Design Document for Bash Zones

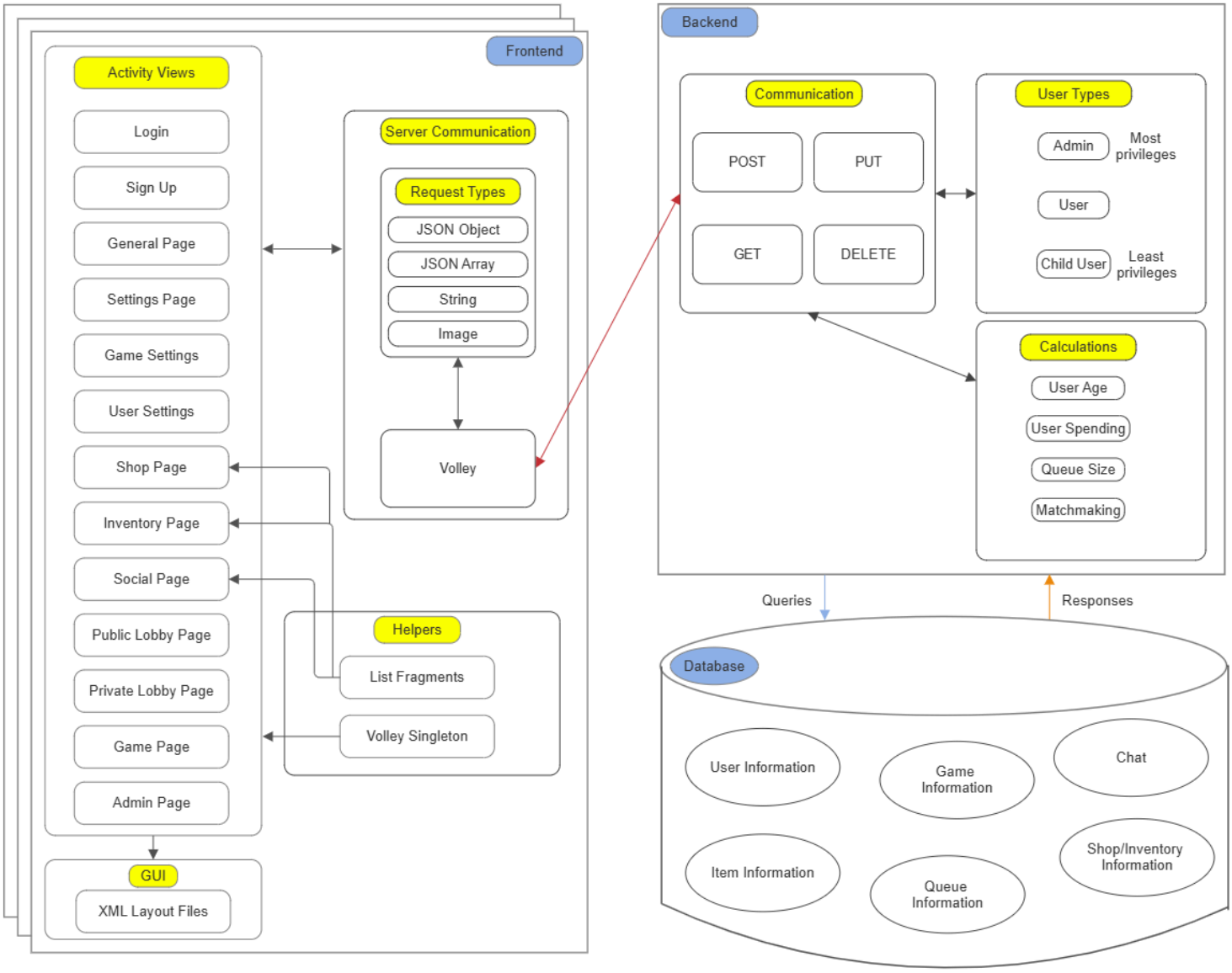
Group 4_Pratik_6

Carson Ferree: 25% contribution

Gabriel Navarro: 25% contribution

Carter Scott: 25% contribution

Joseph Sheets: 25% contribution


Frontend
Activity Views
Login
Sign Up
General Page
Settings Page
Game Settings
User Settings
Shop Page
Inventory Page
Social Page
Public Lobby Page
Private Lobby Page
Game Page
Admin Page
GUI
XML Layout Files
Server Communication
Request Types
JSON Object
JSON Array
String
Image
Volley
Helpers
List Fragments
Volley Singleton
Backend
Communication
POST
PUT
GET
DELETE
User Types
Admin
Most privileges
User
Child User
Least privileges
Calculations
User Age
User Spending
Queue Size
Matchmaking
Queries
Responses
Database
User Information
Game Information
Chat
Item Information
Queue Information
Shop/Inventory Information

Frontend

Sign Up (User, Child User)

- Sign up page has the following elements for information followed by a button:
    - 3 EditText Objects: Username, Password, Email
    - DatePicker: Birthday
- Upon tapping the button “Sign Up,” the values are sent to the server as a POST request. Users are returned to login page.

Shopping (Admin, User, Child User)

- Uses buttons to control scrolling fragments by type of cosmetic item (Hats, Banners, or Tags). Selected type has its items displayed by using GET request when shop is opened.
- Purchasing items (button, then confirmation page) uses a GET request to check user’s currency, then POST request to send item’s information to be connected to user’s inventory.

Friends/Blocked Users (Admin, User)

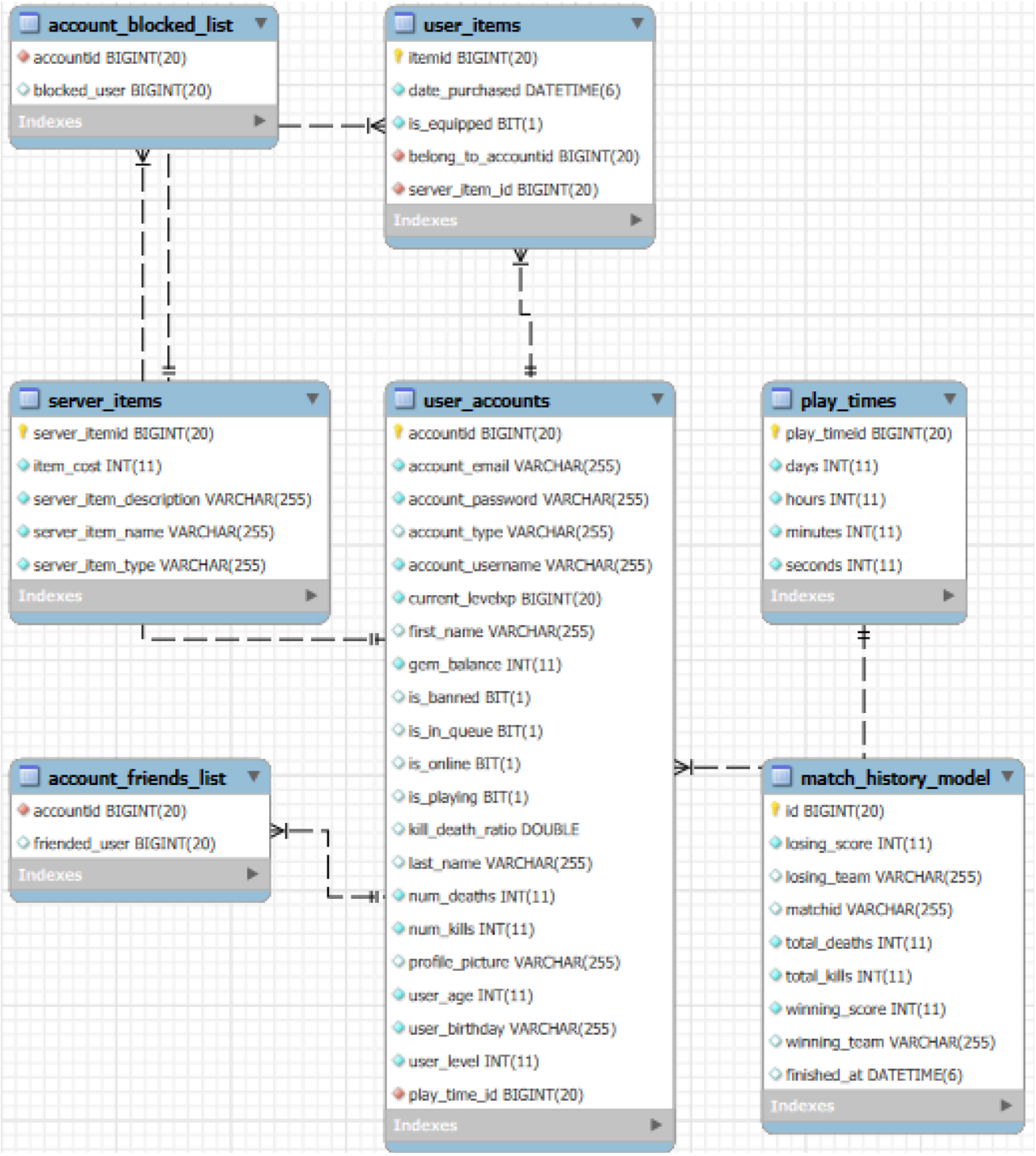
- GET request to access user’s friends, POST requests for blocked users and accepted friend requests.

Backend

The backend receives sign-up information via HTTP POST requests containing username, password, email, and birthday. When a user attempts a purchase, the backend processes a GET request to verify the user's available currency. Upon successful validation, a POST request updates the user's inventory with the purchased item and deducts the corresponding currency amount.

The backend includes functionality to handle friend and blocked user relationships through dedicated RESTful endpoints. At match completion, the game server sends a payload (MatchEndPayloadDTO) back to the backend via a RESTful endpoint. This payload includes the match ID and involved player IDs, upon which the backend updates each user's status to reflect that they are no longer actively playing.

Scheduled health checks periodically test the connection status of the game server via WebSocket. If the server becomes unreachable, the backend automatically resets the status of all actively playing users, ensuring system consistency and reliability.


account_blocked_list
accountid BIGINT(20)
blocked_user BIGINT(20)
Indexes
user_items
itemid BIGINT(20)
date_purchased DATETIME(6)
is_equipped BIT(1)
belong_to_accountid BIGINT(20)
server_item_id BIGINT(20)
Indexes
server_items
server_itemid BIGINT(20)
item_cost INT(11)
server_item_description VARCHAR(255)
server_item_name VARCHAR(255)
server_item_type VARCHAR(255)
Indexes
user_accounts
accountid BIGINT(20)
account_email VARCHAR(255)
account_password VARCHAR(255)
account_type VARCHAR(255)
account_username VARCHAR(255)
current_levelxp BIGINT(20)
first_name VARCHAR(255)
gem_balance INT(11)
is_banned BIT(1)
is_in_queue BIT(1)
is_online BIT(1)
is_playing BIT(1)
kill_death_ratio DOUBLE
last_name VARCHAR(255)
num_deaths INT(11)
num_kills INT(11)
profile_picture VARCHAR(255)
user_age INT(11)
user_birthday VARCHAR(255)
user_level INT(11)
play_time_id BIGINT(20)
Indexes
play_times
play_timeid BIGINT(20)
days INT(11)
hours INT(11)
minutes INT(11)
seconds INT(11)
Indexes
account_friends_list
accountid BIGINT(20)
friended_user BIGINT(20)
Indexes
match_history_model
id BIGINT(20)
losing_score INT(11)
losing_team VARCHAR(255)
matchid VARCHAR(255)
total_deaths INT(11)
total_kills INT(11)
winning_score INT(11)
winning_team VARCHAR(255)
finished_at DATETIME(6)
Indexes