University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 4

Date: August 5, 2024

Group Number: 17

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Brandon Choo	61130613	i3i8a	jbcraft88@gmail.com
Dylan Zhang	50833441	u0n4i	dyzhang131@gmail.com
Alan Wu	99224487	n2p8m	alanwu0004@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)
In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Table of Contents

1. Project Overview

1.1. Link to Repository

https://github.students.cs.ubc.ca/CPSC304-2024S-T2/project_i3i8a_n2p8m_u0n4i

1.2. Initialization SQL FILE:

NOTE: we referenced the example project as a starting point, and so, some files/code remain.
https://github.students.cs.ubc.ca/CPSC304-2024S-T2/project_i3i8a_n2p8m_u0n4i/blob/main/trailback/config/database.sql

1.3. Project Description + What It Accomplished

TrailFinder is a web app that allows users to discover and explore new trails and locations. Users can sign up and create an account via an email and password or through their Google account. On the homepage they have access to a wide array of trails to choose from, including a search option to better cater to their needs. By following a trail widget they can see a page with a plethora of trail information ranging from a description to user reviews. From the profile page users can see and edit their information, see and add friends, and find all the equipment they list they have. This web app is the one-stop shop for all of one's trail finding needs.

1.4. Final Schema Differences

Renamed User ⟶ UserProfile

Denormalized User, Equipment, Trail, and Gear. This is because some of the primary keys of the normalized versions did not allow for duplicate values which was an issue for insertions where non-key attributes overlapped with each other.

For example, User1 had a primary key trailsHiked however it also had DEFAULT 0. This would mean that if two new users registered, the one who registered at a later date would be unable to register and be rejected from insertion into our database (as they both had the same primary key trailsHiked as 0).

Due to the nature of denormalization preventing duplication of otherwise non-primary keys, we were forced to denormalize table User, alongside tables Equipment, Trail, and Gear. However more memory efficient the normalized tables were, they would prove to be challenging to implement if these duplicate attribute values were not allowed.

Coordinate data type was updated from SDO_GEOMETRY into two different attributes, latitude and longitude, with data types Decimal(8,6) and Decimal(9,6) respectively. This was because sdo_geometry was not a recognized data type in the sql insertion.

TimeToComplete and Duration were updated to INTERVAL DAY TO SECOND, for more accurate duration measures.

2. Database Schema

2.1. Copy of Schema

userprofile(userID, name, email, password, trailshiked, experiencelevel, profilepicture, numberoffriends)
userprofile.png - relation data image link
*Note: all the passwords are hashed, and thus safe if extracted. For demo purposes, each user's password is their name with the first letter capitalized. (i.e. user: john.doe@email.com password: John)

equipment(equipmentid, **userID**, type, brand, amount, weight)
equipment.png - relation data image link

transportation(transportid, type, transportcost)
transportation.png - relation data image link

retailer1(retailername, retailerwebsite)
retailer1.png - relation data image link

retailer2(retailerid, **retailername**)
retailer2.png - relation data image link

location(locationname, latitude, longitude, weather)
location.png - relation data image link

trail(**locationname**, **latitude**, **longitude**, trailname, timetocomplete, description, hazards, difficulty)
trail.png - relation data image link

gear(gearname, geartype, **locationname**, **latitude**, **longitude**, **trailname**)
gear.png - relation data image link

preview1(previewid, image)
preview1.png - relation data image link

preview2(**locationname**, **latitude**, **longitude**, **trailname**, **previewid**)
preview2.png - relation data image link

ugc(ugcid, **userID**, **locationname**, **latitude**, **longitude**, **trailname**, dateposted)
ugc.png - relation data image link

review(**ugcid**, rating, description)
review.png - relation data image link

photo(**ugcid**, image)

photo.png - relation data image link

friends(**userID**, **friendid**, datefriended)

friends.png - relation data image link

transportationtolocation(**transportid**, **locationname**, **latitude**, **longitude**, duration, tripcost)

transportationtolocation.png - relation data image link

userhikestrail(**userID**, **locationname**, **latitude**, **longitude**, **trailname**, datehiked, timetocomplete)

userhikestrail.png - relation data image link

retailerfeaturesgear(**retailerid**, **gearname**, productname, productwebsite)

retailerfeaturesgear.png - relation data image link

3. SQL Queries (File Name + Line Numbers)

3.1. INSERT Operation

trailback/services/userService.js 144-199

addFriend — INSERT into friends table by adding a friend via their email

3.2. DELETE Operation

trailback/services/ugcService.js 62-72

3.3. UPDATE Operation

trailback/services/userService.js 30-117

updateProfile — UPDATE userprofile table by editing name, email, or profile picture

3.4. Selection

trailback/services/dataService.js 194-214

selectionTrails — SELECT from trail table tuples based on locationname or trailname

3.5. Projection

traiback/services/dataService.js 443-459

projectAttributesAndTables — selects attributes from a table using drop down selection.

3.6. Join

trailback/services/dataService.js 290-310

getUGC — JOIN user profiles table with ugc table and review table to see who left user-generated content.

3.7. Aggregation with Group By

trailback/services/dataService.js 348-361

findCheapestTransportPerType — GROUP BY transportation type and find minimum cost.

3.8. Aggregation with Having

trailback/services/dataService.js 383-398

findHeaviestEquipmentType — GROUP BY equipment type and find maximum weight for items types with at least 2 instances.

3.9. Nested Aggregation with Group By

trailback/services/dataService.js 463-479

findMaxTransportCostBelowAverageCost — Find the transportation cost for each type right below the average cost of that group.

3.10. Division

trailback/services/dataService.js 481-500

divideForUserAtEveryLocation — DIVIDE to find users who have completed trails at all locations.

4. GUI Functionality

4.1. INSERT Operation

trailfront/src/components/FriendSection.jsx 36-68

addFriend — Inserts friend via email into friends relation. Friend userID is foreign key to userprofile, and if the userID is not found form the email provided, the friend insert is rejected.

BEFORE	DURING	AFTER
4.1) before insert.png	4.1) during insert.png	4.1) after insert.png

4.2. DELETE Operation

trailfront/src/pages/TrailPage.jsx 158-181

handleDeleteReview — Deletes ugc (user generated content) and cascade deletes review

BEFORE	DURING	AFTER
4.2) before delete.png	4.2) during delete.png	4.2) after delete.png

4.3. UPDATE Operation

trailfront/src/pages/ProfileSection.jsx 41-73

handleProfileUpdate — Update and edit profile information

BEFORE	DURING	AFTER
4.3) before update.png	4.3) during update.png	4.3) after update.png

*Node: userID cannot be modified by the user. However, to change the unique attribute email, the userID had to be updated due to various google auth/google userID reasons.

4.4. Selection

trailfront/src/pages/HomePage.jsx 41-62

fetchSelectionTrails — fetch the user selection information

BEFORE	DURING	AFTER
4.4) before selection.png	4.4) during selection.png	4.4) after selection.png

4.5. Projection

trailfront/src/pages/SpecifyProjectionPage.jsx 64-88

submitProjection — submits projection from frontend

BEFORE	DURING	AFTER
4.5) after projection.png	4.5) before projection.png	4.5) during projectio…

4.6. Join

trailfront/src/pages/TrailPage.jsx 113-134

fetchJoinUserUGC — joins UGC and User, the user provides a rating selection

BEFORE	DURING	AFTER
4.6) before join.png	4.6) during join.png	4.6) after join.png.png

4.7. Aggregation with Group By

trailfront/src/pages/HomePage.jsx 102-119

fetchFindCheapestTransportByType — Groups by transportation type and returns minimum cost

BEFORE	DURING	AFTER
4.7) before group_by.png	4.7) during group_by.png	4.7) after group_by.…

4.8. Aggregation with Having

trailfront/src/pages/HomePage.jsx 159-176

fetchFindHeaviestEquipmentType — Finds heaviest equipment for equipment that has duplicate types (equipment name must be unique, type can have duplicates)

BEFORE	DURING	AFTER
4.8) before having.png	4.8) during having.png	4.8) after having.png

4.9. Nested Aggregation with Group By

trailfront/src/pages/HomePage.jsx 121-138

fetchFindMaxTransportCostBelowAverageCost — Finds the max of the transportation cost that is less than the average transportation cost (grouped by type).

BEFORE	DURING*	AFTER
4.9) before nested_gro…	4.9) during nested_group…	4.9) after nested_gr…

*Note: query is executed on component mount.

4.10. Division

trailfront/src/pages/UserHikesTrailSection.jsx 32-56

handleCompleted — Divides all trails to find if the user has hiked on all of them.

BEFORE	DURING*	AFTER
4.10) before division.png	4.10) during division.png	4.10) after division.png