University of British Columbia, Vancouver **Group: 93**
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 4

Date: Nov. 28, 2025

Group Number: 93

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Simon Mattern	85839439	m7r9u	SimonMattern03@gmail.com
Gavin Krebbers	47192521	d4o5r	gkrebbers5@gmail.com
Adrian Leung	52615242	f8e6e	adrian.leung4228@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

University of British Columbia, Vancouver **Group: 93**
Department of Computer Science

Project Description:

Our goal for this project is to create a citizen science application, specifically focusing on tracking and documenting the distribution of marine flora and fauna species. Not only will the application assist in scientific research, such as ecological surveys and documentation of rare species, but it will also provide an outlet for hobbyists to explore, record and collaborate about the dynamic aquatic landscape.

Git Repository:

https://github.students.cs.ubc.ca/CPSC304-2025W-T1/team_93

SQL Script:

Our SQL script can be found in our project, located at app/console/commands/CreateDatabase.php, and if you are having trouble finding i,t the link below contains our SQL script as well.
CreateDatabase.php

Project Scheme:

We made no changes to our initial scheme. We used our schema from milestones 1-3 as the guideline for our Milestone 4 implementation and therefore there was no need to deviate from our initial plan.

Query 1, INSERT:

```
DB::insert(
        'INSERT into observation
    (longitude, latitude, quantity, notes, meanLongitude, meanLatitude, scientificName, email,
date )
    values (?, ?, ?, ?, ?, ?, ?, ?, ?)',
      [
        $validated['longitude'],
        $validated['latitude'],
        $validated['quantity'],
        $validated['notes'] ?? null,
        $validated['meanLongitude'],
```

```
            $validated['meanLatitude'],
            $validated['scientificName'],
            $email,
            Carbon::now(),
        ]
    );
```

This query is on line 46in app/http/controllers/ObservationController.php
We handle the case where there is no record for the selected foreign keys in the front end using the code below (resources/js/Pages/Observations/CreateObservation.jsx - Line 144):

```
    for(const id of form.projectIds) {
        const pres = projects.some(p => p.projectID == id);
        if (!(pres)) {
            addToast("Invalid ID","error");
            return;
        }
    }
```

Query 2, UPDATE:

```
DB::update(
        'UPDATE observation SET
        longitude = ?,
        latitude = ?,
        meanLongitude = ?,
        meanLatitude = ?,
        quantity = ?,
        notes = ?,
        scientificName = ?
    WHERE observationID = ?',
        [
            $validated['longitude'],
            $validated['latitude'],
            $validated['meanLongitude'],
            $validated['meanLatitude'],
            $validated['quantity'],
            $validated['notes'] ?? null,
```

```
            $validated['scientificName'],
            $id
        ]
    );

// This code handles the foreign keys in project observation  to clean up old records and add the new ones following the update
DB::delete('DELETE FROM project_observation WHERE observationID = ?', [$id]);
    $projectIDs = $validated['projectIds'] ?? [];
    foreach ($projectIDs as $projectId) {
        DB::insert(
            'INSERT INTO project_observation (projectID, observationID) VALUES (?, ?)',
            [$projectId, $id]
        );
    }
```

This query is on line 246 in app/http/controllers/ObservationController.php

Query 3, DELETE:

```
DB::delete(
        'DELETE FROM groupChat  where ID = ?', [$groupChatId]);
```

This query is on line 120 in app/http/controllers/GroupChatController.php
This uses cascade on delete to remove all the messages sent in the group chat and removes all users from the groupchat.

Query 4, Selection:

```
$projectIDs = DB::select(
        'SELECT p.projectID
        FROM project p
        WHERE p.name LIKE ?
        AND p.projectID IN (
            SELECT p.projectID
            FROM project p
            LEFT JOIN project_user pu ON p.projectID = pu.projectID
            GROUP BY p.projectID
```

```
        HAVING COUNT(pu.email) >= ?
    )
    AND p.projectID IN (
        SELECT p.projectID
        FROM project p
        LEFT JOIN project_observation po ON po.projectID = p.projectID
        GROUP BY p.projectID
        HAVING COUNT(po.observationID) >= ?
    )
    AND p.projectID IN (
        SELECT p.projectID
        FROM project p
        LEFT JOIN project_observation po ON po.projectID = p.projectID
        LEFT JOIN observation o ON o.observationID = po.observationID
        GROUP BY p.projectID
        HAVING COUNT(DISTINCT o.scientificName) >= ?
    )',
    ['%' . $name . '%', $contributors, $observations, $species]
);
```

This query is on line 26 in app/http/controllers/ProjectController.php
This query selects the project IDs of projects that meet all of the conditions laid out by the user. The user can search for the name of the project, if there have been at least X species observed by all observations in the project, if there are at least X observations in the project, and how many people have contributed to the project.

Query 5, Projection:

```
$observations = DB::select(
    "SELECT $fieldsString
        FROM observation o
        LEFT JOIN location l ON l.meanLatitude = o.meanLatitude AND l.meanLongitude =
o.meanLongitude"
);
```

This query is on line 165 in app/http/controllers/ObservationController.php

While this initially looks like we are allowing the user's input directly into the query, we have it set up so that the only values that could be in $fieldsString are ones decided by us.

Query 6, Join:

```
$messages = DB::select(
        'SELECT user.email, user.username, message.data, message.timeSent, message.id
        FROM message
        JOIN user ON user.email = message.email
        WHERE groupChatId =?
        ORDER BY timeSent ASC',
        [$id]
    );
```

This query is on line 76 in app/http/controllers/GroupChatController.php

Query 7, Aggregation with GROUP BY:

```
$projectObservationCount = DB::select(
        'SELECT p.projectID, COUNT(o.observationID) AS observationCount
    FROM project p
    LEFT JOIN project_observation po ON p.projectID = po.projectID
    LEFT JOIN observation o ON o.observationID = po.observationID
    WHERE p.projectID IN (' . implode(',', $ids) . ')
    GROUP BY p.projectID'
    );
```

This query is on line 72 in app/http/controllers/ObservationController.php

Query 8, Aggregation with HAVING:

```
$great_users = DB::select(
        'SELECT u.email, u.username, COUNT(DISTINCT o.scientificName) AS speciesCount
        FROM user u
        JOIN observation o ON o.email = u.email
        GROUP BY u.email
        HAVING COUNT(DISTINCT o.scientificName) >= 5
```

```
        '
  );
```

This query is on line 48 in app/http/controllers/ExtrasController.php

This query selects users who have seen 5 or more unique species.

Query 9, Nested aggregation with GROUP BY:

```
$projects = DB::select(
        'SELECT p.projectID, p.name, p.description, COUNT(po.observationID) as obs_amount
        FROM project p
        LEFT JOIN project_observation po ON p.projectID = po.projectID
        GROUP BY p.projectID
        HAVING COUNT(po.observationID) <
        (
           SELECT AVG(obs_count)
           FROM
              (
                 SELECT p.projectID, COUNT(po.observationID) AS obs_count
                 FROM project p
                 LEFT JOIN project_observation po ON p.projectID = po.projectID
                 GROUP BY p.projectID
              )
        );
        ',
    );
```

This selects projects with fewer observations than the average project.
This query is on line 14 in app/http/controllers/ExtrasController.php

Query 10, Division:

```
$superUsers = DB::select(
```

```
    'SELECT *
    from user u
    WHERE NOT EXISTS (
      SELECT 1
      FROM project p
      WHERE p.projectID NOT IN (
        SELECT po.projectID
        FROM project_observation po
        JOIN observation o ON o.observationID = po.observationID
        WHERE o.email = u.email
      )
    ) '
);
```

This finds all users who have contributed to every project at least once.
This query is on line 33 in app/http/controllers/ExtrasController.php