Range Finder - Interim Report 2

Team Members

Benjamin Hargett - bharget - C56059977
Jacob Hurd - jhurd - C19777168

Web Portal Description

Our project, Range Finder, is a web portal that is designed to allow users to search a database of mountain peaks in North America as well as track the peaks that they have climbed. This involves a user registering for an account on our site which will then allow them to query the database based off of mountain name, country, state, elevation, latitude and longitude. Once logged in the user will be able to mark a mountain as “visited” as well as leave reviews and ratings about their trip. Users will also be able to search for other user accounts and follow users to see information about their “friends” trips.

There will be an administrative component where a user with admin permissions can login, edit, update, backup, or delete information in the database. Passwords will be encrypted to ensure the integrity and safety of the user’s information.

Database Population

We have already gathered all of the information that we plan to use in our “production” dataset. Currently it is contained in a table named “mountains” and consists of over 64,000 mountains across North America. This data was gathered from openstreetmap.org. We downloaded their map data for North America in XML format and then used their open-source tool, *osmosis,* to scan the XML for any datapoint that was labeled as a mountain peak. We then developed our own ruby script to parse this XML data into a JSON file. The script took the name, elevation, latitude, and longitude from the XML data entry and reverse-geocoded the coordinates to determine the country and state that the mountain is in. We then developed another ruby script to use this newly created JSON file to create a “mountains” table in our database and populate it with the data.

Assumptions about Data

The dataset that we are using to populate our database is rather large so there are a few implicit assumptions that are made to allow users to accurately use the information from our site. We are assuming that the dataset is an accurate representation of mountains in North America. We are assuming that there are no duplicates in our dataset and we are assuming that no fields in the data are null. We have taken steps to try to eliminate any inconsistencies in our dataset, such as removing duplicates, but since the dataset is so large and it comes from an open-source organization with thousands of contributors, it is difficult to easily check the entire set for all assumptions made above.

There are also a few implicit assumptions that are made for the system to properly create a user that can navigate through and use the system properly. The system is designed in a way that forces users to enter in all the necessary information before the account can be created; however, we are assuming that all the information that the user enters is accurate information that reflects that individual. Email and phone validation are not implemented so we are assuming that they entered their information accurately.

Comments also have a few implicit assumptions that relate to Users. We assume that users are logged in when attempting to leave comments. Currently, the website is designed in a way that should not allow users to leave comments unless they are logged in. This design constraint is in place to avoid the creation of null fields within the comments table. We are also assuming that the User enters in accurate information about the Mountain in the comments field to ensure the integrity of the site.

There are also some assumptions about mountain_ratings. It is assumed that a user is logged in for them to be able to create a mountain_rating. It is also assumed that there will not be multiple rows in the table that have the same user_id and mountain_id. There are checks in place to prevent all of these faults from happening.

The assumptions regarding mountain_users are similar to those regarding mountain_ratings. It is assumed that a user is logged in for them to create a mountain_user. It is also assumed that there will not be multiple rows in the table with the same user_id and mountain_id. There are also checks in place to prevent this from happening.

For the Relationship Table, it is assumed that both the follower_id and followed_id in a column represent valid user ids within the User's Tables.

Sample Data

Mountains								
id	name	state	country	latitude	longitude	elevation	created_at	update_at
1	Kelly's Mountain	Nova Scotia	Canada	46.2519258	-60.5281258	240	09/19/17	09/24/17
2	Eliza Rock	Washington	United States of America	48.6437357	-122.5793553	0	09/19/17	09/24/17
3	Mount Whitney	California	United States of America	33.1088014	-117.1551691	527	09/19/17	09/24/17
4	Juniper Butte	California	United States of America	41.7914044	-121.4509942	1287	09/19/17	09/24/17
5	Wildcat Peak	California	United States of America	37.9192307	-122.2621766	370	09/19/17	09/24/17

Users											
id	username	password_hash	first_name	last_ name	age	telephone	email	address	created_a t	updated_ at	admin
1	jhurd	$2y$10$nOwe.B TD.YCQPC...	Jacob	Hurd	20	123-123-1 234	jake@emai l.com	Somew here	09/19/17	09/24/17	true

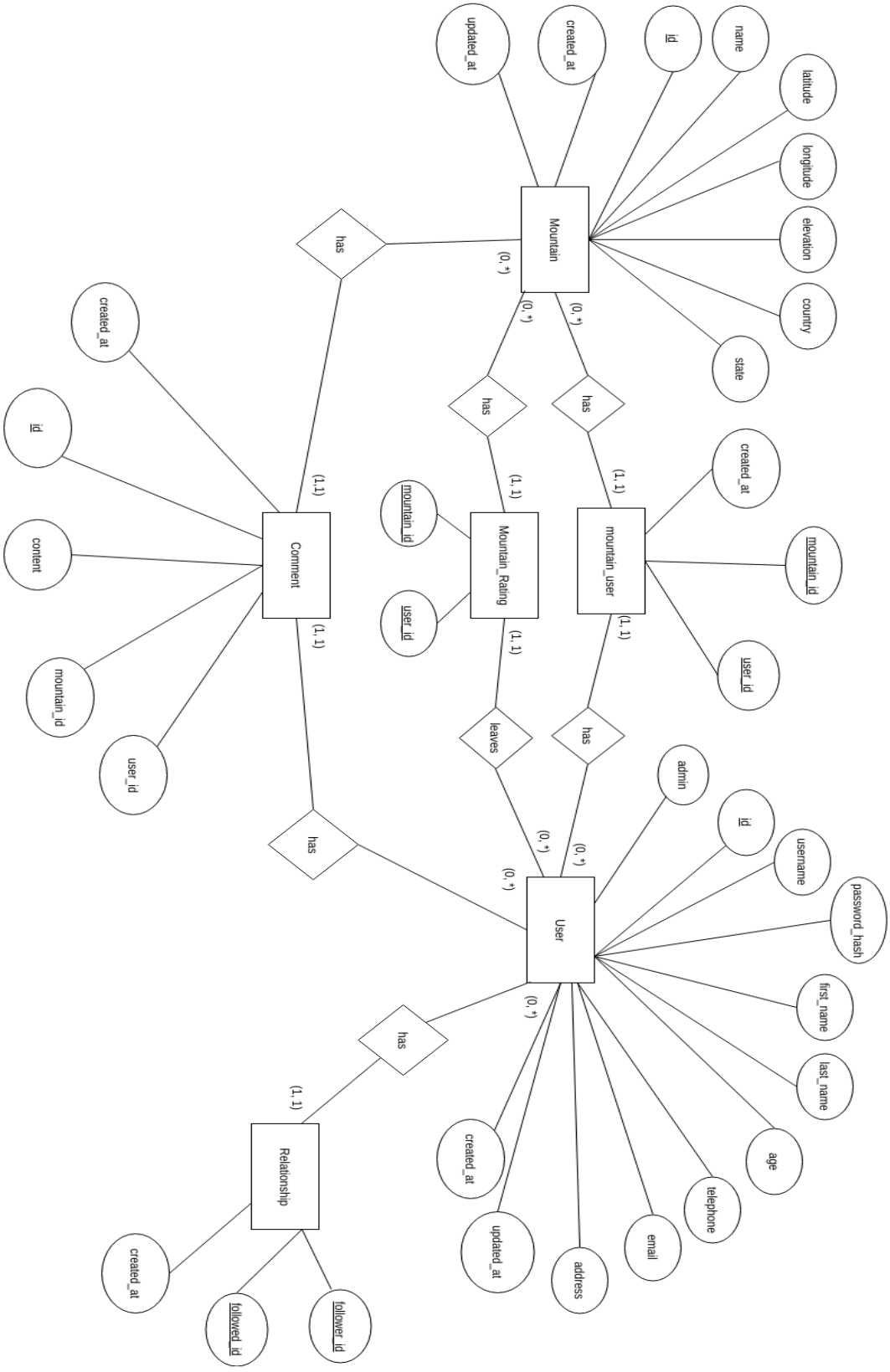
Comments				
id	content	created_at	mountain_id	user_id
1	Sample Content...	2017-10-31 20:14:06	1	1
2	Another Sample Content...	2017-11-31 20:14:06	1	300

Mountain_Ratings	
mountain_id	user_id
1	1
1	300

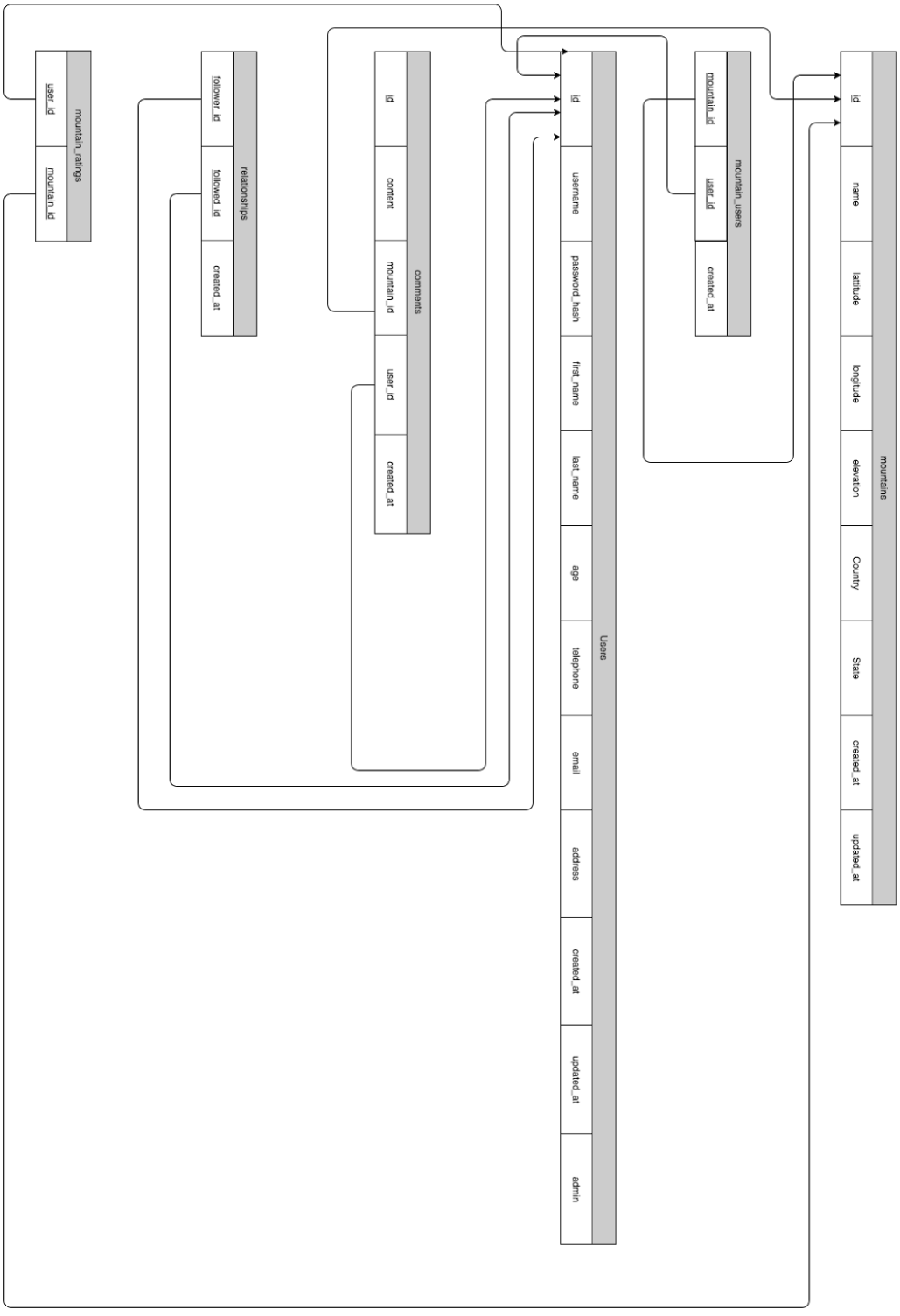
Mountain_Users		
cretated_at	mountain_id	user_id
2017-10-31 20:14:06	1	1
2017-11-18 20:14:06	1	300

Relationships		
created_at	followed_id	follower_id
2017-10-31 20:14:06	1	300
2017-11-18 20:14:06	300	1

ER Diagram


Mountain
id
name
latitude
longitude
elevation
country
state
created_at
updated_at
has
(0, *)
(0, *)
(0, *)
Mountain_Rating
mountain_id
user_id
(1, 1)
(1, 1)
mountain_user
created_at
mountain_id
user_id
(1, 1)
(1, 1)
Comment
id
content
mountain_id
user_id
created_at
(1,1)
(1, 1)
leaves
has
has
has
User
admin
id
username
password_hash
first_name
last_name
age
telephone
email
address
updated_at
created_at
(0, *)
(0, *)
(0, *)
(0, *)
has
(1, 1)
Relationship
created_at
followed_id
follower_id

Database Schema



SQL Queries

Question marks are used to represent values that change in the query depending on what is searched.

Queries dealing with logging in and registering users:

- SELECT id, username, password_hash FROM users WHERE username = ? LIMIT 1
- SELECT password_hash FROM users WHERE id = ? LIMIT 1
- SELECT id FROM users where username = ? LIMIT 1
- INSERT INTO users (username, password_hash, first_name, last_name, age, telephone, email, address) VALUES (?, ?, ?, ?, ?, ?, ?, ?)

Queries dealing with searching for mountains:

- SELECT * FROM mountains WHERE name like % ? %
- SELECT * FROM mountains WHERE state = ?
- SELECT * FROM mountains WHERE country = ?
- SELECT * FROM mountains WHERE latitude like ? %
- SELECT * FROM mountains WHERE longitude like ? %
- SELECT * FROM mountains WHERE elevation <= ?
- SELECT * FROM mountains WHERE elevation >= ?
- SELECT * FROM mountains where id = ? LIMIT 1

Queries dealing with searching for users:

- SELECT * FROM users where id = ? LIMIT 1
- SELECT * FROM users where username = ?

Queries dealing with comments:

- SELECT * FROM comments INNER JOIN users ON comments.user_id = ? WHERE mountain_id = ? ORDER BY created_at DESC
- INSERT INTO comments (content, created_at, mountain_id, user_id) VALUES (?, ?, ?, ?)

Queries dealing with mountain ratings:

- SELECT COUNT(*) FROM mountain_ratings WHERE mountain_id = ?
- SELECT * FROM mountain_ratings WHERE mountain_id = ?
- SELECT * FROM mountain_ratings WHERE mountain_id = ? AND user_id = ?
- INSERT INTO mountain_ratings (user_id, mountain_id, created_at) VALUES (?, ?, ?)

Queries dealing with mountain users:

- SELECT COUNT(*) FROM mountain_users WHERE mountain_id = ?
- SELECT * FROM mountain_users WHERE mountain_id = ?
- SELECT * FROM mountain_users WHERE mountain_id = ? AND user_id = ?
- INSERT INTO mountain_users (user_id, mountain_id, created_at) VALUES (?, ?, ?)

Queries dealing with relationships:

- SELECT * FROM relationships INNER JOIN users ON relationships.follower_id = ?
- SELECT * FROM relationships INNER JOIN users ON relationships.followed_id =?
- SELECT COUNT(*) FROM relationships WHERE follower_id = ?
- SELECT COUNT(*) FROM relationships WHERE followed_id = ?

Web Portal Interface Description

The user's interaction with the portal interface begins with a login screen. The home page of the site is a prompt for the user to enter their username and password to login to Range Finder. If they do not have an account, there is an option to register for an account. If the user is already logged in, then instead of displaying a login prompt, a dashboard will be displayed with pertinent information for that user such as their most recent climbs as well as recent climbs by users that they are following.

Once logged in, they will be able to view all of the mountains that they have marked as climbed. They will also have the option to perform basic database queries using the name of a mountain. They can select the option to do an advanced search which will give them the ability to specify an elevation range, state, country, and coordinates. A list of results will be displayed and the user can select a mountain which will bring up a page with the pertinent information regarding that mountain. There will be an option to mark the mountain as climbed as well as an option to leave a comment. If the user chooses to leave a comment, then they will be prompted to enter text for the body of their comment as well as give their climb a numerical rating.

From the logged-in user's dashboard, they will have the ability to view the list of people that they follow as well as the list of people that follow them. They will also have the option to search the database of users by entering their first name, last name, or username. A list of of results will be displayed and the logged-in user can select a user they wish to view. A page will display that user's publicly viewable information such as their name and username as well as their recent climbs. The logged-in user will then be able to follow or unfollow that user.

If an administrative user logs in to Range Finder, they will be taken to a dashboard very similar to the normal user dashboard. They will be able to search the database for mountains similar to a normal user, but they will have an additional option to edit or delete that mountain's information when viewing the mountain's page. They go through a similar process with users; they are able to search for users but with the additional option to edit/delete them. If editing a mountain or user, they will be prompted with a form where they can input the updated information. The administrative user will also have the ability to delete comments on the selected mountain. In addition to the features listed above, administrators will have the ability to go to a management page from the dashboard where can download a backup of the database.

Completed Tasks

So far we have made really good progress on our project as a whole. We have determined a clear scope/context of our web portal, developed and implemented plans to populate our database using real datasets, and made sure to include a data model that is friendly to both queries and updates of the data. We have also developed an ER Diagram and Relational Schema for our database that has further defined the direction of our project as a whole. With those in place, there is no confusion in how the data will work together and in how their relationships will be integrated into our web portal application. In designing those diagrams, we made it a priority to follow good database design principles and to reduce redundancies throughout the system. Once we laid out a plan for our web portal based on these diagrams, we began implementing the design. We have now populated a database with the real "production" dataset and have begun to incorporate it into our web portal. Starting with a static page to lay out our user interface design, we quickly then moved to designing dynamic pages with embedded PHP that allows a user to register and login to our web portal. In addition to all of that we have also designed some SQL queries to allow users to search and view data based on the relations listed in the ER Diagram and Relational Schema.

Update:

We have now implemented a dynamic search page for querying the database of mountains. Users can search for a mountain using any of its attributes and a paginated listing of the results will be displayed. The user can then browse the results and select a mountain to view. Once a mountain is selected, they are directed to a page displaying information regarding the mountain. This page contains an embedded google map iframe displaying where the mountain is located. It also displays the recent comments left on the mountain and gives the user the option to add a comment. The user can also choose to "like" the mountain or mark it as climbed which correlates to creating entries in the mountain_ratings table and mountain_users table respectively.

In addition to the above progress, we have made improvements to the relational design of our database to better represent our data and we have made improvements to the user interface to make site navigation more intuitive. Lastly, we have also made good progress in designing the rest of the SQL queries that we will need to complete the project.

Future Plans

Week (from current date)	Task - Benjamin	Task - Jacob
1	Create page to dynamically serve search queries for users	Implement following/followed relationships between users
2	Design User information page	Create User dashboard page
3	Add administrative functionality	Add administrative functionality
4	Polishing user interface and testing	Polishing user interface and testing