Design Document for Sorry! Plus

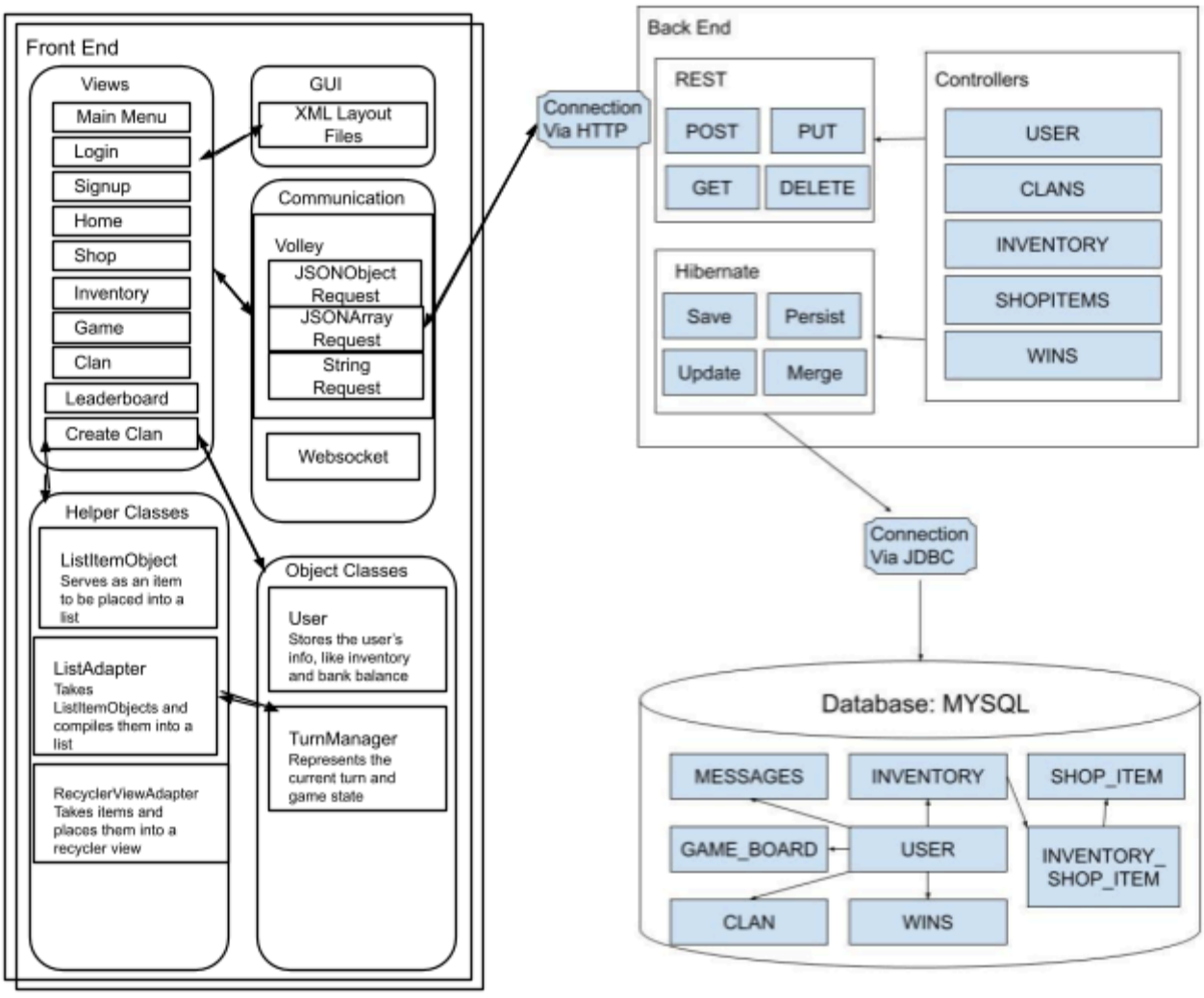
Group MS3_1

Hudson Nebbe: 100% contribution

Jack Olsan: 100% contribution

Nakota Clark: 100% contribution

Rostyslav Sheshenya: 100% contribution


Front End
Views
Main Menu
Login
Signup
Home
Shop
Inventory
Game
Clan
Leaderboard
Create Clan
GUI
XML Layout Files
Communication
Volley
JSONObject Request
JSONArray Request
String Request
Websocket
Helper Classes
ListItemObject
Serves as an item to be placed into a list
ListAdapter
Takes ListItemObjects and compiles them into a list
RecyclerViewAdapter
Takes items and places them into a recycler view
Object Classes
User
Stores the user's info, like inventory and bank balance
TurnManager
Represents the current turn and game state
Connection Via HTTP
Back End
REST
POST
PUT
GET
DELETE
Controllers
USER
CLANS
INVENTORY
SHOPITEMS
WINS
Hibernate
Save
Persist
Update
Merge
Connection Via JDBC
Database: MYSQL
MESSAGES
INVENTORY
SHOP_ITEM
GAME_BOARD
USER
INVENTORY_ SHOP_ITEM
CLAN
WINS

FrontEnd

Module onBindViewHolder (ClanRecyclerViewClass) -

the module inflates each clan item with clan name and level. The following elements are inflated:

- TextView: clanNameLabel
- TextView: clanAvailability (the item is named clanAvailability, however it is used for clan level).

The class SerializableJsonArray -

- This class serializes jSON Arrays. It does so, so json arrays can be transmitted from one activity to another. Even though one can do this with the basic functionality of intents, intents are limited and can only transmit primitive data types. This is why serialization is needed. It is used to transmit files that are not of a primitive type.

The class VolleySingleton -

- This class is used to handle VolleyRequest. VolleyLibrary is a library used in android app development to make communication between frontend and backend easy. It is specifically used to parse JsonArrays which are stored at a backend server. Then this data is decoded and is later displayed on the user's screen.

Backend -

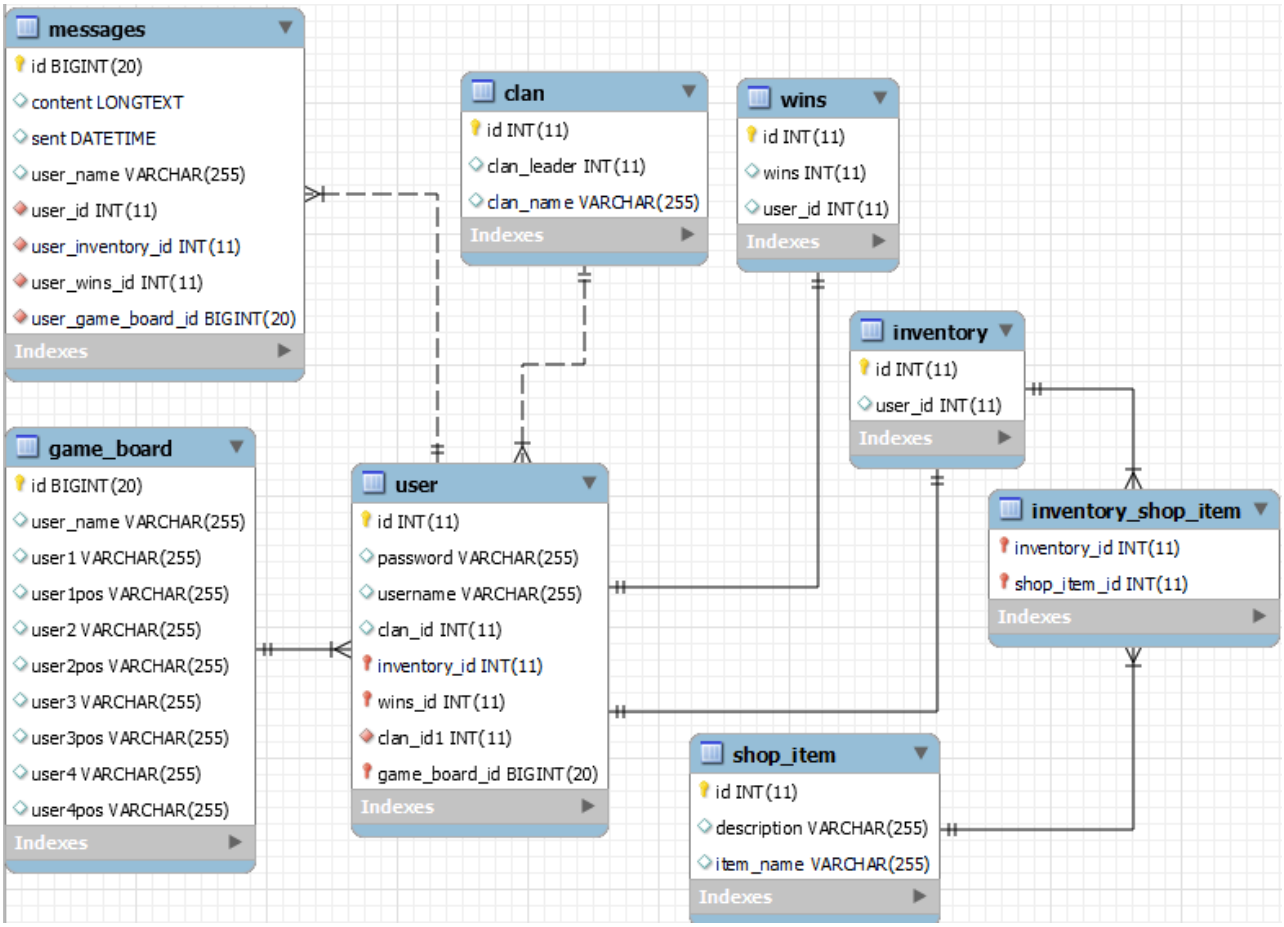
Communication between sides:

We used CRUD operations, and that allowed us to communicate between frontend and our own code. Using JDBC and the TOMCAT server, we are able to make HTTP requests to our frontend using the APIs generated alongside this assignment:

- Post: create new information and save it in the database, whether it is a new account, or a new game instance, this will be used.
- Get: retrieve any information stored on the Database. We have set up quite a few requests, but they cannot access all of the public getter methods that we have created to this date. This is because not all the information should be accessible by the average user.
- Put: put is the real bread and butter here. It does everything that Post does, but we are able to update an entry in the DB if we need to. This means anytime we are changing a score or status in our game, we use Put (we use it a lot)
- Delete: there is not a lot of deletion going on in our game except for after a game is completed the board will be deleted.

Controllers:

The controllers implement the above HTTP methods. We currently have 5, working towards 7 controllers. The more complicated ones are Inventory, which is a many to many connection, and User, which is the lifeblood of the Backend's part of the project.


messages
id BIGINT(20)
content LONGTEXT
sent DATETIME
user_name VARCHAR(255)
user_id INT(11)
user_inventory_id INT(11)
user_wins_id INT(11)
user_game_board_id BIGINT(20)
Indexes
clan
id INT(11)
clan_leader INT(11)
clan_name VARCHAR(255)
Indexes
wins
id INT(11)
wins INT(11)
user_id INT(11)
Indexes
inventory
id INT(11)
user_id INT(11)
Indexes
game_board
id BIGINT(20)
user_name VARCHAR(255)
user1 VARCHAR(255)
user1pos VARCHAR(255)
user2 VARCHAR(255)
user2pos VARCHAR(255)
user3 VARCHAR(255)
user3pos VARCHAR(255)
user4 VARCHAR(255)
user4pos VARCHAR(255)
Indexes
user
id INT(11)
password VARCHAR(255)
username VARCHAR(255)
clan_id INT(11)
inventory_id INT(11)
wins_id INT(11)
clan_id1 INT(11)
game_board_id BIGINT(20)
Indexes
inventory_shop_item
inventory_id INT(11)
shop_item_id INT(11)
Indexes
shop_item
id INT(11)
description VARCHAR(255)
item_name VARCHAR(255)
Indexes