CPSC 304 Project Cover Page

Milestone #: 3
Date: October 25, 2024
Group Number: 93

Name	Student Number	CS Alias (Userid)	Preferred Email Address
Sabrina Lou	17539495	t1e8c	sabrinajwlou@gmail.com
Rohan Shukla	80120785	c7a8v	shuklarohan@live.com
Freddi Li	51907897	f5q5r	winnifredsli38@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 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.

Project Description
Our project is a public transportation system, used within the transportation management sector. It focuses on the operations, logistics, and staff management within a public transit system, based on the public transport available in the Metro Vancouver area. This includes managing routes, vehicles, stations and stops, drivers, riders, and fare payment.

Timeline and Task Breakdown (m4, m5, m6)

Implementation due November 29 (5 weeks from now)

Week 1: Setup
Make revisions based on Feedback from M2 then populate database **(SABRINA/ROHAN)**

Setup basic foundation using tutorial 7 as a guide, create a start/home-page **(FREDDI)**
- Split different query functionality to different pages so that our work doesn't cause issues with each others work
- Once done individual query tasks, add links to home page
- There will be three main pages:
    1. Routes manager
    2. Client manager
    3. Employee/vehicle manager

Week 2- Setup & Query Implementation

Routes manager (SABRINA)
- Query specific or all routes (SELECTION)
    - The user should be allowed to search for tuples using any number of AND/OR clauses and combinations of attributes. Conditions can be based on equality or more complex operations
- With a particular route, be able to look at its (bus stops/ train stations) and change line/route position and the route id(UPDATE)

Client Manager (ROHAN)
- List all clients and have sublists for their paid fares
- Able to add new clients and new fares (INSERT)
- Able to delete clients from the system(DELETE)
- Under the client list prompted to look at fares/scanner relationship and select which attributes you want to see(PROJECTION)

Employee/Vehicle Manager (FREDDI)
- Look at all employees and their assigned vehicles
- Search to find drivers driving specific routes (JOIN)
    - Find all drivers driving "99" .. etc
-

Meet on November 4th to check in and update each other on our progress:
- Brief explanation about each query for entire group comprehension

- Check in on if we have finished or not. If not, what challenges have we faced and how can we mitigate these?
- Looking forward and adjusting our schedule as needed

Week 3 - Continue Query Implementation and Begin GUI Implementation

Meet on November 11th to see our progress from weeks 1 and 2

- What have we completed?
- Looking forward and possibly reworking task splitting based on previous weeks of work

Queries 7–10 may be hardcoded and an interface to execute must be provided (e.g., button)

1. GROUP BY Aggregation
2. HAVING Aggregation
3. Nested Aggregation
4. Division

Routes Manager

- Group by Route id and *count* number of stops/stations (GROUP BY Aggregation) **(SABRINA)**

Client Manager

- Obtain Client/Rider who has purchased and validated a fare at every scanner (DIVISION) (**ROHAN)**
- Group by Scanner ID and *count* number of validated fares HAVING date > specified date (HAVING Aggregation) **(ROHAN)**
- Group by Scanner ID, count number of clients who have scanned, get scanner with MAX # of clients (Nested Aggregation) **(FREDDI)**

Start implementing the GUI

Routes Manager (SABRINA)

- View of query results tables
- View of original tables
- Text input interfaces for SELECTION and UPDATE queries
- Interface + button/dropdown (optional input) for GROUP BY

Client Manager (ROHAN)

- View of query results tables
- View of original tables
- Text input interfaces for INSERT DELETE PROJECTION queries
- Interface + button/dropdown for Division, HAVING, Nested Aggregation queries
-

Employee/Vehicle Manager (FREDDI)

- View of query results tables
- View of original tables
- Text input interfaces for JOIN query

Week 4 - Continue Implementation, Troubleshoot, and Refine Front-End

Meet on November 18th to check in once again

- How is our progress thus far? Are we on schedule?
- What are the biggest challenges we each are facing and how can we help each other?

Continue implementation tasks.

Week 5 - Finalize
Meet on November 25th for the final time
- Is there anything else we need to do at this point?
- Reflect on work for the past month: rose, bud, thorn
- Prep for demo

Possibly meet again before the demo to practice

Possible Challenges/Things Left to do

Translating normal string to sql queries
- This challenge presents itself more than once in our project, probably best to work as a group to create a function or class that handles string input and turns it into valid sql

Asynchronous workflow
- Each of us will be working on our own implementations that shouldn't impact each others work however there's bound to be some overlap. Possible solution is to create 3 developer branches and then each commit to our own branches and we can go through PRs in our scheduled weekly meetings

Front end styling/format
- Closer to the time when we're designing our front end we should come together to gather a general idea of how we want each page to look just so we can have the entire project be uniform in look/feel

Commits
Rohan: add milestone 1
Freddi: add milestone 2
Sabrina: add milestone 3

Repository link
https://github.students.cs.ubc.ca/CPSC304-2024W-T1/project_c7a8v_f5q5r_t1e8c