CS 5200 - Database Management Systems

Group Name: ChavanAKumarRAhireS
Group Members: Akshay Chavan, Rishabh Kumar, Sameer Ahire

RetailHub

1. Introduction

In response to the pressing challenges faced by small-scale vendors and store managers, Our project aims to contribute significantly by developing a versatile software application. This application will serve as an indispensable tool for efficient business management, enabling these enterprises to seamlessly adopt digital solutions. Our focus is on creating a user-friendly and comprehensive tool that streamlines various business operations, including inventory control, employee tracking, vendor interactions, and utility management.

Link to the hosted website : https://retailhub.netlify.app/
Note: The server might be in ideal state and may take time to load. Please wait patiently. Refer the credentials in the README section of the report.

2. README

Technical Specifications for DBMS Project Setup:

Prerequisite:
Ensure that Node.js and React.js are installed. If not, download them from the following link: https://nodejs.org/en/download

Step 1: Setup the Database:
Use the SQL dump file provided in the zip to import the RetailHub database.
Verify the successful import of the dump.
If the dump is not working, add the following line after the CREATE DATABASE statement:
SET SQL_REQUIRE_PRIMARY_KEY = 0;

Step 2: Setup the Application Front-end:
i. Locate the application folder in the zip file, containing "retailhub" and "server" subfolders for frontend and backend, respectively.
ii. Open the terminal and navigate to the "retailhub" folder (frontend).
iii. Use the command: ***cd folderlocation*** to navigate to the specified folder location.
iv. Install front-end dependencies using: ***npm install***
v. Launch the application frontend with: ***npm start***
vi. Manually open the application by navigating to: http://localhost:3000/

Step 3: Setup the Application Backend:
i. Locate the application folder in the zip file, containing "retailhub" and "server" subfolders for frontend and backend, respectively.
ii. Open the terminal and navigate to the "server" folder (backend).
iii. Install server dependencies using: ***npm install***
iv. Run the application server with: ***npm start***
v. Check if the server is running on port 4000.

Step 4: Pointing the Application Code to the Backend Server:

Navigate to: ***application/server/services/databaseConnectionService.js***
Open the file ***databaseConnectionService.js***.
In the **config** object on line number 7, provide the appropriate values for hostname, username, port, password, and database according to the imported database.
Save the file and check if the server is running (refer to **step 3.iv**).
Attempt to log in to the application using the provided credentials.

NOTE: If the application stops working, check if the server is running. If not, restart the server (refer to step 3.iv).

Credentials to Log In:
Login as Owner: Username- **sameer_ahire**, Password- **123**
Login as Manager: Username- **akshaychavan7**, Password- **12345**
Login as Employee: Username- **bob_johnson**, Password- **789**

3. Technical Details

Database Management:
MySQL Workbench: Designed for creating and managing the MySQL database, procedures and triggers.

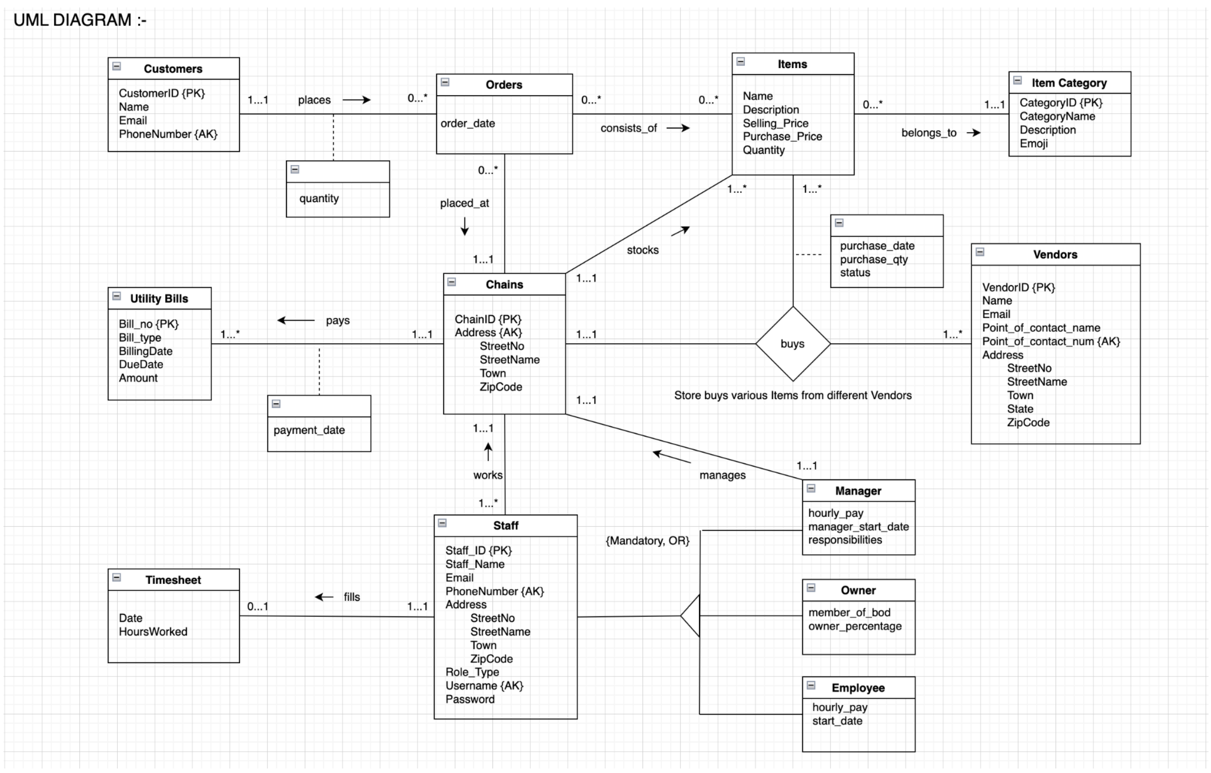
Development Environment:
Visual Studio Code: An integrated development environment for coding and debugging.

Front-end Development:
React: A dynamic JavaScript library for building interactive user interfaces.
Material Design: Offers a structured design system for visual and interactive designs.
Draw.io: A versatile online tool for creating UML and other types of diagrams.

Back-end Development:
Node.js: A scalable JavaScript runtime for building server-side applications.
Express.js: A robust web application framework for Node.js, enabling the creation of APIs and web applications.

Version Control and Collaboration:
GitHub: Facilitating version control and seamless collaboration for software development projects.

4. Database Description

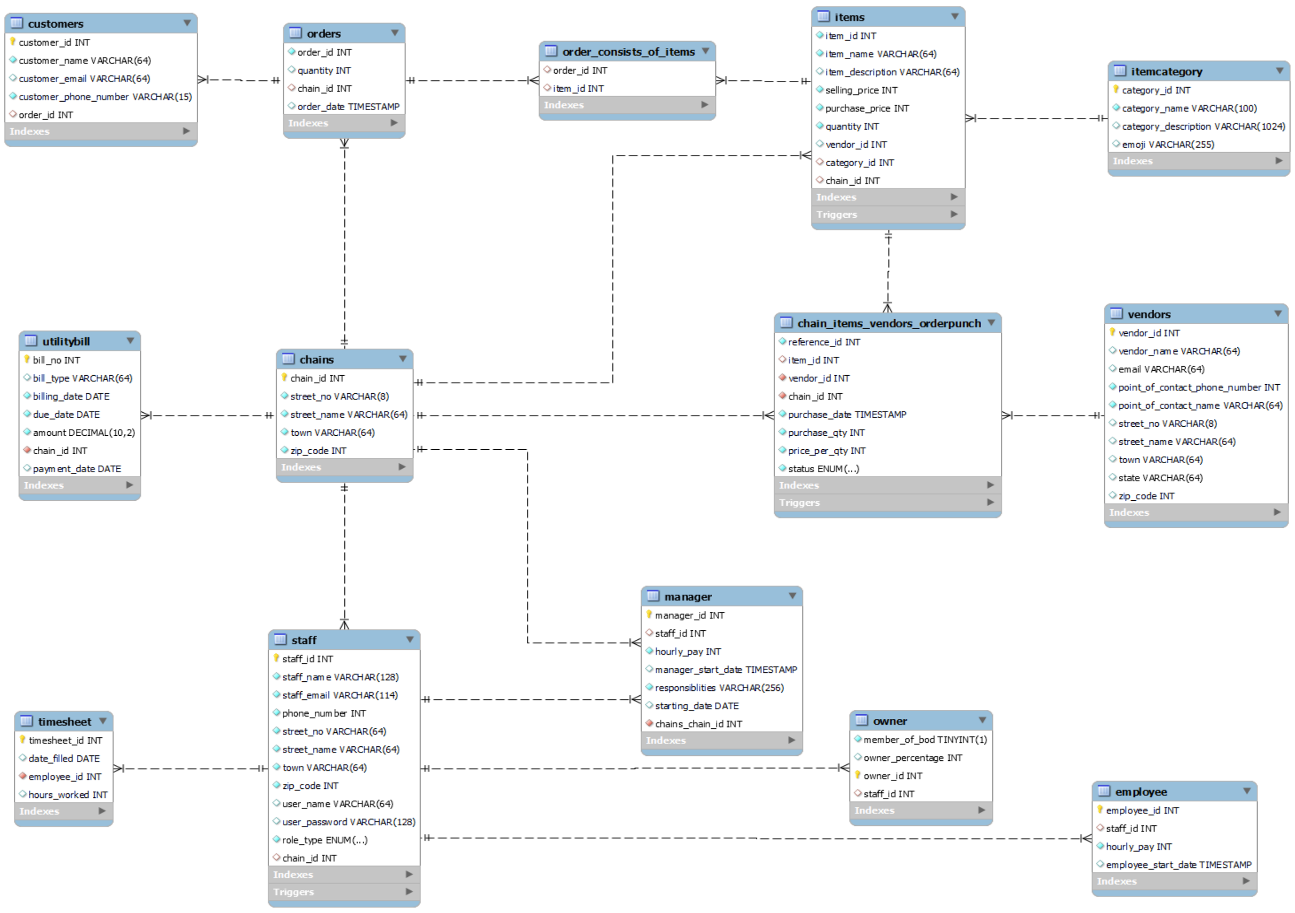


Brief Description:

The project's backbone is a comprehensive database for holistic small business management. It encompasses:

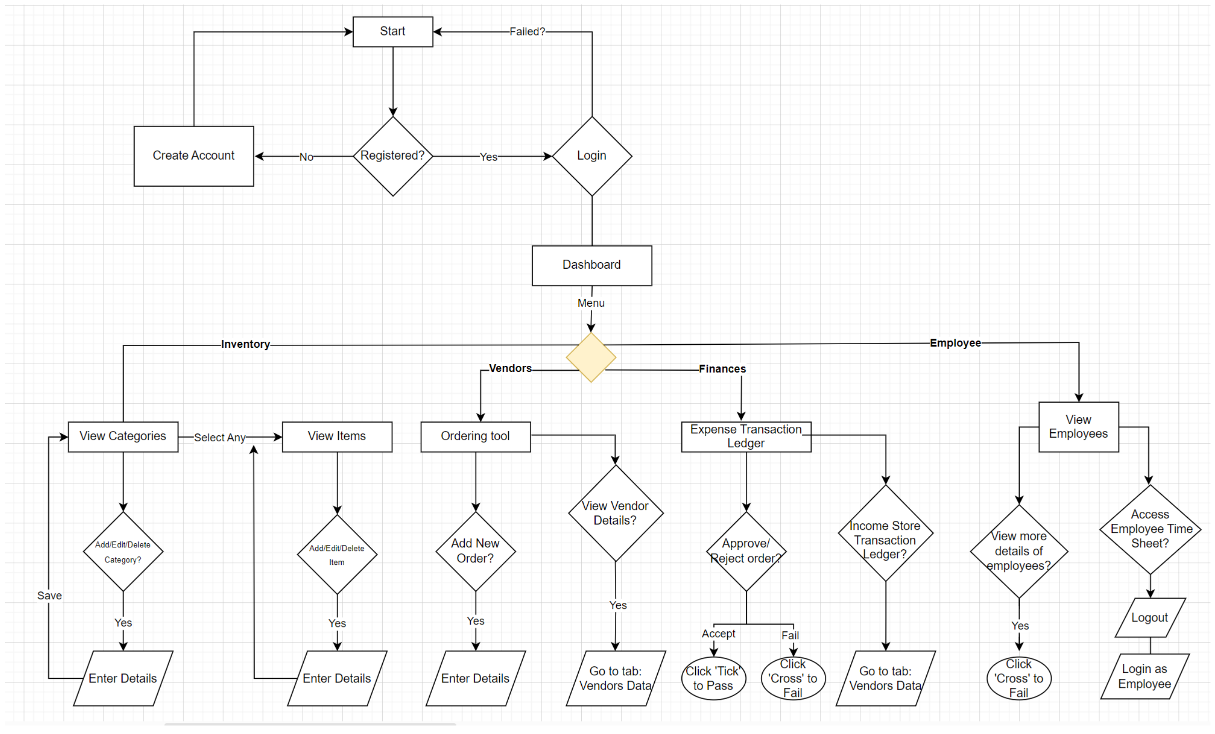
- **Employee Information:** Capturing essential employee details, roles, and responsibilities.
- **Inventory Management:** Efficiently cataloging and monitoring the entire inventory with real-time updates on quantities.
- **Utility Bills Tracking:** Providing a systematic approach to monitor and manage various utility bills and their payment status.
- **Customer Records:** Recording customer details and their interaction history with the business.
- **Store Operations Management:** Secured login credentials tailored for store-specific database operations.
- **Vendor Relations:** Efficiently recording and managing orders placed by the store with various vendors, enhancing vendor interactions.
- **Timesheet and Employee Records:** Accurately tracking employee work hours, responsibilities, and activities within the business structure.

5. Reverse Engineered DB



6. Flow of the Project.

Activity Diagram:



Project Overview

In the realm of store management, our application caters to three distinct user roles: owner, manager, and employee. Each user type experiences a tailored interface designed to streamline their specific responsibilities within the system.

User Authentication and Navigation

While logging in, owners are prompted to select the chain of stores they wish to manage, while managers enjoy a seamless login experience. Notably, the application focuses on store management tasks rather than catering to customer interactions. Employees, upon login, encounter a simplified interface allowing them to input their daily working hours and access a comprehensive chart detailing their working hours for the past month.

Managerial Dashboard

For managers, the dashboard serves as the nerve center of our application. A clean app bar welcomes users with a friendly greeting. The left-side menu offers five key sections: Dashboard, Inventory, Vendors, Finances, and Employees.

Dashboard Insights

The dashboard boasts a rich array of graphs and metrics, presenting vital insights into the store's performance. Six prominent cards dominate the top section, displaying key figures such as total categories, customers, stores, pending orders, employees, and vendors. These values are dynamically fetched from our MySQL database via a robust API. Complementing these cards are four insightful charts, all powered by SQL procedures. Noteworthy examples include a bar chart showcasing the 'Top 3 vendors and their last 5-month orders' and a pie chart detailing 'Categories and their expenses.'

Inventory Management

Navigating to the Inventory section reveals a page adorned with cards representing the various categories within the store, ranging from books to furniture and toys. A user-friendly category add button facilitates the seamless addition of new categories, guiding users through a dialog box for name and description input. Each category tile provides options to edit or delete, with meticulous safeguards in place to prevent accidental deletion of categories housing items. Importantly, all CRUD operations within this section are reflected in real-time within our MySQL database.

Clicking on a category tile opens a dedicated page showcasing detailed information. The top section displays the category name and description, while a comprehensive table beneath provides insights into individual items. Details include the item's vendor, description, price, margin, quantity, and availability. User actions, such as editing, deleting, or adding items, trigger dynamic calculations via SQL procedures, ensuring accurate cost calculations based on manager-defined margins and vendor-supplied cost prices.

Vendor Management

The Vendors section introduces a visually appealing interface with card tiles presenting key metrics for categories, users, pending orders, and vendors. Two tabs, "Place Order with Vendor" and "Vendor Data," offer distinct functionalities. The former empowers managers to create orders with precision, specifying vendor details, desired items, prices, and quantities. The latter tab provides a straightforward table presenting vendor details, seamlessly fetched from the database via an API.

Financial Insights

In the Finances section, users are presented with two insightful tabs: "Expense Transaction Ledger" and "Income Transaction Ledger." The former serves as a comprehensive repository for orders created using the "Place Order with Vendor" tool. Each entry features essential details such as vendor name, item details, quantity, total cost, date and time, payment status, and actionable options for approval or decline. These actions trigger updates in the database, influencing inventory levels accordingly. The "Income Transaction Ledger" tab, on the other hand, provides a read-only view of store transactions, including customer details, quantities, payment modes, and total prices.

Employee Management

The Employees section adopts an intuitive tile format, with each tile representing an employee. Expanding these tiles reveals in-depth details about individual employees, offering a holistic view of the workforce. The application culminates with a secure logout button, ensuring the termination of user sessions with the utmost security and compliance.

7. Lessons Learned.

Technical Expertise Gained:

- We have implemented an efficient schema design to support the seamless functioning of the application.
- Took a comprehensive role in developing the application, contributing to frontend, backend, and overall database design.
- Ensured a cohesive integration of the user interface, backend logic, and the underlying database.
- Developed sophisticated stored procedures and triggers, enhancing the database's functionality.
- Implemented complex logic within the database to support diverse application interactions.

- Adhered to industry best practices, emphasizing the use of stored procedures for database interactions.

8. Future Scope.

While our current application provides a robust and comprehensive solution for store management, there are several avenues for future development and enhancement.

- Explore machine learning algorithms to predict trends and suggest optimization strategies based on historical data.
- Implement a more flexible user role management system, allowing owners to define custom roles with specific permissions tailored to their unique business structures.
- Extend the application's accessibility by developing mobile versions for both iOS and Android platforms.
- Introduce features that facilitate customer interactions, such as loyalty programs, customer feedback, and personalized recommendations.
- Optimize database queries and indexes to enhance overall performance, especially for complex queries generated by advanced analytics features.
- Implement encryption techniques at the database level to ensure the confidentiality and security of sensitive data. Regularly update security protocols to stay ahead of potential threats

By focusing on these aspects from a database perspective, the application can evolve into a more resilient, scalable, and secure system, capable of handling increasing data volumes and meeting the demands of emerging technologies and businesses.

Checkpoint that we have covered:

- ☑ A 3-member group needs at least 10 tables
- ☑ All necessary primary and foreign keys are represented in tables.
- ☑ FKs have ON UPDATE and ON DELETE
- ☑ Field constraints such as NOT NULL, DEFAULT, AUTO_INCREMENT, TABLE constraints with checks.
- ☑ database programming objects created for the database.
- ☑ Many examples of CRUD operations DB and Client.
- ☑ Application can elegantly deal with errors from the user and the database server.
- ☑ UML changes after the professor review .
- ☑ 3 to 5 interesting queries that can be used for analysis or visualization of the data (at the detailed level or summary data) (1-5 points)
- ☑ additional front end functionality such as website or a GUI (1-5 points)
- ☑ Overly complicated translations from user operations to database operations (1-5 points)
- ☑ Complicated schema – user data pull requires multi-joins, or many tables (> 10) due to the complexity of the data domain (1-5 points)
- ☑ Visualization of the data. (1-5 points)
- ☐ Difficult process for data extraction such as web scraping or data cleansing (1-5 points)
- ☑ Application supports multiple user roles (1-5 points)