alexander Schneider

Use Cases of Gacha Game

1. Signup user for new account in database
2. Admin - add/remove card from table of active cards.
3. Admin - add ‘pack’ of cards
4. Admin - remove an entire pack when it’s time frame has expired
5. Add card to database of user inventory(User buying cards)
6. User - add payment information
7. User - purchase packs of new cards, check for duplicates
8. User - trade in card for in-game currency

List of Screens

1. Initial login screen
   1. If they are a new user allow them to make an account
   2. Allow a current user to sign in to an existing account
   3. Allow developers to login with special permissions
2. If admin, take them to a menu where they can add/delete specific cards/packs
   1. Add/Delete specific cards in the game
   2. Delete a user’s account for banning purposes
3. If user,
   1. Buy cards
   2. Assemble a five card deck
   3. Allow them to view the items in their inventory
   4. Trade in cards for in game currency

We ended up not being able to implement some of the use cases described above, as without a GUI the process would be tedious to perform using text based interfaces. We also decided that some of these use cases would be out of scope of the project and should be left up to the game developer. Rather than focusing on game implementation we decided to focus on backend development, not player versus player interaction, or creating fully fledged decks.

The purpose of this project is to create a database that is the foundation of a gacha game, specifically a card-based game (e.g. Hearthstone, Pokémon). We designed the database framework for a game designer to create a functioning gacha game. this includes an inventory system, user account information, store, and payment information. Also included are key user methods necessary for playing the game, such as buying/trading in cards, viewing inventory, and creating a deck and functions for administrator accounts to manage the game, including . adding/deleting cards, and deleting an account. Actual player vs player gameplay was not implemented because it is outside of the foundational framework that we were assigned, this leaves the game designer free reign over how they wish to implement it, rather than having to work around our implementation.

TABLES/KEYS The User table holds the basic information, username(which is unique and used as the primary key), the user’s salt and hash for login, a foreign key to the payment information table.

The PaymentInfo table includes the credit card information, the key is a compound the card number and the account associated with it, as multiple accounts could be associated with the same card. Billing address could have been separated to another table, but we did not as it is not that frequently a card will be associated with multiple accounts. The Inventory table is a junction table consisting of foreign keys from the user table and the card table. The way inventory works is that where all rows where the username is located, that user possesses the card identified by the cardID in the same row.

The Rarity table was created to remove repeating costs of cards in the Card table, the primary key corresponds to the rarity, and the associated cost is what the player gets back when they trade the card in for in game currency.

The card table has a primary key of the cardID number, it also contains the card’s description, special ability, hp, rarity, packID,and name. The rarity corresponds to percentages of cards: 3 = 1%, 2 = 4%, 1 = 20%, 0 = 75%.

The Pack table has a packID, pack description, and pack name. A pack corresponds to a themed set of 100 cards that a user can select to buy random cards from.

The Deck table has a foreign key consisting of a guaranteed unique id, and the card id, and also contains the name of the deck. The deck is differentiated by the assigned deck name and the individual cards are differentiated by the cards from the users inventory and the generated deckID.

LANGUAGE CHOICE We chose to use python because, between the two of us, we have little experience with it, and we wanted to use this project as an opportunity to improve our skills with it, and the connection to the database is much easier to manage.

GROUPWORK For most of the project we practiced pair programming, as it was easier for us to learn and remember the syntax as a pair. This led to us using Github less but, allowed us to communicate with much better efficiency.