Pickup Finder

By Team 2

Tyler Dunning

Cameron Harris

Se Chang Oh

FALL 2023

Table of Contents

Functional Requirements…….3

Functional Requirements…….3

Identification: entities, attributes, multiplicity……6

Entity Relationship Diagram……..……...12

ERD Schema……..13

SQL Workbench Tables…….14

Instructions to Run the Program……26

Application walkthrough……...……28

Lesson Learned……..40

Project Description

Pickup Finder is a database application designed for basketball enthusiasts offering a platform for players to interact with their local courts and fellow players. Our primary goal is to simplify the process of finding pickup times and available courts around them. Users will be able to reserve spots, organize pickup games, and join the games of others. By doing this, we intend to add organization to the unstructured nature of pickup basketball reducing the time and effort required to play currently. Our key stakeholders are the players who utilize the app and our application domain is the consumer market. Ultimately our mission is to make it convenient for basketball enthusiasts to arrange and enjoy their favorite sport.

Functional Requirements

This application provides functionality to users. Users are the customers and they are able to create an account, sign up for events, as well as view and join courts. When a user first launches the application, they will be prompted to create an account or sign in. Once signed into their account, the user will be greeted by the “Browse” menu, which displays all of the courts that are available. They can filter their search by zip code. If the user selects a court, they can see how busy it will be. The user can then select to join the court for a certain window of time so that other users can see that someone will be there. Users have the ability to add other users as friends to see their status.

Create Account:

- Users will enter an email address, username, and password. Once the email is verified through a confirmation email, the user will have full user access to the website

Join Court:

- Users can find a local court from the listings and press the “Join” button. They can select a time frame that they plan on being at the court.
- Users have the option to add themselves to the court anonymously so that other users do not see their username when they view the court.

View Court

- If a user opens a selected court’s “View” page, they will be shown a timeline with the amount of people that are at the court throughout the day.
- An image of the court as well as information such as the condition, number of hoops, and address will be displayed.

Add Friend

- Users can add friends either by username search or by selecting the user from the court View Page. Friends can see each other’s playing history, current court, and they can send each other messages.

Send Message

- Users will have the option to send a direct message to any of their friends or group members. Each private chat will be shown in its own window.
- If a user wants to message a user that they see active on a court, they must first send a friend request and receive confirmation from the other user

Create Event

- Users can create an event for tournaments with scheduled times and a maximum number of players.
- Users can sign up together as a team. Creators of events can optionally require users to sign up as a team of a certain size.
- Events will be given their own page with a description and an option to message the event creator.

Create Team

- For an event that requires a team to sign up, users can create a team
- Each team will have a team captain that creates the group
- Users that want to join the team can send a join request to the captain from the event's page

User Reputation

- The more that a player joins a court and wins or participates in events, they will increase their account level for other users to see.

Browse Courts

- Based on a zip code entered, users can view the courts nearby to them

Court Group Chat

- Each court will have a chat feature where users can communicate with other people that are viewing the court.
- Each chat message will be deleted after 24 hours so that the chat stays relevant for current users

Groups

- Users can create a group for multiple users to be a part of. Groups will have a name, a group chat feature, and group activity postings.
- Users that join the group will be visible to other group members and will have access to the group message box and group activities.
- This feature would allow users to organize larger groups through centralized activities. Users could see the current status of other group members, as well as have the option to send anyone in the group a direct message.
- The group owner can remove users from the group

Group Activities

- Group members can create a group activity for other group members to see
- Activities will show up on the “Activities” tab of the group’s page
- Activities have a title, description,

Identification of entities, attributes, and relationships with multiplicity

Users entity set

The Users entity set is responsible for storing the user information in the database.

- username: The primary key of the User entity. When users create an account, they will check the availability of a username to make every username unique.
- password: The user's password for security.

Messages entity set

The Messages entity set is responsible for storing the messages sent between users or within a group in the database. The entities in this table will be pulled from the table and displayed to a chat box in order of time sent based on the corresponding chat_id. The chat_id will match either a friendship_id, group_id, or court_id. All of these table IDs will have distinguishing factors, such as a unique starting number, so that they will not overlap.

- chat_id: The primary key of the Message entity. A unique chat_id will be given when messages are saved into the database.
- sender_id: The sender who sent the message.
- time: The time that the message was written.
- content: The content of the message.

Courts entity set

The Courts entity set is responsible for storing information about each court in the database.

- court_id: The primary key of the Courts entity. A unique court_id will be given when courts are saved into the database.
- court_name: The name of the court.
- address: The address of the court.
- condition: The overall condition of the court. It will be stored as a number.
- num_of_hoops: The number of hoops in the court.
- hours_of_operation: The operation hours of the court.

Events entity set

The Courts entity set is responsible for storing information about tournament events in the database.

- event_id: The primary key of the Events entity. The unique event_id will be created when events are posted on the website and stored in the database.
- Held_in: foreign key from the court_id in the Courts entity.
- date: The date that the event is held.
- max_teams: The maximum number of teams that can apply for the event.
- team_size: The team size of each team.
- description: The description of the event.

Teams entity set

The Team entity set is responsible for storing team information in the database. Teams are created just for the event, so when the event is over, it will be deleted along with the event.

- event_id: This is from the Event entity set. It shows which team is in which event.
- team_id: The primary key of the Teams entity. It will be created when the team applies for the event
- team_captain: This is from the Users entity set. The username will be used for this attribute.
- team_size: The team size of the team.

Group entity set

The Group entity set is responsible for storing information about groups in the database.

- group_id: The primary key of the Group entity. It will be created when the group is created.
- group_is_private: The attribute that shows if the group is private or public. It will be stored as a boolean type.
- group_owner: The owner of the group.
- group_description: The description of the group.

Group activity entity set

The Group activity entity set is responsible for storing information about the group activities in the database. It is a weak entity set of 'Groups.'

- activity_name: The name of the activity will be the defining characteristic of an activity within a group
- group_id: Each activity will be defined to a group. This way there can be two different groups that have an activity with the same name.
- activity_description: The description of the group activity.
- activity_location: The location where the group activity took place.

UserOnCourt entity set

This table will show which users are signed up for specific times on a court. A user can be reserved for multiple times and at multiple different courts, and a court can have multiple users signed up to use it.

- Username: Primary key from username in Users entity
- Court_id: primary key from court_id in Courts entity

UserInGroup entity set

This table will keep track of what users are in which groups. Groups can hold zero to many users and users can be members of zero to many groups.

- username: primary key as username from User entity
- Group_id: primary key as group_id from Groups entity

Friends entity set

This table will hold information of which users are friends with each other. It is a many to many relationship within the Users entity.

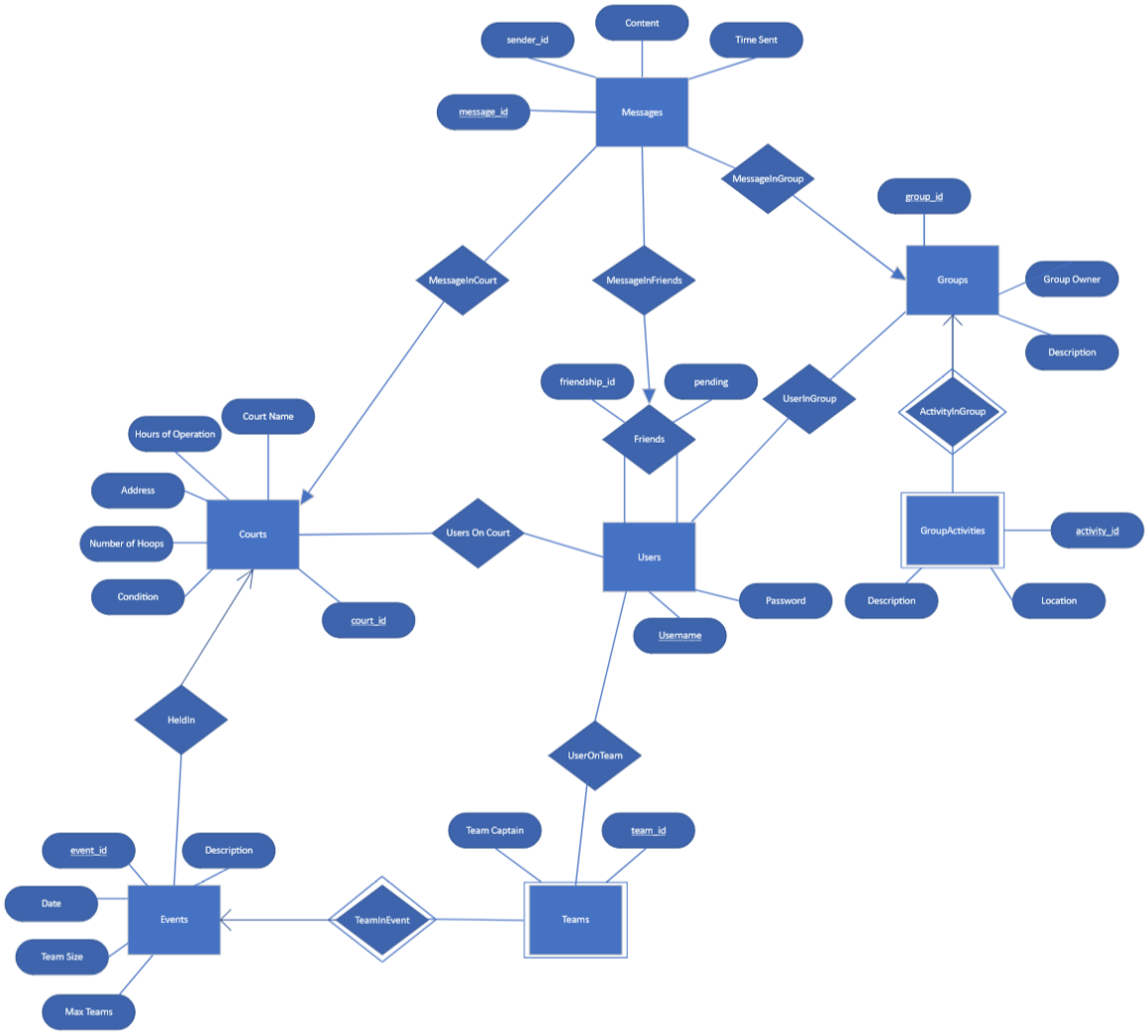
- Username: Primary key from username in Users entity. The name of one user. This will be assigned to the user that sent the friend request
- Friend_id: Primary key from username in Users entity. The name of the other user. This is assigned to the user that accepted the friend request
- Friendship_id: Used for reference with the messaging system
- Pending: Initially set to true, updated to false once the friend request is accepted

UserOnTeam entity set

This table will hold information showing which teams have which users. Users can be on zero to many teams, and teams can hold one to many users.

- Username: Primary key from username in Users entity.
- Team_id: primary key from team_id in Teams entity
- Event_id: primary key from event_id in Events entity

E-R Diagram


Content
sender_id
Time Sent
Messages
message_id
MessageInGroup
group_id
MessageInCourt
MessageInFriends
Groups
Group Owner
Description
friendship_id
pending
UserInGroup
ActivityInGroup
Friends
Court Name
Hours of Operation
Address
Courts
Users On Court
Users
GroupActivities
activity_id
Number of Hoops
Condition
court_id
Password
Username
Description
Location
HeldIn
UserOnTeam
Team Captain
team_id
event_id
Description
Date
Events
TeamInEvent
Teams
Team Size
Max Teams

Schemas

Users(username, password)

Messages(message_id, sender_id, time, content)

MessageInCourt(message_id, court_id)

MessageInFriends(message_id, friendship_id)

MessageInGroup(message_id, group_id)

Courts(court_id, court_name, address, condition, num_of_hoops, hours_of_operation)

PlayerEvents(event_id, court_id, date, max_teams, team_size, description)

Teams(event_id, team_id, team_captain)

PlayerGroups(group_id, group_description)

GroupActivities(activity_name, activity_description, activity_location)

Friends(username, friend_id, friendship_id)

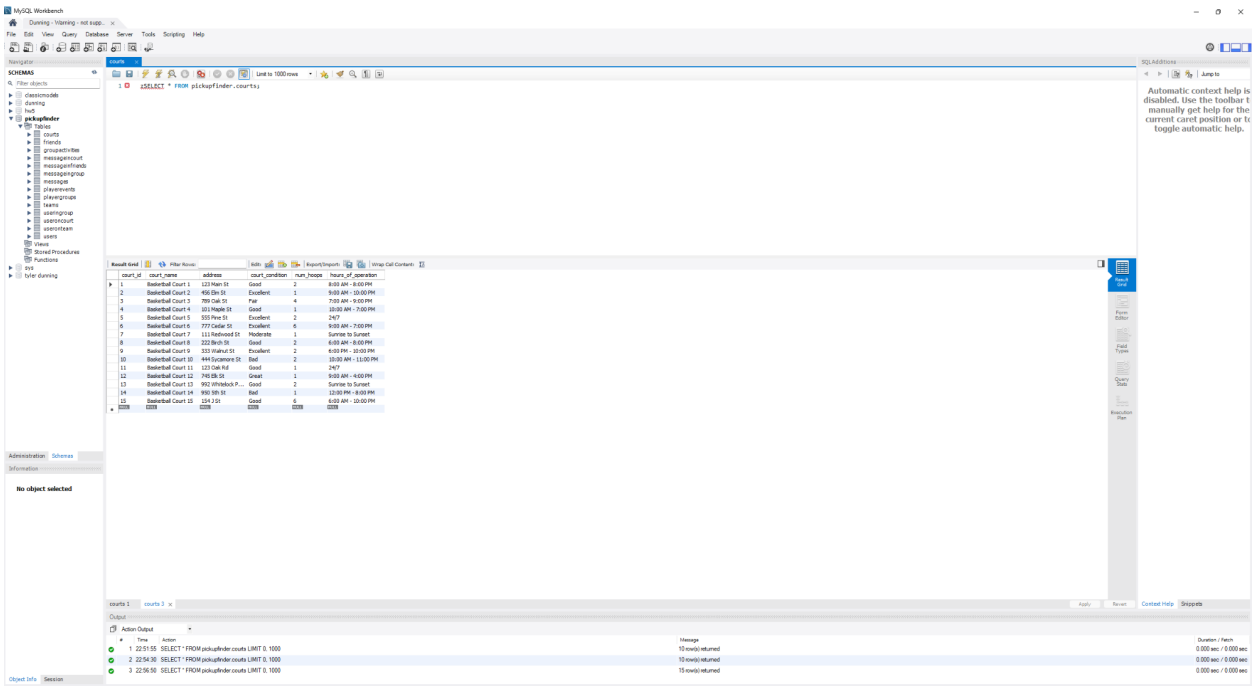
UserInGroup(username, group_id)

UserOnTeam(username, team_id, event_id)

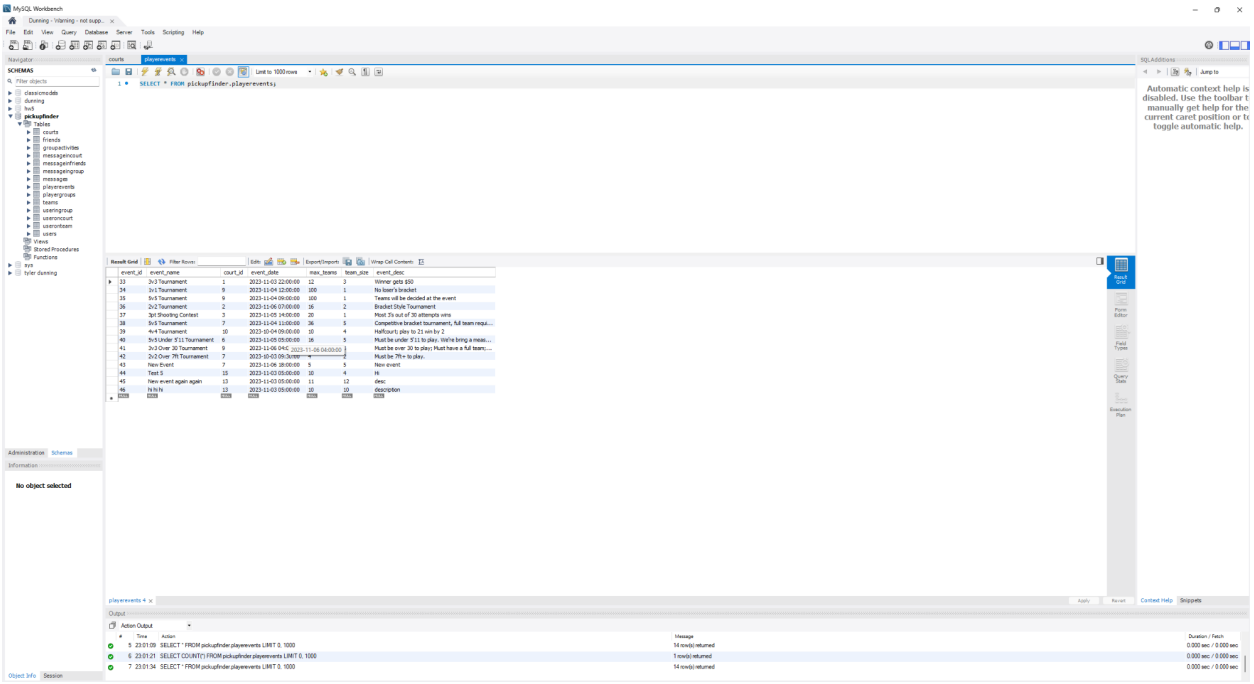
UserOnCourt(username, court_id)

MySQL Workbench Tables

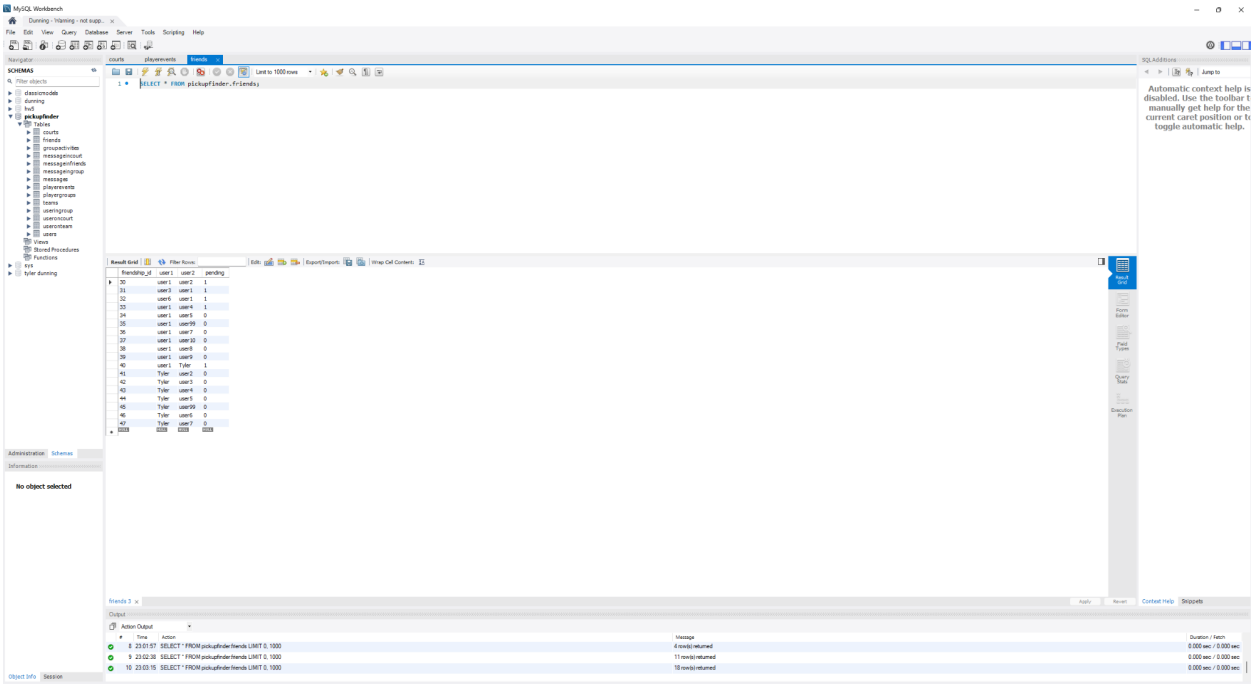
Courts



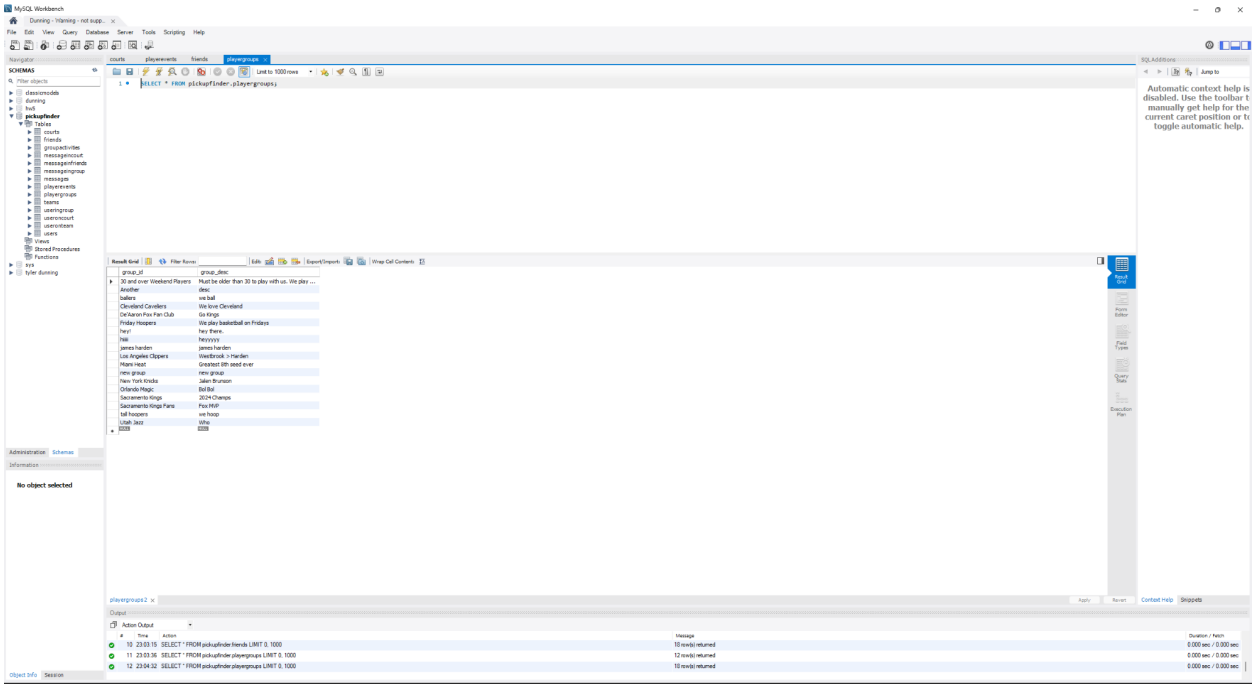
Events


MySQL Workbench
SCHEMAS
pickupfinder
SELECT * FROM pickupfinder.playerevents;
No object selected

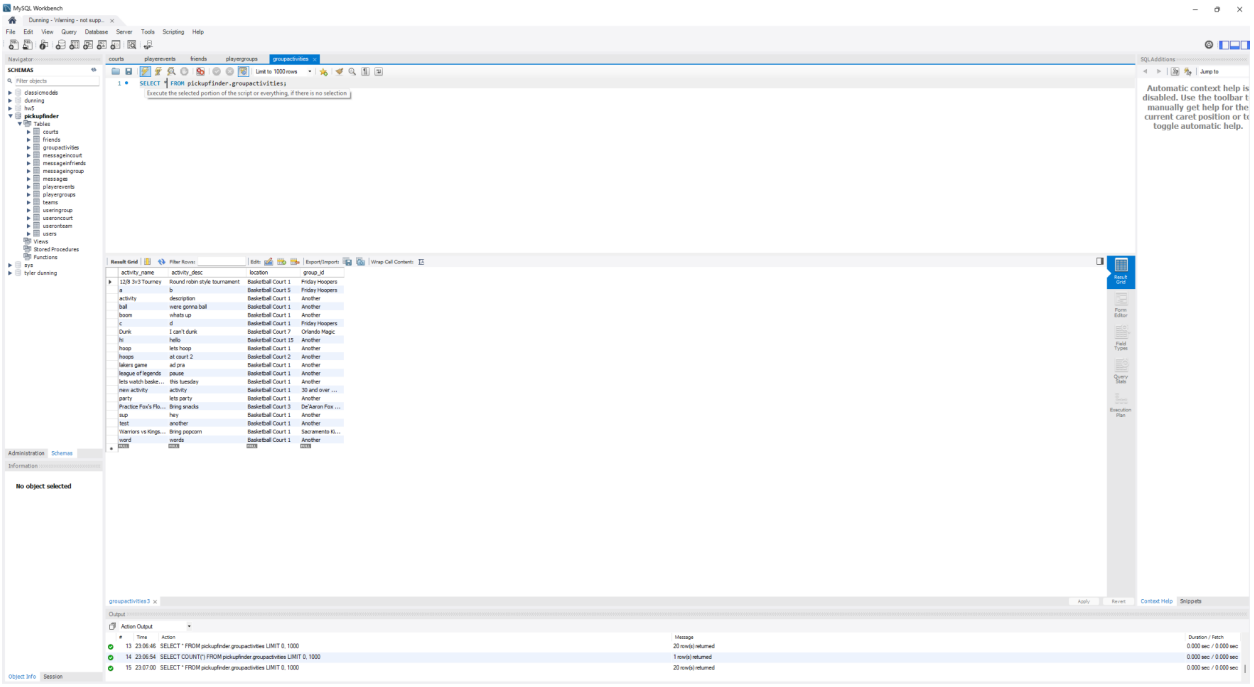
Friends


MySQL Workbench
SCHEMAS
pickupfinder
Tables
courts
friends
groupactivities
messagesincourt
messagesinfriends
messagesingroup
messages
playerevents
playergroups
teams
usersingroup
usersincourt
usersonteam
users
Views
Stored Procedures
Functions
SELECT * FROM pickupfinder.friends;
friendship_id
user1
user2
pending
No object selected
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Action Output
SELECT * FROM pickupfinder.friends LIMIT 0, 1000
4 row(s) returned
11 row(s) returned
18 row(s) returned

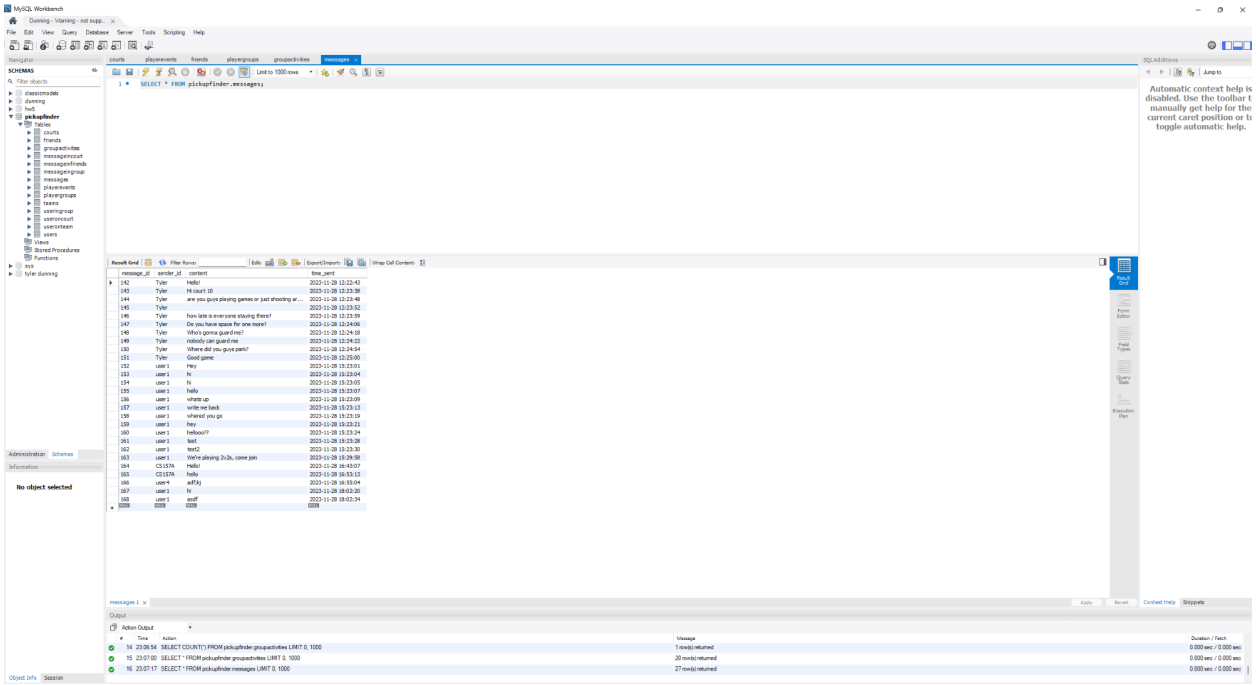
Group


MySQL Workbench
SCHEMAS
pickupfinder
Tables
courts
friends
groupactivities
messagesincourt
messagesinfriends
messagesingroup
messages
playersevents
playergroups
teams
usersingroup
usersincourt
usersinteam
users
Views
Stored Procedures
Functions
SELECT * FROM pickupfinder.playergroups;
group_id
group_desc
30 and over Weekend Players
Must be older than 30 to play with us. We play ...
Another
desc
ballers
we ball
Cleveland Cavaliers
We love Cleveland
De'Aaron Fox Fan Club
Go Kings
Friday Hoopers
We play basketball on Fridays
hey!
hey there.
hiii
heyyyyy
james harden
james harden
Los Angeles Clippers
Westbrook > Harden
Miami Heat
Greatest 8th seed ever
new group
new group
New York Knicks
Jalen Brunson
Orlando Magic
Ball Bia
Sacramento Kings
2024 Champs
Sacramento Kings Fans
Fox MVP
bill hoopers
we hoop
Utah Jazz
Who
No object selected
Action Output

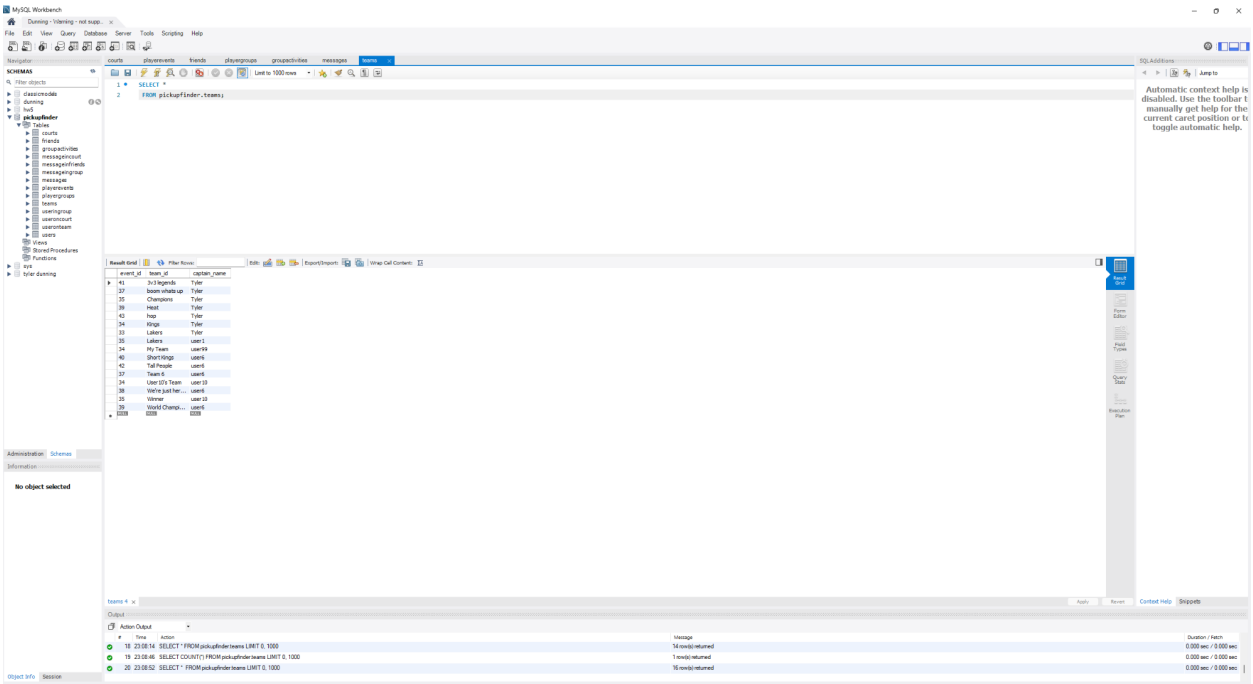
GroupActivities



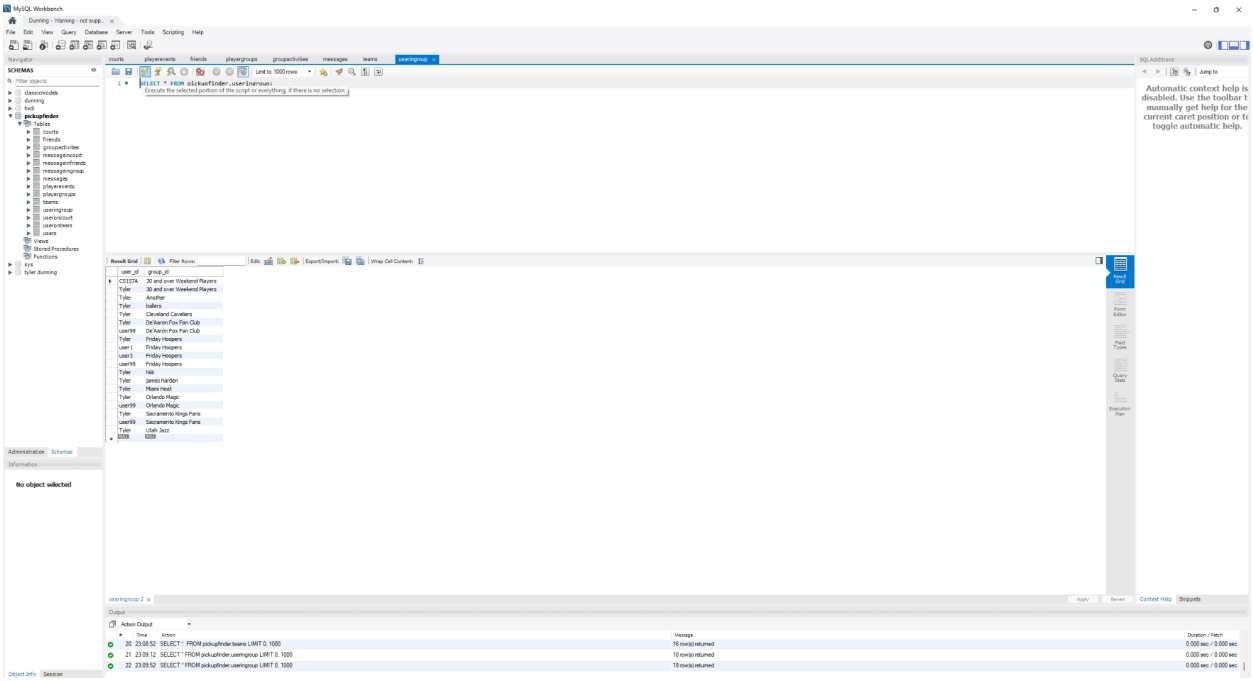
Messages


MySQL Workbench
SELECT * FROM pickupfinder.messages;
Result Grid
message_id
sender_id
content
time_sent
No object selected
Action Output

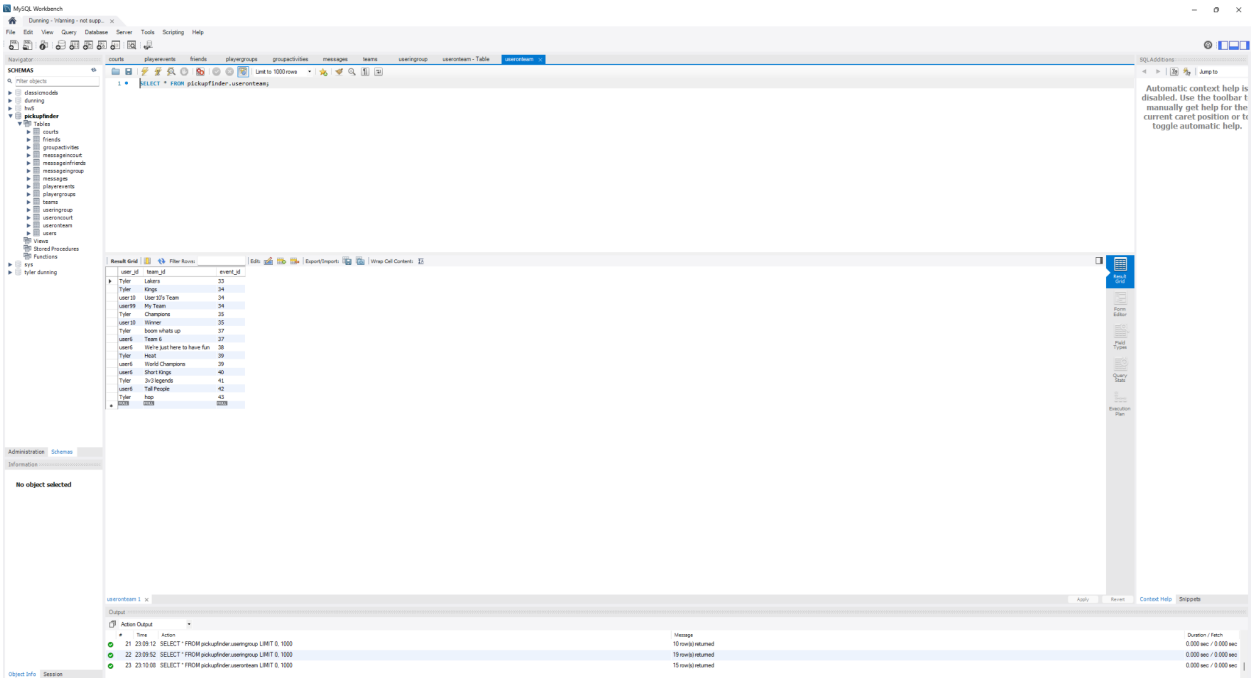
Teams


MySQL Workbench
SELECT *
FROM pickupfinder.teams;
No object selected

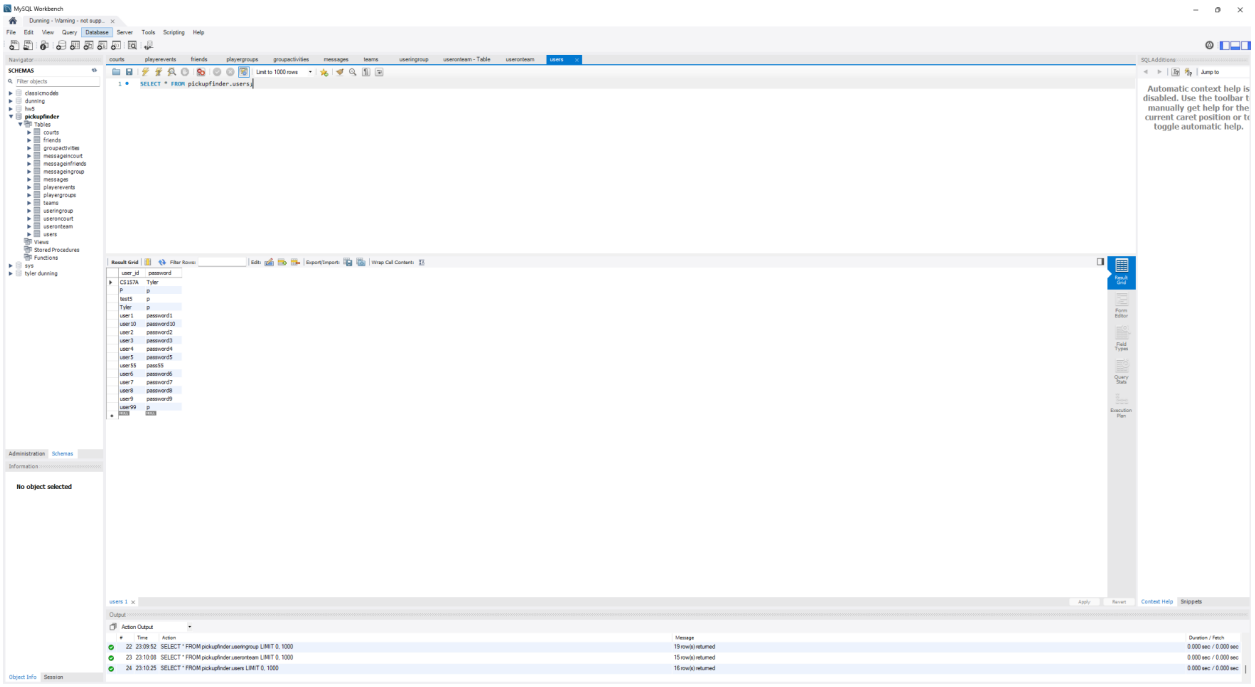
UserInGroup



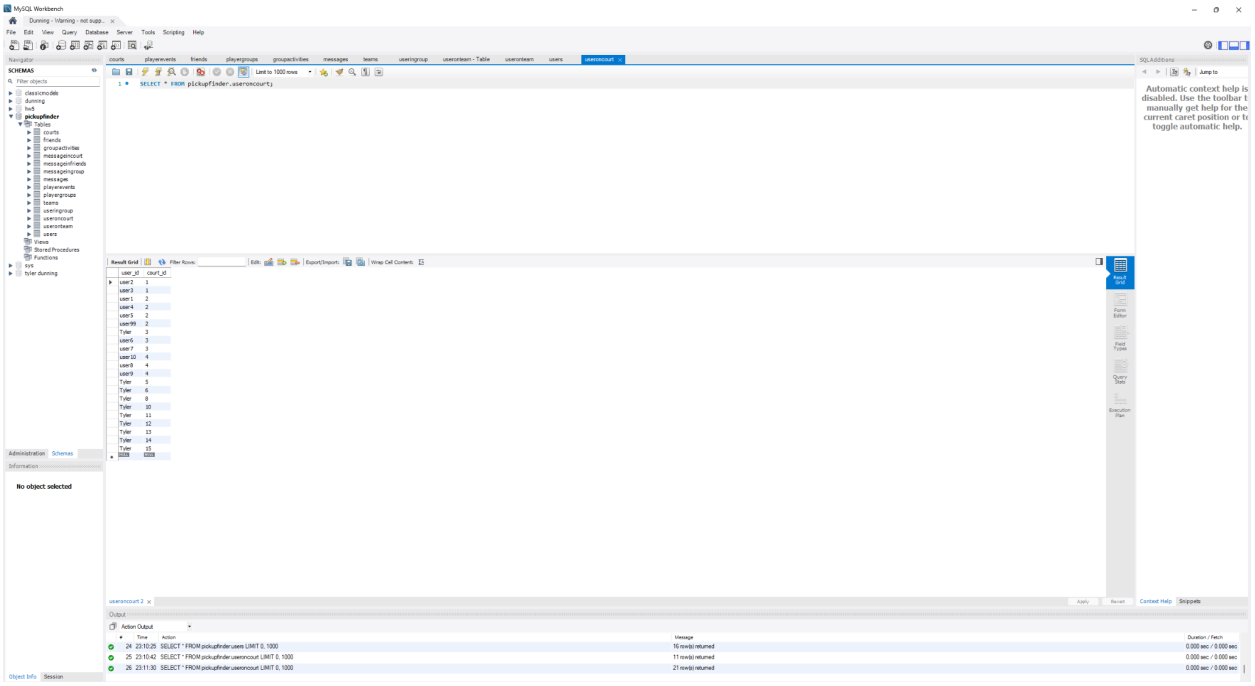
UserOnTeam


MySQL Workbench
SELECT * FROM pickupfinder.useronteam;
Result Grid
Action Output

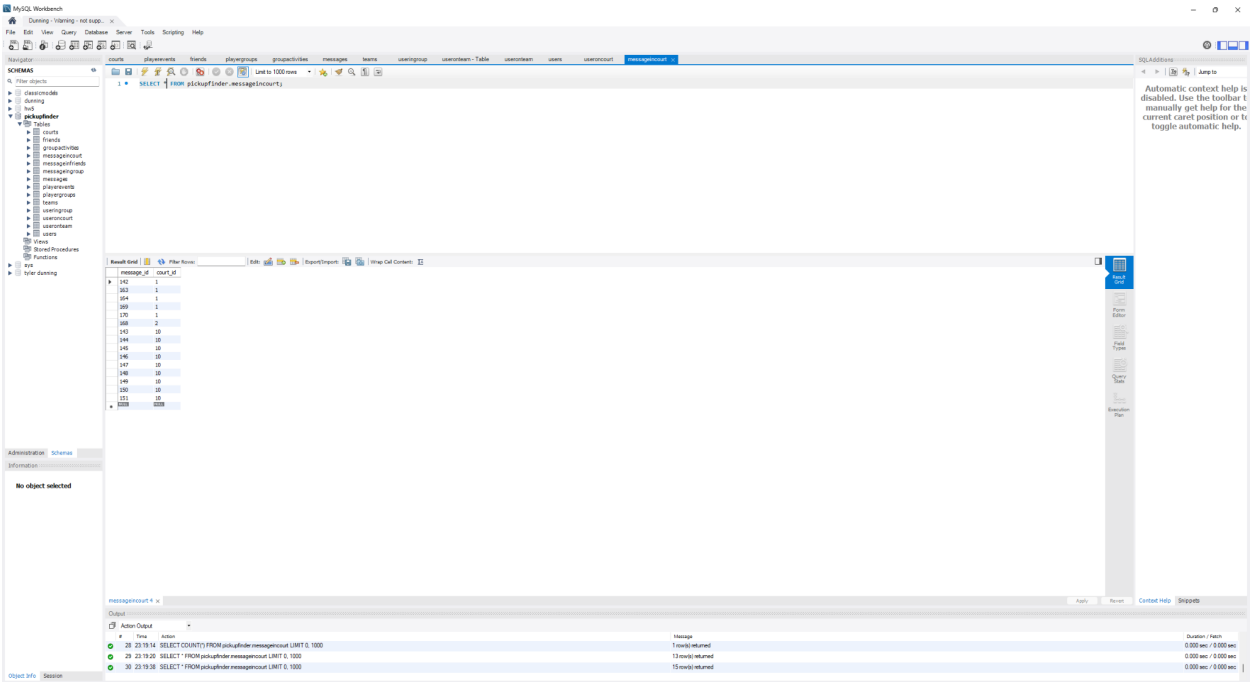
Users



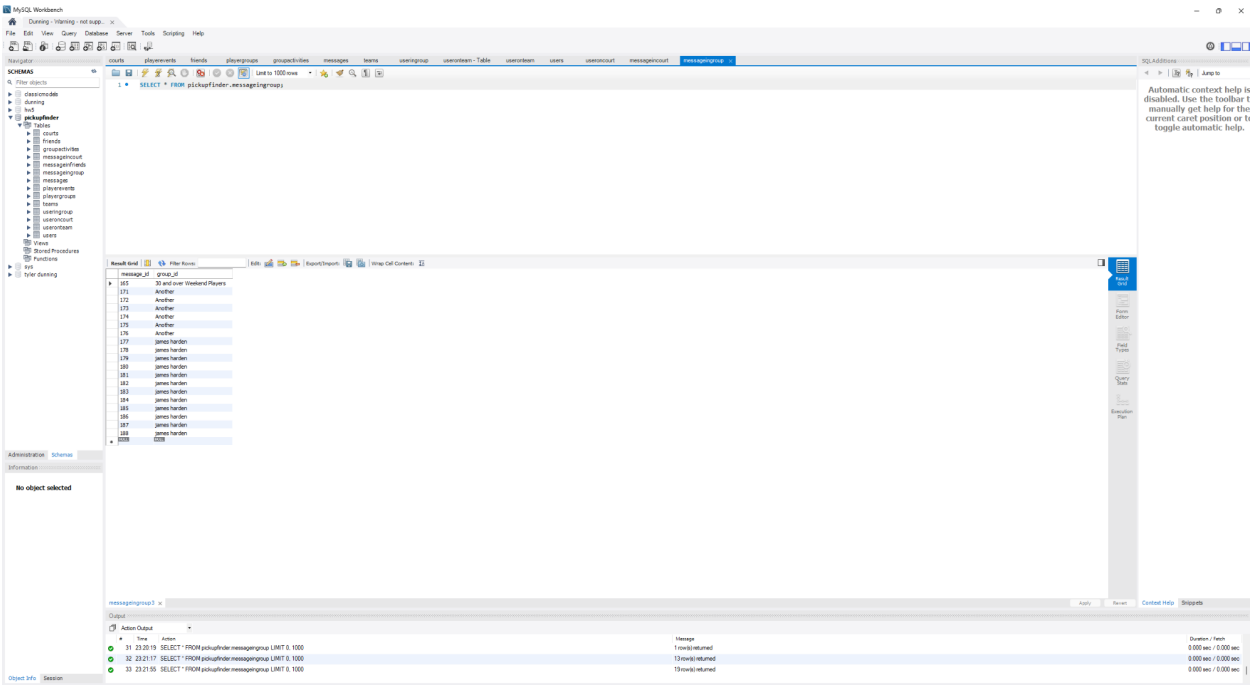
UsersOnCourt



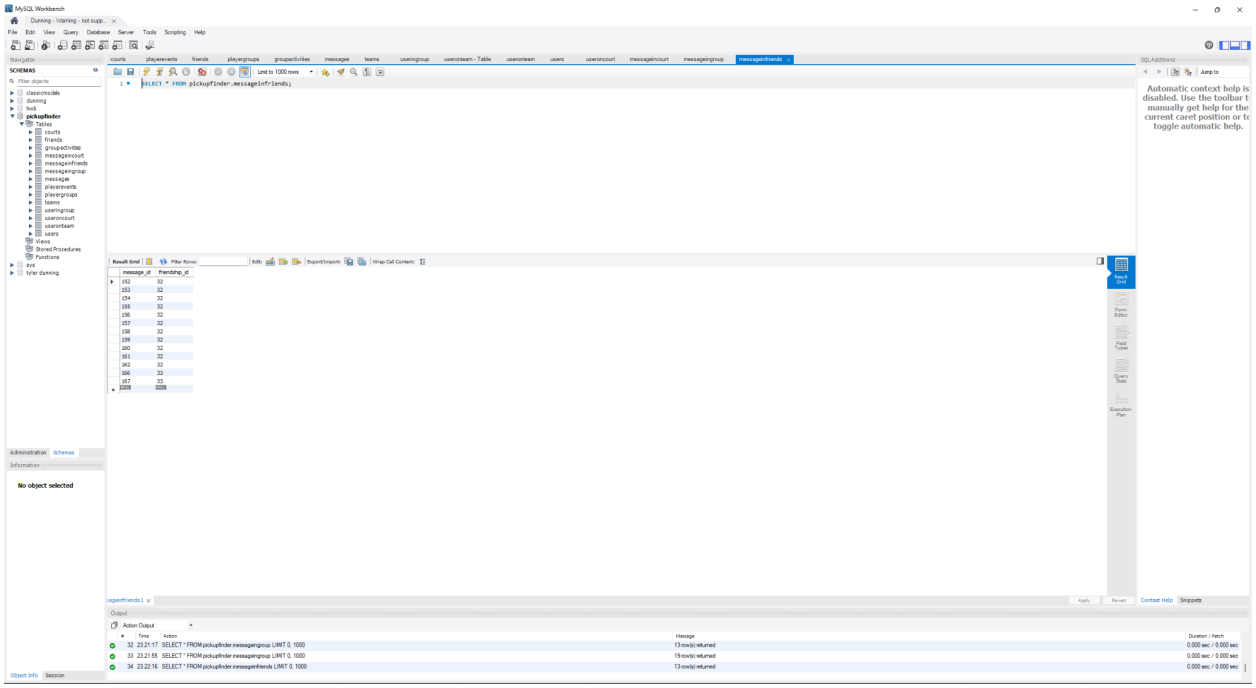
MessageInCourt



MessageInGroup



MessageInFriends



Instructions to Run the Program:

First, either unzip the source code from the zip file or clone our git repository using:

git clone https://github.com/Tyler-Dunning/CS157-Team2.git

To set up your SQL server to match the one that our project uses, we have provided a .sql file containing the commands necessary to copy our database. Run this script in your MySQL workbench or another MySQL terminal. You must also enter your SQL root password into the variable labeled “password” shown here within the file Backend/server.js.

```
const db = mysql.createPool({
    host: "localhost",
    user: "root",
    password: "",
    database: "pickupfinder",
    port: 3306,
    options: {
        trustedConnection: true
    }
})
```

NOTE: We encountered an issue involving the password field and authorization. If the terminal outputs errors after the following step, you may need to change your root password through this command within a MySQL workbench query:

ALTER USER 'your_username'@'your_hostname' IDENTIFIED WITH 'mysql_native_password' BY 'your_password'; FLUSH PRIVILEGES;

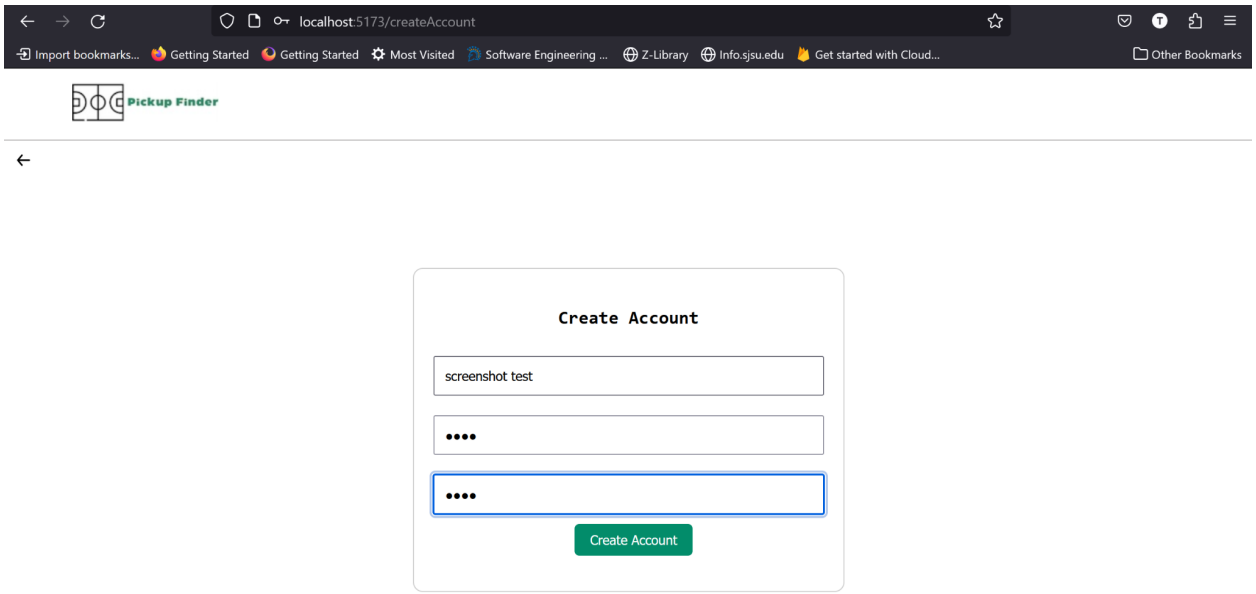
Once this is done, we can deploy our backend. Do this by navigating to the backend folder in a command prompt/terminal using “cd Backend” Once the directory has been selected, type ‘npm start’. (Assuming that npm is installed on the computer)

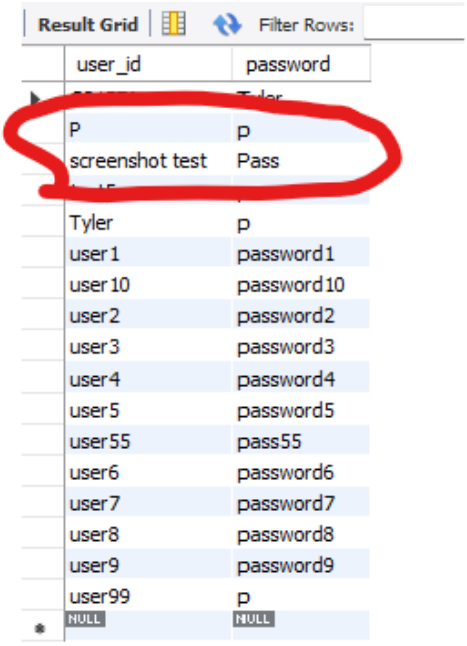
From there, you will need to open another command prompt/terminal to the frontend folder by typing “cd Frontend” You can then type ‘npm i’ followed by ‘npm run dev’.

Once all of these steps have been completed, you should be able to click the link that has been output in the frontend terminal and use the website.

Application Walkthrough (Screenshots):

Creating an account: Entering credentials into the given fields creates a new row in the ‘users’ table, as long as the username is not already taken.

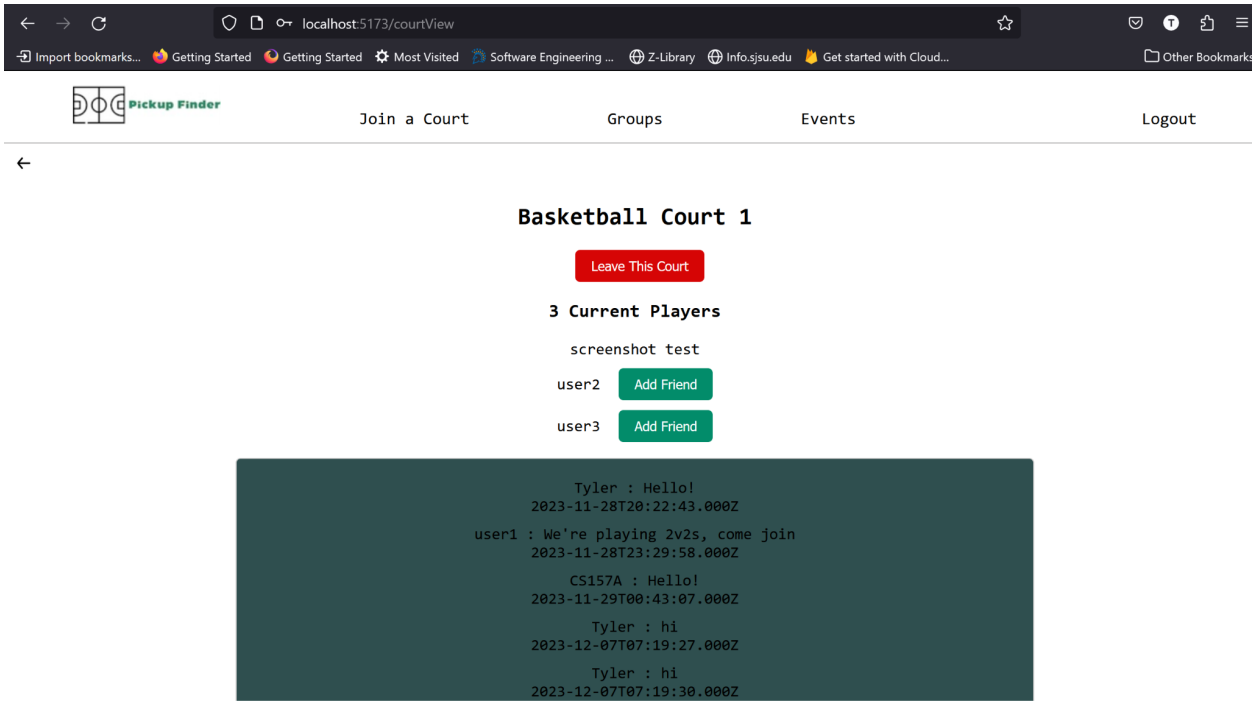


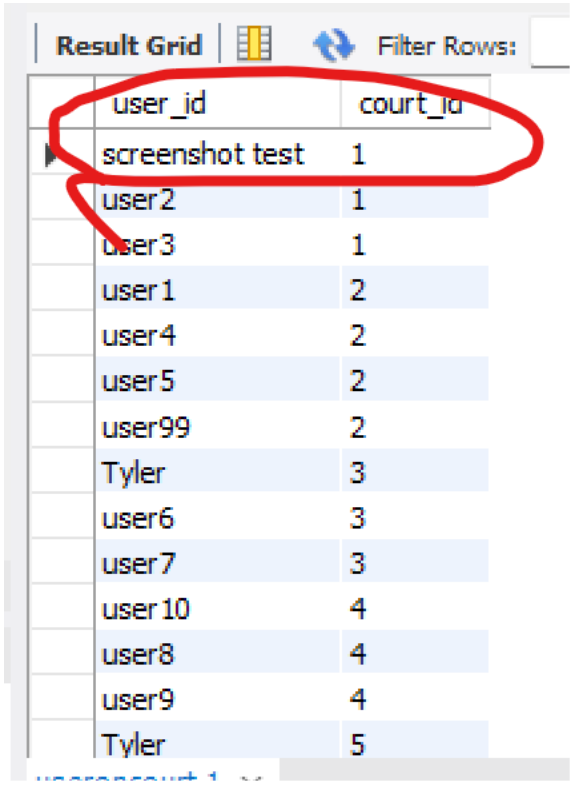


Result Grid | Filter Rows:

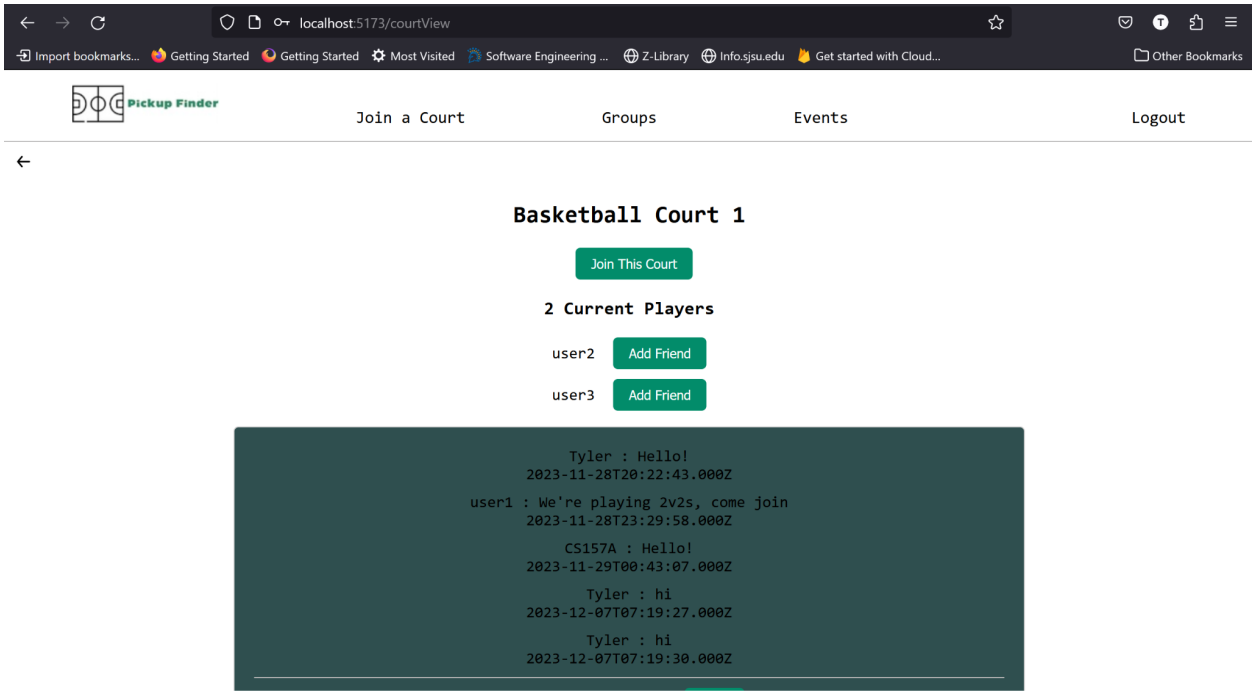
user_id	password
P	p
screenshot test	Pass
Tyler	p
user1	password1
user10	password10
user2	password2
user3	password3
user4	password4
user5	password5
user55	pass55
user6	password6
user7	password7
user8	password8
user9	password9
user99	p
NULL	NULL

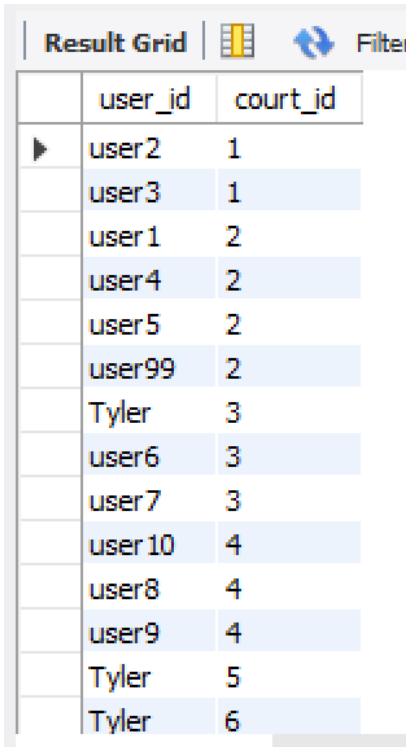
Joining a Court: This screenshot shows the user after pressing “Join This Court.” This will create an entry in the ‘usersoncourt’ table with a reference to our user and the court we are on





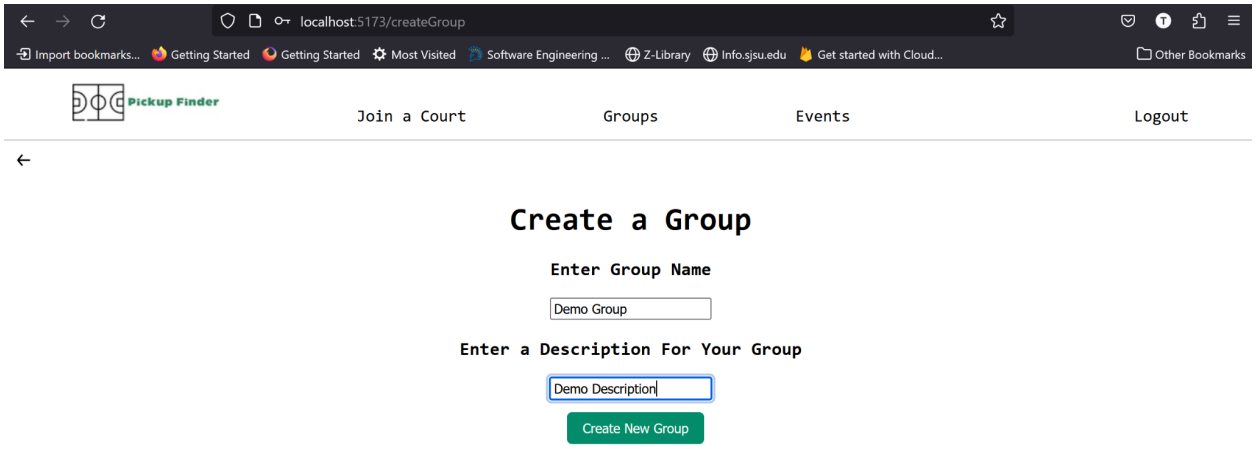
Leaving a Court: Pressing the previously shown “Leave This Court” button, we will delete the entry from the ‘usersoncourt’ table.

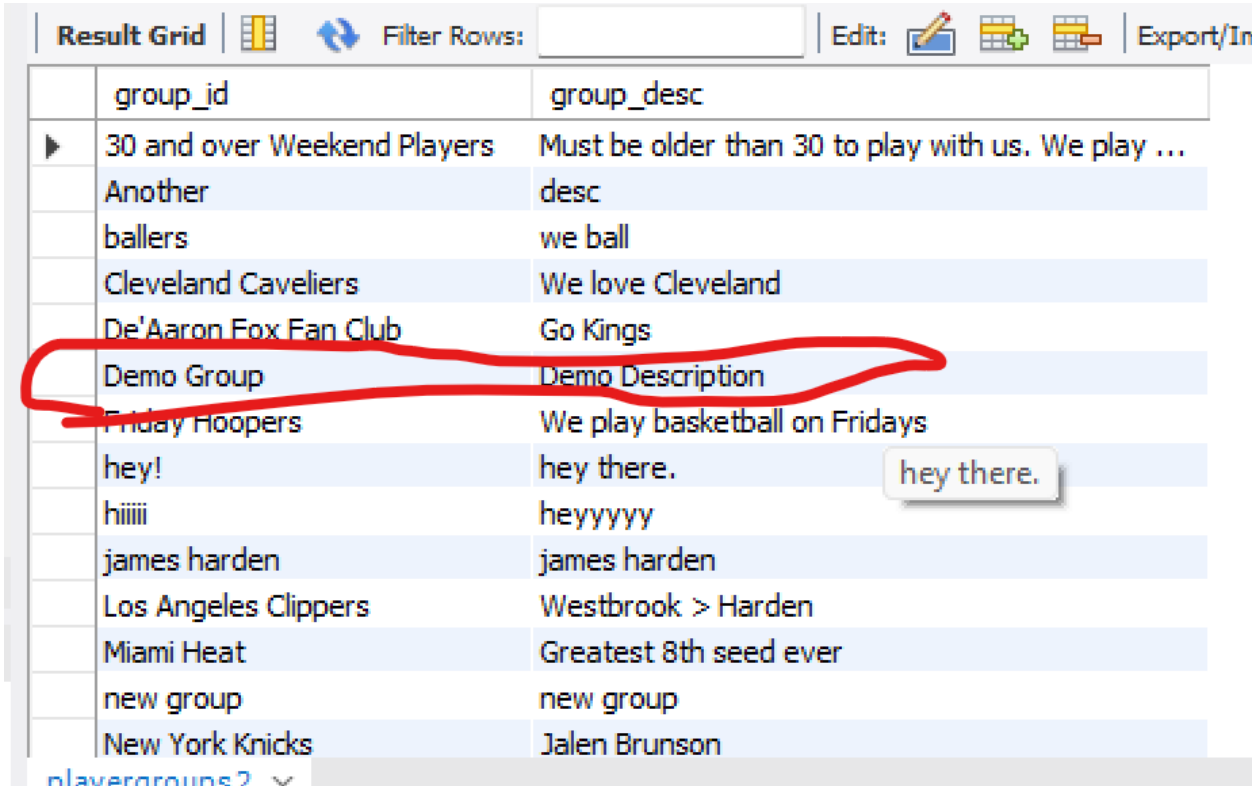




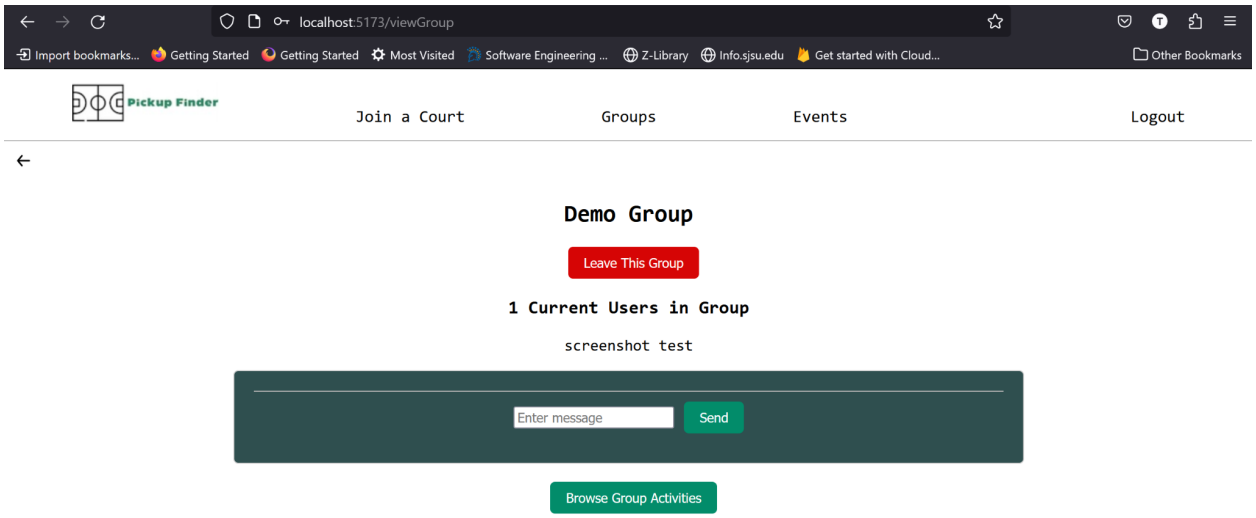
	user_id	court_id
▶	user2	1
	user3	1
	user1	2
	user4	2
	user5	2
	user99	2
	Tyler	3
	user6	3
	user7	3
	user10	4
	user8	4
	user9	4
	Tyler	5
	Tyler	6

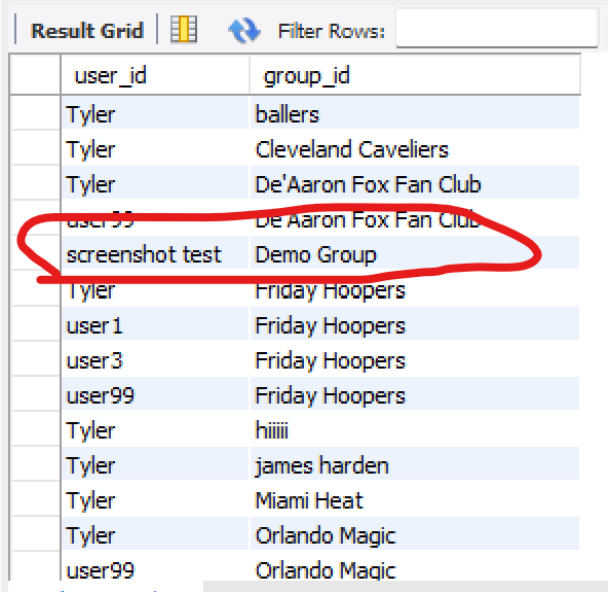
Creating a group: The user can enter the credentials for a new group. It will appear in the ‘playergroups’ table.





Joining a group: When a user joins a group, a row in ‘usersingroup’ will be created with the values of the user and the group that the user joined. Alternatively, a user can leave the group and the row will be deleted, the same way that it would work for the previously mentioned ‘usersoncourt.’

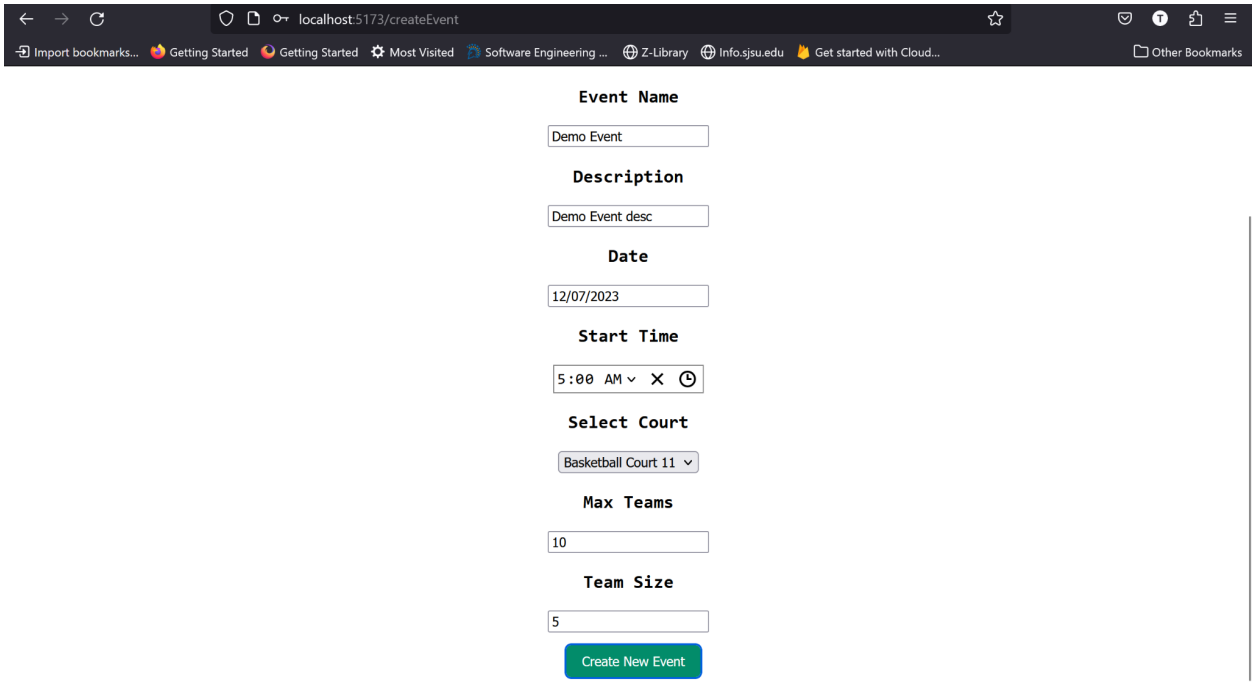




Result Grid | Filter Rows:

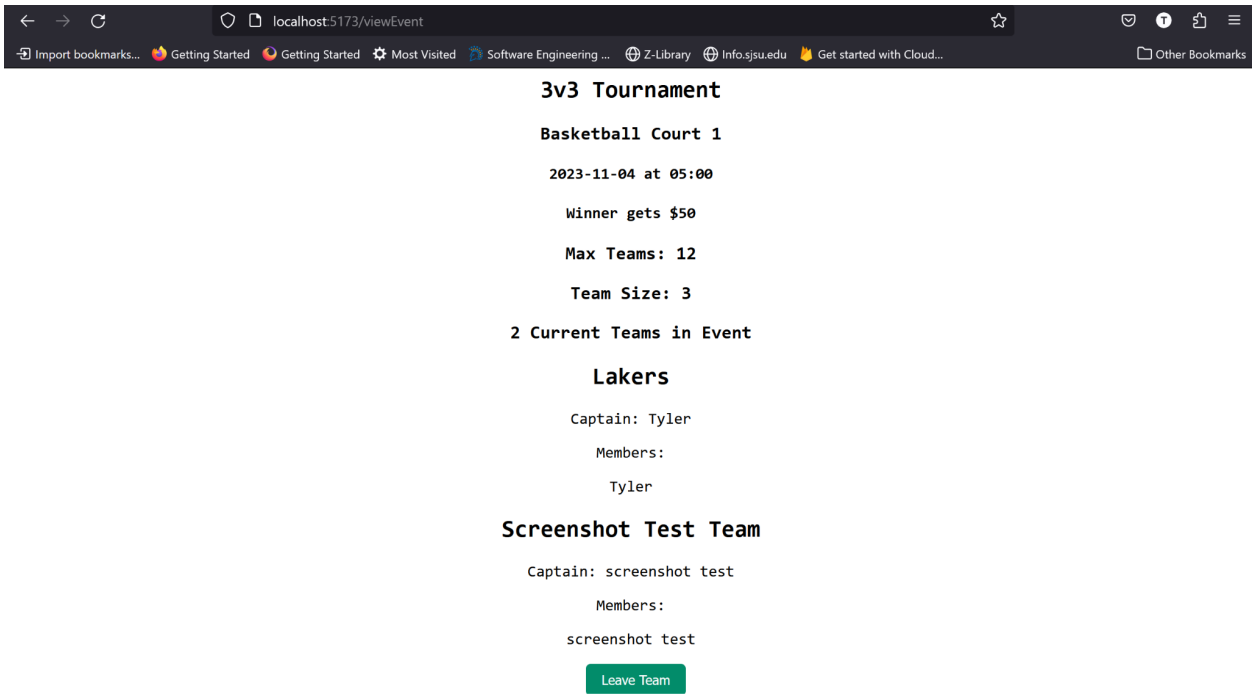
user_id	group_id
Tyler	ballers
Tyler	Cleveland Caveliers
Tyler	De'Aaron Fox Fan Club
[illegible]	De'Aaron Fox Fan Club
screenshot test	Demo Group
Tyler	Friday Hoopers
user1	Friday Hoopers
user3	Friday Hoopers
user99	Friday Hoopers
Tyler	hiiiii
Tyler	james harden
Tyler	Miami Heat
Tyler	Orlando Magic
user99	Orlando Magic

Creating an Event: Users have the option to create their own events if they input the required information. It will insert a new row in ‘playerevents’



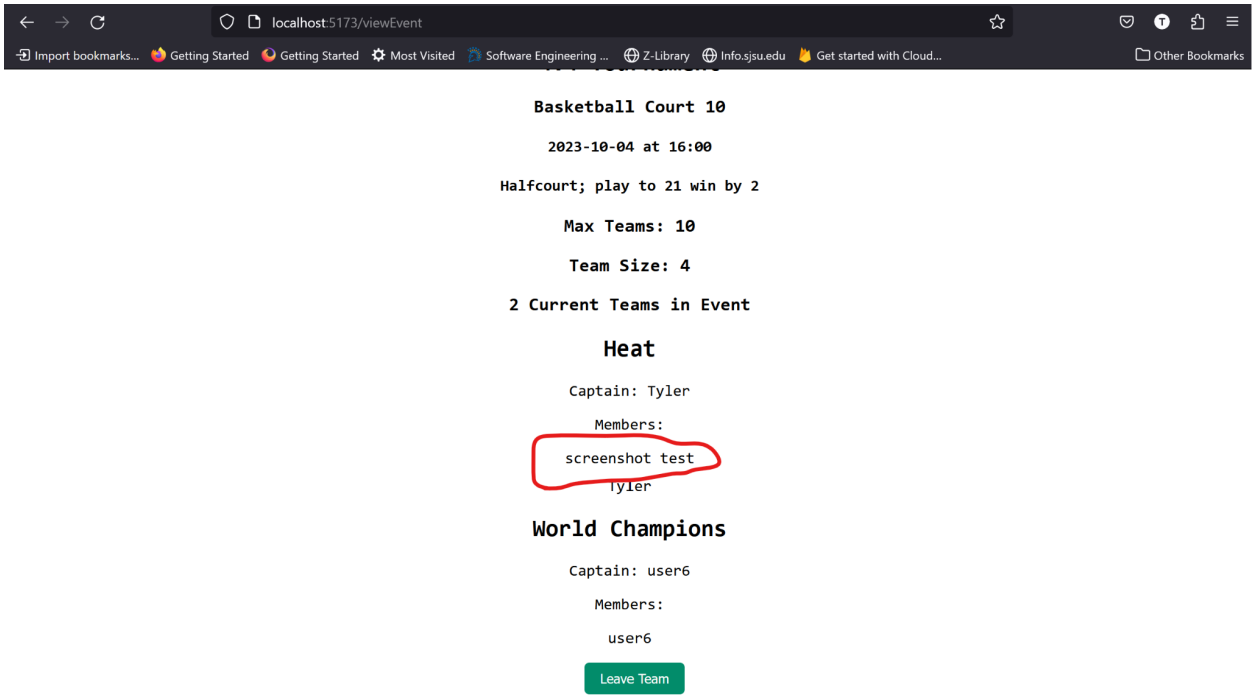
event_id	event_name	court_id	event_date	max_teams	team_size	event_desc
37	3pt Shooting Contest	3	2023-11-05 14:00:00	20	1	Most 3's out of 30 attempts wins
38	5v5 Tournament	7	2023-11-04 11:00:00	36	5	Competitive bracket tournament, full team
39	4v4 Tournament	10	2023-10-04 09:00:00	10	4	Halfcourt; play to 21 win by 2
40	5v5 Under 5'11 Tournament	6	2023-11-05 05:00:00	16	5	Must be under 5'11 to play. We're bring a
41	3v3 Over 30 Tournament	9	2023-11-06 04:00:00	30	3	Must be over 30 to play; Must have a full t
42	2v2 Over 7ft Tournament	7	2023-10-03 09:30:00	4	2	Must be 7ft+ to play.
43	New Event	7	2023-11-06 18:00:00	5	5	New event
44	Test 5	15	2023-11-03 05:00:00	10	4	Hi
45	New event again again	13	2023-11-03 05:00:00	11	12	desc
46	hi hi hi	15	2023-11-03 05:00:00	10	10	description
47	Demo Event	11	2023-11-04 05:00:00	10	5	Demo Event desc
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Creating a team: Teams are a weak entity set of events, users have the opportunity to create a team within an event as long as the event isn't full. They can do so by viewing an event, entering a team name and pressing create team. The user that creates the team will also automatically be placed on the team. This creates a new row in the 'teams' table.



event_id	team_id	captain_name
41	3v3 legends	Tyler
37	boom whats up	Tyler
35	Champions	Tyler
39	Heat	Tyler
43	hop	Tyler
34	Kings	Tyler
33	Lakers	Tyler
35	Lakers	user1
34	My Team	user00
33	Screenshot Test Team	screenshot test
40	Short Kings	user6
42	Tall People	user6
37	Team 6	user6
34	User 10's Team	user10

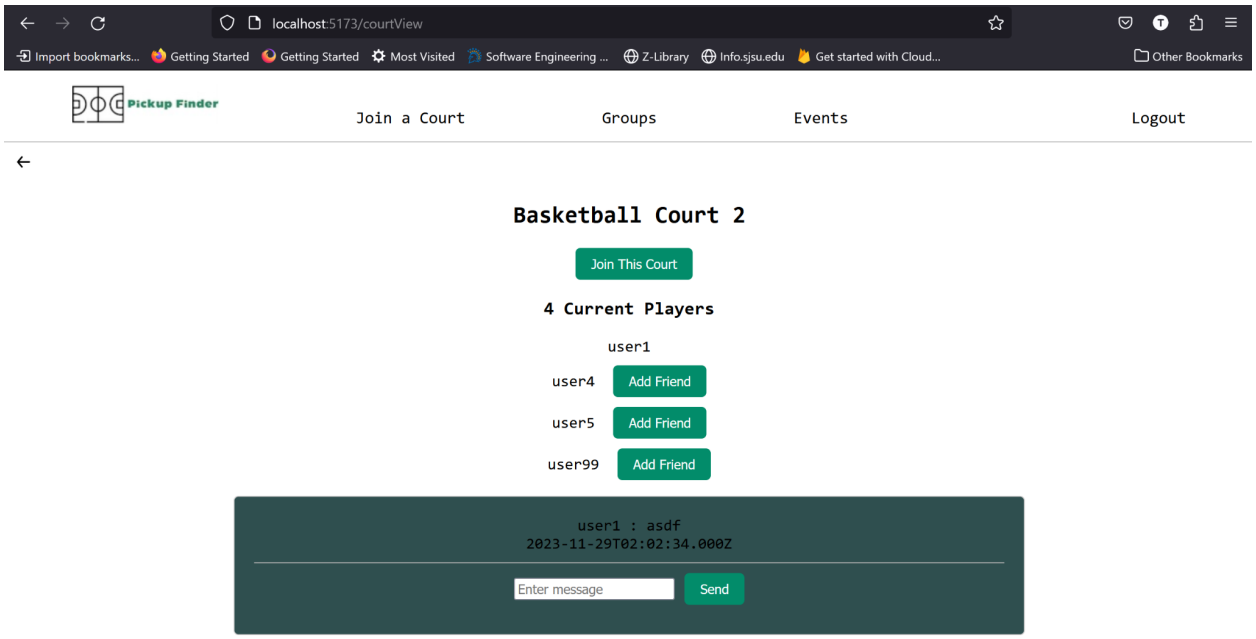
Joining and Leaving Teams: Users can join teams created by other users as long as they are not already on a team in the same event. The can do this by pressing the “Join Team” button under any team that is not full. This action creates a row in the ‘usersonteam’ table with the event id, the user’s id, and the team’s id. The player is also able to leave the team, which will delete that row from the table.

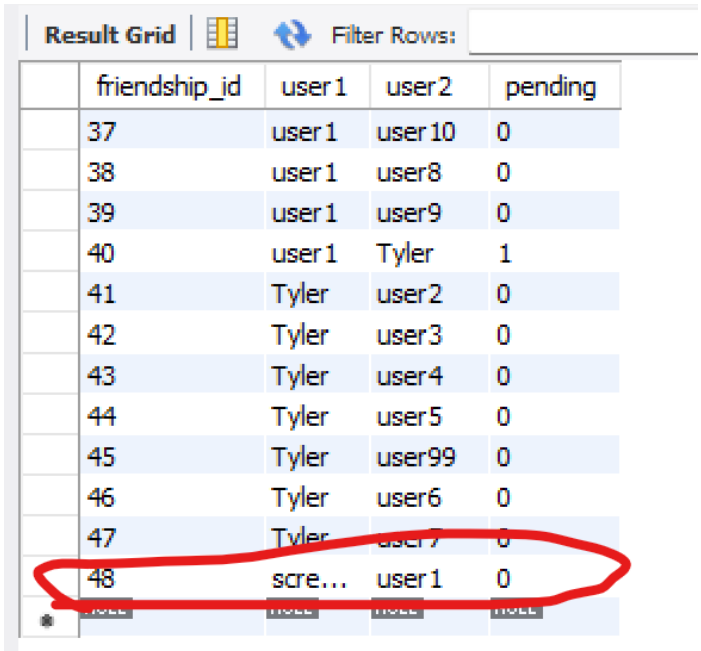


Result Grid | Filter Rows: | Edit:

user_id	team_id	event_id
Tyler	Lakers	33
user10	User10's Team	34
user99	My Team	34
Tyler	Champions	35
user10	Winner	35
Tyler	boom whats up	37
user6	Team 6	37
user6	We're just here to have fun	38
screenshot test	Heat	39
Tyler	Heat	39
user6	World Champions	39
user6	Short Kings	40
Tyler	3v3 legends	41
user6	Tall People	42

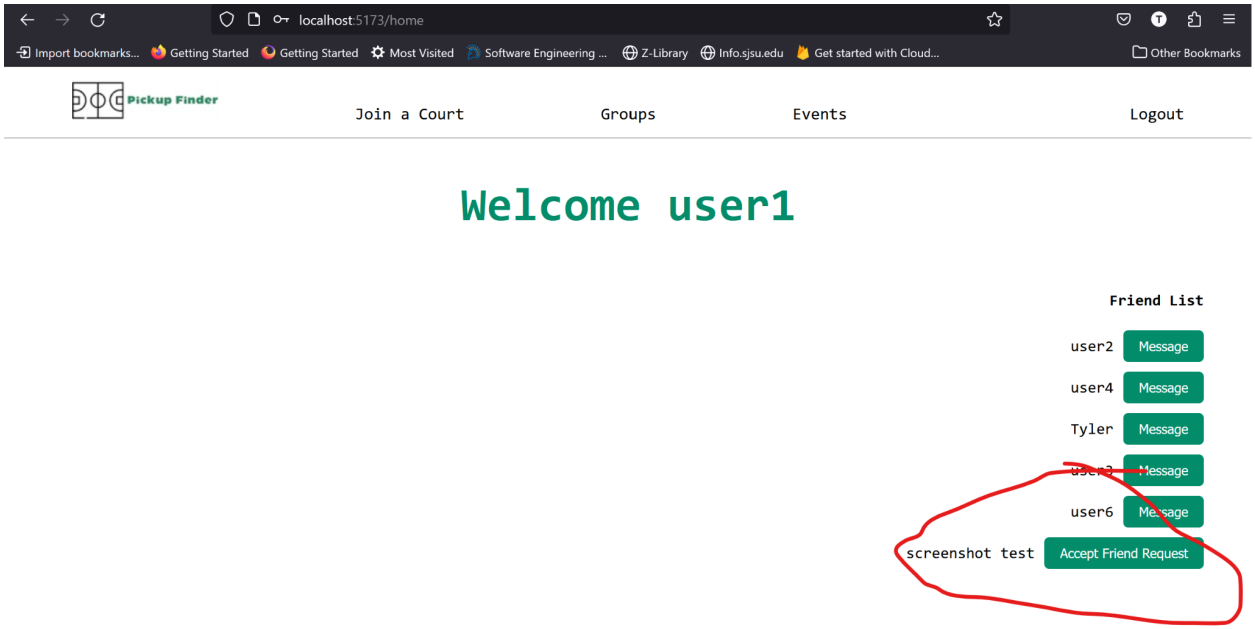
Sending Friend Requests: User's can click on the "Add Friend" button next to any of the players seen on a court or in a group. This will create a new row in the 'friends' table, and the 'pending' attribute will be defaulted to false, implying that the friend request is not complete because it is awaiting confirmation.

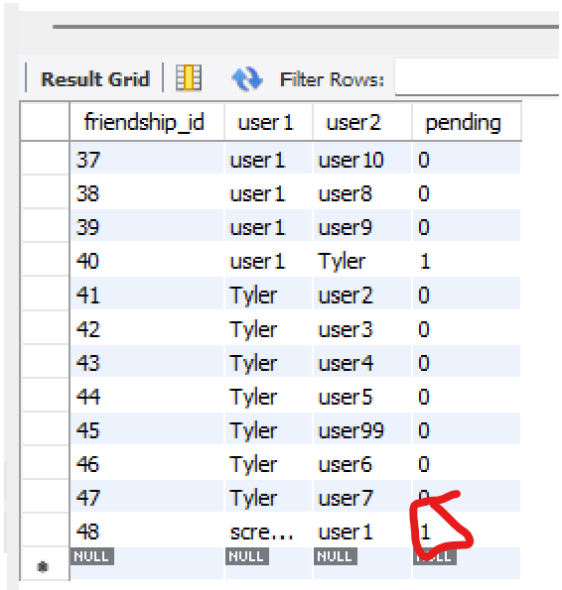




friendship_id	user1	user2	pending
37	user1	user10	0
38	user1	user8	0
39	user1	user9	0
40	user1	Tyler	1
41	Tyler	user2	0
42	Tyler	user3	0
43	Tyler	user4	0
44	Tyler	user5	0
45	Tyler	user99	0
46	Tyler	user6	0
47	Tyler	user7	0
48	scre...	user1	0

Accepting Friend Requests: The user that receives the friend request (this is user1 in the case of the previous example) will see a pending request in their friend list. After pressing "Accept Friend Request," the row that was created when the previous user sent the request will be updated to show that it is no longer pending. Once the friendship is not pending, the two users are allowed to message each other.

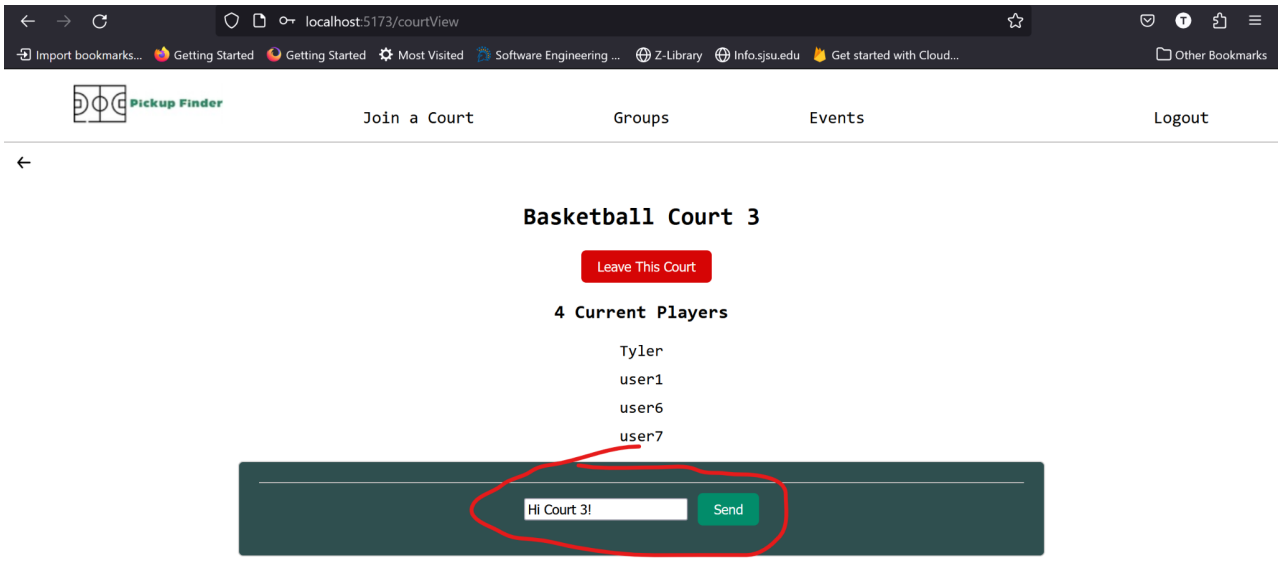




friendship_id	user1	user2	pending
37	user1	user10	0
38	user1	user8	0
39	user1	user9	0
40	user1	Tyler	1
41	Tyler	user2	0
42	Tyler	user3	0
43	Tyler	user4	0
44	Tyler	user5	0
45	Tyler	user99	0
46	Tyler	user6	0
47	Tyler	user7	0
48	scre...	user1	1
NULL	NULL	NULL	NULL

Sending messages: For the sake of repetitiveness, I will show only the messages related to courts. However the process is mostly identical for functionalities in groups and friends.

Once the user is on a page with a chatbox available, they can send a message to other people with access to that chat. When the user writes their message and presses send, a new row will be inserted into messages. It has an auto incrementing primary key, and includes data about the sender, content of the message, and time sent. A row will also be inserted into (in this case) 'messageincourt' with the message id and the court id.



Messages:

message_id	sender_id	content	time_sent
178	Tyler	pee yew	2023-12-06 23:21:01
179	Tyler	p.u.	2023-12-06 23:21:04
180	Tyler	this guy stinks	2023-12-06 23:21:08
181	Tyler	yikes	2023-12-06 23:21:09
182	Tyler	another airball	2023-12-06 23:21:13
183	Tyler	holy smokes	2023-12-06 23:21:25
184	Tyler	hit a shot	2023-12-06 23:21:28
185	Tyler	play defense	2023-12-06 23:21:30
186	Tyler	ruin another franchise	2023-12-06 23:21:40
187	Tyler	booooo	2023-12-06 23:21:42
188	Tyler	this guys the wrost	2023-12-06 23:21:49
189	user1	Hi Court 3!	2023-12-07 00:23:14
NULL	NULL	NULL	NULL

MessageInCourt:

Result Grid | Filter Rows:

message_id	court_id
142	1
163	1
164	1
169	1
170	1
188	2
189	3
143	10
144	10
145	10
146	10
147	10
148	10
149	10

Refresh dat

ssageincourt 1

Lesson Learned:

Tyler-

I learned a lot about developing full-stack applications start to finish. I think that my biggest takeaway was that the initial planning phase can drastically change the development process, and if the project is well planned then development can move much faster. Having a well structured database allowed us to completely switch development platforms without feeling like we completely restarted. I also learned a lot about making dynamic queries for the application. I felt like for each component that I implemented, I had to take a unique approach for querying the database in a way that is most efficient for the user and database.

Cameron-

In this project, I learned a lot about designing a database system. Before taking this course, I treated database tables similarly to objects where each row was an instance of the object. After taking this course I realized that my old database designs held many redundancies and were overall inefficient. I also learned a lot about creating SQL statements, triggers, and views. The combination of these three concepts made for efficient queries and fewer anomalies in our project development. With my new knowledge in database design and querying, I feel confident in my ability to create robust and efficient backends moving forward.

Se Chang-

For the first time, I learned about databases through this class. It was nice to learn how to design a database and use SQL queries. I'm glad I learned the professional way to design a database and use it. This project was the first experience in which I implemented a database into a project. It

was nice that I had a chance to implement what I learned in this class through this project. Also, It was my first time using React.js. I'm happy that I learned another tool that I can use. I learned so many things through this course and project. It will be really helpful to work on projects in the future.