University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 4

Date: April 2, 2025

Group Number: 80

Name	**Student Number**	**CS Alias (Userid)**	**Preferred E-mail Address**
Cheuk-lun Cheung	36251726	d2n8s	cheukluncheung@gmail.com
Ali Wagih	21605143	v8i3u	awagih@outlook.com
Ziad Khalifa	22016142	f5f1b	Khaliifa1@student.ubc.ca

Short project description:

The Property Management Software System is ultimately designed to help manage real estate data. Users are able to add, update, and delete property records, track maintenance requests, as well as view data on property managers, owners, and the property itself.

Description of how final schema is different than Milestone 2 schema and why:

1. The final schema splits requestPriority and requestStatus into their own table: MaintenanceRequestStatus, using requestID as the primary key.
   The purpose of this is to improve normalization by separating concerns, which makes updates and querying more modular and avoids redundancy if status info changes or is reused.

2. The final schema uses nullable foreign keys with ON DELETE actions like SET NULL and CASCADE.
   The purpose of this is to allow more flexible deletion handling. For example, if a property manager is deleted, the property can still exist with a null reference, or if a tenant is deleted, related rental agreements can be automatically removed.

3. The final schema introduces an AmenityType table to separate amenity definitions from amenity instances.
   The purpose of this is to improve normalization and data integrity by avoiding duplication and allowing consistent definitions of amenity types and sizes across properties.

4. The final schema allows null foreign key fields in several tables, unlike the original schema which requires all foreign keys to be non-null.
   The purpose of this is to provide greater flexibility and reduce the likelihood of constraint violations when related entities are not yet available or are removed.

5. The final schema enforces uniqueness on fields like creditScore in the Individual table and uses ON DELETE CASCADE for Individual and Company.
   The purpose of this is to ensure data consistency and simplify cleanup by automatically removing related sub-entity records when an owner is deleted.

List of all the SQL queries and where they can be found (file name and line number(s))

1. INSERT:
   a. File Name: PropertyMgmtUI.php
   b. Line Numbers: 50 - 62
2. DELETE
   a. File Name: PropertyMgmtUI.php
   b. Line Numbers: 64 - 72
3. SELECTION
   a. File Name: PropertyMgmtUI.php
   b. Line Numbers: 74 - 112

4. UPDATE
    a. File Name: PropertyMgmtUI.php
    b. Line Numbers: 115 - 127
5. PROJECTION
    a. File Name: PropertyMgmtUI.php
    b. Line Numbers: 129 - 137
6. JOIN
    a. File Name: PropertyMgmtUI.php
    b. Line Numbers: 139 - 145
7. Aggregation with GROUP BY
    a. File Name: PropertyMgmtUI.php
    b. Line Numbers: 147 - 152
8. Aggregation with HAVING
    a. File Name: PropertyMgmtUI.php
    b. Line Numbers: 153 - 159
9. Nested aggregation with GROUP BY
    a. File Name: PropertyMgmtUI.php
    b. Line Numbers: 161 - 167
10. DIVISION
    a. File Name: PropertyMgmtUI.php
    b. Line Numbers: 169 - 174

A copy of queries 7 to 10 and a short description of what they do.

Query 7 groups the properties to the assigned property manager ID and displays the count in how many properties each manager is responsible for, in order long with the managerID:

```
SELECT propertyMgrID, COUNT(*) AS total_properties
FROM Property
GROUP BY propertyMgrID
ORDER BY total_properties;
```

Query 8 groups properties by their respective owners (ID) and returns the count of properties per owner only if the count is greater than 1. Owner name and number of properties they own are displayed:

```
SELECT ownerID,
    (SELECT ownerName FROM Owner WHERE Owner.ownerID = Property.ownerID)
    AS ownerName,
    COUNT(*) AS num_properties
FROM Property
GROUP BY ownerID
HAVING COUNT(*) > 1;
```

Query 9 calculates the average value of properties by each property manager. The only average property value along with the corresponding manager ID is only returned when the average is equal or higher than the overall average value of all properties in the database.

```
SELECT propertyMgrID, AVG(currentValue) AS avg_value
FROM Property
GROUP BY propertyMgrID
HAVING AVG(currentValue) >= (SELECT AVG(currentValue) FROM Property);
```

Query 10 finds the property managers who manage a residential and commercial property by ensuring that there is no property type missing from their managed properties through division.

```
SELECT DISTINCT pm.propertyMgrID, pm.propertyMgrName
FROM PropertyManager pm
WHERE NOT EXISTS (
   (SELECT 'Residential' FROM dual UNION SELECT 'Commercial' FROM dual)
   MINUS
   (
      SELECT 'Residential' FROM Residential r
      JOIN Property p ON r.propertyID = p.propertyID
      WHERE p.propertyMgrID = pm.propertyMgrID
      UNION
      SELECT 'Commercial' FROM Commercial c
      JOIN Property p2 ON c.propertyID = p2.propertyID
      WHERE p2.propertyMgrID = pm.propertyMgrID
   )
);
```