**Design Document for Sporting Social Media**

Group 2\_NS\_8

Member1 Name: Alex Glass (Relationship Model+ Block diagram + Swagger)

Member2 Name: Hrijul Balayar (Frontend javadoc + Block diagram + Explanation)

Member3 Name: Duong Hoang Vu (Explanation + Block diagram + Backend Javadoc + Swagger + Update Relationship model)

Graphical user interface, application

Description automatically generated

Use this third page to describe complex parts of your design.

= Frontend stuff- Hrijul Balayar

Our app uses the HTML pages to provide the view for the user. It also passes information using Java. The database will store user information, messages, posts, groups etc. The main page includes options to view other sports and select the different activities each user can do like access settings, chat with other users, view different venues for the specific sport they have selected and other features provided. The different API’s from the backend support the function created in the api that support different functionalities. Some of our features use google’s api to pull information on where specific venues are located for the sport the user selects. For example, if the user selects soccer he is able to find information about places to play soccer and then use the other features like post, comment and are able to review a place. There is also a chat service which we are gonna have to figure out what websocket to use so the users could send, receive messages andalso be able to manage those chat rooms. Identity interface will also be implemented to manage user requests so the right profile is able to be accessed.

Duong Vu:

Define: **action function**s - functions in the UserModel that simulate the user’s actions and perform necessary data manipulation on the database.

Backend is organized based on the MVC (Model View Controller) design pattern.

* Views: Interfaces that implement JPA repository to add/modify data in the database
* Controllers: For each Model class (non user), there’s going to be a corresponding controller. That controller is for mapping the user’s action recorded on the frontend to an action function related to that Model in the UserModel.
* Models:

For this assignment, I categorize the models into two types (not actual physical classes).

* Type one, which is the main type, is the User Model. This model contains all the functions the controller needs to perform data retrieving/manipulation. The User Model has functions that can retrieve/modify the user’s data and relations, as well as other related models’ data and relations (if has permission).
* Type two, which is the supporting type, is all the other Models that are not the User Model (Group, Post, Comment). These models contain helper functions which are never called by any other class other than the User Model. These helper functions can retrieve/modify only that model’s data and relations.

The reason I chose to organize the functions this way is because the frontend serves as a gateway for the users to send their request/actions to the backend. For every action the user takes, that action is going to get simulated by an according function in the Controller classes, which will call an according function in the User Model. I believe that by doing this, it makes it easier to keep track of the range of actions the user can take. Hence, easier code-maintainability + debugging.

Alex Glass

Relationship Model: 