

# CS411 Database Systems Final Project Report

## Team members:

Chuanyue Shen

Shengwen (Jeffery) Wang

Yawen Deng

Feiyang Liu

Fall 2021

### • Briefly describe what the project accomplished.

HitchHiking enables you to explore different hiking trails, find travel buddies, and enjoy nearby restaurants. We provide our users the opportunity to search for hiking trails based on their interests, and learn about the features, rating, and location of various trails. Our users can add the trails they have visited or are interested in to design their own travel schedule, which will be used to match travel buddies and suggest nearby restaurants based on their schedule and interests. You can also add your favorite features to your preference to match with other users who share the same interests. In addition, our recommendation page will provide you with the high-rating trails that include your favorite features so that you can find future trails more easily. We also have a ranking system that will show the top hikers of the year based on users' trail history.

### • Discuss the usefulness of your project, i.e. what real problem you solved.

Our project allows nature lovers to enjoy outdoor activities and connect with people. HitchHiking was built during Covid-19, and our main goal is to make it easier for people to go out and explore a lifestyle that is different from quarantine life. We implemented a search function that collected all trails from national parks in the U.S. to show people detailed information about each trail, which would allow users to pick the trails that they are interested in. If a user is only interested in forest, he can design a travel schedule that only includes forest trails. Since it has been difficult to socialize during Covid-19, we also want our users to be able to interact with people who may share the same interests. Therefore, we match travel buddies with you based on your interests and travel schedule, which would allow you to make new friends while also enjoying outdoor activities. To further improve your travel experience, we have a recommendation function that will suggest the trails based on your preferences, and we also provide information about nearby restaurants for you to enjoy a nice dinner meal with your new travel buddies.

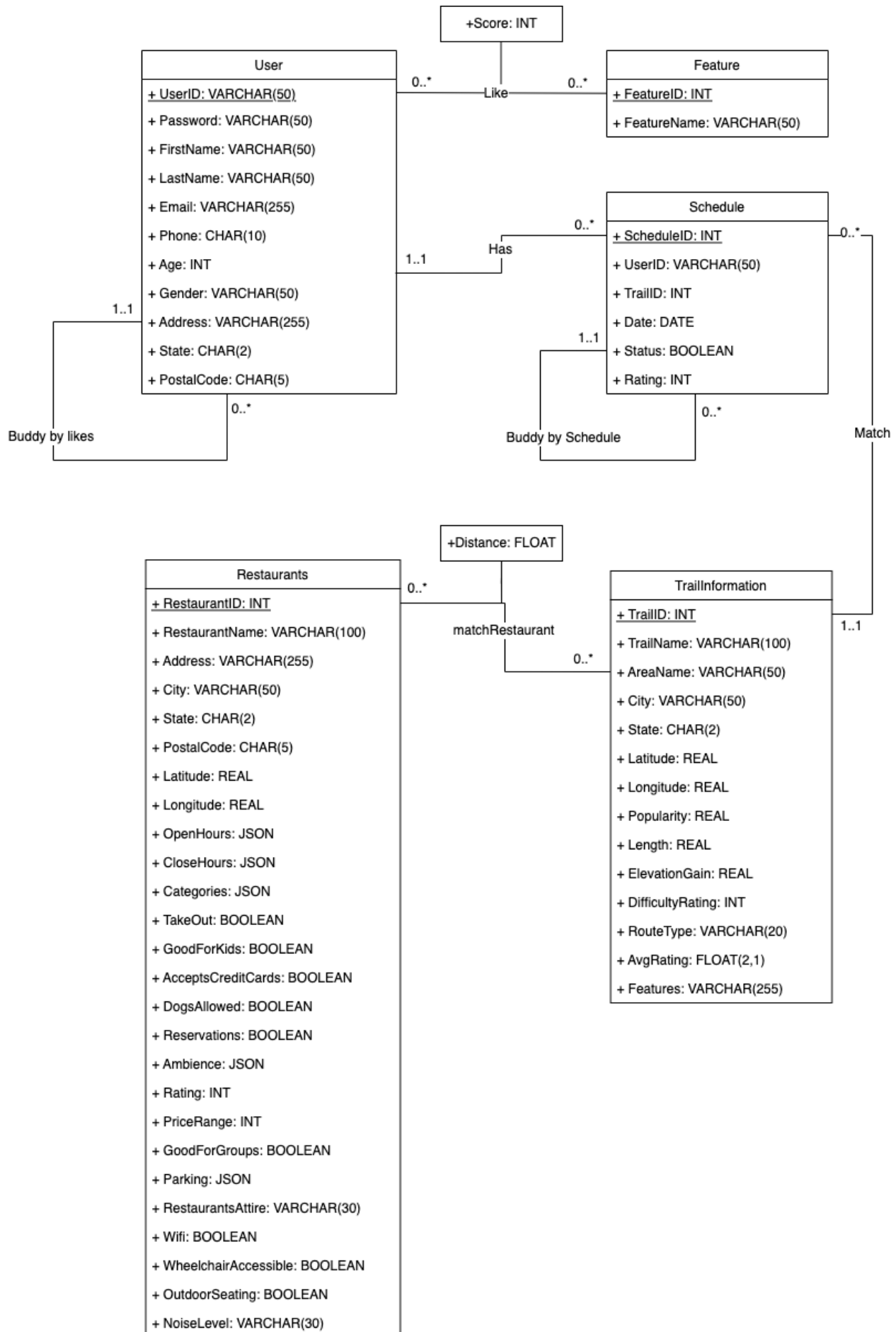
### • Discuss the data in your database.

Our database contains the following information:

- Trail information table: detailed information about trails, including name, location, features, difficulty, rating
- Restaurant table: all the restaurants retrieved from yelp, includes name, location, open hours, ratings, and traits
- User table: user personal information and password

- Feature table: includes all the trails features that a user can select from to show their interests
- Likes table: the liked features by different users
- Schedule table: stores users' personal travel schedule, includes date, rating, and hiking status of users' interested trails
- Recommend table: recommend the trails that users may be interested in, includes name, rating, difficulty, features
- Match restaurant table: matched restaurants to users
- Match buddy table: matched travel buddies to users, includes detailed contact information and similarities
- Like buddy table: the travel buddies you may like, based on shared liked features

- **Include your updated ER/UML Diagram.**



- **Include your final database schema (DDLs) and index design analysis.**

**Auth\_User** ( UserID: VARCHAR(50) [PK], Password: VARCHAR(50), FirstName: VARCHAR(50), LastName: VARCHAR(50), Email: VARCHAR(255), Phone: CHAR(10), Age: INT, Gender: VARCHAR(50), Address: VARCHAR(255), State: CHAR(2), PostalCode: CHAR(5));

**likeBuddy** ( id: INT [PK], Buddy\_name VARCHAR(150), Email: VARCHAR(255), Phone CHAR(10));  
**Like** (UserID: VARCHAR(50) [FK to User.UserID], FeatureID: INT [FK to Feature.FeatureID], Score: INT);

**Feature** ( FeatureID: INT [PK], FeatureName: VARCHAR(50));

**Schedule** ( ScheduleID: INT [PK], UserID: VARCHAR(50) [FK to User.UserID], TrailID: INT [FK to TrailInformation.TrailID], Date: DATE, Rating: INT, State: BOOLEAN);

**likeBuddy** ( id: INT [PK], Trail\_id INT, Trail\_name VARCHAR(150), Buddy\_name VARCHAR(150), Email: VARCHAR(255), Phone CHAR(10));

**TrailInformation** ( TrailID: INT [PK], TrailName: VARCHAR(100), AreaName: VARCHAR(50), City: VARCHAR(50), State: CHAR(2), Latitude: REAL, Longitude: REAL, Popularity: REAL, Length: REAL, ElevationGain: REAL, DifficultyRating: INT, RouteType: VARCHAR(20), AvgRating: FLOAT(2,1), Features: VARCHAR(255));

**matchRestaurant** ( Trail\_id: INT [FK to TrailInformation.TrailID], Trail\_name VARCHAR(150), Rest\_id: INT [FK to Restaurants.RestaurantID], RestaurantName VARCHAR(100), Address VARCHAR(255), City VARCHAR(50), State CHAR(2), PostalCode INT, Openhour VARCHAR(10), Closehour VARCHAR(10), Rating DECIMAL(2,1), distance\_in\_miles FLOAT);

**Restaurants** ( RestaurantID: INT [PK], RestaurantName: VARCHAR(100), Address: VARCHAR(255), City: VARCHAR(50), State: CHAR(2), PostalCode: CHAR(5), Latitude: REAL, Longitude: REAL, Ambience: JSON, Hours: JSON, TakeOut: BOOLEAN, GoodForKids: BOOLEAN, AcceptsCreditCards: BOOLEAN, DogsAllowed: BOOLEAN, Rating: INT, Reservations: BOOLEAN, Categories: JSON, PriceRange2: INT, GoodForGroups: BOOLEAN, Parking: JSON, RestaurantsAttire: VARCHAR(30), Wifi: BOOLEAN, WheelchairAccessible: BOOLEAN, OutdoorSeating: BOOLEAN, NoiseLevel: VARCHAR(30));

index design analysis: same as before (please copy from there)

- **Briefly discuss from where you collected data and how you did it (if crawling is automated, explain how and what tools were used).**

Among the data collections in our application, trail information and restaurant information tables are real data downloaded from Kaggle. Our implementation uses National Park trails from <https://www.kaggle.com/planejane/national-park-trails>, and the restaurant data is from <https://www.kaggle.com/khushishahh/fast-food-restaurants-across-us>. Feature table is a subset of trail information table, where only trail features are extracted to form a standalone table.

The other user related data including user information table and likes table are artificial data that were populated randomly.

Tables such as schedule table, recommend table, match restaurant table, and match buddy table are generated data from the application based on the tables.

- **Briefly discuss your application design and the features involved. Clearly list the functionality of your application (feature specs).**

The intention of this application is to be more accessible and bring more integrated features for users who love hiking in resolving the limitations of similar applications in the market.

The core functionality of this application is to improve the hiking experience. Imaging a user hiking journey, we designed the three main features to support the functionality.

- 1) recommend hiking trails based on the user's preference. Users can choose their preferred trail features (e.g., lake, forest, dog/no-dog) and assign a scale between 1 and 5 at which they value an individual feature. Based on the preferences and scores, this application precisely recommends relevant hiking trails.
- 2) connect hiking buddies who share similar interests and/or experience. This application connects users in two ways: matching hiking buddies by same future hiking schedules, and suggesting making friends by similar preferences.
- 3) suggest food spots to cool down after hiking. Imagining that users need to recharge after a joyful hike with matched buddies, this application recommends food spots near the hiking places according to the physical distances.

- **Include your advanced database program. Discuss your choice for the advanced database program and how it is suitable for your application. For example, if you choose transaction+trigger, explain how this choice is suitable for your application, the isolation level of your transaction, etc.**

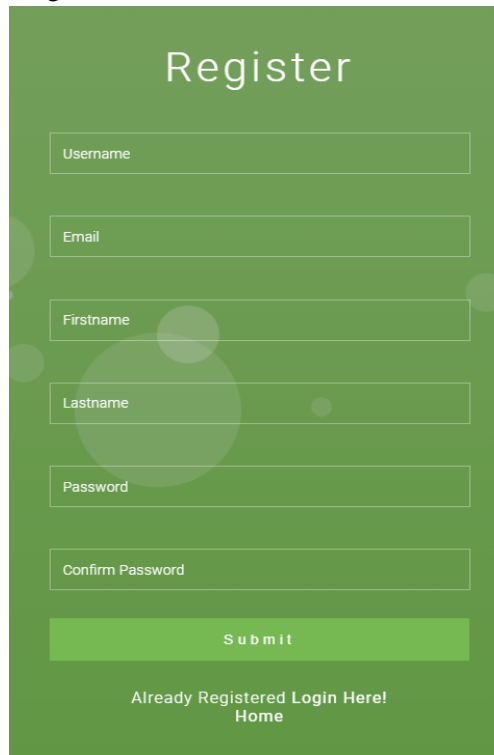
The design of the advanced database program is to improve the users' experience by matching potential hiking buddies and suggesting nearby restaurants. There are two ways to search for buddies. The first method is based on future hiking plans. The system will match people who plan to visit the same trail on

the same day. The second recommendation is based on users' interests. The program will match users who like the same features with a score higher than 2. Moreover, to set a standard for the matching buddies, the system will filter out users that went to less than two trails in the history in both methods. It would always be great to have a meal with friends after hiking. Therefore, the system will sort a list of restaurants by distance based on the matched trails.

To fulfill the above function, our team implemented a trigger function to check the logic of the schedule table and design a stored procedure that can complete the three suggestions. The trigger function will prevent users from marking a future hiking plan as hiked unrealistically, which hinders the matching buddy functionality. Also, when a past schedule is changed to not hiked, the rating of the trail will be updated to 0. We use stored procedures because we need to pass parameters to a complex query for each recommendation and the queries must be executed under a certain order. The other reason is that we need to loop through all the trails of each user to find buddies and restaurants that satisfy our suggested criteria. Combining the trigger and stored procedures, we can make sure the schedules are ready for finding partners and the queries can be executed without disturbance.

- **List and briefly explain the dataflow, i.e. the steps that occur between a user entering the data on the screen and the output that occurs (you can insert a set of screenshots).**

- Our new user should first register, and the information will be stored for user login



Register

Username

Email

Firstname

Lastname

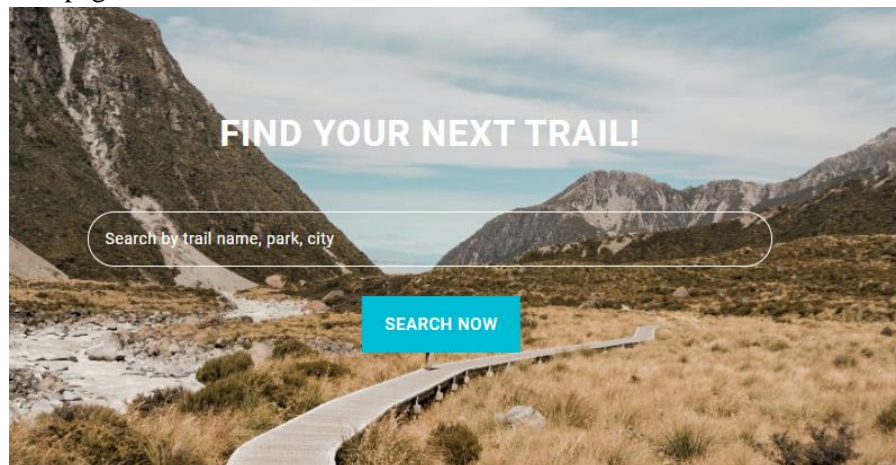
Password

Confirm Password

Submit

Already Registered Login Here!  
Home

- In the home page, the user can search for trail information that will show all the matched trails



## Trails You Are Looking For!

### Coyote Creek Trail

- AreaName: Yellowstone National Park
- City: Yellowstone National Park
- Length: 22369.826
- RouteType: out and back
- AvgRating: 3.0
- Features: dogs-no,river,wildlife,backpacking,hiking



ADD

### Seven Mile Hole Trail


- AreaName: Yellowstone National Park
- City: Yellowstone National Park
- Length: 15610.598
- RouteType: out and back
- AvgRating: 4.5
- Features: dogs-no,forest,hot-springs,river,views,wildlife,backpacking,camping,fishing,hiking,nature-trips,trail-running



ADD

- The user can user “add” button to enter specific trail and date into their schedule

TrailName: 'Coyote Creek Trail'

schedule date:  

## My Schedule

TrailName	AreaName	State	Date	Rating	Hiked	Delete	Update
Parkman to Sargent to Maple Springs via Carriage Road	Acadia National Park	ME	June 6, 2019	5	Yes	DELETE	UPDATE
Paradise Glacier Caves via Skyline Trail Spur	Mount Rainier National Park	WA	Dec. 30, 2021	0	No	DELETE	UPDATE
Ridgecrest to Death Valley OHV Trail	Death Valley National Park	CA	Aug. 8, 2019	3	Yes	DELETE	UPDATE
Beach 1 Nature Trail	Olympic National Park	WA	Dec. 10, 2021	0	No	DELETE	UPDATE
Vernal and Nevada Falls via the Mist Trail	Yosemite National Park	CA	June 19, 2021	5	Yes	DELETE	UPDATE

YOU WANNA IMPROVE YOUR HIKING EXPERIENCE?

- User can use update and delete buttons to edit their schedule

Hiked:

YES

Schedule Date:

06/24/2021

Trail Rating:

5

OK

CANCEL

- In the Preference page, the user can also enter and change their preference, which will correspondingly adjust their matched buddies and recommended trails

## My Preference

Features	Fancy Degree	Delete
backpacking	5	DELETE
city-walk	5	DELETE
hiking	3	DELETE
horseback-riding	5	DELETE
views	3	DELETE

Features: Select an Interest Rating: 0 LIKE



## What are the recommended trails for YOU?

TrailName	State	DifficultyRating	AvgRating	Features
Alder Trail to The Anvil	ME	1	5.0	beach,forest,kids,views,wildlife,hiking,walking
Beach Trail 2	WA	1	5.0	beach,dogs-no,kids,views,wild-flowers,wildlife,birding,hiking,nature-trips,trail-running,walking
Birch Harbor Mountain Bike Path	ME	1	5.0	forest,views,wildlife,hiking,biking,nature-trips,trail-running,walking

## Buddies with Same Plan

Name	Email	Phone	Trail
Janeliz	iWtdSCLBu375@gmail.com	1747819257	Paradise Glacier Caves via Skyline Trail Spur

## Buddies You May Like

Name	Email	Phone
Marquis	FJmQk809@hotmail.com	1818481297
Monet	twWMJevqh805@yahoo.com	1424195768
Neima	apezlqAr132@yahoo.com	1661896451
Casen	gADViOckENo792@gmail.com	1279093742
Kline	WPwGzZXf762@hotmail.com	1858821705

- **Describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or where to maintain your project.**

We used the Django framework for our project. One technical challenge that we met during our project is to pass data between templates and views. For example, in our search.html, we show the results of trails searching based on keywords. For each record, we add the ‘add’ button to trigger the action of adding records to schedule. In this process, we need to get the trailid of each trail, and pass it to the “add schedule” page, then get it in the views to insert the schedule record to the database.

We use a html tag <form> to define the action when the user clicks the button of each record, but we couldn’t get the trailid without <input>. In the end, we solve this problem by passing trailid through the action parameter of <form> tag. The format is: <form action=actionName?trailid={trailid} method="POST">, then use GET to get trailid in the views, although the method of <form> is POST.

Based on our experience, we get some tips to pass data between templates to views. First, we should work through the flow of data, then decide the way to trigger the function. Second, use POST and GET flexibly. We get data in the form by POST, but we still can get data in the URL by GET. It can be helpful if some teams don't have much experience in JavaScript. Third, in views, when we want to redirect to a page which need to get data from the backend, we can just return the related function.

- **State if everything went according to the initial development plan and proposed specifications, if not - why?**

The majority of the work went smoothly according to the initial development plan and proposed specification. One deviation from the initial plan is that we reasonably refined the relations between the tables based on a better understanding of users experience and their behaviors. For example, we originally defined a "recommend" relation between Restaurant table and TrailInformation table. This relation was removed to avoid redundancy as the advanced query can output the recommendation results without creating a new storage table. Besides, we added several temporary tables which were not specified initially in the plan to facilitate the core functionality in improving the user's hiking experience.

- **Describe the final division of labor and how well you managed teamwork.**

The labor division followed the initial plan. Below is a rough work distribution. The team members were actively helping each other in all parts of the work. The work amount was evenly distributed.

- Data collection: Chuanyue, Shengwen, Feiyang, Yawen
- MySQL database: Shengwen, Chuanyue, Feiyang, Yawen
- API development and testing: Yawen, Shengwen, Chuanyue, Feiyang
- Frontend: Feiyang, Yawen, Shengwen
- Cloud: Chuanyue
- Documentation: Chuanyue, Shengwen, Feiyang, Yawen