University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 4

Date: 04/03/2025

Group Number: 43

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Cian Geldenhuys	18406314	b1s1e	cgeldenh@student.ubc.ca
David Huang	44803658	m5i5x	dhuang13@student.ubc.ca
Rani Naser	35459338	e0u3r	ranuz23@student.ubc.ca

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

2. SQL Script

Our setup.sql script can be found in our GitHub repository, linked to the Canvas submission.

3. Description of Final Project

The final project is a hiking application that allows users to create an account and manage trail information in British Columbia. Users can insert, update, delete, search, and filter trails across British Columbia, allowing them to decide and plan which trails are best suited for them and help others in their trail adventures. In addition, users can also view regions in which trails are located, allowing them to get a better sense of the kinds of trails in British Columbia. Furthermore, our group feature promotes an outdoor community, allowing beginners, intermediates, or even pros to join a trail group made up of people of all types of backgrounds. BC HikeHub is designed to help those explore all that British Columbia has to offer.

4. Final Schema vs Original Schema

Since Milestone 2, our schema did not change much. Notable changes are denormalizing some tables such as Region for simplicity in the code. In the final project, some relations are not accessible through the GUI, but we have included enough to make a complete final project meeting all requirements. Many of the tables the client interacts with refer to tables they do not interact with, and these integrity constraints still hold.

Some integrity constraints that we would have wanted could not be implemented. Namely, ON UPDATE CASCADE is not supported by Oracle (a full breakdown of where we would have used this is included in the repository in Milestone 2). Also, full participation constraints are not enforced because of a lack of assertion examples.

A summary of our schema (as output from setup.sql) can be found in the Milestone 4 folder on our repository.

5. SQL Queries:

- **Insert, appService.js, Line 192**: INSERT INTO LocatedIn_Trail_2 (TrailID, RegionID, TrailName, Length_km, Sport, TerrainType, StartLoc, EndLoc) VALUES (:bindTrailId, :bindRegionId, :bindName, :bindLength, :bindSport, :bindTerrain, :bindStart, :bindEnd)

- **Update, appService.js, Line 248:** UPDATE LocatedIn_Trail_2 SET ${queryOn.join(', ')} WHERE TrailID = :bindTrailId

- **Delete, appService.js, Line 264:** DELETE FROM LocatedIn_Trail_2 WHERE TRAILID = :bindTrailId

- **Selection, appService.js, Line 464:** SELECT * FROM LocatedIn_Trail_2 ${sqlWhere}

- **Projection, appService.js, Line 480:** SELECT ${colStr} FROM LocatedIn_Trail_2

- **Join, appService.js, Line 544:** SELECT t.TrailName, t.Length_km, t.Sport, t.TerrainType FROM Trail t, Region r WHERE t.RegionID = r.RegionID AND r.City LIKE '%Vancouver%

- **Aggregation with GROUP BY, appService.js, Line 426:** SELECT g.GroupID, g.Username AS Leader, g.Experience, COUNT(p.Username) FROM Leads_Group g, PartOf p WHERE g.GroupID = p.GroupID AND g.Experience LIKE '%pro%' GROUP BY g.GroupID, g.Username, g.Experience;

    - This query finds all Groups that have professional-level experience. Since there may be different ways this is written, the LIKE condition is not case-sensitive and allows for text before or after. It also counts the number of members in each professional Group.

- **Aggregation with Having, appService.js, Line 404:** SELECT g.GroupID, g.Username AS Leader, g.Experience, COUNT(p.Username) FROM Leads_Group g, PartOf p WHERE g.GroupID = p.GroupID GROUP BY g.GroupID, g.Username, g.Experience HAVING COUNT(p.Username) = (SELECT MAX(Users) FROM (SELECT COUNT(*) AS Users FROM PartOf GROUP BY GroupID));

    - This query finds the largest Group(s). It joins the Group with PartOf (which stores Users in the group) and then aggregates the number of Users in the Group. It then compares this to the MAX number of Users found in any Group and only returns groups HAVING this COUNT of Users.

- **Nested aggregation with GROUP BY, appService.js, Line 568:** SELECT au.Username, au.Hometown, COUNT(p.GroupID) FROM AppUser au, PartOf p WHERE au.Username = p.Username GROUP BY au.Username, au.Hometown HAVING COUNT(p.GroupID) = (SELECT MAX(Groups) FROM (SELECT COUNT(GroupID) AS Groups FROM PartOf GROUP BY Username));

    - This query finds the User who is a part of the most Groups. It joins AppUser and PartOf and only keeps AppUsers HAVING the same COUNT of Groups as the MAX COUNT of Groups which is found through a nested query which returns the COUNT of each Group.

- **Division, appService.js, Line 610:** SELECT t.RegionID FROM LocatedIn_Trail_2 t GROUP BY t.RegionID HAVING COUNT(DISTINCT t.Sport) = (SELECT COUNT(DISTINCT Sport) FROM LocatedIn_Trail_2);

- This Query returns all RegionIDs that offer every sport found in the database. It performs a classic division operation by comparing how many distinct sports each region offers with the total number of distinct sports across all trails. Only regions that support all available sports are included in the result. This query helps us identify regions with the most diverse trail offerings.