CRAVE ON CAMPUS (EXPANDED) PROJECT SCOPE

DSBA-6160 | 11/15/2021

OVERVIEW

GitHub: https://github.com/acamlibe/Crave-on-Campus-Expanded

GitHub Pages: https://acamlibe.github.io/Crave-on-Campus-Expanded/

Dictionary: https://acamlibe.github.io/Crave-on-Campus-Expanded/Dictionary/#/

1. Team Members

- Saurabh Vijay
- Shina Dhingra
- Sammie Srabani
- Ali Camlibel
- Lakshmi Medineni

2. Project Background and Description

This project was initially created by students in the past, studying at UNC Charlotte, to create a database system that was designed to handle deliveries on campus during the COVID-19 pandemic. Due to the increased risk of spreading the COVID-19 virus, the students at UNC Charlotte wanted deliveries to be handled by faculty staff and other students, which the university keeps a record of vaccinations and positive tests. This way, only staff and students who were known to NOT be COVID-19 positive would be the ones to deliver, drastically lowering the risk of catching the virus.

Now, currently in 2021, my team and I have continued this project to add additional functionality to the database for data access, data entry, and reporting on ratings (from 1 - 5) for the different drivers AND restaurants that students may order from. This way, students could be informed of the driver associated with their order and may cancel their order if he or she has a low rating. Students may also view ratings for restaurants, so they may base their decision on ordering depending on the restaurants rating.

We have added tables, views, procedures, and functions to this database to extend the functionality.

3. Project Scope

The main purpose of this college/university restaurant management system is to show the production of reports, inventories, staff/student/driver/location/faculty/vehicle records/ratings, and a list of orders. This system is useful for users who enjoy ordering food through delivery in order to reduce the spread of COVID-19. The user then has information on other students' interactions with the drivers and restaurants that are a part of the system.

The ERD system consists of the 1st Level Dataflow Diagram and Second Level Dataflow Diagram. The 1st Level Dataflow Diagram indicates that the system is divided into sub-systems such as vehicle details, drivers' details, people involved-Student/Staff, Location-faculties for students and staff, orders, and restaurant details/Ratings.

Each unit deals with one or more data flow. All of them combined to give the functionality of the college/university restaurant management system.

In the 1st level Dataflow Diagram, the main entities and outputs include processing the user records and generating a report of user categories (Student, Staff, Drivers - ratings), processing all orders category records and producing a report of all orders, and processing locations of all orders made and reports of those locations (Different faculties). It also supports processing details of vehicles used in the delivery of orders in different locations (faculties), processing delivery details and generating a report on all deliveries made, processing orders made by the student and producing reports, and processing orders made by staff and producing reports.

The 2nd level Data Flow Diagram gives deeper information of the college/university restaurant management system. It contains more details of the hotel staff, customers, means of payments, exact locations, order IDs, Driver IDs, Driver's employment details, Rating about the driver, restaurant and services and Details of student graduation year. The 2nd level Dataflow Diagram functionalities where admin can log in and manage all the functionalities of the system. It also allows the admin to add, remove, and update student, staff, or driver details and provide the ratings to food delivery ratings. The other functionality is that it allows the admin to generate accurate reports of all restaurant sections-faculties, orders, staff, drivers, and location, delivery, or payment details.

4. Use Cases

US-001	**Secure Food Delivery on Campus**	
Dependencies	OBJ 001: To manage secured food delivery within a campus	
	OBJ 002: Access to various local restaurants	
	OBJ 003: Restricted Drivers and accessible restaurants system	
	OBJ 004: Delivery food to correct person and location	
	OBJ 004: Timely and safe deliveries for students.	
Description	The system will capture what the user orders, as well as the rating he or she gives to the driver and restaurant. The user may view his or her assigned driver's average ratings to assess whether they want to change the order.	
Precondition	The person who delivers the food and the driver who is going to deliver the food have been inserted into the system and identified by their ID. The order must be inserted into the order table followed with an optional rating for that specific order's restaurant and driver.	
Ordinary Sequence	Step	Action
	1	User finds a specific restaurant of his or her choice.
	2	User may then sign in or register (if no account found) for an account to save his or her payment info and delivery address.
	3	The system provides a list of food and drinks, as well as the deals associated with the food, found at this restaurant
	4	User picks what he or she wants to order.

	5	The system displays the total price and delivery charge.
	6	User provides payment information.
	7	System confirms food, delivery times, and payment information
	8	The system sends the information to the restaurant and the drivers who are nearby.
	9	A driver picks up the order.
	10	User views driver's average ratings and decides whether to switch order/delivery driver.
	11	The restaurant validates the order and prepares the food. Once the food is ready, the system will send the completion confirmation to the driver and the user.
	12	The driver will then pick up the food and deliver to the given address and update the order in the system.
	13	The system sets the order as complete.
	14	The user is then prompted to leave a rating for the driver and the restaurant.
Postcondition	The system can insert ratings for that specific driver and restaurant into the system for future users to access.	
Exceptions	If the restaurant is closed, or not accepting deliveries, the user will be refunded, and the driver will get a cancellation notification.	
Comments	Every transaction should be handled in a timely manner, as to avoid time wasted for the driver and user.	

5. Business Rules

The new system must include the following:

- Ability to easily retrieve ratings on both drivers and restaurants that are a part of the system.
- Ability to see information about the individual drivers and restaurants in our system.
- Ability to see statistical analysis on ratings, such as average, min, max, count, and sum.
- Ability to see aggregated/grouped reports on restaurants and drivers, so that the viewer can easily see the best restaurants, most expensive restaurants, and metrics on ratings for both restaurants and drivers.
- Ability to query all related data within a reasonable time. Query times should not exceed 5 seconds.

6. Deliverables

- Use Case, Business Rules, Enhanced ERD, Schema (Due 11/18/2021)
- New Rating Table, Test Data (Due 11/25/2021)
- Queries (Due 12/2/2021)
- Final Project (Due 12/8/2021)

7. Implementation Plan

Implementing this plan will require the following:

1. Spin up a MySQL RDS instance on Amazon AWS.

2. Initialize private Git repo on GitHub for version control and collaboration.
3. Import the current design, data, and database design of the existing database schema.
4. Adjust the setup to work with Amazon AWS instance.
5. Normalize any tables, if necessary. All tables should be in 3rd normal form.
6. Break apart any many-to-many relationships that may exist.
7. Generate an ERD diagram of the current, or normalized, implementation.
8. Add in rating tables for both drivers and restaurants.
    - Should be in 3rd normal form with the proper foreign key constraints.
    - Should not have any many-to-many relationships.
    - Should contain auto-incrementing surrogate keys for the tables' primary keys. No composite keys should be used.
    - Make sure common metadata within the design is refactored out and joined together.
9. Generate an ERD diagram of the expanded design.
10. Insert in fake, but reasonable, data into the new rating tables.
    - Can be done manually by generating and importing data from generatedata.com
    - Can be done in Python to automatically generate and run the proper INSERT statements.
11. Create functions to the schema to allow quick access to the data.
12. Create stored procedures to the schema to allow quick access to the retrieval and insertion of common and statistical data.
13. Create views that will be used for reporting on various metrics such as quality, costs, and average ratings.
14. Index the views.
15. Update the setup script with all the new creations to allow destroying and recreating the database schema whenever needed.
16. Record a video demo of the system and the data associated with it.