University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 3

Date: October 24, 2024

Group Number: 116

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Quang Duy Do	49549538	n6k8c	jackydo1974@gmail.com
Hai Son Vu	23411960	f0f5u	vuhaison16@gmail.com
Sabir Shaikh	65129090	c0d1a	mohammedsabirshaikh2@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

1. Brief project description

- A Charity Donation management database, where the best securities are applied for organizations to conduct collecting money and creating funds (which requires verification from Admin and statements from Financial Companies).
- Users and organizations can donate money to funds.
- Finance Companies are the storage of the money, and providing statements about the use of money
- Organizations need to provide legally lawful documents and verification from admins in order to start and open a Fund.
-

2. Task timeline

TASK	Expected deadline (23:59 pm on the day of)	Assigned person
Project environment setup and configuration (We use **Node**.js with **Express** for back end and **React**.js for the front end along with SQL **Oracle**). This includes: - Setting up Node on every teammates' local computers - Setting up Oracle SSH - Creating the package.json and install the important dependencies	10/31/2024	Jack Do
Writing **SQL Scripts to create the tables** in the database	10/31/2024	Jack Do
Writing the raw SQL queries to INSERT data into each table. The queries must be	11/06/2024	Sabir

runnable, and must be tested on the terminal. Definition of Done: a command with **INSERT** and the information that put in must be inserted into the appropriate table.		
Writing the raw SQL queries to **UPDATE** data into each table. The queries must be runnable, and must be tested on the terminal. Only the non-primary key attributes can be updated.	11/06/2024	Jack Do
Writing the raw SQL queries to **DELETE** data into each table. The queries must be runnable, and must be tested on the terminal. Definition of DONE: A DELETE query must delete a single tuple with a provided primary key. If not found, throw an error.	11/06/2024	Son
Writing the raw SQL queries to **retrieve SINGLE tuple** in each table. The query must join appropriate tables, and must return a single tuple by a provided primary key in the request body.	11/08/2024	Sabir
PROJECTION: Writing the raw SQL queries to **retrieve requested attributes** in each table. The query must return the attributes values of the table with requested attributes from users. If an input attribute is not found, throw an error.	11/08/2024	Sabir
SELECTION, JOIN, AGGREGRATION WITH GROUP BY, HAVING, NESTED AGGREGATION, DIVISION queries Writing the raw SQL queries to retrieve	11/14/2024	Jack Do, Sabir, Son

MULTIPLE tuples in each table. The query must join appropriate tables, and must return a single tuple by a provided primary key in the request body. Definition of DONE: the query must return multiple tuples based on all types of searching queries: comparison, AND/OR wildcards, include,...		
POPULATING the database The database tables are populated with 5-20 tuples/records for each table. The records must not be too similar, so checking on retrieving queries can be more accurate.	11/16/2024	Sabir
Develop the Node.js back-end code: - Creating the data models - each table must have a class/interface representing it in the code. - Creating the data access object. Implement the database connection.	11/20/2024	Jack Do
Develop the Node.js back-end code: - Creating the **REPOSITORY pattern.** This repository pattern will run appropriate SQL queries and reduce the total time used to implement each API.	11/24/2024	Jack Do
Develop the Node.js back-end code: - Implementing the basic APIs to interact with the database. The APIs should use the written SQL queries, and must correctly impact the database.	11/26/2024	Jack Do
Develop the AUTHENTICATION and AUTHORIZATION feature with JSON WEB TOKEN and Cookies	11/27/2024	Jack Do
Develop the global Error handling classes for	11/27/2024	Jack Do

the back end		
FRONT END		
Set up the UI framework (React). Implement key pages (basic: frames, head, body of the page- like a web) : ● Home,Transactions, Admin, Funds, Volunteer Organizations, Finance Companies. Handle API calls (using Axios or Fetch). Implement authentication and session management.	11/10/2024	Son
Basic one-page GUI with ALL buttons and input form to interact with each table in the database.	11/18/2024	Son
More pages GUI with login/signup pages, viewing pages, updating pages,...	11/27/2024	Son
FINISHING UP		
Writing the project PDF file submission	11/27/2024	Jack, Son, Sabir
Citing the work, preparing the project presentation.	11/30/2024	Jack, Son, Sabir
Deploying the website to Railway/ AWS EC2	11/30/2024	Jack

3. Repository URL

https://github.students.cs.ubc.ca/CPSC304-2024W-T1/project_c0d1a_f0f5u_n6k8c