

# **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**

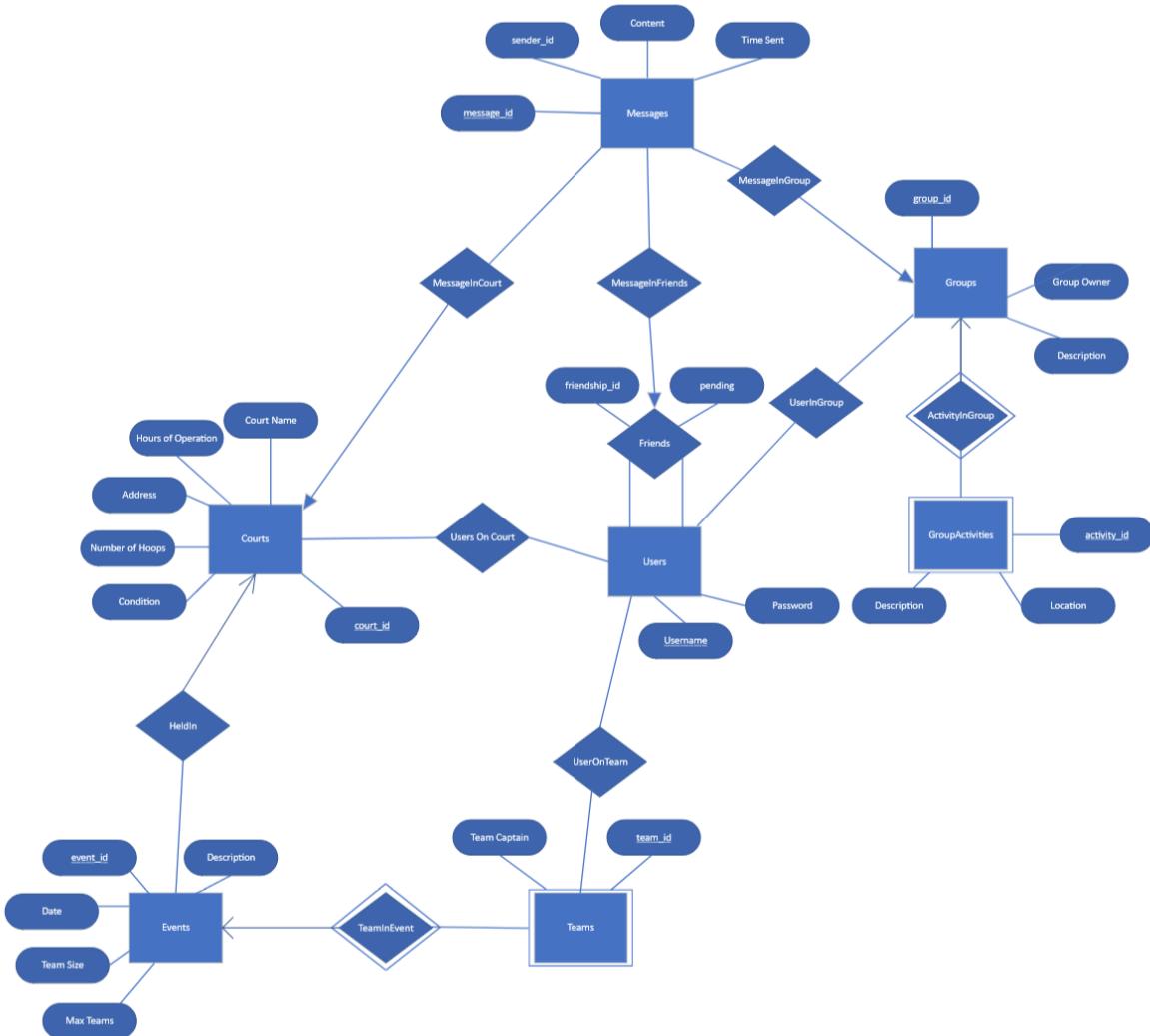
This table will hold information of which users are friends with each other. It is a many to many relationship within the Users entity.

- 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**

## 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

The screenshot shows the MySQL Workbench interface with the 'pickupfinder' database selected. The 'Schemas' tree on the left shows the 'pickupfinder' schema expanded, revealing tables like 'courts', 'events', 'friends', 'groups', 'messages', 'users', and 'views'. The 'courts' table is selected in the main query editor window.

**Result Grid:**

court_id	court_name	address	court_condition	num_holes	hours_of_operation
1	Basketball Court 1	123 Main St	Good	2	8:00 AM - 8:00 PM
2	Basketball Court 2	456 Elm St	Fair	3	8:00 AM - 8:00 PM
3	Basketball Court 3	789 Oak St	Poor	4	7:00 AM - 9:00 PM
4	Basketball Court 4	234 Pine St	Med	1	8:00 AM - 7:00 PM
5	Basketball Court 5	555 Hwy St	Excellent	2	7:00 AM - 10:00 PM
6	Basketball Court 6	777 Cedar St	Excellent	5	8:00 AM - 7:00 PM
7	Basketball Court 7	333 Birch St	Med	1	8:00 AM - 8:00 PM
8	Basketball Court 8	222 Bee St	Good	2	8:00 AM - 8:00 PM
9	Basketball Court 9	666 Cedar St	Excellent	2	8:00 AM - 10:00 PM
10	Basketball Court 10	444 Spruce St	Bad	2	10:00 AM - 1:00 PM
11	Basketball Court 11	123 Oak Rd	Good	1	24/7
12	Basketball Court 12	555 Pine St	Good	1	8:00 AM - 4:00 PM
13	Basketball Court 13	992 Whelock P...	Good	2	Sunrise to Sunset
14	Basketball Court 14	333 Birch St	Med	1	12:00 PM - 8:00 PM
15	Basketball Court 15	555 Hwy St	Good	6	12:00 PM - 8:00 PM
16	Basketball Court 16	222 Cedar St	Good	6	12:00 PM - 8:00 PM

**Output Tab:**

```

Action Output
Time Action
1 22:51:55 SELECT * FROM pickupfinder.courts LIMIT 0,1000
2 22:52:30 SELECT * FROM pickupfinder.courts LIMIT 0,1000
3 22:52:40 SELECT * FROM pickupfinder.courts LIMIT 0,1000
Message
10 rows returned
10 rows returned
10 rows returned
Duration / Fetch
0.000 sec / 6.000 sec
0.000 sec / 6.000 sec
0.000 sec / 6.000 sec

```

## Events

MySQL Workbench

Dunning - Training - not supp... X

File Edit View Query Database Server Tools Scripting Help

Connections Databases Schemas Tables Functions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid | Filter Rows: SELECT \* FROM pickupFinder.playerevents

event_id	event_name	cost	start_time	end_time	max_teams	team_size	event_desc
13	3v3 Tournament	1	2023-11-03 22:00:00	22			Winner gets \$50
34	3v3	9	2023-11-04 12:00:00	100	1		No team's bracket
24	3v3	4	2023-11-04 12:00:00	100	4		Must be over 18 to register at the event
36	2v2 Tournament	2	2023-11-06 07:00:00	30	2		Bracket Style Tournament
37	3v3 Shooting Contest	3	2023-11-05 14:00:00	20	1		Must 3s out of 30 attempts wins
38	3v3	7	2023-11-05 14:00:00	20	3		Must be over 18 to register at the event. Full team required
39	4v4 Tournament	10	2023-04-09 06:00:00	30	4		Halfhour play to 21 win by 2
40	3v3 Over 20 Tournament	10	2023-04-09 06:00:00	30	3		Must be over 18 to register at the event. Must bring a team
41	3v3 Over 20	9	2023-11-04 04:00:00	2023-11-05 04:00:00	3		Must be over 20 players. Must have a full team
42	2v2 Over 20 Tournament	7	2023-05-06 06:00:00	30	2		Must be 20+ to play
43	Test 5	18	2023-11-03 06:00:00	30	4		
44	Test 5	15	2023-11-03 06:00:00	30	14		
46	Hi Hi	13	2023-11-03 06:00:00	30	20		description

Addressation: Schemas Information

No object selected

playerevents + X

Output

Action Output

2 rows(s) returned

5 23/01/09 SELECT \* FROM pickupFinder.playerevents LIMIT 0, 1000

6 23/01/21 SELECT COUNT(\*) FROM pickupFinder.playerevents LIMIT 0, 1000

7 23/01/34 SELECT \* FROM pickupFinder.playerevents LIMIT 0, 1000

1 row(s) returned

14 row(s) returned

0.000 sec / 0.000 sec

0.000 sec / 0.000 sec

0.000 sec / 0.000 sec

## Friends

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The "friends" schema is selected.
- Tables:** The "friends" table is shown with columns: friends\_id, user1, user2, pending.
- Query Grid:** A result grid displays 46 rows of data from the "friends" table.
- Output Tab:** Shows three recent queries:
  - 21:51:57: SELECT \* FROM pickupFinder.friends LIMIT 0, 1000 (4 rows returned)
  - 21:52:38: SELECT \* FROM pickupFinder.friends LIMIT 0, 1000 (11 rows returned)
  - 23:03:15: SELECT \* FROM pickupFinder.friends LIMIT 0, 1000 (10 rows returned)

## Group

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** playerfinder
- Query Editor:** A query is running: `SELECT * FROM pickupfinder.playergroups;`
- Result Grid:** The results show various group names and their descriptions. Some entries are highlighted in blue.
- Output:** The execution details are listed as follows:

Action	Time	Message	Duration / Error
10	23:53:15	SELECT * FROM pickupfinder.friends LIMIT 0, 1000	0.000 sec / 0.000 sec
11	23:53:36	SELECT * FROM pickupfinder.playergroups LIMIT 0, 1000	0.000 sec / 0.000 sec
12	23:54:32	SELECT * FROM pickupfinder.playergroups LIMIT 0, 1000	0.000 sec / 0.000 sec

## Group Activities

The screenshot shows the MySQL Workbench interface with the following details:

- File**: Dunning\_Vining - not supp.
- Server**: Tools Scripting Help
- Schemas**: Shows databases: counts, departments, levels, players, procedures.
- Procedures**: Shows a procedure named getActivities.
- Functions**: Shows a function named getActivities.
- Tables**: Shows a table named activities.
- Data**: Displays data from the activities table:

activity_id	activity_desc	location	group_id
1	1-on-1 T-Ball	Round Rock Park	T-Ball Hoopers
2	b	Basketball Court 1	Friday Hoopers
3	activity	Basketball Court 1	Basketball Hoopers
4	description	Basketball Court 1	Another
5	ball	were gonna ball	Basketball Court 1
6	base	white up	Basketball Court 1
7	dunk	d	Basketball Court 1
8	Dunk	I can't dunk	Basketball Court 7
9	hoop	hoop	Basketball Court 1
10	hoop	lets hoop	Basketball Court 1
11	Hooper	lets play	Basketball Court 1
12	activity	activity	Basketball Court 1
13	league game	ad pro	Basketball Court 1
14	league of legend	peuse	Basketball Court 1
15	team	team	Basketball Court 1
16	new activity	new	Basketball Court 1
17	activity	30 and over ...	Basketball Court 1
18	part	party	Basketball Court 1
19	Practice Field Pk.	Bring wacks	Basketball Court 3
20	pk	pk	Basketball Court 1
21	team	team	Basketball Court 1
22	Warriors v. Vikings	bring popcorn	Basketball Court 1
23	team	team	Basketball Court 1

- Logs**: Shows three recent log entries:

Time	Action	Message	Duration / Fetch
13 23:06:46	SELECT * FROM `pk_dunring_groupactivities` LIMIT 0, 1000	20 rows returned	0.000 sec / 0.000 sec
14 23:06:54	SELECT * FROM `pk_dunring_groupactivities` LIMIT 0, 1000	21 rows returned	0.000 sec / 0.000 sec
15 23:07:00	SELECT * FROM `pk_dunring_groupactivities` LIMIT 0, 1000	22 rows returned	0.000 sec / 0.000 sec

- Right Sidebar**: Includes tabs for 'SQL Additions', 'Automatic Context Help' (disabled), 'Code Editor', 'Query Plan', and 'Execution Plan'.

## Messages

The screenshot shows the MySQL Workbench interface with the 'messages' table selected. The table has three columns: message\_id, sender\_id, and content. The data is as follows:

message_id	sender_id	content	time_sent
143		hi	2023-11-28 12:22:43
145	Tyler	H out!	2023-11-28 12:33:38
146		we shu gone playing games or just shooting or...	2023-11-28 12:33:52
147	Tyler	huh like is there some shooting there?	2023-11-28 12:33:59
148		oh yeah we gonna shoot some more now!	2023-11-28 12:34:06
149	Tyler	Who's gonna guard me?	2023-11-28 12:34:18
150		where do you gonna park?	2023-11-28 12:34:44
151	Tyler	Good game.	2023-11-28 12:35:00
152		you're welcome	2023-11-28 12:35:00
153	user1	hi	2023-11-28 12:35:00
154		hello	2023-11-28 12:35:00
155	user1	Hello	2023-11-28 12:35:00
156	user1	what up	2023-11-28 12:35:00
157	user1	whats up see back	2023-11-28 12:35:00
158	user1	where you go	2023-11-28 12:35:19
159		hey	2023-11-28 12:35:19
160	user1	heloooo??	2023-11-28 12:35:24
161	user1	user1	2023-11-28 12:35:24
162	user1	test2	2023-11-28 12:35:24
163	user1	We're playing 2v2s, come jan	2023-11-28 12:29:58
164		ok	2023-11-28 12:29:58
165	C15738	Hello	2023-11-28 16:53:13
166		Hi	2023-11-28 16:53:13
167	user4	Hi	2023-11-28 16:53:20
168	user1	asif	2023-11-28 18:02:24
169	user1	asif	2023-11-28 18:02:24

The bottom pane shows the output of a query:

```
Output
=====
14:23:56.54 | [Info] 0 rows affected, 1 row(s) returned
14:23:57.00 | [Info] 1 row(s) returned
14:23:57.17 | [Info] 20 rows(s) returned
14:23:57.17 | [Info] 0 rows(s) returned
14:23:57.17 | [Info] 27 rows(s) returned
14:23:57.17 | [Info] 0 rows(s) returned
```

## Teams

The screenshot shows the MySQL Workbench interface with a query editor window open. The query is:

```
SELECT *
FROM pickupFinder.teams;
```

The results grid displays the following data:

event_id	team_name
11	3x3 tagup
27	bison whata up
28	biggopop
29	biggopop
39	Heed
43	ho
54	Tyler
33	Lakers
55	Lakers
34	My Team
40	My Team
42	My Team
27	Team 6
34	Team 6
38	We're just here ...
29	World Champs
39	World Champs
2018	0000

The bottom pane shows the execution history:

- 18 23:08:14 SELECT \* FROM pickupFinder.teams LIMIT 0, 1000
- 18 23:08:46 SELECT COUNT(\*) FROM pickupFinder.teams LIMIT 0, 1000
- 26 23:08:52 SELECT \* FROM pickupFinder.teams LIMIT 0, 1000

## UserInGroup

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is `pickupfinder`.
- Query Editor:** A query is selected: `SELECT * FROM pickupfinder.user; (empty)`.
- Result Grid:** The results of the query are displayed in a grid format. The columns are `user_id` and `group`. The data includes:
 

user_id	group
20	30 and over Weekend Players
21	30 and over
22	Tyler
23	baffin
24	Devin Cawthon
25	De Aron Fox Fan Club
26	De Aron Fox
27	user99
28	Fridays Hoopers
29	Fridays Hoopers'
30	Friday Hoopers
31	Hall
32	Jayson harden
33	Marcus Head
34	Orlando Magic
35	Sacramento Kings Fans
36	Utah Jazz
37	2020
- Object Info:** Shows the session information.

## UserOnTeam

MySQL Workbench

Running - 'Training' - not supp... X

File Edit View Query Databases Server Tools Scripting Help

Connections Friends Properties Messages Items useringroup useronteam Table Search

SCHEMAS: pickupfinder

Q Filter objects

- classmodels
- datatypes
- domains
- functions
- help
- procedures
- roles
- schemas
- views
- users
- Stored Procedures
- Functions

Q Filter objects

Result Grid | Filter Rows: Edit Export/Imports View Cell Context

User_ID	Team_ID	event_id
Tyler	Kings	32
Tyler	Knights	34
user19	Knights Team	34
user19	My Team	34
Tyler	Champions	35
user19	Knights	35
Tyler	boom what's up	37
user19	Knights	37
user19	We're just here to have fun	38
Tyler	Knights	39
user19	World Champions	39
user19	Short Kings	40
Tyler	Knights	41
user19	Tall Knights	42
Tyler	Knights	42
user19	Knights	42

Administration Schemas Information

No object selected

useronteam 1 x

Output

Action Output

- Drop Table
- Add Column
- 21 23:09:12 SELECT \* FROM pickupfinder.useronteam LIMIT 0, 1000
- 22 23:09:52 SELECT \* FROM pickupfinder.useronteam LIMIT 0, 1000
- 23 23:10:08 SELECT \* FROM pickupfinder.useronteam LIMIT 0, 1000

Memory 10 rows returned 0.000 sec / 0.000 sec

Memory 19 rows returned 0.000 sec / 0.000 sec

Memory 15 rows returned 0.000 sec / 0.000 sec

Object Info Session

## Users

The screenshot shows the MySQL Workbench interface with the 'users' table selected. The table has two columns: 'User\_Id' and 'password'. The data contains 100 rows, each with a unique User\_Id from user1 to user100 and a corresponding password.

User_Id	password
user1	p
user2	p
user3	p
user4	p
user5	p
user6	p
user7	p
user8	p
user9	p
user10	p
user11	p
user12	p
user13	p
user14	p
user15	p
user16	p
user17	p
user18	p
user19	p
user20	p
user21	p
user22	p
user23	p
user24	p
user25	p
user26	p
user27	p
user28	p
user29	p
user30	p
user31	p
user32	p
user33	p
user34	p
user35	p
user36	p
user37	p
user38	p
user39	p
user40	p
user41	p
user42	p
user43	p
user44	p
user45	p
user46	p
user47	p
user48	p
user49	p
user50	p
user51	p
user52	p
user53	p
user54	p
user55	p
user56	p
user57	p
user58	p
user59	p
user60	p
user61	p
user62	p
user63	p
user64	p
user65	p
user66	p
user67	p
user68	p
user69	p
user70	p
user71	p
user72	p
user73	p
user74	p
user75	p
user76	p
user77	p
user78	p
user79	p
user80	p
user81	p
user82	p
user83	p
user84	p
user85	p
user86	p
user87	p
user88	p
user89	p
user90	p
user91	p
user92	p
user93	p
user94	p
user95	p
user96	p
user97	p
user98	p
user99	p
user100	p

## UsersOnCourt

The screenshot shows the MySQL Workbench interface with the 'usersoncourt' table selected. The table has two columns: 'User\_Id' and 'court\_Id'. The data contains 100 rows, each with a unique User\_Id and a court\_Id value ranging from 1 to 15.

User_Id	court_Id
user1	1
user2	1
user3	2
user4	2
user5	2
user6	2
user7	3
user8	3
user9	4
user10	4
user11	4
user12	5
user13	5
user14	5
user15	5
user16	6
user17	6
user18	6
user19	7
user20	7
user21	8
user22	8
user23	9
user24	9
user25	10
user26	10
user27	11
user28	11
user29	12
user30	12
user31	13
user32	13
user33	14
user34	14
user35	15
user36	15
user37	15
user38	15
user39	15
user40	15
user41	15
user42	15
user43	15
user44	15
user45	15
user46	15
user47	15
user48	15
user49	15
user50	15
user51	15
user52	15
user53	15
user54	15
user55	15
user56	15
user57	15
user58	15
user59	15
user60	15
user61	15
user62	15
user63	15
user64	15
user65	15
user66	15
user67	15
user68	15
user69	15
user70	15
user71	15
user72	15
user73	15
user74	15
user75	15
user76	15
user77	15
user78	15
user79	15
user80	15
user81	15
user82	15
user83	15
user84	15
user85	15
user86	15
user87	15
user88	15
user89	15
user90	15
user91	15
user92	15
user93	15
user94	15
user95	15
user96	15
user97	15
user98	15
user99	15
user100	15

## MessageInCourt

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is 'pickupfinder'.
- Tables:** The 'messagecourt' table is selected.
- Query Editor:** The query is: `SELECT * FROM pickupfinder.messagecourt;`
- Result Grid:** The grid displays the following data:

message_id	court_id
140	1
363	1
141	1
369	1
170	1
168	2
143	10
144	10
145	10
146	10
147	10
148	10
149	10
200	10
151	10

- Output Panel:** Shows the execution history with three entries:

  - 28 23:19:14: SELECT COUNT(\*) FROM pickupfinder.messagecourt LIMIT 0, 1000
  - 29 23:19:20: SELECT \* FROM pickupfinder.messagecourt LIMIT 0, 1000
  - 30 23:19:38: SELECT \* FROM pickupfinder.messagecourt LIMIT 0, 1000

## MessageInGroup

The screenshot shows the MySQL Workbench interface with the following details:

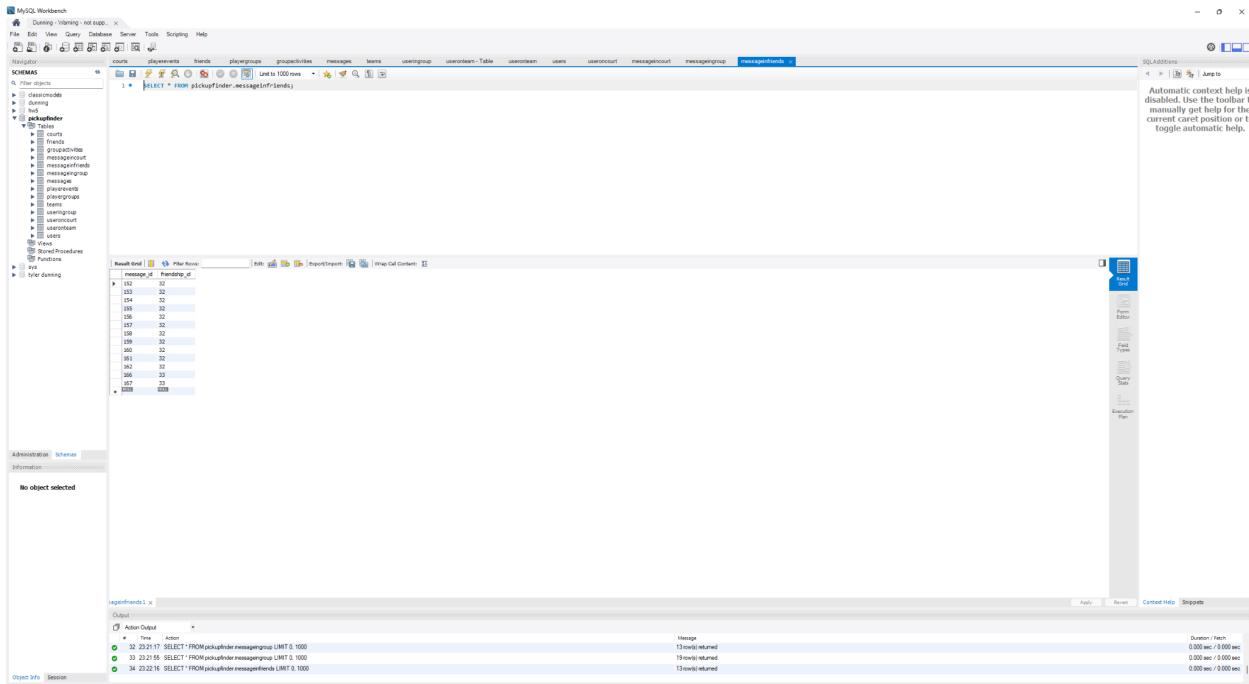
- Schemas:** The current schema is 'pickupfinder'.
- Tables:** The 'messagegroup' table is selected.
- Query Editor:** The query is: `SELECT * FROM pickupfinder.messagegroup;`
- Result Grid:** The grid displays the following data:

message_id	group_id
165	30 and over Weekend Players
171	Another
172	Another
173	Another
174	Another
175	Another
176	Another
177	Another
178	james harden
179	james harden
180	james harden
181	james harden
182	james harden
183	james harden
184	james harden
185	james harden
186	james harden
187	james harden
188	james harden

- Output Panel:** Shows the execution history with three entries:

  - 31 23:20:19: SELECT \* FROM pickupfinder.messagegroup LIMIT 0, 1000
  - 32 23:21:17: SELECT \* FROM pickupfinder.messagegroup LIMIT 0, 1000
  - 33 23:21:55: SELECT \* FROM pickupfinder.messagegroup LIMIT 0, 1000

## 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;
```

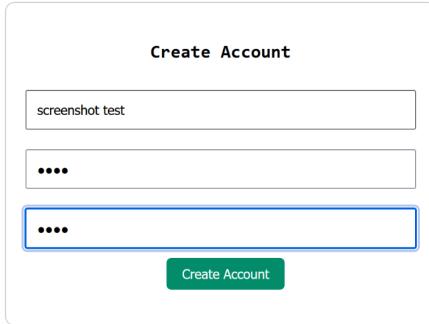
Once this is done, we can deploy our backend. Do this by navigating to the backend folder in a command prompt/terminal using “cd Backend” Once the directory has been selected, type ‘npm start’. (Assuming that npm is installed on the computer)

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’.

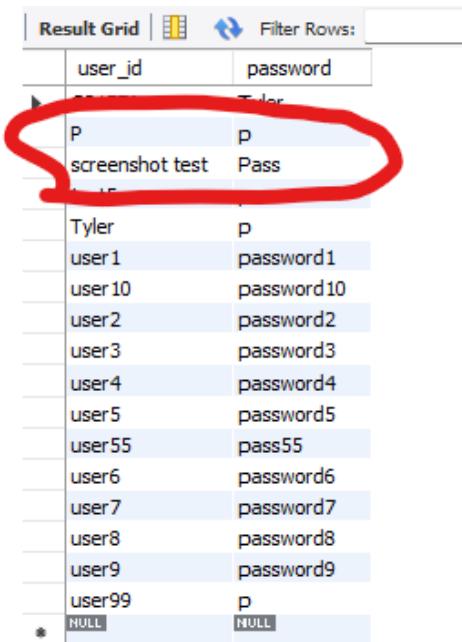
Once all of these steps have been completed, you should be able to click the link that has been output in the frontend terminal and use the website.

**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.



The screenshot shows a web browser window with the URL `localhost:5173/createAccount`. The page title is "Pickup Finder". The main content is a "Create Account" form with three fields: "username" containing "screenshot test", "password" containing "\*\*\*\*", and "confirm password" also containing "\*\*\*\*". The "confirm password" field is highlighted with a blue border. Below the form is a green "Create Account" button.



Result Grid		Filter Rows:
	user_id	password
	Tyler	P
	screenshot test	Pass
	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

The screenshot displays two main parts of the Pickup Finder application.

**Top Part (Court View):**

- Browser Header: localhost:5173/courtView
- Page Title: Pickup Finder
- Navigation: Join a Court, Groups, Events, Logout
- Section: Basketball Court 1
- Buttons: Leave This Court (red)
- Text: 3 Current Players
- Player List: screenshot test, user2, user3
- Message Box (Dark Green):
  - Tyler : Hello!
  - 2023-11-28T20:22:43.000Z
  - user1 : We're playing 2v2s, come join
  - 2023-11-28T23:29:58.000Z
  - CS157A : Hello!
  - 2023-11-29T00:43:07.000Z
  - Tyler : hi
  - 2023-12-07T07:19:27.000Z
  - Tyler : hi
  - 2023-12-07T07:19:30.000Z

**Bottom Part (Result Grid):**

user_id	court_id
screenshot test	1
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

A red circle highlights the first row of the result grid, specifically the 'screenshot test' entry.

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

The screenshot shows a web browser window for 'localhost:5173/courtView'. The title bar says 'localhost:5173/courtView'. The page header includes a logo, 'Join a Court', 'Groups', 'Events', and 'Logout'. Below the header is a back arrow. The main content area is titled 'Basketball Court 1' with a 'Join This Court' button. It displays '2 Current Players': 'user2' and 'user3', each with an 'Add Friend' button. A dark green sidebar contains a chat log:

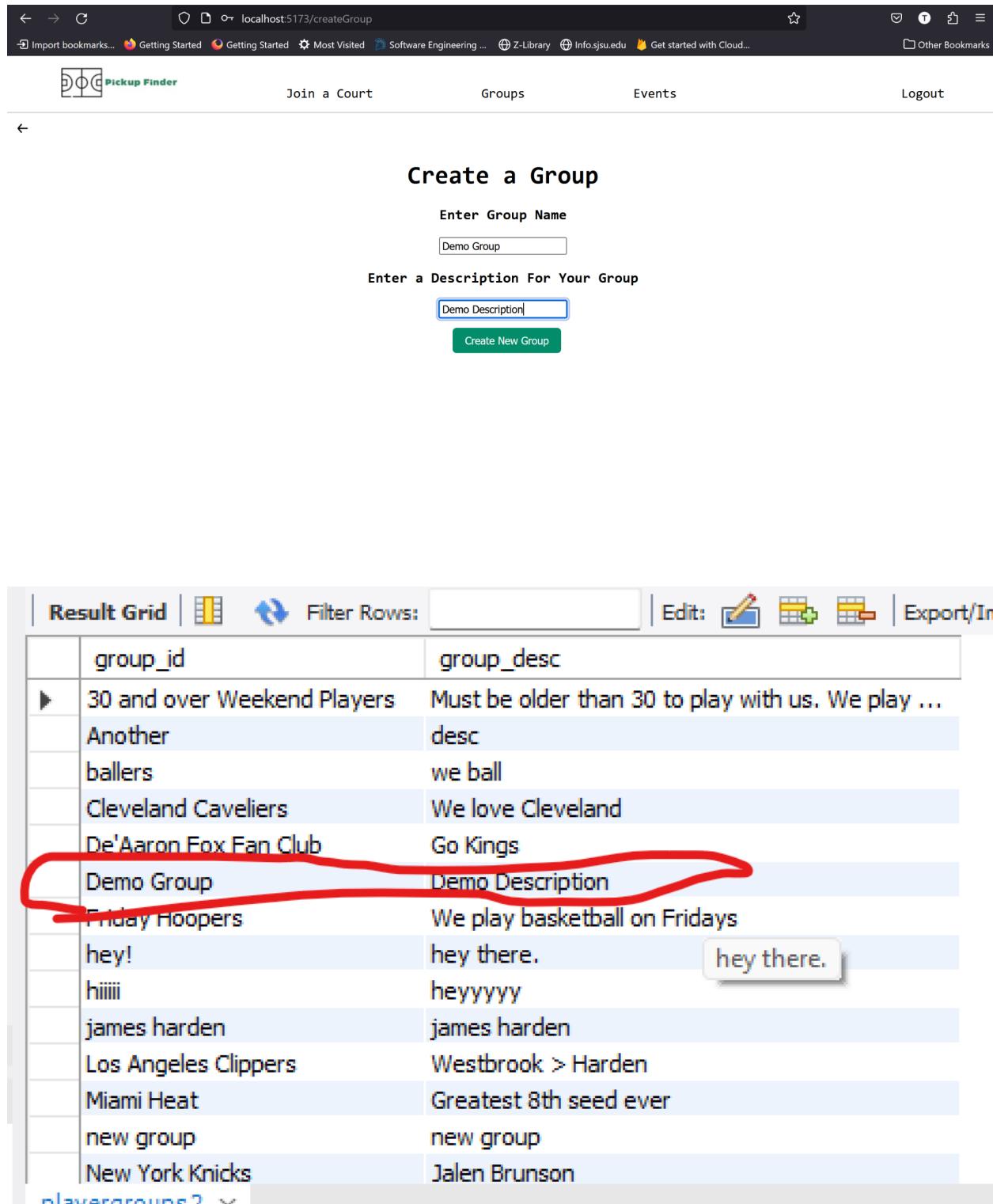
```

    Tyler : Hello!
    2023-11-28T20:22:43.000Z
    user1 : We're playing 2v2s, come join
    2023-11-28T23:29:58.000Z
    CS157A : Hello!
    2023-11-29T00:43:07.000Z
    Tyler : hi
    2023-12-07T07:19:27.000Z
    Tyler : hi
    2023-12-07T07:19:30.000Z
  
```

Below this is a 'Result Grid' section with a 'Filter' button. The grid has columns 'user\_id' and 'court\_id'. The data is as follows:

	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

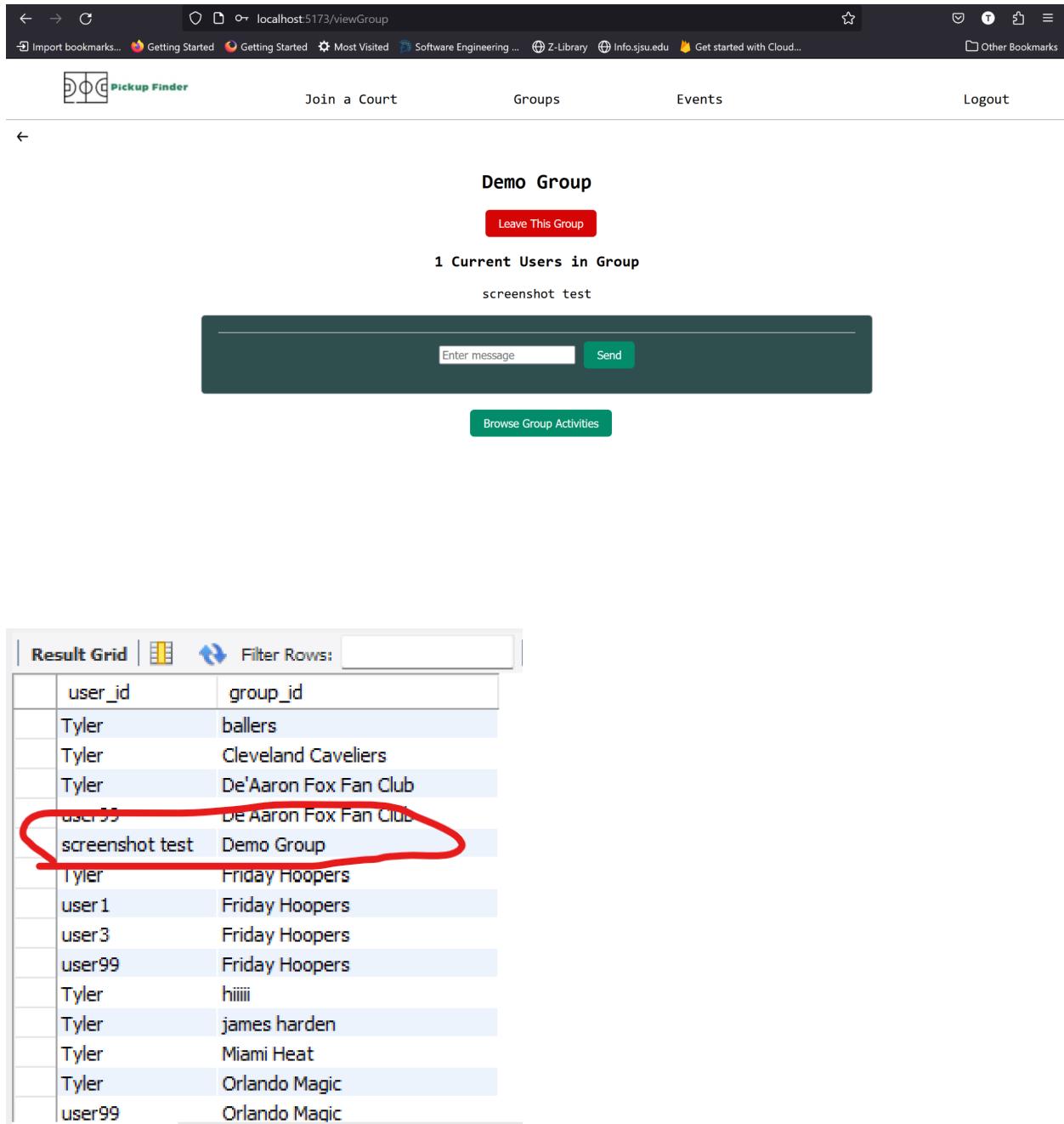
**Creating a group:** The user can enter the credentials for a new group. It will appear in the ‘playergroups’ table.



The screenshot shows the Pickup Finder application interface. At the top, there is a navigation bar with links for 'Join a Court', 'Groups', 'Events', and 'Logout'. Below this is a form titled 'Create a Group' with fields for 'Enter Group Name' (containing 'Demo Group') and 'Enter a Description For Your Group' (containing 'Demo Description'). A green button labeled 'Create New Group' is at the bottom of the form. The main area shows a 'Result Grid' table with columns for 'group\_id' and 'group\_desc'. The table contains several rows of data, with the row for 'Demo Group' circled in red. The circled row has 'group\_id' 'Demo Group' and 'group\_desc' 'Demo Description'. Other visible rows include '30 and over Weekend Players', 'Another', 'ballers', 'Cleveland Caveliers', 'De'Aaron Fox Fan Club', 'Friday Hoopers', 'hey!', 'hiiiii', 'james harden', 'Los Angeles Clippers', 'Miami Heat', 'new group', and 'New York Knicks'.

	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 Caveliers	We love Cleveland
	De'Aaron Fox Fan Club	Go Kings
	Demo Group	Demo Description
	Friday Hoopers	We play basketball on Fridays
	hey!	hey there.
	hiiiii	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

**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 ‘useroncourt.’



The screenshot shows two main parts of the Pickup Finder application.

**Top Section (Demo Group Page):**

- Header: localhost:5173/viewGroup
- Navigation: Join a Court, Groups, Events, Logout
- Section Title: Demo Group
- Buttons: Leave This Group
- Text: 1 Current Users in Group
- Text: screenshot test
- Input Field: Enter message
- Button: Send
- Link: Browse Group Activities

**Bottom Section (Result Grid):**

	user_id	group_id
Tyler	ballers	
Tyler	Cleveland Caveliers	
Tyler	De'Aaron Fox Fan Club	
user99	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 Maqic	

A red circle highlights the row for 'screenshot test' with 'Demo Group'.

**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’

localhost:5173/createEvent

**Event Name**  
Demo Event

**Description**  
Demo Event desc

**Date**  
12/07/2023

**Start Time**  
5:00 AM

**Select Court**  
Basketball Court 11

**Max Teams**  
10

**Team Size**  
5

**Create New Event**

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
*	HULL	HULL	HULL	HULL	HULL	HULL

**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.

localhost:5173/viewEvent

3v3 Tournament

Basketball Court 1

2023-11-04 at 05:00

Winner gets \$50

Max Teams: 12

Team Size: 3

2 Current Teams in Event

Lakers

Captain: Tyler

Members:

Tyler

Screenshot Test Team

Captain: screenshot test

Members:

screenshot test

Leave Team

	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	Smart Kings	user6
	42	Tall People	user6
	37	Team 6	user6
	34	User10'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. They 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.

The screenshot shows a web browser window with the URL `localhost:5173/viewEvent`. The page displays event details for "Basketball Court 10" on "2023-10-04 at 16:00". It specifies "Max Teams: 10" and "Team Size: 4". There are two current teams listed: "Heat" and "World Champions". The "Heat" team has a captain "Tyler" and members "screenshot test" and "tyler". The "World Champions" team has a captain "user6" and member "user6". A "Leave Team" button is visible for the "Heat" team.

Below the event details, there is a "Result Grid" table showing the 'usersonteam' data. The columns are `user_id`, `team_id`, and `event_id`. The data includes rows for various users joining different teams, with one specific row for "screenshot test" joining the "Heat" team highlighted with a red circle.

<code>user_id</code>	<code>team_id</code>	<code>event_id</code>
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.

The screenshot shows a web browser window for 'localhost:5173/courtView'. The page title is 'Basketball Court 2'. At the top, there are navigation links: 'Join a Court', 'Groups', 'Events', and 'Logout'. Below the title, a section titled '4 Current Players' lists four users: 'user1', 'user4', 'user5', and 'user99'. Each user has an 'Add Friend' button next to their name. A message box at the bottom shows a message from 'user1' with the content 'asdf' and timestamp '2023-11-29T02:02:34.000Z'. There is an input field 'Enter message' and a 'Send' button.

	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.

The screenshot shows the Pickup Finder application interface. At the top, there's a navigation bar with links like 'Join a Court', 'Groups', 'Events', and 'Logout'. Below the navigation bar, the main content area has a teal header 'Welcome user1'. To the right, under 'Friend List', there's a list of users with 'Message' buttons. A red oval highlights the 'Accept Friend Request' button for 'screenshot test'. In the bottom-left corner, there's a 'Result Grid' section showing a table with columns: friendship\_id, user1, user2, and pending. A red arrow points to the 'pending' column of the last row, which has a value of 1.

	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

Result Grid				Filter Rows:
	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

**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.

The screenshot shows a web browser window for the 'Pickup Finder' application. The URL in the address bar is 'localhost:5173/courtView'. The page title is 'Basketball Court 3'. At the top, there is a navigation bar with links for 'Join a Court', 'Groups', 'Events', and 'Logout'. Below the navigation bar, there is a section titled '4 Current Players' listing 'Tyler', 'user1', 'user6', and 'user7'. At the bottom of the page, there is a dark green chatbox containing a text input field with the placeholder 'Hi Court 3!' and a green 'Send' button. A red oval has been drawn around this chatbox area.

## 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	boooooo	2023-12-06 23:21:42
188	tyler	this guys the worst	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
	180	2
	189	3
	143	10
	144	10
	145	10
	146	10
	147	10
	148	10
	149	10

ssadeincourt 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.