University of British Columbia, Vancouver
Department of Computer Science**CPSC 304 Project Cover Page**

Milestone #: ___4____

Date: _Apr 02, 2025_____

Group Number: ____110_____

Name	**Student Number**	**CS Alias (Userid)**	**Preferred E-mail Address**
Frank Yang	11753753	j6l4t	ffyang@student.ubc.ca
Xingyang Zheng	57446361	c5i2r	xingyang2027@gmail.com
Shiyu Zhou	27214782	z9y5k	szhou49@outlook.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
Department of Computer Science**CPSC 304 Project Cover Page**

Description

final project:

The project is a comprehensive Parking Management System designed to handle various aspects of residential and visitor parking operations. It features a robust database structure with multiple interconnected tables to manage users (residents and visitors), vehicles, parking lots, violation tickets, payments, and visitor passes. The system supports essential functionalities including user registration and authentication, vehicle registration and tracking, visitor pass management with different duration options (8-hour, 24-hour, and weekend passes), parking violation tracking and payment processing, and comprehensive reporting capabilities. The application implements all required database operations (INSERT, UPDATE, DELETE, SELECT) with proper foreign key relationships and cascading effects. It includes advanced query features such as JOIN operations, aggregations with GROUP BY and HAVING clauses, and nested aggregations. The system provides a user-friendly interface for both residents and administrators, allowing them to manage parking spaces, track violations, process payments, and generate various reports about parking usage, violations, and revenue. The implementation ensures data integrity through proper constraints and transaction management.

init.sql: see backend/init.sql

difference from schema:

Only key changes (e.g., type changes) and obvious changes are listed, case shifts (e.g., lotId becomes LOT_ID) are not listed

1.Parkinglot

a) from lotId: Integer to lotId: number

   reason: get larger range

b) from address: Varchar(100) to address: Varchar2(200)

   reason: get larger range

c) from capacity to TOTAL_SPACES

reason: easy to read

d) from currentOccupancy to AVALABLE_SEATS

reason: easy to read

e) Delete currentRemain

reason: currentRemain can get from TOTAL_SPACES - AVALABLE_SEATS

f) Add key 'LOT_NAME'

reason: improve information

2. User

To make it easier for the back-end to call the sql code to process the matter, we merge User, Resident, and Visitor into a new table, User, and add new keys and constraints to implement the original functionality

a) from userId: Integer to ID: number

reason: get larger range

b) from phone: Integer to phone: varchar2(20)

reason: some phone number may begin with '0'. If we use Integer, we will lose this '0'

c) from name: varchar(20) to varchar2(200)

reason: get larger range

d) add key 'Password'

reason: improve appropriate information for "Log in" and "Register" function

e) add key 'Role' and constraint 'role must be user or admin'

reason: improve information, make sure whether or not a user is an admin

f) add key 'USER_TYPE' and constraint 'USER_TYPE must be resident or visitor'

reason: after merge User, Resident, Visitor, we make this change to implement the original functionality

g) add key 'UNIT_NUMBER' and 'HOST_INFORMATION'

reason: after merge User, Resident, Visitor, we make this change to implement the original functionality

h) add key 'CREATED_AT'

reason: easy to track the create time

3. Vehicle

a) add key 'VEHICLE_ID' as a primary key

reason: easy to track and use the corresponding sql codes

b) add key 'USER_ID'

reason: easy to track the owner

c) from parkingUntil: datetime to parkingUntil: timestamp

reason: datetime can store information from year 1000 to 9999, we do not need such large range

d) add key 'CURRENT_LOT_ID'

reason: easy to track which parking lot does it park now

e) add key 'CREATED_AT'

reason: easy to track the create time

4. Staff

we change table name Admin to Staff for easier reading

a) from staffId: integer to staffId: number

reason: get larger range

b) delete key ‘name’, and key ‘USER_ID’

reason: connect table Staff and User, enhanced database connectivity

c) add key ‘CREATED_AT’

reason: easy to track the create time

5. Report

We deleted this table, because we found we could use table Violation and table Payments to directly get the violation records for a specified period of time

6. Violation

we change table name ViolationTicket to Violation for simplicity

a) from ticketId: integer to ticketId: number

reason: get larger range

b) from time: datetime to time: timestamp

reason: datetime can store information from year 1000 to 9999, we do not need such large range

c) add key ‘STATUS’ and a constraint that ‘status must be pending, paid, or appealed’

reason: improve information

d) add key ‘CREATED_AT’

reason: easy to track the create time

7. Payments

a) from payId: integer to payId: number

   reason: get larger range

b) from amount: integer to amount: number(10,2)

   reason: having a fractional part makes the data closer to reality

c) from cardNumber: integer to cardNumber: varchar2(20)

   reason: if some card number begin with '0', using integer will lose this '0'

d) add key 'TICKET_ID'

   reason: connect Payments to Violation, make it easier to track the corresponding violation ticket (if ticket_id is not null)

e) add key 'CREATED_AT'

   reason: easy to track the create time

8. VisitorPasses

a) from visitorPassId: integer to visitorPassId: number

   reason: get larger range

b) from validTime: datetime to validTime: number

   reason: after change, we store duration in hour (8, 24, 48). It is easy to fix a valid time and make it close to reality

c) add key 'STATUS' and a constraint that 'status must be in active or expired'

   reason: make it easier to track whether a visitor pass can be used

d) add key 'CREATED_AT'

   reason: easy to track the create time

e) add key 'VISITOR_PLATE'

reason: make it easier to track whether a visitor pass can be used

schema and screenshots:

1) User(ID:number, PHONE: varchar2(20), PASSWORD: varchar2(100), NAME: varchar2(100), ROLE: varchar2(20), USER_TYPE:varchar2(20), UNIT_NUMBER: number, HOST_INFORMATION:varchar2(200))

```
SQL> select * from Users;

        ID PHONE                PASSWORD        NAME                 ROLE       USER_TYPE    UNIT_NUMBER HOST_INFORMATION          CREATED_AT
---------- -------------------- --------------- -------------------- ---------- ------------ ----------- ------------------------- ---------------------------
########## 1234567890           password        Admin User           admin      resident                                           02-APR-25 06.57.07.705415 PM
########## 9876543210           password        Resident User        user       resident      ##########                           02-APR-25 06.57.07.706989 PM
########## 1112223333           password        John Smith           user       resident      ##########                           02-APR-25 06.57.07.707887 PM
########## 2223334444           password        Emily Johnson        user       resident      ##########                           02-APR-25 06.57.07.708648 PM
########## 3334445555           password        Michael Brown        user       resident      ##########                           02-APR-25 06.57.07.709383 PM
########## 4445556666           password        Sarah Wilson         user       visitor                   Visiting Unit 101        02-APR-25 06.57.07.710125 PM
########## 5556667777           password        David Lee            user       visitor                   Visiting Unit 102        02-APR-25 06.57.07.710833 PM
########## 6667778888           password        Jessica Garcia       user       visitor                   Visiting Unit 103        02-APR-25 06.57.07.711521 PM

8 rows selected.
```

2) ParkingLot(LOT_ID: number, TOTAL_SPACES: number, AVAILABLE_SPACES: number, ADDRESS: varchar2(200), LOT_NAME: varchar2(100))

```
SQL> select * from ParkingLot;

    LOT_ID TOTAL_SPACES AVAILABLE_SPACES
---------- ------------ ----------------
ADDRESS
--------------------------------------------------
LOT_NAME
--------------------------------------------------
         1          100              100
123 Main St, Vancouver, BC
Maple Grove Estates

         2           80               80
456 Oak Ave, Vancouver, BC
Oakwood Heights

         3          120              120
789 Pine St, Vancouver, BC
Pine Ridge Gardens

         4           90               90
321 Maple Dr, Vancouver, BC
Maplewood Village

         5          150              150
654 Park Rd, Vancouver, BC
Parkview Meadows

         6          110              110
987 Waterfront Blvd, Vancouver, BC
Harbourview Residences

         7           95               95
147 Mountain View Dr, Vancouver, BC
Mountainview Heights

         8          130              130
258 Ocean Park Ave, Vancouver, BC
Oceanview Estates


8 rows selected.
```

3) Vehicles(VEHICLE_ID: number, **USER_ID:** number, PROVINCE: varchar2(10), LICENSE_PLATE: varchar2(20), PARKING_UNTIL: timestamp, **CURRENT_LOT_ID**: number, CREATED_AT: timestamp)

```
SQL> select * from Vehicles;

VEHICLE_ID    USER_ID PROVINCE   LICENSE_PLATE        PARKING_UNTIL                                                                 CURRENT_LOT_ID CREATED_AT
---------- ---------- ---------- -------------------- ----------------------------------------------------------------------------- -------------- -----------------------------------------------------------------------------
         1          2 CA         TEST123              03-APR-25 06.57.07.778253 PM                                                           1 02-APR-25 06.57.07.778256 PM
         2          3 NY         ABC123               04-APR-25 06.57.07.779411 PM                                                           1 02-APR-25 06.57.07.779413 PM
         3          4 TX         XYZ789               05-APR-25 06.57.07.780130 PM                                                           2 02-APR-25 06.57.07.780132 PM
         4          2 ON         EXPIRED              01-APR-25 06.57.07.780822 PM                                                           1 02-APR-25 06.57.07.780823 PM
         5          3 BC         OLD123               31-MAR-25 06.57.07.781472 PM                                                           2 02-APR-25 06.57.07.781473 PM
```

4) VisitorPasses(PASS_ID:number, **USER_ID**: number, VALID_TIME: number, STATUS: varchar2(20), CREATED_AT: timestamp, VISITOR_PLATE: varchar2(30))

```
SQL> select * from VisitorPasses;

   PASS_ID    USER_ID VALID_TIME STATUS              CREATED_AT                    VISITOR_PLATE
---------- ---------- ---------- ------------------- ----------------------------- ------------------------------
         1          2          8 active              02-APR-25 06.57.07.819464 PM  BC-AB123CD
         2          2         24 active              02-APR-25 06.57.07.820569 PM  WA-KDA1233
         3          2         48 active              02-APR-25 06.57.07.821256 PM  CA-FSD1234
         4          2          0 expired             02-APR-25 06.57.07.822000 PM  ON-OFN2312
         5          3          8 active              02-APR-25 06.57.07.822820 PM  NY-FAB7680
         6          3         24 active              02-APR-25 06.57.07.823484 PM  AB-CD123AD
         7          4         48 active              02-APR-25 06.57.07.824169 PM  SK-FA123DF
         8          4          0 expired             02-APR-25 06.57.07.824839 PM  MO-1A3489
```

5) Violations(TICKET_ID: number, **LOT_ID**:number, PROVINCE: varchar2(10), LICENSE_PLATE:varchar2(20), REASON: varchar2(200), TIME: timestamp, STATUS:varchar2(20), CREATED_AT: timestamp)

```
SQL> select * from Violations;

 TICKET_ID     LOT_ID PROVINCE   LICENSE_PLATE
---------- ---------- ---------- --------------------
REASON
--------------------------------------------------------------------------------------------------------------------------------
TIME                                                                            STATUS              CREATED_AT
------------------------------------------------------------------------------- ------------------- -----------------------------
         1          1 BC         ABC123
No Valid Visitor Pass
05-JAN-25 10.30.00.000000 AM                                                    pending             02-APR-25 06.57.07.864364 PM

         2          1 ON         DEF456
Expired Pass
12-JAN-25 02.45.00.000000 PM                                                    paid                02-APR-25 06.57.07.865626 PM

         3          2 AB         GHI789
Unauthorized Parking Area
18-JAN-25 09.15.00.000000 AM                                                    pending             02-APR-25 06.57.07.866414 PM

         4          1 QC         JKL012
Blocked Access
23-JAN-25 04.20.00.000000 PM                                                    appealed            02-APR-25 06.57.07.867142 PM

         5          3 BC         MNO345
No Valid Visitor Pass
02-FEB-25 11.10.00.000000 AM                                                    pending             02-APR-25 06.57.07.867886 PM

         6          2 AB         PQR678
Expired Pass
08-FEB-25 01.25.00.000000 PM                                                    paid                02-APR-25 06.57.07.868571 PM

         7          1 ON         STU901
Unauthorized Parking Area
15-FEB-25 08.40.00.000000 AM                                                    pending             02-APR-25 06.57.07.869301 PM

         8          3 BC         VWX234
Blocked Access
21-FEB-25 05.55.00.000000 PM                                                    paid                02-APR-25 06.57.07.869997 PM

         9          2 QC         YZA567
Other
03-MAR-25 10.05.00.000000 AM                                                    pending             02-APR-25 06.57.07.870654 PM
```

```
        10          1 BC         BCD890
No Valid Visitor Pass
09-MAR-25 03.30.00.000000 PM                                        paid                 02-APR-25 06.57.07.871348 PM

        11          3 ON         EFG123
Expired Pass
14-MAR-25 09.45.00.000000 AM                                        pending              02-APR-25 06.57.07.872054 PM

 TICKET_ID     LOT_ID PROVINCE   LICENSE_PLATE
---------- ---------- ---------- --------------------
REASON
--------------------------------------------------------------------------------------------------------------------------------
TIME                                                                STATUS               CREATED_AT
------------------------------------------------------------------------------ -------------------- -------------------------------

        12          2 AB         HIJ456
Unauthorized Parking Area
20-MAR-25 12.15.00.000000 PM                                        appealed             02-APR-25 06.57.07.872740 PM

        13          1 QC         KLM789
Blocked Access
25-MAR-25 04.40.00.000000 PM                                        pending              02-APR-25 06.57.07.873524 PM

        14          3 BC         NOP012
No Valid Visitor Pass
30-MAR-25 11.20.00.000000 AM                                        paid                 02-APR-25 06.57.07.874263 PM
```

Staff(STAFF_ID: number, **USER_ID**: number, **LOT_ID**: number, CREATED_AT: timestamp)

```
SQL> select * from Staff;

  STAFF_ID    USER_ID     LOT_ID
---------- ---------- ----------
CREATED_AT
---------------------------------------
         1          1          1
02-APR-25 07.40.32.261652 PM

         2          2          2
02-APR-25 07.40.32.263221 PM

         3          3          3
02-APR-25 07.40.32.264267 PM

         4          4          4
02-APR-25 07.40.32.265288 PM

         5          5          5
02-APR-25 07.40.32.266329 PM
```

Payments(PAY_ID:number, **USER_ID**:number, AMOUNT: number(10,2), PAYMENT_METHOD: varchar(50), CARD_NUMBER: varchar2(20), **LOT_ID**: number, **TICKET_ID**: number, STATUS: varchar2(20), CREATED_AT: timestamp)

University of British Columbia, Vancouver
Department of Computer Science**CPSC 304 Project Cover Page**

```
SQL> select * from Payments;

    PAY_ID    USER_ID     AMOUNT PAYMENT_METHOD                                     CARD_NUMBER            LOT_ID  TICKET_ID STATUS              CREATED_AT
---------- ---------- ---------- ------------------------------------------------- -------------------- ---------- ---------- ------------------- ---------------------------
         1          2         50 Credit Card                                       4111111111111111              1          2 completed           14-JAN-25 09.30.00.000000 AM
         2          3         75 Debit Card                                        5555555555554444              1          4 completed           25-JAN-25 02.15.00.000000 PM
         3          4         40 Credit Card                                       3782822463100005              1         10 completed           10-MAR-25 11.45.00.000000 AM
         4          2         60 PayPal                                            4111111111111111              2          6 completed           10-FEB-25 10.20.00.000000 AM
         5          3         45 Apple Pay                                         5555555555554444              2         12 failed              22-MAR-25 04.30.00.000000 PM
         6          4         80 Credit Card                                       6011111111111117              3          8 completed           23-FEB-25 01.10.00.000000 PM
         7          2         55 Debit Card                                        4242424242424242              3         14 completed           01-APR-25 09.45.00.000000 AM
         8          3         25 Credit Card                                       4111111111111111              1            completed           18-JAN-25 11.25.00.000000 AM
         9          4         30 PayPal                                            5555555555554444              2            completed           05-FEB-25 03.40.00.000000 PM
        10          2         35 Apple Pay                                         3782822463100005              3            completed           28-FEB-25 10.15.00.000000 AM
        11          3         40 Credit Card                                       4242424242424242              1            completed           15-MAR-25 01.50.00.000000 PM
        12          4         45 Debit Card                                        6011111111111117              2            pending             27-MAR-25 04.30.00.000000 PM

12 rows selected.
```

Repository link

https://github.students.cs.ubc.ca/CPSC304-2024W-T2/project_c5i2r_j6l4t_z9y5k

University of British Columbia, Vancouver
Department of Computer Science**CPSC 304 Project Cover Page**

List of all SQL queries

1. INSERT: registerVehicle() in backend/appService.js 404

   for checking whether the userId exist: backend/appService.js : line 388

```
          `INSERT INTO Vehicles(USER_ID, PROVINCE, LICENSE_PLATE,
CURRENT_LOT_ID,PARKING_UNTIL)

          VALUES(:userId, :province, :licensePlate,:lotId
,TO_TIMESTAMP(:parkingUntil, 'YYYY-MM-DD HH24:MI:SS'))`,
```

2. UPDATE: createPayment() in backend/appService.js : 1022

```
              `UPDATE Violations SET STATUS = 'paid' WHERE TICKET_ID = :1`,
```

3. DELETE: deleteVehicle() in backend/appService.js :459

```
          `DELETE FROM Vehicles WHERE PROVINCE = :province AND LICENSE_PLATE =
:licensePlate`,

           { province, licensePlate },

           { outFormat: oracledb.OUT_FORMAT_OBJECT }
```

4. SELECTION: registerUser(name, phone, password, userType, unitNumber, hostInformation, role) in backend/appService.js :199

```
`SELECT * FROM Users WHERE phone = :phone AND password = :password`
```

5. Projection getAllVehiclesWithProjection() in backend/appService.js : 1780

```
         SELECT ${selectedCols.join(', ')}

         FROM Vehicles v

         JOIN Users u ON v.USER_ID = u.ID

         ORDER BY v.VEHICLE_ID
```

6. Join: adminLogin() in backend/appService.js :1088

```
            s.STAFF_ID as staffId,

            u.NAME,

            s.LOT_ID as lotId

        FROM Staff s

        JOIN Users u ON s.USER_ID = u.ID

        WHERE s.STAFF_ID = :1 AND u.PASSWORD = :2 = :1
```

7. AGGREGATION with GROUP BY: getUserVisitorPassQuota(userId) in backend/appService.js: 715

   Counts the number of active visitor passes grouped by their validity duration (VALID_TIME) for a specific user.

```
               `SELECT

                    VALID_TIME,

                    COUNT(*) AS active_count

                FROM VisitorPasses vp

                WHERE vp.USER_ID = :userId

                AND vp.STATUS = 'active'

                AND CURRENT_TIMESTAMP < vp.CREATED_AT +
NUMTODSINTERVAL(vp.VALID_TIME, 'HOUR')

                GROUP BY VALID_TIME`
```

8. AGGREGATION with HAVING: getAllParkingLots() in backend/appService.js: 809

   Retrieves details of all parking lots, including their total capacity, available spaces, current occupancy, and the number of parked vehicles.

```
SELECT
    p.LOT_ID as lotId,
    p.TOTAL_SPACES as capacity,
    p.AVAILABLE_SPACES as currentRemain,
    (p.TOTAL_SPACES - p.AVAILABLE_SPACES) as currentOccupancy,
    COUNT(DISTINCT v.VEHICLE_ID) as currentVehicles
FROM ParkingLot p
LEFT JOIN Vehicles v ON v.CURRENT_LOT_ID = p.LOT_ID
GROUP BY p.LOT_ID, p.TOTAL_SPACES, p.AVAILABLE_SPACES
HAVING MAX(p.TOTAL_SPACES) >= MIN(p.TOTAL_SPACES)
ORDER BY p.LOT_ID
```

9. Nested AGGREGATION with GROUP BY: getParkingLotById(lotId) in backend/appService.js: 835

   Fetches detailed information for a specific parking lot, including total capacity, current occupancy, and all vehicles currently parked there, using a nested aggregation.

```
SELECT
    sub.lotId,
    sub.capacity,
    sub.currentRemain,
    sub.currentOccupancy,
```

```
        v.PROVINCE,

        v.LICENSE_PLATE,

        v.PARKING_UNTIL

    FROM (

        SELECT

            p.LOT_ID AS lotId,

            p.TOTAL_SPACES AS capacity,

            p.AVAILABLE_SPACES AS currentRemain,

            (p.TOTAL_SPACES - p.AVAILABLE_SPACES) AS currentOccupancy,

            COUNT(*) AS count1

        FROM ParkingLot p

        WHERE p.LOT_ID = :1

        GROUP BY p.LOT_ID, p.TOTAL_SPACES, p.AVAILABLE_SPACES

    ) sub

    LEFT JOIN Vehicles v

        ON v.CURRENT_LOT_ID = sub.lotId
```

10. Division getUserViolations(userId, startDate, endDate) in backend/appService.js: 882

    Retrieves all violation tickets for a specific user and period, ensuring the user has violations related to every one of their vehicles.

```
    SELECT

        v.TICKET_ID as ticketId,

        v.REASON,
```

```
        v.TIME,

        v.LOT_ID as lotId,

        v.PROVINCE,

        v.LICENSE_PLATE,

        v.STATUS

    FROM Violations v

    JOIN Vehicles ve ON v.PROVINCE = ve.PROVINCE

                    AND v.LICENSE_PLATE = ve.LICENSE_PLATE

    WHERE ve.USER_ID = :1

    AND NOT EXISTS (

        SELECT ve2.VEHICLE_ID

        FROM Vehicles ve2

        WHERE ve2.USER_ID = ve.USER_ID

        MINUS

        SELECT ve3.VEHICLE_ID

        FROM Vehicles ve3

        WHERE ve3.USER_ID = ve.USER_ID

    )

    ORDER BY v.TIME DESC
```