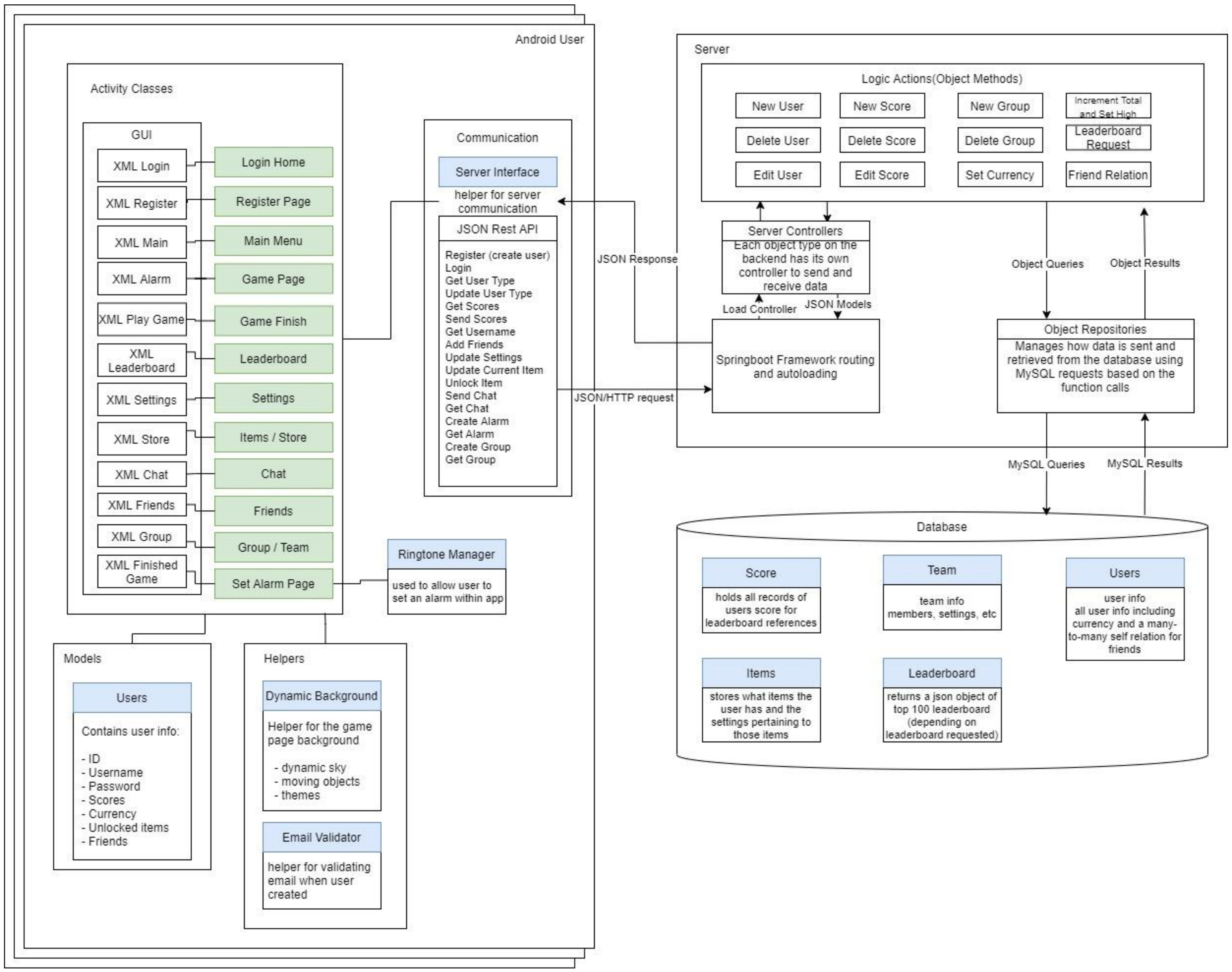
Wake Up Battle

Block Diagram
2_hb_5

David Helmick: 25%
Dawood Ghauri: 25%
Neha Maddal: 25%
Maisy Millage: 25%


Android User
Activity Classes
GUI
XML Login
Login Home
XML Register
Register Page
XML Main
Main Menu
XML Alarm
Game Page
XML Play Game
Game Finish
XML Leaderboard
Leaderboard
XML Settings
Settings
XML Store
Items / Store
XML Chat
Chat
XML Friends
Friends
XML Group
Group / Team
XML Finished Game
Set Alarm Page
Ringtone Manager
used to allow user to set an alarm within app
Communication
Server Interface
helper for server communication
JSON Rest API
Register (create user)
Login
Get User Type
Update User Type
Get Scores
Send Scores
Get Username
Add Friends
Update Settings
Update Current Item
Unlock Item
Send Chat
Get Chat
Create Alarm
Get Alarm
Create Group
Get Group
Models
Users
Contains user info:
- ID
- Username
- Password
- Scores
- Currency
- Unlocked items
- Friends
Helpers
Dynamic Background
Helper for the game page background
- dynamic sky
- moving objects
- themes
Email Validator
helper for validating email when user created
JSON Response
JSON/HTTP request
Server
Logic Actions(Object Methods)
New User
New Score
New Group
Increment Total and Set High
Delete User
Delete Score
Delete Group
Leaderboard Request
Edit User
Edit Score
Set Currency
Friend Relation
Server Controllers
Each object type on the backend has its own controller to send and receive data
Load Controller
JSON Models
Springboot Framework routing and autoloading
Object Queries
Object Results
Object Repositories
Manages how data is sent and retrieved from the database using MySQL requests based on the function calls
MySQL Queries
MySQL Results
Database
Score
holds all records of users score for leaderboard references
Team
team info members, settings, etc
Users
user info all user info including currency and a many-to-many self relation for friends
Items
stores what items the user has and the settings pertaining to those items
Leaderboard
returns a json object of top 100 leaderboard (depending on leaderboard requested)

Design Descriptions

Android Activity Classes

The application has about 12 activity classes that will have corresponding XML files. Most logic is done within java activity classes but does use helper classes like for DynamicBackground. The game logic is in Game class, keeping track of lives, and points collected.

Android Communication

Most activity pages will send or get data through the server. A server interface, Network Calls, allows the JSON commands to the server to be handled. State updates between the android application and server are done synchronously. A current user object is created in android containing all information for the user upon login. As fields are updated on the local side, the "Network Calls" abstracts out state updates between back-end and front-end with methods that can be used to generate the necessary volley calls.

Android Models

There is one primary model used within the application which is the user model. The user model consists of information like the user ID, username, password, scores/points earned from playing the game, currency collected to buy items, unlocked items, and friends. Having this user model helps collect and update information as the user uses the application.

Android Helpers

The android helpers include an email validator and dynamic background. The dynamic background is used throughout the application to help with displaying a sunset and sunrise background. The email validator is used to help validate an email the user includes when creating their account for the application.

Springboot Controllers

The springboot controllers are split into three; the first addresses the main user entity and all of its relevant functions. The second deals with the score object, which is a subobject of the user. This score object stores all the user's score information, and it is one-to-one mapped to the user (it's ID is shared with the user; the controller for the score will alter those fields depending on the state update on game end). Lastly, the team controller addresses the in-game team feature functionality where users can get together, based on friend status, to play a game together. Therefore, it is mapped many-to-one, that is, many users can be mapped to one team. The controller has methods to create teams, alter team settings, and delete teams.

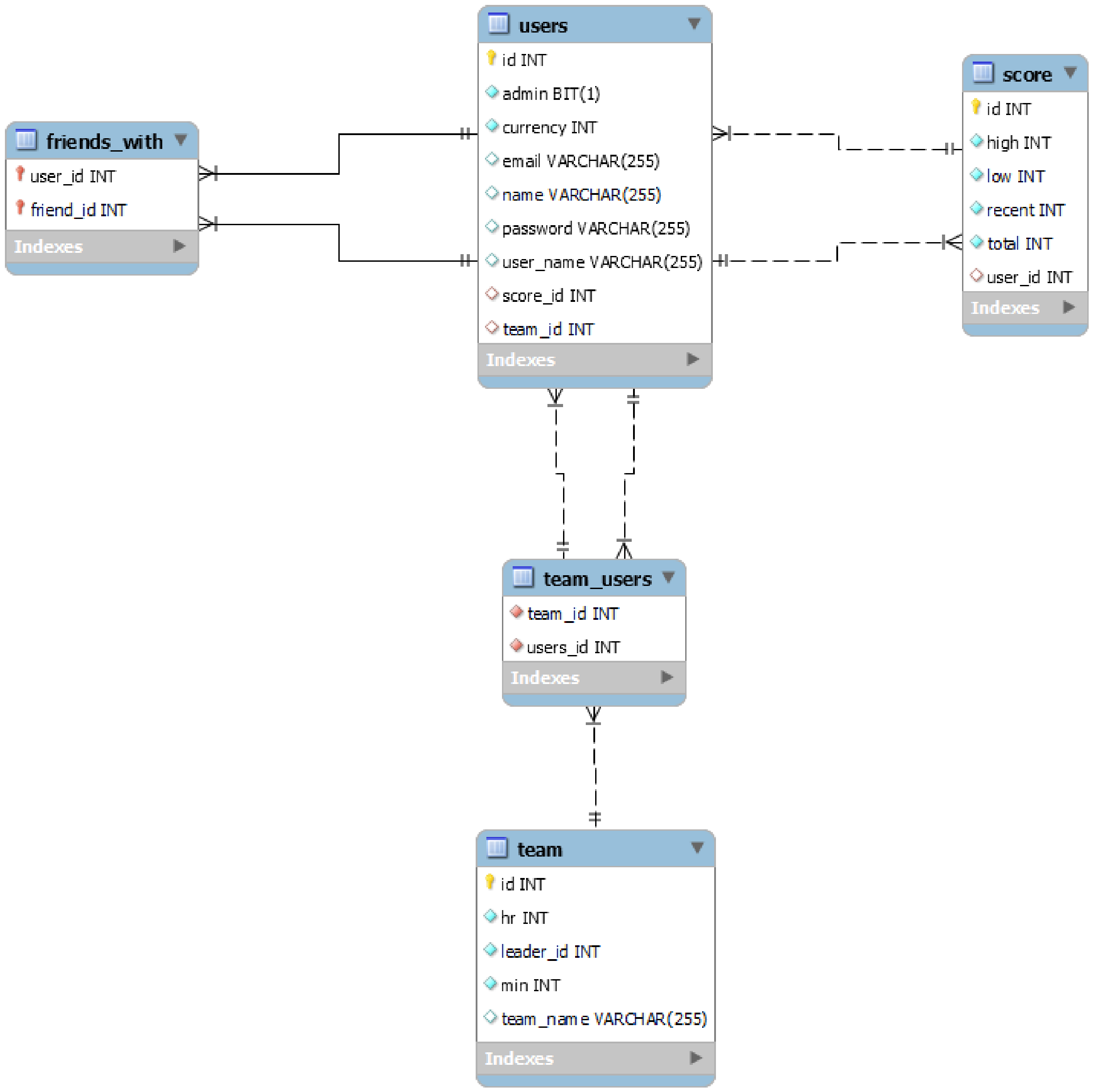
Database:

The user database table contains columns that are relevant directly to the user. These are the user ID, admin status, currency of that user, their name, password, username, score ID, and a group/team ID.

The score database table contains columns that are relevant directly to the score (and by extension, to the user that score is tied to). These are the score ID, and the high/total/recent scores. Only the recent score needs to be updated on the front-end and sent by a volley call to the back-end; high and total will be updated automatically as described above in the springboot controllers section.

The team database table contains columns denoting the relations between users and their created team. Up to four users can be in one team, so the many-to-one mapped relation will automatically generate the columns of team ID (this is specific to the team in question; primary uses for it may be for administrative or user-specific actions), leader ID, and team name.

The friends-with database table contains columns mapping users and their friends. This is a many-to-many relationship as many users can have multiple friends. There are two column entries included: user ID and friend ID.



For whatever reason, I had to generate the relationships between the tables manually and it is very restricted on how you can do that so I'll explain each relationship.
User-Score has a 1:1 (1 score object for each user object(done like this for easy table sorting of scores on backend))
User-Team has a Many:1 (many users per team)
User-User has a Many:Many (friends with other users)