CS 5200 Project Report

08.12.2022

—

Team name: BudhirajaPSiddiquiS
Team members: Pratik Budhiraja, Shamekh Siddiqui

Overview

This E-commerce application allows a customer to browse products offered by suppliers and place an order for home delivery. We implement a relational database that tracks customers and sellers while storing corresponding real time information on products and orders .

The ideal users of this database are customers who want to place and track products related to an order while the supplier gets user and order information for analytics . A customer can add themselves to the database by signing up or logging in and searching through the product catalog offered by respective sellers. A customer then adds all the products and creates an order upon hitting the confirm button.

README

1. **Create the database using provided Dump:** The first step is to create and insert data into the database. We have provided the data dump which can be used to populate the data.
    a. Import provided dump and execute in MySQL workbench
2. **Python version:** The python version used in our program is 3.9.12 however we believe that any version upwards of 3.8 should be sufficient
3. **Flask and dependencies**: please install flask and dependencies using the command line or where you run your project using the following pip install commands:
    a. $pip install flask
    b. $pip install flask-login
    c. pip install flask-mysql
4. **Change username and password:** In the **_init_.py** python file (in website folder/package), please change USERNAME, PASSWORD, HOSTNAME on line 7-9 to your workbench localhost username and password
5. **Installing Werkzeug:** Even though this comes with the flask framework, we still recommend you try to install this just in case [pip install -U Werkzeug]
6. **Recommended Browser:** Safari, Any browser outside chrome
7. **!!* Run application *!! :** run **main.py** file (in the root folder/package) and you get **Running on "http://127.0.xyz"** output on the command line. Copy this website url and paste in your browser (**except chrome** it may lead to bugs)

Design and Features

1. The user is given an intuitive GUI website to interact with.
2. We allow user more than one way to perform CRUD operations on various tables via the application:

   a. **CREATE**
      i. The customer can add profile details (INSERTS into users table)
      ii. The customer can add an address (INSERTS into address table)
      iii. The customer can add products to cart and place an order (INSERTS into order and order_product_details table)

   b. **READ**
      i. The product catalog is displayed to the customer (SELECT from product table)
      ii. The customer can view orders summary (SELECT from orders table)
      iii. The customer can view orders details (SELECT from the consolidated view we created using multiple joins)

   c. **UPDATE**
      i. The customer can update address later (UPDATE address table based on user id)

   d. **DELETE**
      i. The customer can delete profile (DELETE from users table and cascades deletes onto other related tables)
      ii. The customer can delete orders (DELETE from orders and cascades deletes onto other related tables)

3. The application uses MVC design which modularised code as well as database programming objects
4. On the database end, we provide 9 tables in 3rd Normal Form while carefully considering the ON delete and ON update constraints for foreign keys
5. Our application provides user intuitive feedback if an error has been thrown
6. We have a consolidated view which performs multiple joins are gives us a master look at all the related data

Technical Specifications

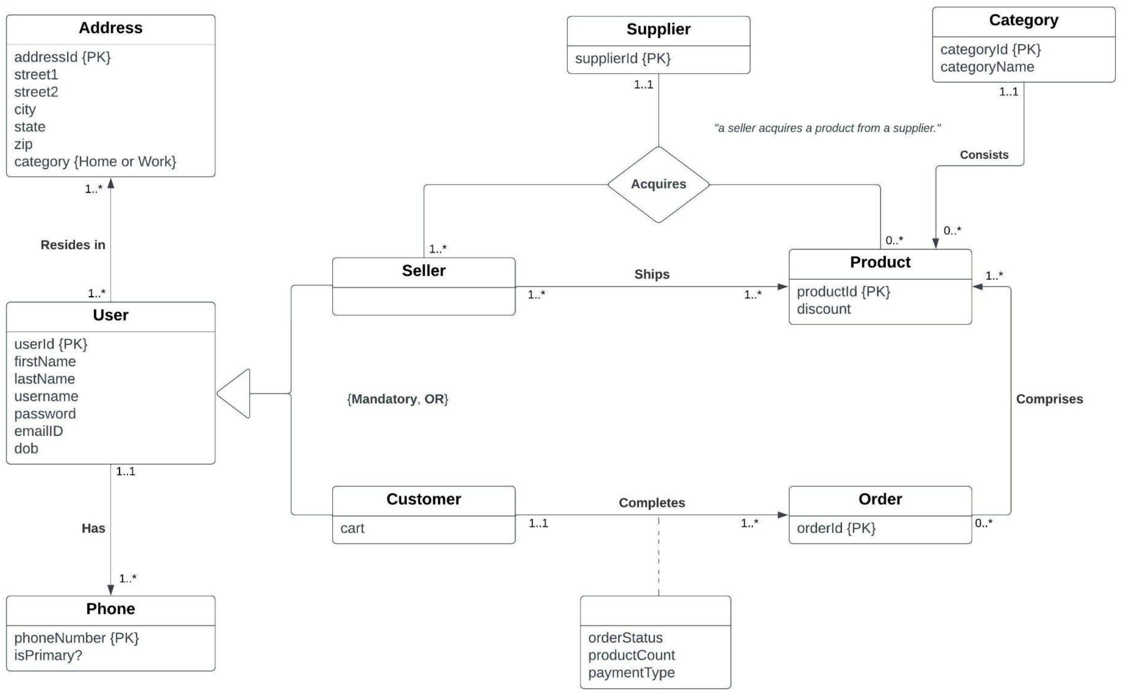
Software: MySQL Workbench

Output Screen: GUI

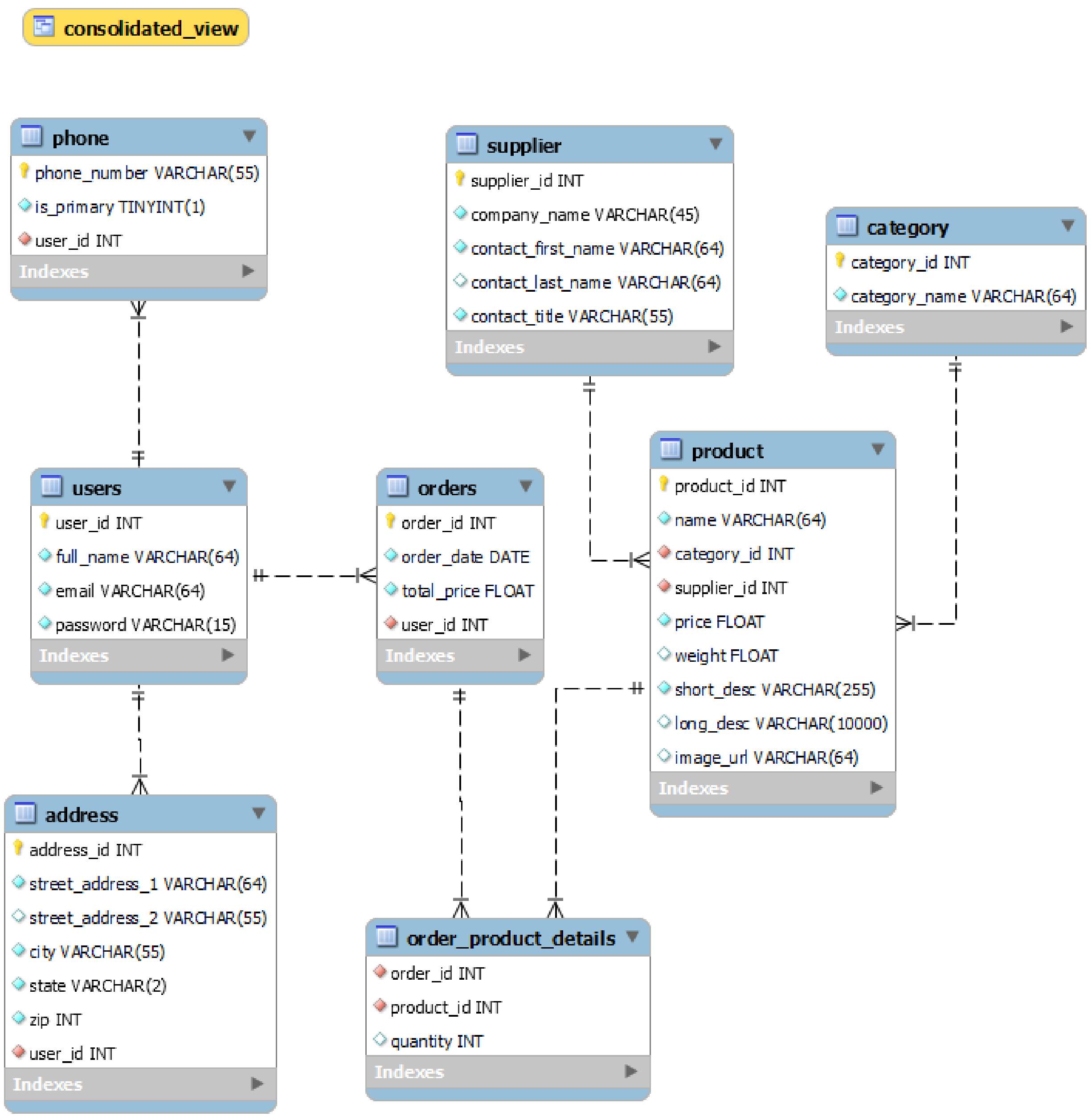
Language/Tools:

- Python for backend and application host language for connecting the user to the database
- HTML for frontend i.e dynamic web pages for intuitive user interaction
- SQL to manage the database while MySQL workbench acts as the database manager
- Flask as web framework to connect to database and provide user related functionality
- Jinja as a web engine template service for rendering pages to server

Conceptual Design

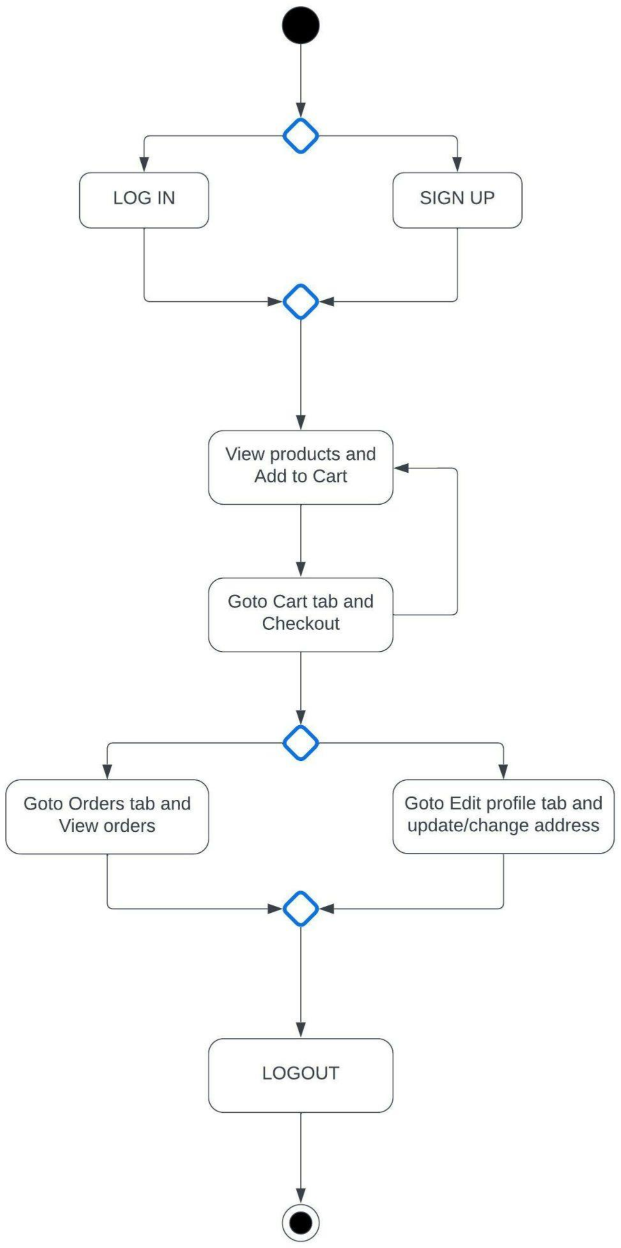


Logical Design



User Flow Activity Chart

A customer can add themselves to the database by signing up or logging in and searching through the product catalog offered by respective sellers. A customer then selects the quantity of the products and creates an order upon hitting the confirm button. This order then is mapped to our database and the user has the ability to view past orders.


LOG IN
SIGN UP
View products and Add to Cart
Goto Cart tab and Checkout
Goto Orders tab and View orders
Goto Edit profile tab and update/change address
LOGOUT

Lessons Learnt

I. Technical expertise gained

A. The project helped us understand the relation between user input and data manipulation on the backend while always adhering to the provided constraints and the relation between the data

B. We gained knowledge on how much access to provide the user based on business needs and the safest way for them to update data without corrupting existing data using stored procedures

C. On the frontend we were able to use python and flask to create a mvc design for the UI. This helped us modularise our code and provide an intuitive user interaction

D. Connecting the application to the database while keeping track of user information was a challenge which we overcame using session ids

II. Insights

A. User privacy is an important concept as it is very easy for different types of users to be privy to sensitive customer information if the valid constraints are not present

B. Views are useful for business and data analysts to make calculated decisions and should be used to make optimal usage of time rather than creating complex join queries from the scratch

C. This project helped us understand the amount of collaboration and visibility needed between the database and the application

III. Alternative design

A. Our alternate design included having a trigger for automatically inserting into the order_product_details table

B. We also thought about keeping track of the supplier address and login for further analytics. The supplier would have had to login as well with different functionality like including product into the catalog

IV. Unused Code

A. We have not provided end to end functionality for the supplier features like inserting a product into the catalog.

B. We have a consolidated view created in the database which can further be used by business and data experts and stored procedures which were not directly implemented

Future work

I. Planned uses

A. The reason we chose this domain was due to its prevalence and widespread use cases, this project could be expanded on a low level scale for suppliers based out of their home

B. Implement a fully functioning delivery system based on order date and address along with inventory management

II. Functionality / Areas of Improvement

A. More privacy based on user type. We will implement more procedures in the database to allow our application/user less access to the database

B. Inventory management can be achieved by keeping real time track of the quantity