

PROJECT STEP 6

Team Name:

Group 8

Team Members:

Chuck Loughin, Harvey Ng

URL to HTML pages: <http://flip3.engr.oregonstate.edu:4438/>

Note: must be connected to VPN

Executive Summary:

- Made gyms optional/nullable within sessions to allow users to log training sessions that happen outdoors and to delete the gym entry if it was entered in error for the given session.
- Created an entity-relationship diagram, as we originally included only a schema.
- Originally had primary keys set as NOT NULL and unique with auto_increment, removed NOT NULL and unique after feedback highlighted the redundancy.
- Made attempts optional for a given session, allowing a user to create a session (they otherwise would have been required to create an attempt at the same time as a session).
- Updated sample data to reflect nullable relationships.
- Added dropdowns for selecting foreign keys.
- Updated forms for users table to separate name fields and make placeholder text more human readable.
- Added a primary key to the users_gyms table to make CRUD operations easier to implement.
- Updated the queries used in the CRUD implementation for tables in the DML.sql file to reflect the actual queries used in the application.
- Updated sample data to showcase all edge-case relationships between entities.
- Fixed issue with editing a session where gym_name is null.
- Restricted users_gyms page to only allow edits to the gym_name.
- Added an error page if the database administrator tries to add or edit in a duplicate of an existing relationship within the users_gyms table.
- Removed delete from gyms based on internal discussion around the potential business logic.
- Added constraint to make email unique in users table, and an associated error page if the database administrator tries to add or edit in a user with the same email as an existing user.
- Added a README to the project with a basic introduction and broad citations.
- Added some additional text/captions to make pages more descriptive, based on feedback.

- Made updates to our project outline, entity diagram, schema and example data in accordance with feedback and all above changes and to ensure consistency across the design document.

Project Outline:

Project Title:

Climbing Tracker

Project Overview (Problem to be solved by a website with a backend database):

John Doe goes to a rock climbing gym twice per week and spends about 60-90 minutes attempting various routes each session. Climbing routes are assigned, by convention, a difficulty grade that ranges from V0-V15+. John generally climbs about 15 problems of difficulty V1 per session, and is a member of 3 different climbing gyms. Each climbing gym has over 50 routes available to climb at any given time. John climbs with several friends and would like to compare his progress and attempts relative to his friends, so he wants multiple users to be able to join the same system. Ultimately, John would like an app that would allow him and his friends to log their climbing sessions and attempts, giving them the ability to track stats like: total number of problems completed per session, average difficulty grade of problems completed per session, hardest lifetime difficulty completed, ratio of number of times reaching the top vs number of times attempted, etc.

Database Outline:

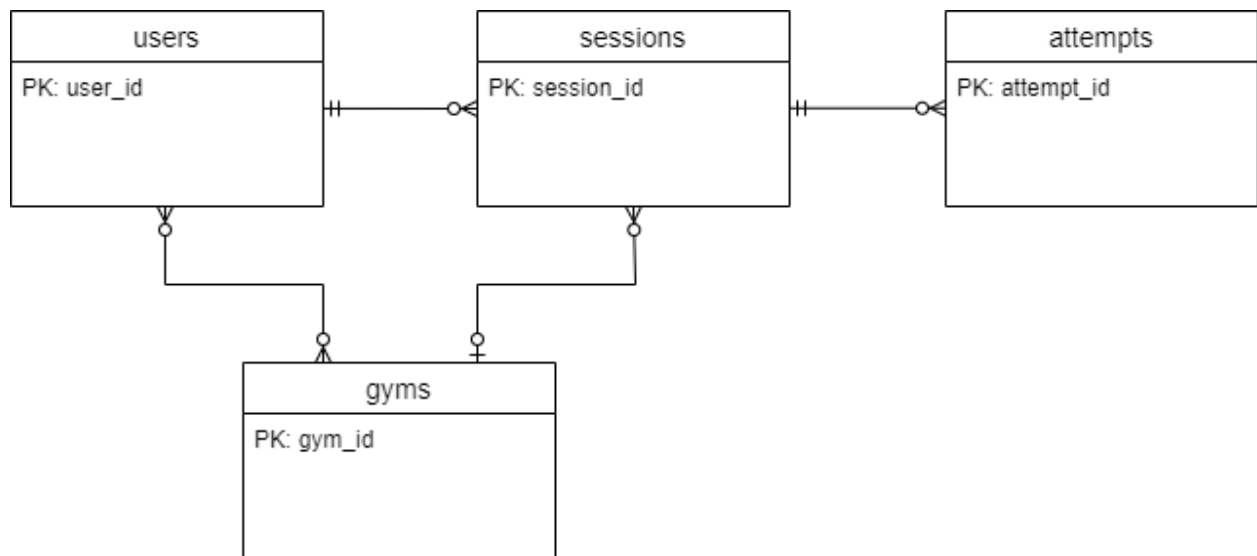
- **users:** records details of the user of the app
 - user_id: INT, auto_increment, PK
 - first_name: VARCHAR, not NULL
 - last_name: VARCHAR, not NULL
 - email: VARCHAR, not NULL
 - birth_date: DATE
 - **Relationships:**
 - There is a M:N relationship between users and gyms, which is implemented with gym_id and user_id as foreign keys inside of the users_gyms intersection table. A single user can climb at zero, one, or multiple gyms, and a single gym can have zero, one, or multiple users. This will serve to show which gyms a user has climbed at.
 - There is a 1:M relationship between users and sessions, which is implemented with user_id as a foreign key inside of sessions. A single user can have zero, one, or multiple sessions logged, and a single session must be linked to one user only.

- **sessions:** records details of the training sessions logged by the user. Each session record represents a unique training session, even if the user logs multiple sessions on a given date.
 - session_id: INT, auto_increment, PK
 - date: DATE, not NULL
 - user_id: INT, not NULL, FK
 - gym_id: INT, FK
 - **Relationships:**
 - There is a 1:M relationship between sessions and attempts, which is implemented with session_id as a foreign key inside of the attempts table. Each session is associated with zero or more attempts, and each attempt is unique.
 - There is a M:1 relationship between sessions and users, which is implemented with user_id as a foreign key inside of sessions. Zero, one, or multiple sessions can be logged for a single user. Each session must be linked to one, and only one, user.
 - There is a M:1 relationship between sessions and gyms, which is implemented with gym_id as a foreign key inside of sessions. Each gym can have zero, one, or many sessions that took place at the gym. Each session can be linked to zero or one gym.
- **attempts:** records details of the climbing attempts logged by the user
 - attempt_id: INT, auto_increment, PK
 - grade: INT, not NULL
 - is_success: TINYINT, not NULL
 - session_id: INT, not NULL, FK
 - **Relationships:**
 - There is a M:1 relationship between attempts and sessions, which is implemented with session_id as a foreign key within the attempts table, that links each attempt to a particular session. A session can have zero, one or many attempts that were logged during the session.
- **gyms:** records details about gyms where users climb and sessions are located
 - gym_id: INT, auto_increment, PK
 - gym_name: VARCHAR, not NULL
 - **Relationships:**
 - There is a M:N relationship between gyms and users, which is implemented with gym_id and user_id as foreign keys inside of the

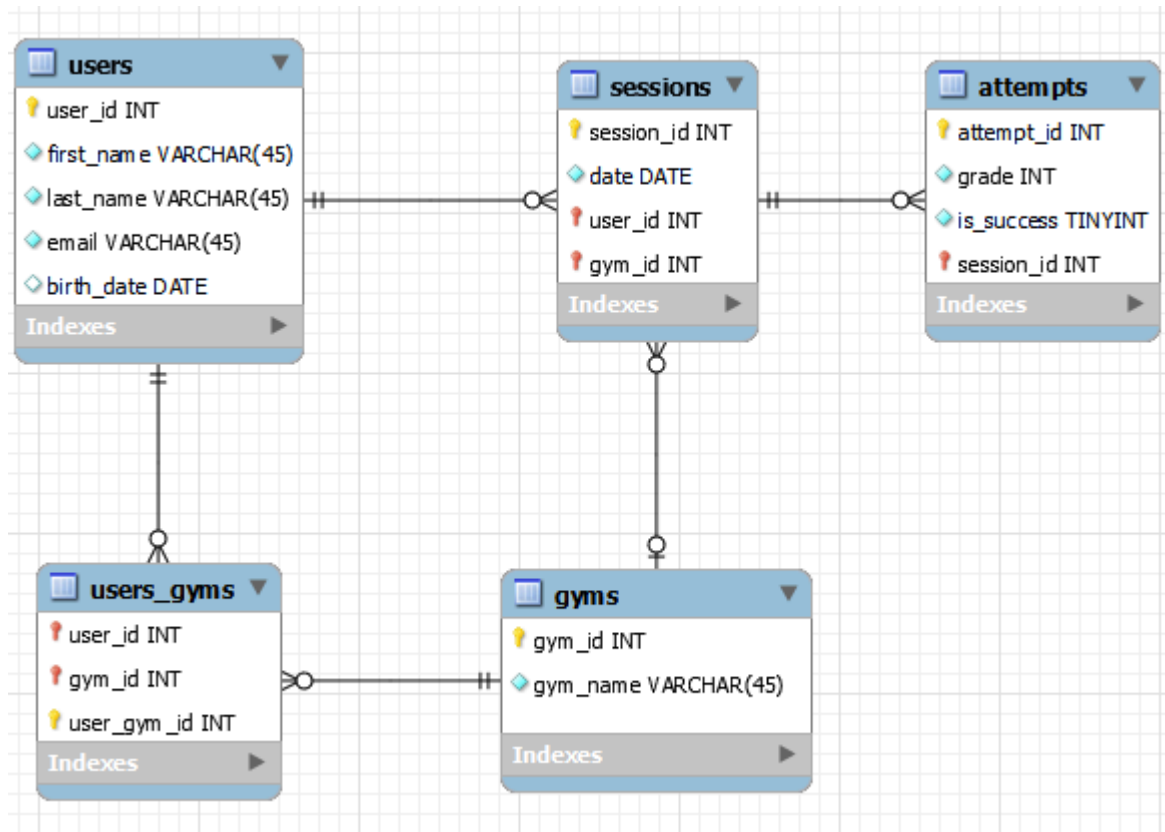
users_gyms intersection table. A single user can climb at zero, one, or multiple gyms, and a single gym can have zero, one, or multiple users.

- There is a 1:M relationship between gyms and sessions, which is implemented with gym_id as a foreign key inside of sessions. Each gym can have zero, one, or many sessions that took place at the gym. Each session can be linked to zero or one gym.
- **users_gyms:** intersection table to map the many to many relationship of users to gyms
 - user_gym_id: INT, not NULL, PK
 - user_id: INT, not NULL, FK
 - gym_id: INT, not NULL, FK

Entity-Relationship Diagram:



Schema:



Example Data (Updated):

users table:

user_id	first_name	last_name	email	birth_date
1	John	Doe	john.doe@example.com	1990-05-15
2	Jane	Smith	jane.smith@example.com	1985-08-22
3	Michael	Johnson	michael.johnson@example.com	1978-11-10
4	Emily	Brown	emily.brown@example.com	1993-02-28
5	David	Wilson	david.wilson@example.com	NULL

sessions table:

session_id	date	user_id	gym_id
1	2024-01-01	1	1
2	2024-01-01	1	2
3	2024-01-02	2	2
4	2024-01-03	3	3
5	2024-01-04	4	NULL
6	2024-02-02	5	NULL

attempts table:

attempt_id	grade	is_success	session_id
1	1	1	1
2	3	1	1
3	2	1	2
4	3	0	2
5	3	0	3
6	7	1	4
7	5	0	4
8	2	1	5
9	3	1	5

gyms table:

gym_id	gym_name
1	Gym A
2	Gym B
3	Gym C
4	Gym D
5	Gym Z

users_gyms table:

user_gym_id	user_id	gym_id
1	1	1
2	1	2
3	2	2
4	3	3
5	4	3
6	4	4

CRUD UI Screen Captures:

USERS PAGES:

READ/DELETE users page:

users

[Home](#) [users](#) [gyms](#) [users_gyms](#) [sessions](#) [attempts](#)

Available functionality: Create, Read, Update, Delete

Note: The email for each user must be unique.

[Add New](#)

user_id	first_name	last_name	email	birth_date		
1	John	Doe	john.doe@example.com	1990-05-15	Edit	Delete
2	Jane	Smith	jane.smith@example.com	1985-08-22	Edit	Delete
3	Michael	Johnson	michael.johnson@example.com	1978-11-10	Edit	Delete
4	Emily	Brown	emily.brown@example.com	1993-02-28	Edit	Delete
5	David	Wilson	david.wilson@example.com	None	Edit	Delete

EDIT users page:

Editing row in users table:


user_id	first_name	last_name	email	birth_date
5	David	Wilson	david.wilson@example.com	None

Edit user

first_name:

last_name:

email:

birth_date: 

ADD users page:

users


[Home](#) [users](#) [gyms](#) [users_gyms](#) [sessions](#) [attempts](#)

Add user

first_name:

last_name:

email:

birth_date: 

GYMS PAGES:

READ gyms page:

gyms

[Home](#) [users](#) [gyms](#) [users_gyms](#) [sessions](#) [attempts](#)

Available functionality: Create, Read, Update

[Add New](#)

gym_id	gym_name	
1	Gym A	Edit
2	Gym B	Edit
3	Gym C	Edit
4	Gym D	Edit
5	Gym Z	Edit

ADD gyms page:

gyms

[Home](#) [users](#) [gyms](#) [users_gyms](#) [sessions](#) [attempts](#)

Add gym

gym_name:

EDIT gyms page:

Editing row in gyms table:

gym_id	gym_name
5	Gym Z

Edit gym

gym_name:

USERS_GYMS PAGES:

READ/DELETE users_gyms page (* this fulfills deletion from M:N relationship requirement):

users_gyms

[Home](#) [users](#) [gyms](#) [users_gyms](#) [sessions](#) [attempts](#)

Available functionality: Create, Read, Update, Delete

Note: Each user, gym relationship must be unique.

[Add New](#)

user_gym_id	first_name	last_name	gym_name		
1	John	Doe	Gym A	Edit	Delete
2	John	Doe	Gym B	Edit	Delete
3	Jane	Smith	Gym B	Edit	Delete
4	Michael	Johnson	Gym C	Edit	Delete
5	Emily	Brown	Gym C	Edit	Delete
6	Emily	Brown	Gym D	Edit	Delete

ADD users_gyms page:

users_gyms

[Home](#) [users](#) [gyms](#) [users_gyms](#) [sessions](#) [attempts](#)

Add user_gym

full_name: ▼

gym_name: ▼

EDIT users_gyms page (* this fulfills M:N relationship updates requirement):

Editing row in users_gyms table (only the gym may be edited):

user_gym_id	first_name, last_name	gym_name
3	Jane Smith	Gym B

Edit user_gym

full name:

gym_name:

SESSIONS PAGES:

READ sessions page:

sessions

[Home](#) [users](#) [gyms](#) [users_gyms](#) [sessions](#) [attempts](#)

Available functionality: Create, Read, Update

[Add New](#)

session_id	date	first_name	last_name	gym_name	
1	2024-01-01	John	Doe	Gym A	Edit
2	2024-01-01	John	Doe	Gym B	Edit
3	2024-01-02	Jane	Smith	Gym B	Edit
4	2024-01-03	Michael	Johnson	Gym C	Edit
5	2024-01-04	Emily	Brown	None	Edit
6	2024-02-02	David	Wilson	None	Edit

ADD sessions page:

sessions

[Home](#) [users](#) [gyms](#) [users_gyms](#) [sessions](#) [attempts](#)

Add session

date:

full_name:

John Doe

gym_name:

Gym A

Add session

Cancel

EDIT sessions page (* this fulfills NULLable relationship requirement, gym field can be set to NULL):

Editing row in sessions table (only the gym may be edited; gym can be null, satisfying nullable requirement):

session_id	date	full_name	gym_name
1	2024-01-01	John Doe	Gym A

Edit session

date:

full_name:

gym_name: ▼

Edit session

Cancel

ATTEMPTS PAGES:

READ attempts page:

attempts

[Home](#) [users](#) [gyms](#) [users_gyms](#) [sessions](#) [attempts](#)

Available functionality: Create, Read

[Add New](#)

attempt_id	grade	is_success	session_id	session_date
1	1	1	1	2024-01-01
2	3	1	1	2024-01-01
3	2	1	2	2024-01-01
4	3	0	2	2024-01-01
5	3	0	3	2024-01-02
6	7	1	4	2024-01-03
7	5	0	4	2024-01-03
8	2	1	5	2024-01-04
9	3	1	5	2024-01-04

ADD attempts page:

attempts

[Home](#) [users](#) [gyms](#) [users_gyms](#) [sessions](#) [attempts](#)

Add attempt

grade1

is_success1

session_id, session_date1, 2024-01-01

Add attempt

Cancel