University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 4

Date: November 26, 2024

Group Number: 4

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Angela Felicia	51190304	r5e5u	angelafc@students.cs.ubc.ca
Qinying Li	10959491	o3e0n	carina00@students.cs.ubc.ca
Leo Shang	52598174	c3r3z	lshang04@students.cs.ubc.ca

University of British Columbia, Vancouver
Department of Computer Science

Repository Deliverable Link

https://github.students.cs.ubc.ca/CPSC304-2024W-T1/project_c3r3z_o3e0n_r5e5u

SQL DDL, INSERT, and DROP statements

It's in the repository
project_c3r3z_o3e0n_r5e5u/LaundryMachineManager/sql-scripts/setup.sql

Project Purpose and Description

This application is designed to manage and track laundry services across university campus residences, making the laundry experience more convenient for students and efficient for maintenance teams.

The database will primarily track and display the real-time status and availability of laundry machines in campus residences. This allows users to easily check machine availability, plan their laundry sessions, and monitor the progress or completion status of their laundry, providing a more convenient and efficient laundry experience. The database also handles user management, enabling users to register and link their accounts to specific residences. Additionally, it tracks washing cards for machine payments, technician repairs to ensure accountability, and residence managers responsible for machine oversight. This model can be applied in real-life situations like university dorms, where students need easy access to laundry services, and maintenance teams need clear machine and repair tracking systems to ensure smooth operation.

University of British Columbia, Vancouver
Department of Computer Science

Final Schema - The schema which we updated

NOTE:

- *primary keys are underlined and foreign keys are bolded*
- *For primary keys, I did not put them under not null, unique, and candidate keys since they are implicitly candidate keys, not null, and unique anyways.*
- *Updated parts have been highlighted by pink color*

UserLivesIn (uid: UUID, **bid**: INT, uname: VARCHAR[20], uemail: VARCHAR(50), upassword: VARCHAR[50])

- **bid** references CampusResidence.
- uname default = “User”
- UNIQUE:
    - uemail
- NOT NULL:
    - bid
    - uname
    - uemail
    - upassword

ResidenceLaundryMachine (**bid**: INT, lid: INT, brand: VARCHAR[15], model:VARCHAR[20], washing status: VARCHAR[20])

- **bid** references CampusResidence(bid)
- brand default = “Coinamatic”
- NOT NULL
    - washing_status

LoadsWashingCard (cid: UUID, **uid**: UUID, balance: INT)

- balance default = 0
- uid references UserLivesIn
- NOT NULL:
    - balance
    - uid

ResidenceManager (mid: INT, mname: VARCHAR[20], mphone: VARCHAR[15], memail: VARCHAR[30], mpassword: VARCHAR[50])

- NOT NULL:
    - mname

- memail
- mphone
- mpassword

- Candidate Key
  - memail
  - mphone
- Unique
  - memail
  - mphone

Description of how our final schema differed from the schema we turned in

UserLivesIn

- changed the data type of **uid** from INT to **UUID** because we want to make a unique identifier for users in the application
- added **uemail VARCHAR(50) NOT NULL UNIQUE** and **upassword VARCHAR(50) NOT NULL** for user login function

ResidenceLaundryMachine

- changed **washing_status** from VARCHAR(10) to **VARCHAR(20)** because we need more characters to modify

LoadWashingCard

- changed the data type of **cid** from INT to **UUID** because we want to make a unique identifier for cards in the application

ResidenceManager

- added **mpassword VARCHAR(50) NOT NULL** for manager login function

University of British Columbia, Vancouver
Department of Computer Science

A list of all SQL queries

*All queries are in project_c3r3z_o3e0n_r5e5u/LaundryMachineManager/sql-scripts/Query for Grading.sql

- INSERT (LINE 3-6)
    - Register a user to a campus residence building
- INSERT (LINE 8-11)
    - Register a washing card for the user
- UPDATE (LINE 17-21)
    - Load the washing card by the user ID
- DELETE (LINE 27-30)
    - Remove the washing card from the system by the user ID
- DELETE (LINE 32-35)
    - Remove the user from the system by the user ID
- SELECTION (LINE 41-45)
    - View the specific laundry information by the building ID
- SELECTION (LINE 47-51)
    - View the specific transaction information by the card ID
- SELECTION (LINE 53-58)
    - View the user's information if they log in successfully
- PROJECTION (LINE 64-68)
    - View the card ID by the user ID
- PROJECTION (LINE 70-74)
    - View the card balance by the user ID
- JOIN (LINE 80-93)
    - View the information of all laundry machines by building
- JOIN (LINE 96-111)
    - View the washer machines by the specific building

- JOIN (LINE 113-128)
  - View the dryer machines by the specific building

- AGGREGATION WITH GROUP BY (LINE 134-149)
  - View the number of the laundry machines by the buildings

- AGGREGATION WITH HAVING (LINE 155-169)
  - View the buildings which have one or more available laundry machines

- NESTED AGGREGATION WITH GROUP BY (LINE 175-208)
  - View the laundry machines which are most frequently used
  - *Frequently used means this laundry machine using times are more than all laundry machines' average using times*

- DIVISION (LINE 214-242)
  - View the users who have filed feedback for every laundry machine in their building

For query 7-10, copy of SQL query and a maximum of 1-2 sentences describing what that query does.

/* 7. Aggregation with GROUP BY */

-- View the number of the laundry machines by the buildings

```
SELECT
  c.bid,
  c.bname,
  c.address,
  COUNT(r.lid) AS machine_count
FROM
  CampusResidence c,
  ResidenceLaundryMachine r
WHERE
  c.bid = r.bid
GROUP BY
  c.bid,
  c.bname,
  c.address;
```

/* 8. Aggregation with HAVING */

-- View the buildings which have one or more available laundry machines

```
SELECT
    c.bname AS Building_Name,
    r.washing_status AS Status,
    COUNT(r.lid) AS AvailableMachines
FROM

    ResidenceLaundryMachine r, CampusResidence c
WHERE
    r.washing_status = 'Available'
    AND c.bid = r.bid
GROUP BY
    r.bid, c.bname, r.washing_status
HAVING
    COUNT(r.lid) > 0;
```

/* 9. Nested aggregation with GROUP BY */

-- View the laundry machines which are most frequently used

-- Frequently used means this laundry machine using times
-- are more than all laundry machines' average using times

```
SELECT
    rlm.bid,
    cr.bname,
    rlm.lid,
    COUNT(p.cid) AS usage_count
FROM
    ResidenceLaundryMachine rlm,
    Pays p,
    CampusResidence cr
WHERE
    rlm.bid = p.bid AND rlm.lid = p.lid
    AND rlm.bid = cr.bid
GROUP BY
    rlm.bid,
    cr.bname,
    rlm.lid
HAVING
    COUNT(p.cid) > (
        SELECT AVG(machine_usage_count)
        FROM (
            SELECT
                COUNT(*) AS machine_usage_count
            FROM
                Pays
            GROUP BY
```

```
                bid,
                lid
        ) AS machine_usage_counts
    )
```

/* 10. Division */

-- View the users who have filed feedback for every laundry machine in their building

```
SELECT
    u.uid,
    u.uname,
    u.uemail,
    u.bid,
    c.bname AS bname,
    c.address AS address,
    lwc.cid
FROM
    UserLivesIn u,
    CampusResidence c,
    LoadsWashingCard lwc
WHERE
    u.bid = c.bid
    AND u.uid = lwc.uid
    AND NOT EXISTS (
        SELECT 1
        FROM ResidenceLaundryMachine r
        WHERE r.bid = u.bid
        AND NOT EXISTS (
            SELECT 1
            FROM ReportsFeedback rf
            WHERE rf.uid = u.uid
            AND rf.bid = r.bid
            AND rf.lid = r.lid
        )
    );
```