



**DALHOUSIE  
UNIVERSITY**

**Faculty of Computer Science**

**CSCI 4140**

**Advanced Database Systems**

**Final Report**

**August 14<sup>th</sup>, 2021**

**Submitted By:**

Adama Camara (B00757421)

Clarizze Santos (B00730508)


Sijia Han (B00734818)

## Table of Contents

<b>1. Project Overview</b>	<b>4</b>
<b>2. Purpose of the project</b>	<b>5</b>
2.1 Business requirements and rules	5
2.1.1 Database design:	5
2.1.2 Business rules	5
<b>3. Tech Stack and Software</b>	<b>5</b>
3.1 Frontend	6
3.2 Backend	6
3.3 Tools used	6
<b>4. API Performance</b>	<b>6</b>
<b>5. ER Model Diagram</b>	<b>7</b>
<b>6. Test Plans</b>	<b>8</b>
6.1 Features to be tested	8
6.2 Acceptance criteria	8
6.4 Open issues	10
<b>7. Software Organization</b>	<b>10</b>
7.1 Database	10
7.2 Framework used for UI development	10
7.2.1 Language	10
7.2.2 Design pattern	10
7.3 Website pages (Front-end) designs	10
7.3.1 Login & Registration page	10
7.3.2 Dashboard	11
7.3.3 Profile page	12
7.3.4 Orders page	13
7.3.5 Books page	13
7.3.6 About us page	14
<b>8. Future Work</b>	<b>15</b>
2PC Utilization	15
<b>9. Submission Excellence</b>	<b>16</b>
<b>10. Project Repository</b>	<b>17</b>
<b>11. Running Project Code/Script</b>	<b>17</b>
<b>12. Reference:</b>	<b>18</b>



# 1. Project Overview

In this project we aim to create a client-agent web application that provides a user interface to agents for managing incoming orders and updating the availability of parts in the warehouse. This document contains all the project development for the web application READ  IT.

## 2. Purpose of the project

### 2.1 Business requirements and rules

#### 2.1.1 Database design:

The organization hired our team to develop a database with a front-end application for their agents to manage incoming orders and adding/updating availability of parts that are stored in the company warehouse. The application that we created have a client front-end where placing of order would be made and client will be able to check their order status and their balance. [4]

Our database design was setup in a manner that changing the price of a part would automatically change the price stored in the purchase order table. Currently, there are four processes in the website: created order, pending order, processing and cancelled. This influenced the efficiency of the business as it informs the agents which process is the order. In compensation for the possible inconvenience, we have charge \$5.00 for each cancelled order. If the order is created or pending and the client decides to cancel the order, they will not get charged of the \$5.00 cancellation fee. However, once the order is in process and the client decides to cancel their orders, they will be charged the \$5.00 cancellation fee. [11]

#### 2.1.2 Business rules

There are two types of user role available upon signing up: client and agent. Every purchase from a client will create a purchase order. The agent will then oversee completion of purchase order and confirm the order by making sure the parts being requested by the client are available and the order can proceed. [13]

We added triggers and attributes to part entity to make sure that purchase order will only be processed depending on the availability of part in the company current inventory. By using the attributes in parts, it notifies the company if the quantity on hand of their part is below the minimum always required of part available. The triggers added to the database are currently allowing clients to create purchase orders and add items by creating lines. The triggers implemented are currently changing the quantity on hand of part whenever a client create a new line. The price of the purchase order is also changing depending on the product of  $\text{lineUnit} * \text{linePrice}$ . The amount that client owed is only updated when the purchase order has been processed. We also added charging a processing fee of \$5 when client cancel a purchase order that was already processed.

### 3. Tech Stack and Software

We used model-view-controller design pattern for the overall the architecture of our project. We have a front-end, where it contains all the design and information structure of the website we will be creating and a backend that is connected to the database that stores all the information requested by the user. Below are further details of the different frameworks that we will be using to establish our project. [11]

#### 3.1 Frontend

Our team decided to use HTML, CSS, and a bit of ReactJS for our front-end. We chose these for our project because all the team members have adequate experience with HTML and CSS and using ReactJS complements along with the other frameworks we will use for our backend framework and the database. We expect a minimal learning curve with using ReactJS, since we have few functionalities needed to connect the frontend to the backend. To enhance further the layout and design of our project, we will also be using Bootstrap framework. [11]

#### 3.2 Backend

We used **NodeJS** and boilerplate the back end by generating an Express app. The Express app is connected to the tester local machine **MySQL**. The front-end uses **axios** to make the http requests, and the library **React-uuid** to generate random and unique identifiers for the creation of purchase order. [11]

#### 3.3 Tools used

In terms of IDEs and software, we used Virtual Studio Code as most team members are familiar with using it. We also used MySQL Workbench to generate the ER Diagram shown above. The codes for the back end and front-end are stored under the GitLab link provided in the next section. [11]

#### 3.4 Project dependencies

In terms of IDEs and software, we used Virtual Studio Code as most team members are familiar with using it. We also used MySQL Workbench to generate the ER Diagram shown above. The codes for the back end and front-end are stored under the GitLab link provided in the next section. [11]

### 4. API Performance

Most of the GET request for listing parts, clients, lines, and purchase orders are around 44ms.

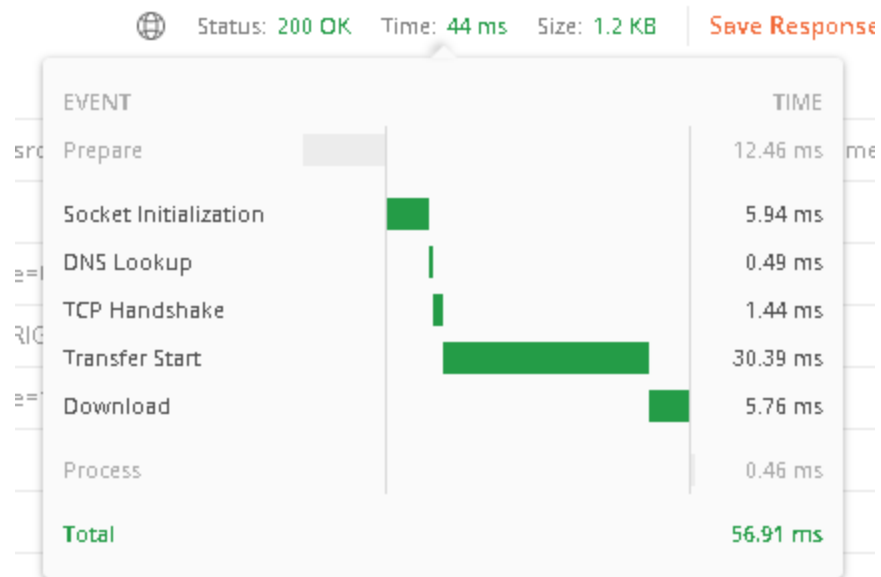


Figure 1. API Performance from Postman. [5][12]

## 5. ER Model Diagram

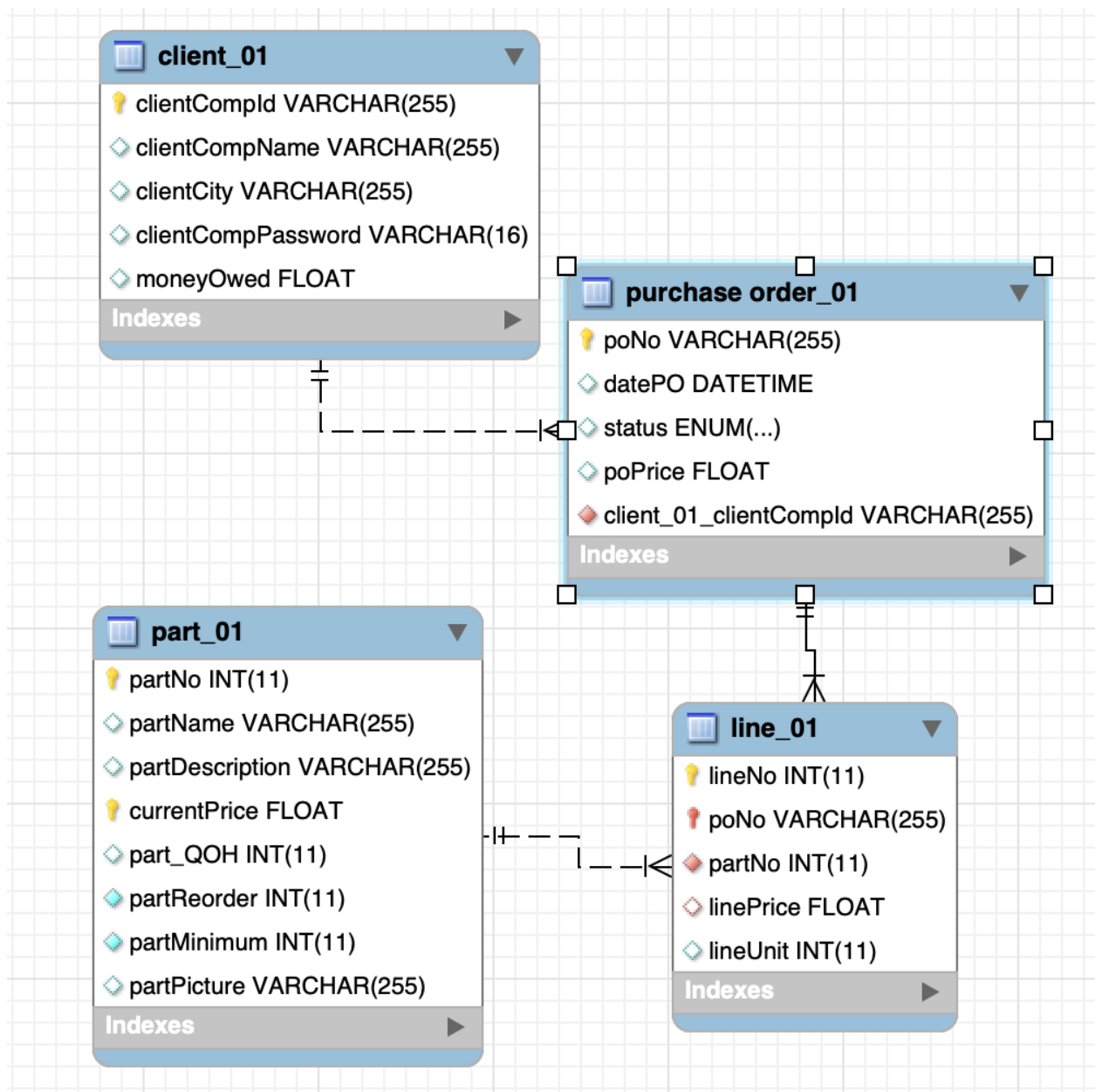


Figure 2. Improved ERD from Assignment 3. [12]

Figure 2 is the ERD that we have been using since Assignment 2. We had good feedback about our design and was suggested to just add few more attributes after the ERD we have for Assignment 1 for improvement.



## 6. Test Plans

For testing purposes, the application currently has the following two testing accounts.

The agent account:

- Username: acamara
- Password: password

The client account:

- Username: demo\_client
- Password: password

### 6.1 Features to be tested

Concurrent Testing of Software for Companies W, X, and Y Each member, who is logged at one of the companies W, X, and Y, will perform each of the following operations. After each operation, I shall ask one of the members to share a screen and show the results of the operation.

- List information about all clients
- List information about all parts (products)
- Change price and quantity for a part (given part number)
- Submit a purchase order successfully
- Process a purchase order successfully
- Process a PO unsuccessfully
- Tests as requested

Testing Software for Company Z (headquarters)

- List information about all clients
- List information about all parts (fetched from W, X, and Y)
  - Test case when each of W, X, and Y have parts with the same part number
  - (If the above does not work, then a test case when W, X, and Y do not have any parts with the same part number)
- Submit a purchase order successfully
- Process a purchase order successfully
- Submit a PO with problems (error checking)
- Process a PO unsuccessfully (insufficient parts)
- Concurrently at each of W, X, and Y, retrieve a list of parts
- Tests as requested

## 6.2 Acceptance criteria

The team agreed on the following acceptance criteria for the features that were being implemented since Assignment 3.

- C1:
  - **Scenario** - List information about all clients
  - **Given** - a valid agent user is logged in
  - **When** - the agent goes to the dashboard and click on the client tabs
  - **Then** – a list of clients should appear with their username, balance, and company names.
- C2:
  - **Scenario** - List information about all parts(products)
  - **Given** - a valid agent or client user is logged in
  - **When** - the user goes to the book page
  - **Then** – a list of parts should appear with their name, price, pictures, and option to purchase them.
- C3:
  - **Scenario** - Change price and quantity for a part (given part number)
  - **Given** - a valid agent user is logged in
  - **When** - the agent goes to the dashboard and opens the inventory tab to access all parts
  - **Then** – the agent enters a new price in the part form and saves the changes.
- C4:
  - **Scenario** - Submit a purchase order successfully
  - **Given** - a valid agent or client user is logged in
  - **When** - the users go to the order page and click on the process button on an order
  - **Then** – the order status gets changed to processed.
- C5:
  - **Scenario** - Process a purchase order successfully
  - **Given** - a valid agent or client user is logged in
  - **When** - the users go to the order page and click on the process button on an order
  - **Then** – the order status gets changed to processed.
- C6:
  - **Scenario** - List information about all parts (fetched from W, X, and Y)
  - **Given** - a valid agent user is logged in
  - **When** - the users go to the dashboard and click on inventory tabs
  - **Then** – then a list of parts should be displayed from 3 companies.
- C7:
  - **Scenario** - Create new account
  - **Given** – a username, role, password passed into the register interface
  - **When** – the register button is pressed

- **Then** – the API creates a new account and gives feedback to registration front-end
- C8:
  - **Scenario** - Sign in existing account
  - **Given** – a username, password passed into the login interface
  - **When** – the login button is pressed
  - **Then** – the API response back with a token and keep the user logged in for 24 hours.
- C9:
  - **Scenario** - Remove item from order
  - **Given** – a created order with item in it
  - **When** – the user opens their order and clicks on the remove button on an item
  - **Then** – the item should be removed from the order and the price of the order should decrease with a trigger set in the database.
- C10:
  - **Scenario** - Add item to order
  - **Given** – a created order
  - **When** – the user goes into the Book page and add item to order
  - **Then** – the item should be added to the order and the price of the order should increase with a trigger set in the database.

## 6.4 Open issues

The software currently has an open issue with the following features:

- **Change price and quantity for a part (given part number)**
- **Issue with PriceChange.js**
- Connection error when GET request is sent in line 38.

## 7. Software Organization

### 7.1 Database

#### 7.1.1 Functions:

- **getPartQTY\_001(id INT)**: it returns the quantity of a product by id
- **checkPartAvailability\_001 (qty INT, minimum INT)**: it returns the difference between the quantity on hand of a specific part and the minimum allowed of part present in inventory.
- **getPoAmountByPoNo\_001 (poNo INT)**: it returns the amount of a purchase order based on the purchase id passed in the stored procedures.
- **getMoneyOwnedByClientID\_001 (clientCompId: VARCHAR ())**: it returns an INT of the money owned by a client after submitting purchases.

### 7.1.2 Procedures:

- **updateClientAmountOwnedById\_001 (clientCompId INT):** it updates the client due amount
- **updatePOAmountByPoNo\_001 (poNo varchar):** it updates the purchase order total amount.
- **lineRemove (poNo varchar):** This procedure will remove all the lines from a cancel purchase order. This procedure needs to be call when client cancel an order. Doing so, will free the memory and increase the quantity on hand of parts.

### 7.1.3 Triggers:

- **Trigger#1 UpdatePartQuantityOnHand\_01:** This trigger will update the number of quantities on hand when part is added in purchase order.
- **Trigger#2 trg\_increasePoAmount\_01\_on\_insert\_line:** This trigger will update the amount for the purchase order that just received a new selected part. Happen when new line is inserted.
- **Trigger#3 update\_QOH\_on\_lineUnit\_01:** This trigger will update the number of part quantity on hand on the Part table when user reduce or increase lineUnit.
- **Trigger#4 updateReorder\_01:** This trigger will check if all part quantity on hand is above the minimum required number of products. If QOH < MINIMUM, we notify the agent.
- **Trigger#5 update\_QOH\_on\_line\_removed:** This trigger will update the quantity on hand from the Part table. This happen when client delete a selected product from their purchase order (removing line)
- **Trigger#6 trg\_decreasePoAmount\_on\_delete\_line\_01:** This trigger will update the amount for the purchase order that just removed a selected part. Happen when line is deleted.
- **Trigger#8 trg\_on\_purchase\_order\_cancellation\_fees:** This trigger will update the amount for the purchase order that just removed a selected part. Happen when line is deleted.

## 7.2 Framework used for UI development

### 7.2.1 Language


The main languages we used for the front-end of our project are JavaScript, HTML, and CSS. In addition, we applied Reat Bootstraps for the design of the project. The application stack can be called MERN, which are MySQL, Express.js, React.js and Node.js. [13]

### 7.2.2 Design pattern

The application follows the model view controller design pattern. In the folder src/views, we have the model for the Part and Order entity. The controllers are in the folder src/forms. They take the data model of Part and Order as props. Most data are passed down to their child component using props, and the loading of each screen was optimized for a great user experience. [13]

## 7.3 Website pages (Front-end) designs

### 7.3.1 Login & Registration page

- The user obtains complete website access of READ  IT by completing fill-in the information in the figure below. In addition, the user will choose to fill in "client" or "agent" in the registration interface to obtain different access rights. [13]



The sign-up page for READ IT features a central form with the following elements:

- Logo: READ  IT
- Text: Create an account
- Form fields:
  - Username
  - Name
  - ex: Halifax
  - client or agent
- Sign Up button (blue)
- Link: Already have an account?

Figure 3. READ  IT's Sign-up Page. [13]

- Clients and agents will access the website by verifying the registered usernames and matching passwords. [13]



The sign-in page for READ IT features a central form with the following elements:

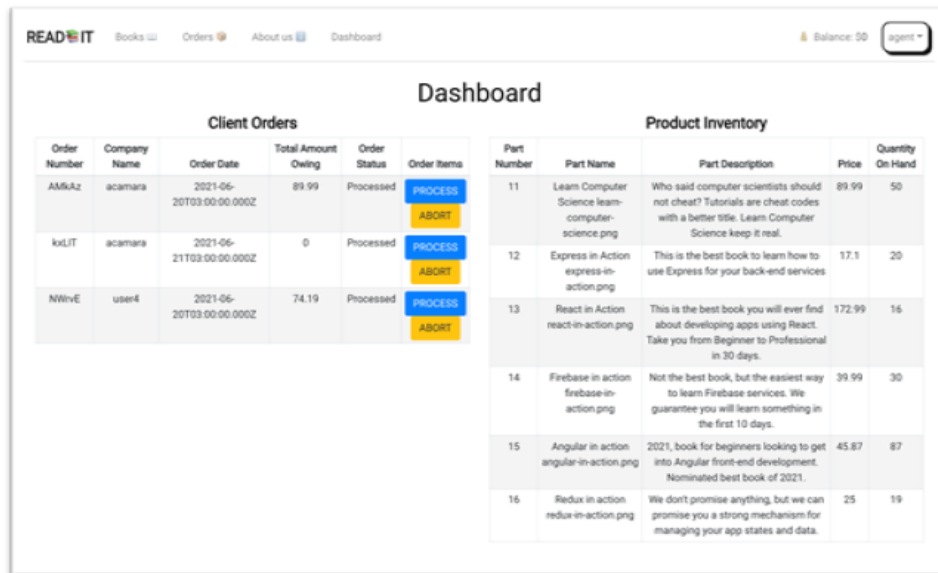
- Logo: READ  IT
- Text: Please sign in
- Form fields:
  - user123
  - password
- Login button (blue)
- Link: Don't have an account yet?

Figure 4. READ  IT's Sign-In Page. [13]

### 7.3.2 Dashboard

- The dashboard can only be accessed if you are registered as Agents. It includes the client orders list and the product inventory list. The agents are available to handle the clients' orders by choosing to process the orders or abort the orders in the dashboard. Also, it is convenient for

agents to check the QOH (quantity on hand) of each product in the stock through the dashboard. [13]



The screenshot shows the 'Agents Dashboard' with two main sections: 'Client Orders' and 'Product Inventory'. The 'Client Orders' table lists three orders with columns for Order Number, Company Name, Order Date, Total Amount Owling, Order Status, and Order Items. The 'Product Inventory' table lists 16 products with columns for Part Number, Part Name, Part Description, Price, and Quantity On Hand. Each order in the 'Client Orders' table has 'PROCESS' and 'ABORT' buttons. The 'Product Inventory' table shows various books and their quantities.

Client Orders					
Order Number	Company Name	Order Date	Total Amount Owling	Order Status	Order Items
AMkAz	acamara	2021-06-20T03:00:00.000Z	89.99	Processed	PROCESS ABORT
kuLIT	acamara	2021-06-21T03:00:00.000Z	0	Processed	PROCESS ABORT
NWivE	user4	2021-06-20T03:00:00.000Z	74.19	Processed	PROCESS ABORT

Product Inventory				
Part Number	Part Name	Part Description	Price	Quantity On Hand
11	Learn Computer Science learn-computer-science.png	Who said computer scientists should not cheat? Tutorials are cheat codes with a better title. Learn Computer Science keep it real.	89.99	50
12	Express in Action express-in-action.png	This is the best book to learn how to use Express for your back-end services	17.1	20
13	React in Action react-in-action.png	This is the best book you will ever find about developing apps using React. Take you from Beginner to Professional in 30 days.	172.99	16
14	Firebase in action firebase-in-action.png	Not the best book, but the easiest way to learn Firebase services. We guarantee you will learn something in the first 10 days.	39.99	30
15	Angular in action angular-in-action.png	2021, book for beginners looking to get into Angular front-end development. Nominated best book of 2021.	45.87	87
16	Redux in action redux-in-action.png	We don't promise anything, but we can promise you a strong mechanism for managing your app states and data.	25	19

Figure 5. Agents Dashboard. [13]

### 7.3.3 Profile page

- The profile page displays user's information and the account balance. [13]

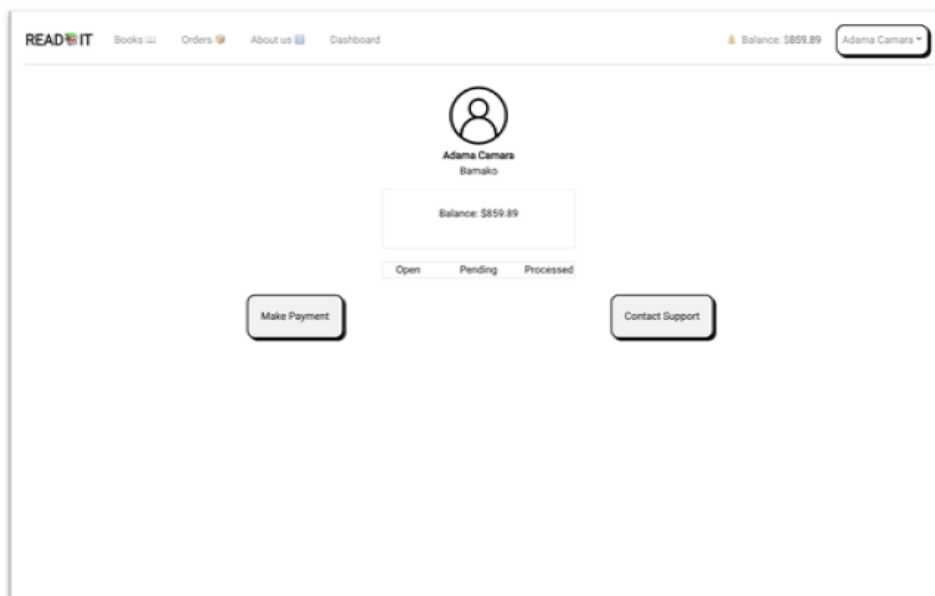


Figure 6. Client and Agents Profile Page. [13]

### 7.3.4 Orders page

- The orders page will have different accessibilities for Agents and Clients. For Clients, the page will display their orders from READ IT. Clients are able to edit or cancel their orders here. For Agents, the page will show the orders for agents to restock the product inventory. [13]

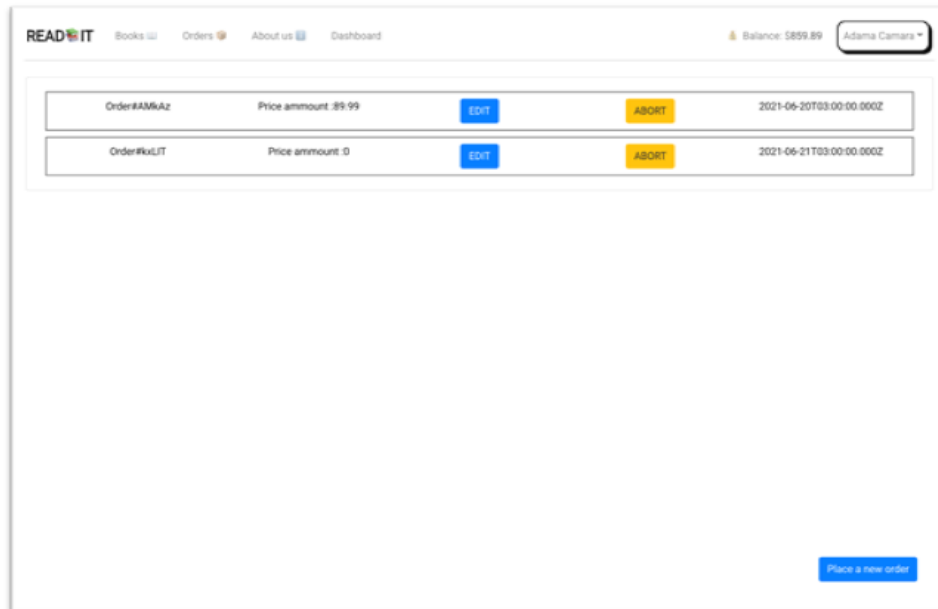


Figure 7. Clients and Agents Orders Page. [13]

### 7.3.5 Books page

- The books page is available for both Agents and Clients. Additionally, customers can browse the books that are available to be ordered. [13]

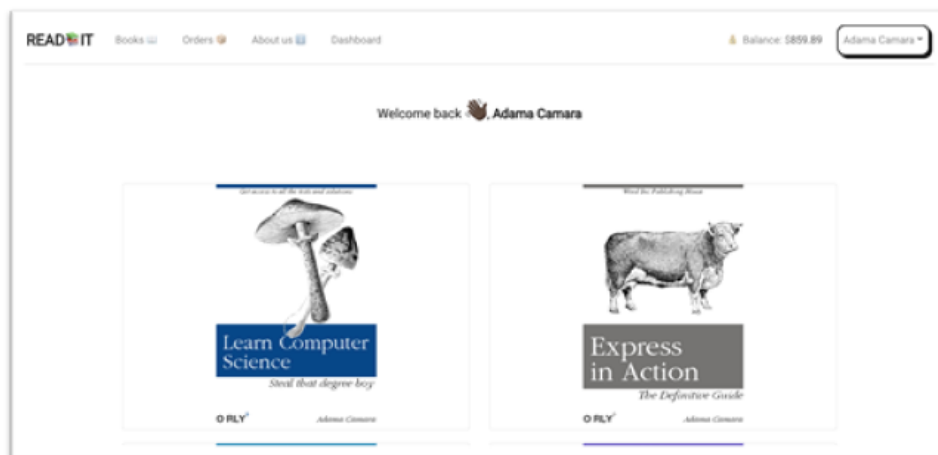


Figure 8. Books Page. [13]

- To add the book to the order, customers need to click on the book which they want to buy and select the number of books they want. [13]

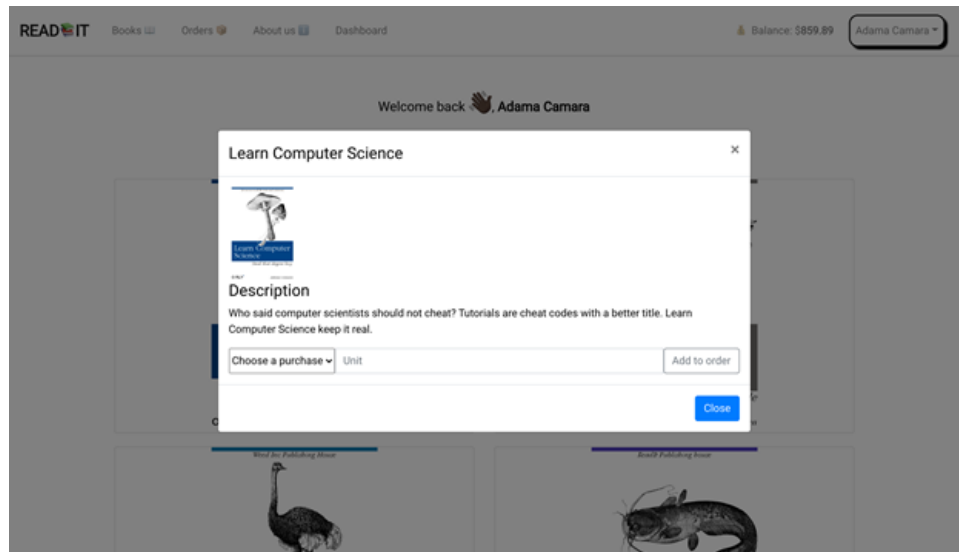


Figure 9. Books Page Modal to Order Book. [13]

### 7.3.6 About us page

- About us page displays the information related to our company and website. This page is accessible for both clients and agents. [13]

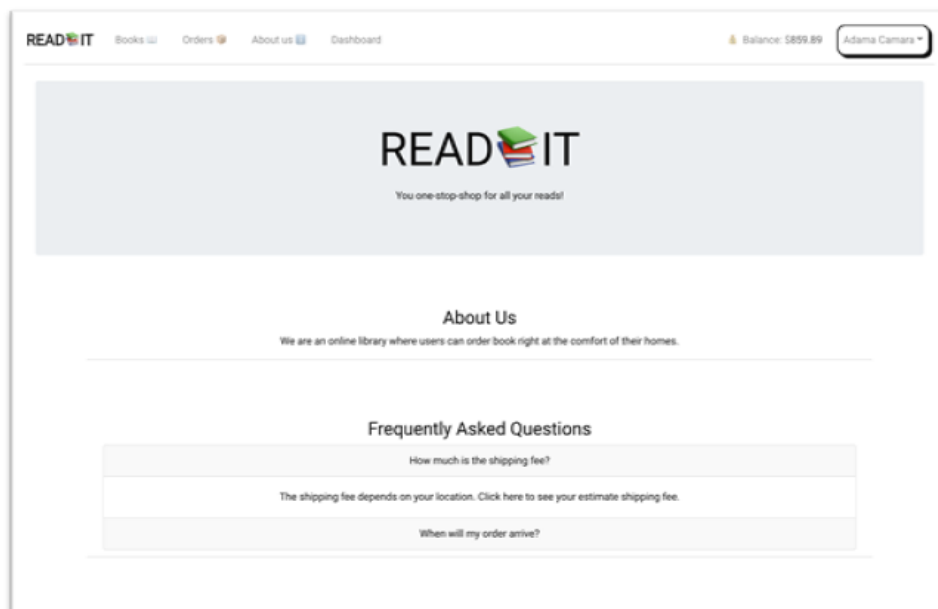


Figure 10. About Us Page. [13]



## 8. Future Work

### 8.1 2PC Utilization

- The current state of the software is set to rely on asynchronous replication techniques to enforce data integrity. The team attempted to configure the Dal FCS servers to enable a master-slave type of configuration. The idea was to use the slaves as snapshots that are mirror replicas of the master database.
- Our initial goal was to have any action of update or deleting data, will get coordinated using the SQL\*Net
- A new table (pending\_operation) would be needed to log result of each operation. For example, if an operation from a transaction is issued and the database either crashed or front-end lose connection with the API, the table "pending\_operation" will log the information needed to perform a debugging.
- The policy for the 2PC environment would be to have the entire transaction rollback if any operation of the transaction fails.

### 8.2 Front-end tasks:

- We will add more error handlers in the future to deal with potential bugs.

## 9. Submission Excellence

Our team believes that we have went above and beyond that deserves the A- - A+ grade in this course. We believe this is the case because in our every assignment:

We have explicitly explained the changes and the reasonable justifications on why we did the necessary changes, considering the given the feedback that we have received every submission.

We were able to demonstrate project and programming principles like MVC AND MERN in our stack organization and design pattern which helped in efficiency of our overall project.

The documentation that we submit had logos, proper format, labels, and reference that were consistent all throughout the course and submission of our assignments. The screenshots that we have added have their labels and we have properly referenced, even the software and websites that we use to complete the assignments. We have also always added a bit of information from our previous assignments to demonstrate the progress that we have made and the efficiency and overall understanding of each assignment report.

We were always ahead of the project requirements. We have been working on REST APIs since the first assignment and was able to get it done even before the submission for the next assignment. This allowed us to prepare for the following assignments and to plan tasks accordingly. Our work being advanced was also evident when we were not able to demonstrate what happens when a purchase order processed unsuccessfully. We were not able to do so because we have designed our database so that when a product being ordered is out of stock, it will not show on the website and eliminates the possibility of an order being processed and the client being charged.

We were ahead of the assignment requirements despite the learning curve present since most of the members were not familiar with React and Node JS.

Despite having minimal supervision from the TAs and varied assignment descriptions, we had created business logic that we find is suitable for our chosen products and company and have made a website that is reasonable and works well.

Lastly, A- - A+ have been the range of grades that we have been receiving in the assignments. We have not received a grade below this range and had minimal changes and improvements for the future assignments, which gives evidence that we have been doing our assignments beyond expectation and hence explains the excellence that we have performed in this course project.

## 10. Project Repository

GitLab Repository: [https://git.cs.dal.ca/apaul/4140s21\\_group1.git](https://git.cs.dal.ca/apaul/4140s21_group1.git)

## 11. Running Project Code/Script

### Required Software to run Group 1's application:

- Node.js, Cisco VPN Networks
  - \*\* Warning \*\*
    - Testing the app requires to run the server first, then the front-end.
    - The server is set to use port 3000, and the front-end will automatically set its own port to 3001.
1. Clone the repository onto local machine ([https://git.cs.dal.ca/apaul/4140s21\\_group1.git](https://git.cs.dal.ca/apaul/4140s21_group1.git)).
  2. Connect to Dalhousie server with Cisco VPN Networks.
  3. Open a command-line tool.
  4. Change to A3 folder using the command `'cd A3'`
  5. Install Node.js if your computer doesn't have it using `'npm install'` to install new dependencies.
  6. Enter `'npm start'` to start the application process
  7. Server will be running on <http://localhost:3000/>
  8. Open a new terminal and cd to Front-end/app
  9. Install the dependencies using `"npm install"`
  10. Then do `"npm start"` to start the front-end application.

## 12. Reference:

- [1] “MySQL Workbench,” *MySQL*. [Online]. Available: <https://www.mysql.com/products/workbench/>. [Accessed: 15-May-2021].
- [2] “Heroku,” Cloud Application Platform. [Online]. Available: <https://www.heroku.com/>. [Accessed: 22-May-2021]. The team used a free hosting service provided by Heroku for previewing the database. 18669531515
- [3] “Tutorial –MySQL Workbench.” Faculty of Computer Science, Dalhousie University CSCI 4140 Summer 2021, Halifax, 03-May-2021.
- [4] “CSCI 4140 Advance Database Systems – Assignment 1.” Faculty of Computer Science, Dalhousie University CSCI 4140 Summer 2021, Halifax, 12-Jun-2021.
- [5] “Postman,” *Postman*. [Online]. Available: <https://www.postman.com/>. [Accessed: 12-Jun-2021].
- [6] “Lucidchart,” *Lucidchart*. [Online]. Available: <https://www.lucidchart.com>. [Accessed: 12-Jun-2021].
- [7] “Undraw Open Source” Factory. [Online]. Available: <https://undraw.co/illustrations> [Accessed: 03-July-2021]
- [8] “O RLY Parody Book” Pictures of book cover [Online]. Available: <https://dev.to/rly> [Accessed: 20-July-2021]
- [9] “Lottie Files”. [Online]. Available: <https://lottiefiles.com/54639-boy-studying-science> [Accessed: 20-July-2021]
- [10] “The collaborative interface design tool.,” *Figma*. [Online]. Available: <https://www.figma.com/>. [Accessed: 24-Jul-2021].
- [11] “CSCI 4140 Advance Database Systems – Assignment 2.” Faculty of Computer Science, Dalhousie University CSCI 4140 Summer 2021, Halifax, 12-Aug-2021.
- [12] “CSCI 4140 Advance Database Systems – Assignment 3.” Faculty of Computer Science, Dalhousie University CSCI 4140 Summer 2021, Halifax, 12-Aug-2021.
- [13] “CSCI 4140 Advance Database Systems – Assignment 4.” Faculty of Computer Science, Dalhousie University CSCI 4140 Summer 2021, Halifax, 12-Aug-2021.