CPSC 304 Project Cover Page

Milestone #: 3

Date: March 11 , 2024

Group Number: 36

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Wendy Li	38036695	lwqz	wendyli.org@gmail.com
Antonia Tykei	82340886	atykei	antoniatykei@gmail.com
Wendy Tso	34159368	wendytso	wendywktso@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

CPSC304 Project Repository: project_i0b5b_k9f0w_q9m1b
https://github.students.cs.ubc.ca/CPSC304-2023W-T2/project_i0b5b_k9f0w_q9m1b

1. Project Summary

The project focuses on mechanical keyboard customization, providing an informational database and ordering system. Users can design, organize, and place orders for customized keyboards through the platform, which aims to streamline the customization process.

2. Timeline (entire project due April 5)

*As a group we aim to meet 2-3 times a week. Typically we meet for 1-2 hours after lectures on Tuesdays and Thursdays where we will ensure that we keep eachother on track and are gradually working towards completing the tasks required for this project. We also typically book an additional online meeting (as needed) to work on the project.

PREPARATION/ ADMINISTRATIVE ITEMS

Task	Assigned To	Deadline	Status
Clone repository, set up/access oracle account	ALL	Mach 15	*TODO*
Review SQl/Oracle resources and materials	ALL	March 15 (refer as needed)	*TODO*
Complete Tutorial 5: SQL Plus	ALL	March 8	Complete
Complete Tutorial 6: Java/Oracle	ALL	March 15	*TODO*
Complete Tutorial 7: PHP/Oracle	ALL	March 15	*TODO*
Ensure milestone 4 pdf is complete *refer to rubric*	ALL	April 2	*TODO*
• Cover page • Short Project Description • Final Schema • Screenshots for what data is	ALL	April 2	*TODO*

present in each relation after SQL script ● A list of all SQL queries + where it is found in the code ● Screenshots for demonstrating functionality of every query using GUI (before/during/after progression) ● README.txt			
Submit/ commit milestone 4	ALL	April 4	*TODO*
Milestone 5 demo run through (preparation)	ALL	April 7	*TODO*

BACKEND

Task (Q = query)	Assigned To	Deadline	Status	Details/Notes/Comments (as needed)
SQL script to create all the tables and data in the database	ALL	Mach 14	*TODO*	Include sufficient user data to demonstrate project functionality
Q: INSERT	WT	March 14	*TODO*	Implement SQL query for inserting data
Q: DELETE	WL	March 14	*TODO*	Implement SQL query for deleting data
Q: UPDATE	AT	March 14	*TODO*	Implement SQL query for updating data
Q: Selection	WT	March 14	*TODO*	Implement SQL query for data selection
Q: Projection	WL	March 14	*TODO*	Implement SQL query for data projections
Q: Join	AT	March 14	*TODO*	Implement SQL query for table/data joins
Q: Aggregation with Group By	WT	March 19	*TODO*	Implement SQL query for aggregation with group by

Q: Aggregation with Having	WL	March 19	*TODO*	Implement SQL query for aggregation with having
Q: Nested aggregation with group by	AT	March 19	*TODO*	Implement SQL query for nested aggregation with group by
Q: Division	WL	March 21	*TODO*	Implement SQL query for division
Review	ALL	March 21	*TODO*	All group members will review each others work

FRONTEND + DEMO PREP

Task	Assigned To	Deadline	Status	Details/Notes
Refer to/ review any resources for GUI development	ALL	March 15	*TODO*	- For Java, we recommend using JSwing library (much like what you worked with in CPSC 210). For PHP, your group will have to learn CSS+HTML. w3schools is a good site to learn the fundamentals of CSS and HTML.
Overview for GUI: **For each button:** **1) create button + table** **2) place button** **3) place data to table** **4) refresh table when button is triggered**	ALL (review)	-	*TODO*	- Many of the projects submitted in the past have multi page navigation as compression all queries onto one page is not the best for holistic design
Plan/discuss/sketch outline of what we would like GUI to look like (consider tools like Figma)	ALL	March 17	*TODO*	- When designing a GUI, think of a perspective of a user unfamiliar with CS - simple but adequate interface; it doesn't have to be fancy
• Insert	WT	Mach 21	*TODO*	- should be user friendly - EX "create new account" could be name of a button run for the 'insert query' - (more appropriate than 'insert query button' - The user should be able to specify what values to insert. The insert operation

				should affect more than one table (i.e., an insert should occur on a table with a foreign key) - Query should make sense given context - The INSERT operation should be able to handle the case where the foreign key value in the tuple does not exist in the table that is being referred to. This "handling" can either be that the tuple is rejected by the GUI with an appropriate error message or that the values that are being referred to are inserted. - The tables that the insert operation will run on can be pre-chosen by the group.
● Delete	WL	March 21	*TODO*	- Implement a cascade-on-delete situation (or an alternative that was agreed to by the TA if the DB system doesn't provide this). The user should be able to choose what values to delete. - The tables that the delete operation will run on can be chosen by the group. - The chosen query and table(s) should make sense given the context of the application.
● Update	AT	March 21	*TODO*	- The user should be able to update any number of non-primary key attributes in a relation. - The relation used for the update operation must have at least two non-primary key attributes. At least one non-primary key attribute must have either a UNIQUE constraint or be a foreign key that references another table. - The application should present the tuples that are available so that the user can select which tuple they want to update. - The chosen query and table(s) should make sense given the context of the application.
● Select button	WT	March 21	*TODO*	- The user is able to specify the filtering conditions for a given table. That is, the user is able to determine what shows up in the WHERE clause. It is fine to do these based only on equality, but more complex operations (e.g., less than) are also fine. - The user should be allowed to search for

				tuples using any number of AND/OR clauses. It is fine to implement this as a dropdown of and/or options; you don't have to cover all possible cases. - The group can choose which table to run this query on. The query and chosen table(s) should make sense given the context of the application.
● Projection button	WL	March 21	*TODO*	- The user is able to choose any number of attributes to view from any relation in the database. Non-selected attributes must not appear in the result. - One or more tables must contain at least four attributes. - The application must dynamically load the tables from the database (i.e., if we were to insert a new table in the database at the start of the demo, this new table should also show up as a possible option to project from). - You do not have to allow projection over multiple relations. Projecting over individual relations is sufficient for this rubric item. - In other words: we want you to do your projection by accessing the database to find the names of the tables and then the names of the attributes.
● Join button	AT	March 21	*TODO*	- Create one query in this category, which joins at least 2 tables and performs a meaningful query, and provide an interface for the user to execute this query. The user must provide at least one value to qualify in the WHERE clause (e.g. join the Customer and the Transaction table to find the names and phone numbers of all customers who have purchased a specific item). This must be able to be done in the GUI. The group can choose which tables will be affected by the query. The query and chosen table(s) should make sense given the context of the application.
● Aggregation with Group By	WT	March 26	*TODO*	- Create one query that requires the use of aggregation (min, max, average, or count are all fine), and provide an interface (e.g., HTML button/dropdown, etc.) for the user

				to execute this query. The group can choose which table to run this query on. - The schema can be statically set but the tuples used in the query cannot be predetermined. For example, you can specify that you want to find the total revenue per month but you cannot hardcode which exact months will be in the answer. If the TA asks to add in a sale for a new month, then that new month must be included in the returned results. - The query and chosen table(s) should make sense given the context of the application.
● Aggregation with Having	WL	March 26	*TODO*	- Create one meaningful query that requires the use of a HAVING clause, and provide an interface (e.g., HTML button/dropdown, etc.) for the user to execute this query. - The schema can be statically set but the tuples used in the query cannot be predetermined. - The query and chosen table(s) should make sense given the context of the application.
● Nested aggregation with group by	AT	March 26	*TODO*	- Create one query that finds some aggregated value for each group (e.g., use a nested subquery, such as finding the average number of items purchased per customer, subject to some constraint). Some examples for the Sailors table are given in the project specs. - The schema can be statically set but the tuples used in the query cannot be predetermined. - Note the difference between this query and the above Aggregation Query. You must use separate distinct queries for this criterion and the Aggregation Query (i.e., do not double dip). - It is fine to use a view to get the desired behaviour. - The query and chosen table(s) should make sense given the context of the application.
● Division	WL, AT	March 26	*TODO*	- Create one query of this category and

				provide an interface (i.e., HTML button, etc.) for the user to execute this query (e.g., find all the customers who bought all the items). - The schema can be statically set but the tuples used in the query cannot be predetermined. For example, you can specify that division will happen between this relation with x attributes and that relation with y attributes but the actual values used in the division operation should be dynamic. - The query and chosen table(s) should make sense given the context of the application.
Security	ALL	March 26	*TODO*	- Values from the user are not directly used in the database. Basic security practices to prevent injection and rainbow attacks have been followed. - Encryption is not required. But cover cases as discussed in class, e.g., https://xkcd.com/327/
Error Handling	ALL	March 26	*TODO*	- The user receives notifications about user errors such as trying to insert a duplicate value, invalid input (e.g., invalid characters or an int when only strings are allowed etc.), etc.
User Notification	ALL	March 26	*TODO*	- The user will receive a success or failure notification upon the completion of an insert, update, delete action and will have a way to verify the action's effect on the database.
Review	ALL	April 1	*TODO*	- All group members will review each others implementations

3. Description of challenges/ things left to do

- Competing commitments
    - **Suggestions/Solutions:** our group seems to get along well and we communicate well with each other. We will continue to keep one another up to date on our progress, any challenges we encounter or questions we have. If any member is struggling to complete any portions we can schedule time to work together and help one another or we may reassess and change some of our deadlines (if needed)

- Lack of familiarity with Oracle/SQL/PHP
    - **Suggestions/Solutions**: Consult resources shared by the teaching team as well as online tutorials as needed. Attend office hours
    - We are aiming to each have a query and the main SQL script prepared next week so that we are a) starting early and b) can address any challenges early on.
- Integrating our individual work
    - **Suggestions/Solutions**: we plan to work together during a number of sessions throughout the week, uses branches in git and test to ensure our project works as a whole
    - Also we will plan and design the architecture together which should aid with integration
- Debugging and Testing
    - **Suggestions/Solutions:** we can help review each other's work to try to find bugs and similarly test each other's work
    - For complex situations we should all aim to attend office hours
- Seek Feedback and Guidance
    - **Suggestions/Solutions:** reach out to our project TA, OH TAs or instructors if we find we need some guidance (after we have thoroughly reviewed project FAQs, piazza and other resources)