==========

A HIKER'S GUIDE TO THE GALAXY
FALL 2017 DATABASE DESIGN
GROUP 20
QUINCY ELS
Ho Yat Aaron Ng

==========

USED: Apache 2.4 and MySQL

INSTALLATION:

1. Install XAMPP v3.2.* or later. (Bundled installer recommended)

   https://www.apachefriends.org/download.html

2. Follow installation instructions:

   Windows - https://www.apachefriends.org/faq_windows.html

   OS X - https://www.apachefriends.org/faq_osx.html

3. After installing XAMPP, make sure to start Apache and MySQL. You should see a start button next to either of their names,

   if you are using OS X you may need to first start XAMPP on the main screen, then go to the services tab to start the two services.

4. Extract Hikers_Guide.zip and place the extracted folder in your XAMPP installation directory under xampp\htdocs

   If you use a mac you may need to go under Volumes -> Mount -> Explore in order to find htdocs

5. Access PHPMyAdmin by going to localhost/PHPMyAdmin or, if you aren't hosting on localhost, replace localhost with the IP address found in the General tab.

   If you are hosting on something besides localhost you may need to disable PHPMyAdmin security. The instructions can be found on the following page:
https://www.apachefriends.org/faq_osx.html

6. Import the database under the import tab.

Select choose file, choose "hikers_guide_create"

Hit "Go"

7. (Optional) Use data import on data_dump file to import sample data.

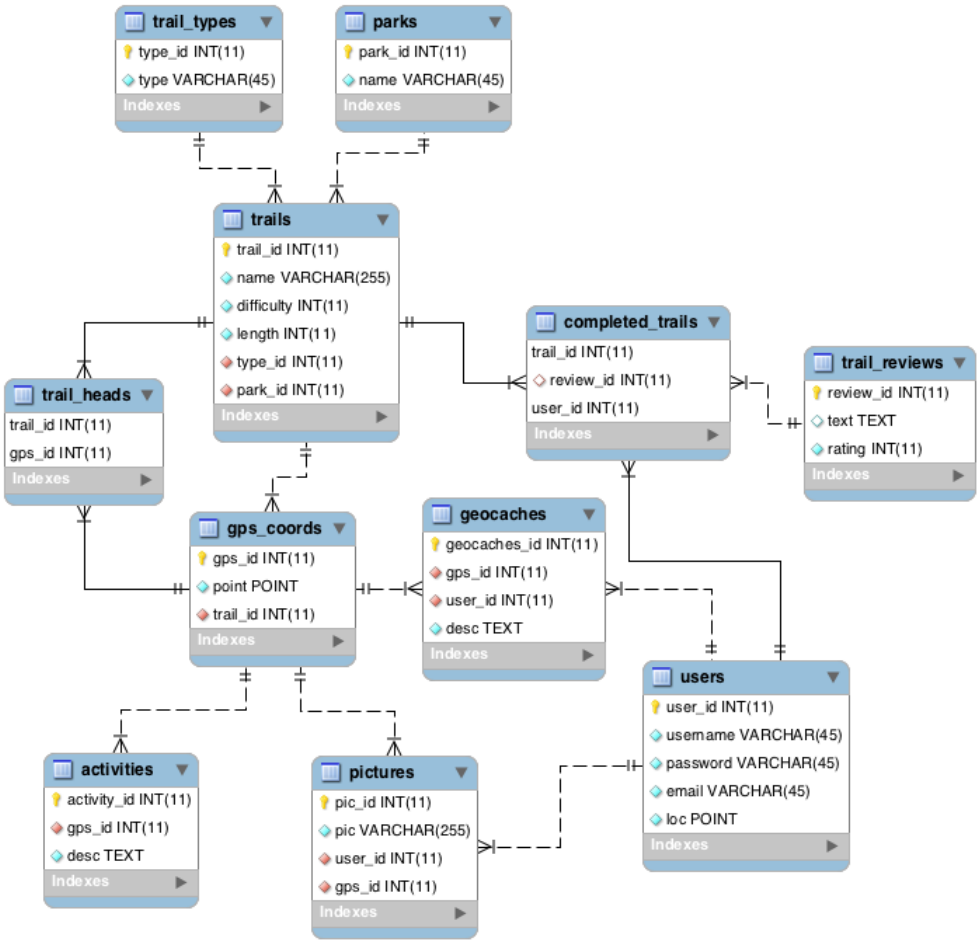
8. Go to localhost/hikers_guide/Hikers_Guide.php (if you are using OS X replace localhost with the ip address found under the general tab)

USING:
Admin Info:
Username: admin
Password: pass
Admin Controls Accessed Through the 42 Button on the Front Page.

Technical Specifications

In order to physically build this proposal we would need to utilize PHP, HTML, and MySQL. The PHP and HTML would be used to make the interface and the MySQL would be for the database. This would allow users to access the database from any web browser.

Schema and Design Choices

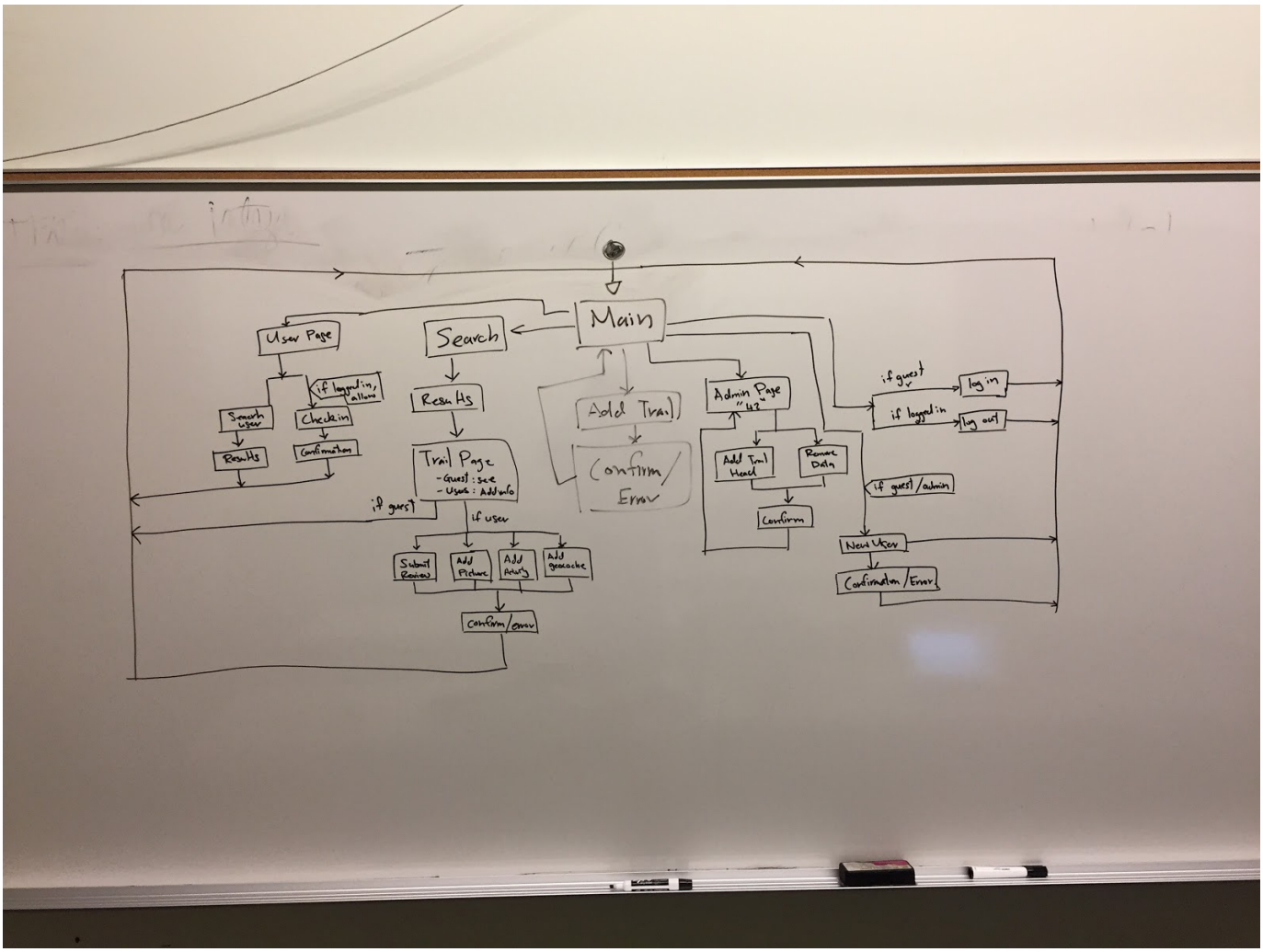


Design choices in our schema.

- In the pictures table, the column pic is a varchar instead of a blob because it is storing the pictures url on the php website.
- In trail_reviews, text is nullable because a user must give a rating to the trail however the user does not need to give a text review of the trail.
- In completed_trails, the foreign key review_id is nullable since users can decide not to review a trail.

4. Provide the final user flow of the system. List the commands or method the users performs to interact with the system.

User Flow Diagram



Lessons learned:

1. Importing data and scraping the internet for specific data is extremely hard. When trying to insert data without already creating existing procedures is time consuming, since data must be inserted in the correct order.
2. Data formatting. Even when we did find data online it would usually be of limited use, for example we would be given the street address of a trail head but we store it as a GPS coordinate so we would need to convert it.
3. Mastered using SQL. The project required us to create a variety of SQL functions and procedures for front end usability.
4. Our first initial approach at designing the schema was nowhere near our current design of the schema. For example we decided to create a subset of gps_coords specifically designated for the head of a trail, this new table is now called trail_heads.

5. Increased technical expertise on PHP. Having to design an entire PHP website to handle SQL was not easy and took us a couple days to understand how to call SQL functions and procedures.
6. PHP also caused other various challenges, like getting the location of a user. We also ran into some limitations, for example, because we only used PHP (which is server side) it was hard to do things that involved client side code, like we could not run code at the click of a button without redirecting the page or trying to use AJAX.
7. Web hosting. Due to certain limitations with the school's wifi we struggled to setup the website at first. Even after getting it working we were stuck with certain bugs. For example, on the school's wifi I can't have anyone access the website due to porting issues. However, if I try to get the IP address of the computer that hosts the website than the IP is "localhost" meaning I can't track it for a GPS update. This made testing extremely hard.
8. Underestimating the time it would talk to create the front end as well as the back end. After the first initial day of transferring our design to an actual schema we realized how long this process would take. We initially believed it would be speedy process of creating a schema and adding procedures and functions however we encountered certain bugs that delayed this process.

Future work:

1. In the future it would be best if we could create a cleaner more well designed user interface. As well as porting the front end as mobile apps so that users can access this database in the middle of trail.
2. The current database is planned for crowdsourcing trails, once it reaches a critical mass of users this database will be able self updating and self sustaining.
3. Later on, it would be best if we could implement google's map api so that trail heads can be shown on an actual map instead of just looking at the gps coordinates.
4. Another api we wish to implement in the future would be a self check in gps api, where the user's current location will be automatically updated in our database, rather than the manual check in accessible from the user page. Furthermore with this api we would be able to confirm if a user has actually completed the trail or not.

Member Contribution

Group member 1: Ho Yat Aaron, Ng

Quincy and I both contributed equally into the creation of the schema, and when changes were needed for the schema we would both discuss about it and come up with a solution. I primarily worked on the backend schema, procedures and functions. I was in charge of making sure the backend was clean and functional for our front end. The majority of the calculations, procedures and function was written by me.

Group member 2: Quincy Els

After the initial design and creation of the main procedures and functions my main responsibility was the front end and integration of the PHP and MySQL. I designed and built the website while Aaron continued the back end work. My main tasks were figuring out how to host and build a website, trying to build security procedures in PHP so that non-admins cannot access pages they aren't supposed to (like if they open the php file for Admin_Controls they should see "Access Denied" unless they are logged in as the Admin. Finally, I did a good amount of testing and bug checking.

List of procedures, triggers, functions, transactions, error handling:

1. Adding data / Modifying data:
    1. (Procedure) Add_activities: adds a activity based on the given: trail_id, text description, longitude and latitude of an activity. If the gps coordinates does not exist it will create a new one to be used.
    2. (Procedure) Add_completed_trail: adds a row into completed_trails based on the given: username and trail_id. Sets the trail as completed by a user.
    3. (Procedure) Add_geocache: Adds a new row into geocache based on the given: username, description, longitude, latitude and a trail_id. Creates a geocache on a specific trail.
    4. (Procedure) Add_picture: Adds a new row into pictures based on the given: username, longitude, latitude and picture url. Creates a picture and stores the url of the picture.

5. (Procedure) Add_review: Adds a new row into review based on the given: username, trail_id, trail rating and trail review. Creates a review that includes the users rating and review of the trail, also updates the associated completed_trails table to the new reviews id.
6. (Procedure) Add_trail_head: Adds a new into trail_head based on the given: trail_id, longitude and latitude. Creates a new trail starting point.
7. (Procedure) Add_trails: Adds a new trail based on the given: name of the trail, difficulty of a trail, length of the trail, type of trail and the park name.
8. (Procedure) Add_user: Adds a new user based on the given: username, password, email, longitude and latitude. Creates a new user and sets their current position to the given gps coordinate.
9. (Procedure) Set_user_gps: Updates the gps coordinate of user to the given: longitude and latitude.

2. Functionalities (for example: search based on certain criteria, return the distance of two points)
    1. (Procedure) Get_activities: Gets the list of activities based on the given trail id.
    2. (Function) Get_completed_trails: Gets the number of completed trails of a certain user based on their username.
    3. (Procedure) Get_geocaches: Gets a list of geocaches on a certain trail, takes in the trail_id.
    4. (Procedure) Get_geocaches_by_gps: Gets a list of geocaches based on a gps position and the maximum distance of the position and the geocaches.
    5. (Procedure) Get_pictures_on_trail: Gets a list of pictures based on the given trail id.
    6. (Procedure) Get_pictures: Gets a list of pictures that has been uploaded by a specific user, takes in the username.
    7. (Procedure) Get_reviews: Gets the table of reviews and the user that submitted said reviews based on a trail id.
    8. (Function) Get_distance: Gets the distance in kilometers between two gps coordinates.
    9. (Function) Get_gps_point: Gets the specific GPS point value based on the given GPS_ID. This is a helper function for larger procedures.
    10. (Function) Get_nearest_head_distance: Gets the nearest head distance of a trail to the user, does this by taking in the trail id and user id.

11. (Function) Get_nearest_head_point: Gets the nearest head point of a specific trail from a point, takes in the trail id and the point.
12. (Function) Get_rating: Gets the average rating of a trail, takes in the trail id.
13. (Function) Get_user_id: Gets the user_id based on the given username.
14. (Procedure) Search: Finds trails that match its search criteria. This search criteria takes in 10 fields.

3. Remove Data:
    1. (Procedure) Remove_activity: Removes an activity based on the gps id and the description of the activity.
    2. (Procedure) Remove_geocache: Removes a geocache based on the gps id and the username.
    3. (Procedure) Remove_trail: Removes a trail and everything related to that trail using trail id.
    4. (Procedure) Remove_user: Removes a user and everything the user has uploaded or trail he/she has reviewed using the username.
4. Error Handling:
    1. Most of the errors are handled through the front end. For example, the HTML forms have fields that cannot be null set as required, so users cannot submit the form without filling in the information. The PHP also checks to be sure values are in accordance with the schema.
    2. In general, the PHP is set up in a way where an action is executed on a button, then the user is brought to a page which either gives a success message or renders the MySQL error in HTML.
    3. The PHP security system is (for the most part) admin_check.php which makes sure the user is logged in as an admin. This allowed me to simply include “include “admin_check.php”” on any page which I wanted to make admin access only.