Database Final Report

CS5200 Introduction to database management

Group Members: **Lin Zhu**

Di Xie

Sunan Jiang

1. Problem statement

League of Legends is a fast-paced, competitive online game that blends the speed and intensity of an RTS with RPG elements. Two teams of powerful champions, each with a unique design and playstyle, battle head-to-head across multiple battlefields and game modes. With an ever-expanding roster of champions, frequent updates and a thriving tournament scene, *League of Legends* offers endless replay ability for players of every skill level.

However there is **no social network community** for this theme, players are hard to find friends in the game, also they don't have a platform to share their happy moment with their friends in the game.

Besides, after each game, they may want to **search themselves or their friends to check how well are they in the game**. Player may want to check their recently matched games, and the detail in each game. Also for rank game, they may want to know more details. A ranked game is a draft mode game type available to level 30 summoners with at least 16 champions available (NOT including free week champions). Summoners are placed on a game ladder and ranked against one another according to a hidden Matchmaking Rating. Players are divided into Challenger, Master, Diamond, Platinum, Gold, Silver and Bronze tiers, with each tier further subdivided into 5 divisions, apart from Challenger and Master, which only have one tier. Ranked games are considered the

competitive alternative to normal games.

2. Proposed solution.

Consider the above problems, we've built a social online communication community website. In our website, you can post your mood for the day, and share your favorite moment to your friends. You can find your friends to join your game here. You can also check your friends profile in this community, it'll let you get to know your friends better.

You can also search your or your friends' recent game and check your game details for each game. Also you can check your rank stat in this website, it can give your total overview in all your rank game, such as total winning rate. Also it can show you your winning rate, KDA (kills, deaths, assists) for each champion you played in a ranked game.

3. Architecture

Login page:

In this page, if you don't have an account in the community, you can register as a new user in this community, after you register, you'll go to your own homepage. If you already have an account here, you can login as well.

Homepage:

You can post you mood here, also you can see your friends post as well. Also you can comment and like them.

Profile page:

In this page, you can create your own profile, you can also edit or hide them after you submit.

Friend page:

You can add friends in this page.

Privilege page:

You can create user, update user, delete user, find one user, and find all users here using JWS.

Search page:

You can search you or your friends in this page by their league of legends username. You can see their recent matched game and ranked games in this page.

Contact us page:

You can leave a message here, so that we can contact you.

Logoff:

You'll logout when click "logoff" button.

4. APIs Discuss

API Used:
League of Legends Official API:
https://developer.riotgames.com/api/method/

Champion Picture:
http://ddragon.leagueoflegends.com/cdn/img/champion/splash/DrMundo_0.jpg

Summoner_icon:
http://avatar.leagueoflegends.com/na/yourdadddd.png

Items_icon: http://ddragon.leagueoflegends.com/cdn/4.19.3/img/item/3124.png

Summoner_spell icon:
http://loldb.gameguyz.com/images/sqlite_images/summoner_spell/13.png

The API provided by the LOL Official Website is kept in different categories, which are interconnected by IDs, such as Summoner ID, Match ID, Champion ID, and so on.

https://na.api.pvp.net/api/lol/na/v2.2/matchhistory/47642700?api_key=e7082900-18ec-4f82-af81-0987641a9f1a
is used to retrieve all 10 matches detail which only about the summoner we point to. Matches are returned as an ObjectArray. Every Array contains several objects that include a match ID.

To get all players detail on single match, we use
https://na.api.pvp.net/api/lol/na/v2.2/match/1543373608?api_key=e7082900-18ec-4f82-af81-0987641a9f1a

Mentioned that all the reflections between ID and real username or champion name are store in a separate category:
So I need to translate IDs to names before I pass my List to .jsp. Also the reverse reflection is needed when users type in the username and search it by IDs.
All icons, such as summoner spells icons, item_icons and champion_icons, are taken from 3 different APIs, where we can retrieve respective pictures by replacing the vector in the URL with ID in need.

To test if we get the correct icons with respective IDs, we are using following URL to check the result: http://loldevelopers.de.vu.

5. Technology stack that you used

1. Java ORM :JPA and JPQL
   We use JPA and JPQL to build our database.
2. JavaScript & HTML
   We use JSP and HTML to build our website, and use servlet to pass variable between them.
3. Boostrap and other CSS
   We use Boostrap and other CSS to make our website like more professional
4. JQuery
   We use JQuery to add dynamic elements in our website. Such as hide, show and toggle.
5. API Utilization and Implementation
6. JWS and Ajax
   JWS is a function that is used to modify our database directly on the web page which does not need to go through the servlet.

   We implement GET to find a specific user by their username or find all user with blank form and PUT to create a new user, POST to modify a exist user and DELETE to delete a existing record.

   We design a web page which has four buttons to trigger different function in AJAX and implement distinct method that we create in our JWS file and DAOs that correspond to specific tables.

6. Use cases

Use Case: Visit League of Legends Official Website
Description: User wants to visit LOL official website
Steps:
[Actor] – User sends “visit LOL website” request on any page
[System] – System jumps to LOL homepage

Use Case: Sign Up
Description: New user wants to register an account by submitting certain information
Precondition: Application must have a LOL account
Steps:
[Actor] – User fills in and submits the form on welcome page
[System] – System creates new row in user table
Post condition: User authorized

Use Case: Complete personal profile
Description: New user wants to complete other personal information on profile page after resigter
Precondition: User authenticated
Steps:
[Actor] – User fills in and submits the form on personal profile page
[System] – System creates new row in profile table and its compositional tables
Post condition: Personal profile page obtained

Use Case: Edit Basic Information/Phone/Email/Work/Education
Description: User wants to update certain personal profile information
Precondition: User authenticated
Steps:
[Actor] – User edits and submits personal profile information on profile page
[System] – System updates relevant profile data in profile table or its compositional tables
Post condition: New data showed on profile page

Use Case: Reset Basic Information/Phone/Email/Work/Education
Description: User wants to delete certain personal profile information
Precondition: User authenticated
Steps:
[Actor] – User sends “delete” request on personal profile page
[System] – System deletes relevant data in profile table or its compositional tables
Post condition: Relevant block emptied on profile page

Use Case: Reset All Personal Information
Description: User wants to delete all personal information
Precondition: User authenticated

Steps:
[Actor] – User sends “delete all” request on personal profile page
[System] – System deletes all relevant data in profile table and its compositional table
Post condition: All block emptied on profile page

Use Case: Hide Basic Information/Phone/Email/Work/Education
Description: User wants to hide certain information block on personal profile page
Precondition: User authenticated
Steps:
[Actor] – User sends “hide” request on personal profile page
[System] – System hides relevant on profile page
Post condition: Certain block disappears on profile page

Use Case: View Blogs List
Description: User wants to view blog list on blog page
Precondition: User authenticated
Steps:
[Actor] – User opens blog page
[System] – System lists all blogs on blog page
Post condition: Blogs list showed

Use Case: View Friends List
Description: User wants to view friends list on friend page
Precondition: User authenticated
Steps:
[Actor] – User opens friend page
[System] – System lists all blogs on blog page
Post condition: Friends list showed

Use Case: Add Friend
Description: User wants to add new friend
Precondition: User authenticated
Steps:
[Actor] – User sends “add this friend” request on add friend page
[System] – System creates new friends relationship in user_friends table
Post condition: New friend’s post showed on homepage

Use Case: Post “What’s up”
Description: User wants to post state on personal homepage
Precondition: User authenticated
Steps:
[Actor] – User types some message and submits on personal homepage
[System] – System creates new row in post table
Post condition: New post showed on homepage

Use Case: Comment on Post
Description: User wants to comment on friend's post on friend's homepage
Precondition: User authenticated, Friends relation
Steps:
[Actor] – User types some message and submits on friend's homepage
[System] – System creates new row in comment table
Post condition: New comment showed on friend's homepage

Use Case: Delete "What's up"
Description: User wants to delete state on personal homepage
Precondition: User authenticated
Steps:
[Actor] – User send "delete this post" request on personal homepage
[System] – System deletes relevant rows in post table and comment table
Post condition: One post and its comments disappeared on personal homepage

Use Case: Like Post
Description: User wants to like friend's post
Precondition: User authenticated
Steps:
[Actor] – User send "like this post" request on personal homepage
[System] – System creates relationship row in thumbup table
Post condition: "Like" information showed on personal homepage

Use Case: Search Summoner
Description: User wants searcher LOL summoner information
Precondition: User authenticated, Summoner authenticated
Steps:
[Actor] – User enters summoner name and sends "search by summoner name" on search page
[System] – System connects LOL database and sends "request data", then receives data
Post condition: Summoner Information showed on search page
Alternate paths: Summoner Unauthenticated→ No data returns

Use Case: Check Summoner Profile/Match/Rank
Description: User wants to take a look at summoner's profile/match/rank
Precondition: User authenticated, Data returned
Steps:
[Actor] – User opens relevant subpage in search page
[System] – System jumps to relevant page
Post condition: Relevant page showed

Use Case: Order Summoner Match by Kills/Deaths/Asists/Gold Earned/ Creeps
Description: User wants to order summoner's match

Precondition: User authenticated, Data returned
Steps:
[Actor] – User sends "order match by" request on match page
[System] – System orders match by kills/deaths/asists/gold earned/ creeps
Post condition: New order showed on match page

Use Case: Order Summoner Rank by Wins/Losses/Win%/KDA/Kills/Deaths/Asists/Creeps
Description: User wants to order summoner's rank
Precondition: User authenticated, Data returned
Steps:
[Actor] – User sends "order rank by" request on rank page
[System] – System orders rank by Wins/Losses/Win%/KDA/Kills/Deaths/Asists/Creep
Post condition: New order showed on rank page

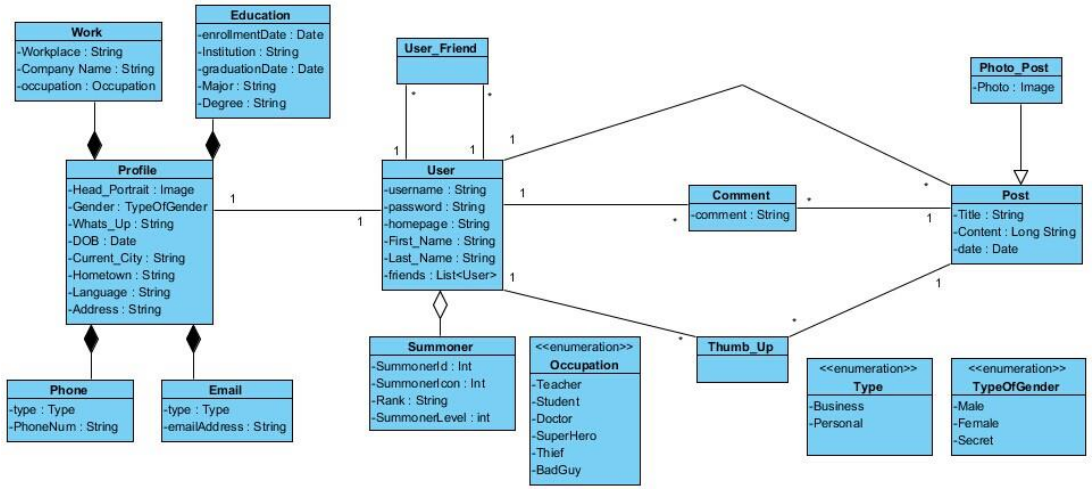
Use Case: View Match Details
Description: User wants to view details of one match
Precondition: User authenticated, Data returned
Steps:
[Actor] – User sends "view details" request on match page
[System] – System jumps to relevant match details page
Post condition: Relevant match details page showed

Use Case: Logoff
Description: User wants to logoff
Precondition: User authenticated
Steps:
[Actor] – User sends "logoff" request
[System] – System logs user off and returns to welcome page
Post condition: User unauthenticated

Use Case: JWS
Description: Administrator wants to manage users in database
Precondition: User authenticated
Steps:
[Actor] – Administrator fills in relevant form and sends create/update/delete/find all/find one request
[System] – System create/update/delete/find all/find one user
Post condition: Relevant data changed in User table

7. Data model

Final UML class diagram:



Generalizations: Table photo_post inherits fields from table post. They share same attributes title, content and date. Photo_post has one more attribute photo.

Aggregation: Table summoner is a part of table user, every user will have one or more summoners.

Composition: Table profile owns table phone, email, work and education. Phone, email, work cannot exist without profile.

Association Table: Table comment is an association table. It shows relationship between user and post, also has another attribute comment.

Self-reference table: User table is a self-reference table, the friends attribute reference to user itself.

Many to Many relationships: Table User and friends.

Many to One relationships: Table Post and table User

One to One relationships: Table User and Profile.

Enumeration table: Table TypeOfGender, Type, Occupation.

Reify: User has a self-association relationship which is many to many. A table user_friend is used to reify this many to many relationship. User_friend use a compound key which is two foreign key of table user as its primary key.

Table thumb_up here is to reify many to many relationship between user and post. Thumb_up has a compound key which is made up of one foreign key of user and one foreign key of post as its primary key.