SW Engineering CSC648/848 Spring 2021

Project title: "Campus Cantina"
Milestone 2
Team 04

Member Name	**Member Role**
Rajdeep Singh	Team Lead (*rsingh12@mail.sfsu.edu*)
Rinay Kumar	Backend Lead
Bhavani Goruganthu	Frontend Lead / Document Editor
Frederick White	Github Master
German Perez	Frontend Team
Henzon Zambrano	Backend Team

Revisions History:

Version Number	**Date Created/Revised**	**Date Submitted**	**Instructor Comments**
Version 1	March 14 2021	March 19 2021	

1) Executive Summary

“Life is what happens when you are busy making other plans”

- John Lennon

As digital technology continues to progress, people are migrating towards online platforms for their various needs such as shopping, entertainment as well as ordering a meal for the day. College campuses represent some of the most concentrated markets for certain types of delivery food and many restaurants find that delivering food items to local college campuses can exponentially increase their order volume when college is in session. In these current times, college students become more occupied from school and work, but they all still need to consume food and that takes up too much time as customers have to drive to restaurants, order and wait for the food. The solution created is “Campus Cantina”. The motivation of Campus Cantina is to be able to provide a convenient food delivery service to SFSU college students, staff and faculty within the campus.

Through an online food ordering system, SFSU customers (students, staff and faculty) can easily access a restaurant's menu in a hassle-free manner. As lives get busier, more SFSU customers want to order the taste of the restaurant at home and eat within their comfortable confines. New ordering and food delivery options serve students and faculty who increasingly want their meals when and where it’s most convenient for them.

“Campus Cantina” is a unique and user-friendly online food ordering system for exclusive use by SFSU students, staff and faculty. This web-based service is easily accessible from your handy devices i.e. Laptop, Tablet & Mobile. All you need to do is to register and login using your sfsu email id and start hunting for your favorite food from the nearby restaurants.

By creating a web-based service exclusive for SFSU students and faculties, we aim to provide a variety of food cuisines that our users will be able to choose from. Our service will be able to help busy college students that have no time to physically go to their desired restaurants that they want to order in. Our users will be able to see different restaurants on our website as well as weekly special deals. Our service will also open opportunities for restaurant owners/vendors to be able to advertise their menu and services. Our system shall be competitive in terms of providing services from on-campus restaurants.

"Campus Cantina" shall have many outstanding features, out of which the ability to order from multiple nearby restaurants to get your meal delivered to any specific location at SFSU campus is the dominant one. Additionally, Campus Cantina shall provide a user-friendly UI experience for SFSU students & faculty to allow ease of ordering food online using a detailed map of the SFSU campus and surrounding areas. Our team plans to have a search feature where we shall be simplifying browsing of food menus using cuisine filtering.

Our team is not just a group of individuals but is one strong effort put by six innovative minds. We have different strengths in different areas, but we continuously help and motivate each other as a team. Our goal is not only to finish the project we have but also we hope to learn new things and help each other grow.

2) List of Main Data Items and Entities *(subject to change)*

1. **Registered User**

   1.1 **Definition**: Registered Users of the app shall have a login with their SFSU email and password stored on the server. We shall also store their full name, address, phone number, and store what type of account and privileges they have.

   1.2 **Sub-items**:

   1.2.1 **ID**: A unique user ID for the database.
   1.2.2 **Full name:** First and last name of the user.
   1.2.3 **Address:** Delivery address, office, or dorm depending on user type.
   1.2.4 **Type:** Either student/faculty/staff
   1.2.5 **Phone:** Contact phone number.
   1.2.5 **Email:** Email used for registration and login.
   1.2.6 **Password:** Password used for login.
   1.2.7 **Status:** Active/Inactive status of the user (can be changed by Admin).
   1.2.8 **Recent Addresses:** Maximum 3 addresses per user.

2. **Restaurants**

   2.1 **Restaurant**

   2.1.1 **Definition:** Restaurant information shall be stored in the database, including name, location, cuisine type, and photos to be used to populate restaurant offerings on the app. Provided by the restaurant owner and approved by the admin.

   2.1.2 **Sub-items:**

   2.1.2.1 **ID:** A unique ID for the restaurant for the database.
   2.1.2.2 **Restaurant Name:** The name of the restaurant.

2.1.2.3 **Address:** Street address of the restaurant.
2.1.2.4 **Phone:** Contact phone number.
2.1.2.5 **Cuisine:** Cuisine type, such as Italian, Indian, Mexican, etc.
2.1.2.6 **Tags:** Tags for the restaurant, such as type and menu item names.
2.1.2.7 **Logo Pic**: Logo pic for the restaurant to be displayed in searches.
2.1.2.8 **Banner Pic:** A larger pic to be displayed on the restaurant page.
2.1.2.9 **Price level:** Cost of food, from one $ to four $$$$.
2.1.2.10 **Lat:** The latitude coordinate of the restaurant for GoogleMaps.
2.1.2.11 **Lng:** The longitude coordinate of the restaurant for GoogleMaps.
2.1.2.12 **Approved:** The approval status of the restaurant. Default is false.

2.2 **Restaurant Owner Profile**

2.2.1 **Definition:** We shall store restaurant owner information for restaurant owners to register and login, including email, password, and contact info.
2.2.2 **Sub-items**:
2.2.2.1 **ID:** A unique ID for the restaurant owner for the database
2.2.2.2 **Name:** Name of the restaurant owner
2.2.2.3 **Phone:** Contact phone for the restaurant owner
2.2.2.4 **Email:** Restaurant owner email used for registration and login.
2.2.2.5 **Password:** Password used for login

2.3 **Restaurant Menu Items**

2.3.1 **Definition:** We shall store each menu item available at the restaurant along with its description, price, and pictures, provided by the restaurant owner.
2.3.2 **Sub-items:**
2.3.2.1 **ID:** A unique ID for the menu item.
2.3.2.2 **Restaurant ID:** The ID for the restaurant the item belongs to.
2.3.2.3 **Name:** The name of the menu item.
2.3.2.4 **Description:** A short description of the menu item.
2.3.2.5 **Price:** The price of the item.
2.3.2.6 **Picture:** A picture of the menu item.
2.3.2.7 **Custom Options:** Customizations for the menu item.

3. **Food Cuisines**

3.1 **Definition**: A table in the database used to contain all the cuisine types available on the app, used to populate cuisine drop-down lists
3.2 **Sub-items:**
3.2.1 **ID:** A unique ID for the cuisine type for the database
3.2.3 **Cuisine:** The name of the cuisine type, such as Indian, Mexican, etc.

4. **Driver**
   4.1 **Definition:** We shall store information on the drivers on the platform for registration, login, and assigning orders.
   4.2 **Sub-items:**
      4.2.1 **ID:** A unique ID for the driver for the database.
      4.2.2 **Name:** The full name of the driver.
      4.2.3 **Phone:** A contact phone number for the driver.
      4.2.4 **Email:** The driver's email used for login.
      4.2.5 **Password:** A password used for login.
      4.2.6 **Vehicle:** The type of vehicle they have to make deliveries (car, bike, etc).

5. **Admin**
   5.1 **Definition:** We shall store login information for admins, who shall login to approve/deny restaurants.
   5.2 **Sub-items:**
      5.2.1 **ID:** A unique ID for the admin for the database.
      5.2.1 **Username:** A username used to login.
      5.2.1 **Password:** A password used to login.

6. **Orders**
   6.1 **Definition:** We shall store order information for each order a registered user makes, to be seen by the restaurant owners, drivers, and in the user order history.
   6.2 **Sub-items:**
      6.2.1 **ID:** A unique ID for each order for the database.
      6.2.2 **Restaurant ID:** The ID of the restaurant the order was placed from.
      6.2.3 **Restaurant Name:** The name of the restaurant the order was placed from.
      6.2.4 **Restaurant Address:** The address of the restaurant, for drivers or pickup.
      6.2.5 **User ID:** The ID of the user that placed the order.
      6.2.6 **User Name:** The name of the user that placed the order.
      6.2.7 **Delivery Location:** The address to where the order shall be delivered.
      6.2.8 **Order Contents:** The menu items added to the order.
      6.2.9 **Tip:** The tip added by the user for the driver.
      6.2.10 **Delivery Fee:** The fee charged to the user for the delivery.
      6.2.11 **Service Fee:** The service fee charged to the user to process the order.
      6.2.12 **Total:** The total amount for the order charged to the user.
      6.2.13 **Delivery ETA:** The estimated time of arrival for the delivery.

3) Functional Requirements - Prioritized

Priority 1:

1. Unregistered Users shall be able to search for local restaurants and their services.
2. Unregistered Users shall be able to view details of a particular restaurant and browse through their menu.
3. Unregistered Users shall have access to a map of the area surrounding campus with restaurants.
4. Unregistered Users shall register and login with SFSU credentials to order food.
5. SFSU faculty, students and staff shall register for seeking the delivery service of 'Campus Cantina' and shall login with their credentials.
6. Registered Users shall either request for delivery on the campus or pick up from the restaurant directly.
7. Restaurant Owners shall be able to register to advertise their services.
8. Restaurant Owners shall request approval before their restaurants can go live.
9. Restaurant Owners shall be able to add menu items along with photos and prices.
10. Driver shall be able to login and check the orders assigned to him/her.
11. Driver shall be able to see customer order items.
12. Driver shall be able to see the customer drop-off location.
13. The site Administrator shall approve the restaurants before being searchable by the user.
14. The site Administrator shall be able to delete or remove inappropriate users, drivers, owners.

Priority 2:

15. Unregistered Users shall be able to enter a chat to request help.
16. Local Favorites shall be displayed on the Home page.
17. Registered Users shall be able to add menu items from multiple restaurants to the cart.
18. Registered Users shall be able to manage the items added to the cart.
19. Registered Users shall be able to update their profile and delivery location details. (Delivery in SFSU campus)

20. Personal favorites shall be displayed on the Home screen.

Priority 3:

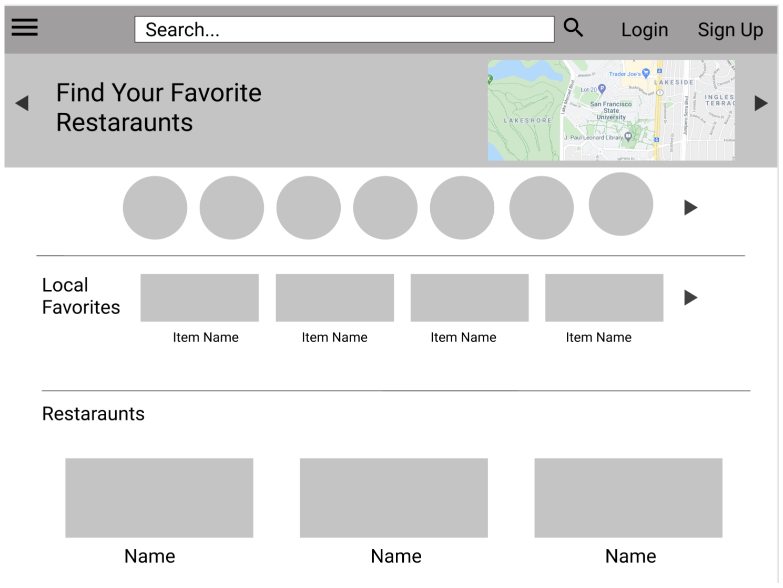
21. Registered Users shall have access to order history.
22. Restaurant Owners shall be able to view and manage the meal orders and update the status accordingly.
23. Driver shall be able to chat with customers.
24. Drivers shall be able to share their location after signing in so that an order can be assigned to him/her**.**

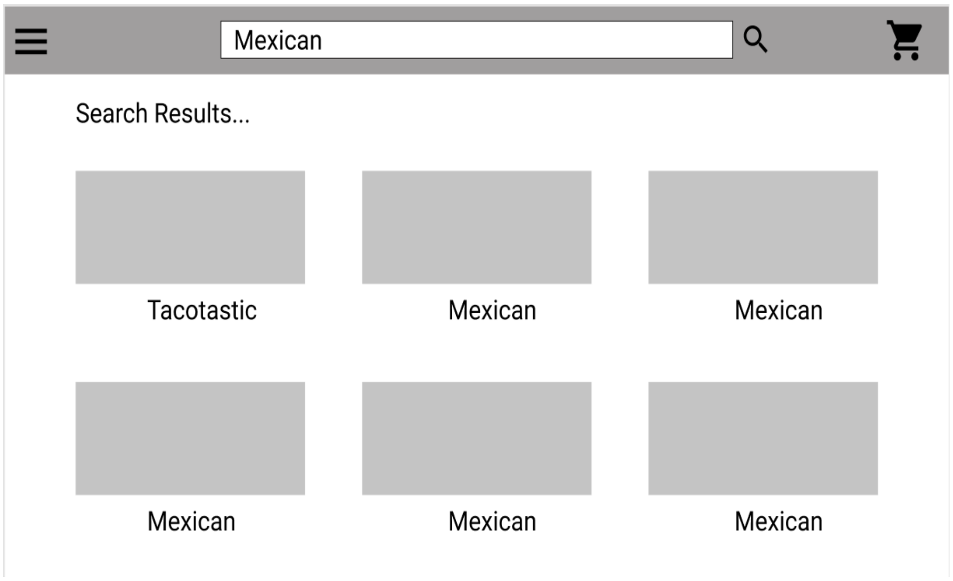
4) UI Mockups and Storyboards

Storyboard 1: Ordering Food

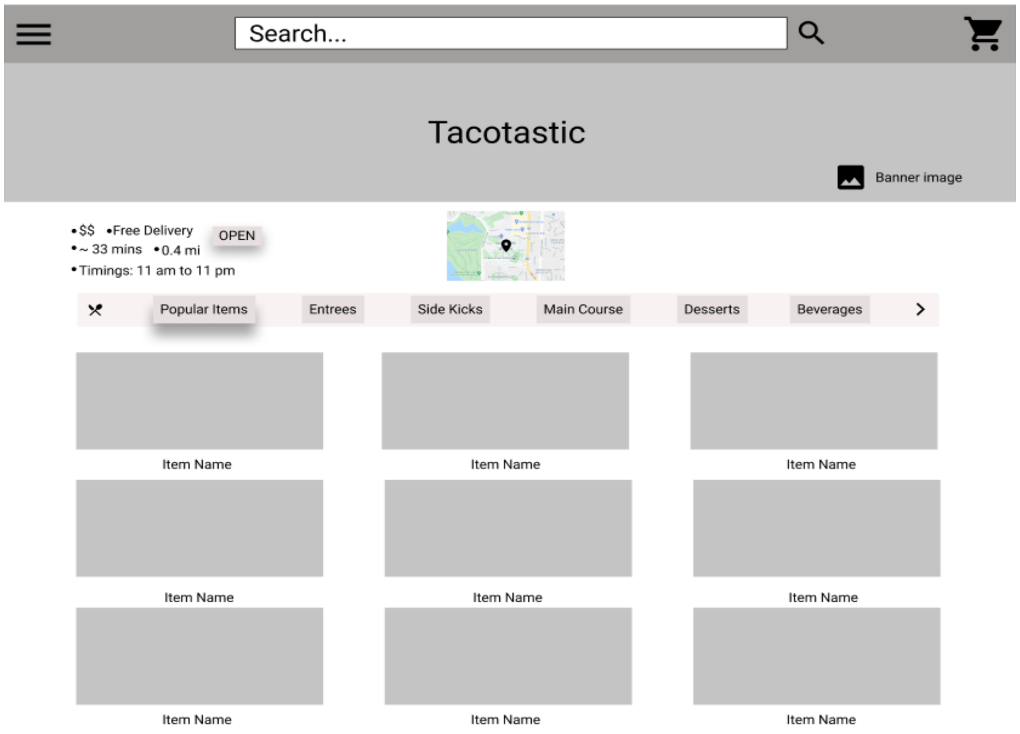
Kim gets on Campus Cantina to order some food.

Since she's craving tacos, she types *Mexican* in the search bar (located at the top of the screen). She is shown a list of Mexican restaurants in her area.

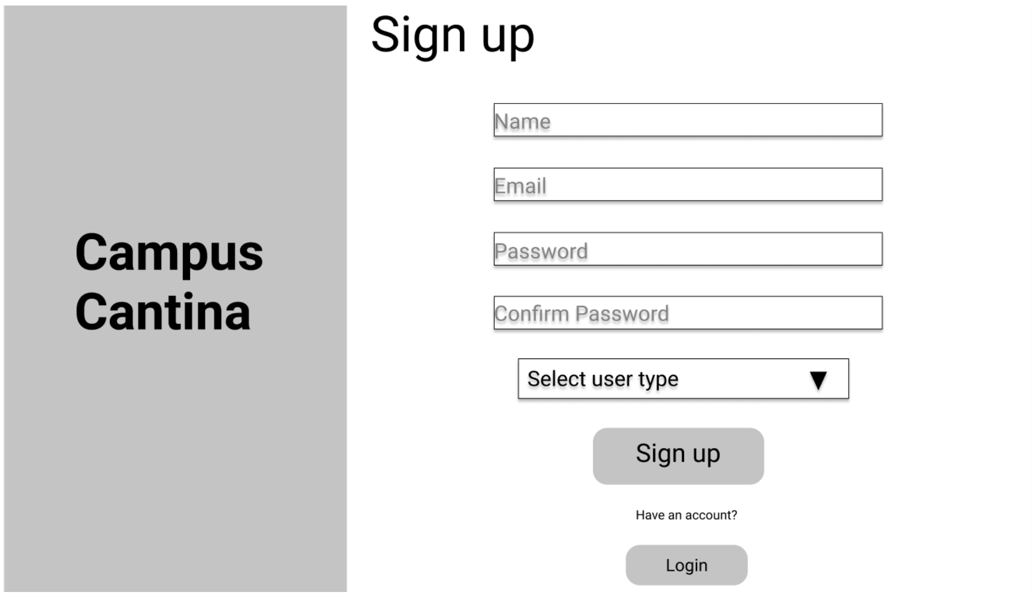


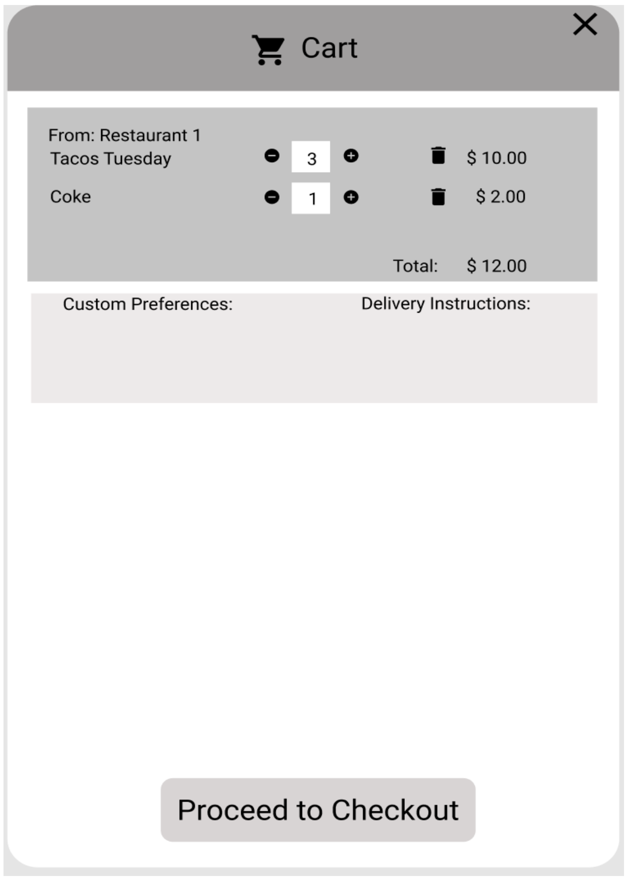


She settles on Tacotastic. As she browses through the categories, she finds a Taco Tuesday deal in the Specials section.



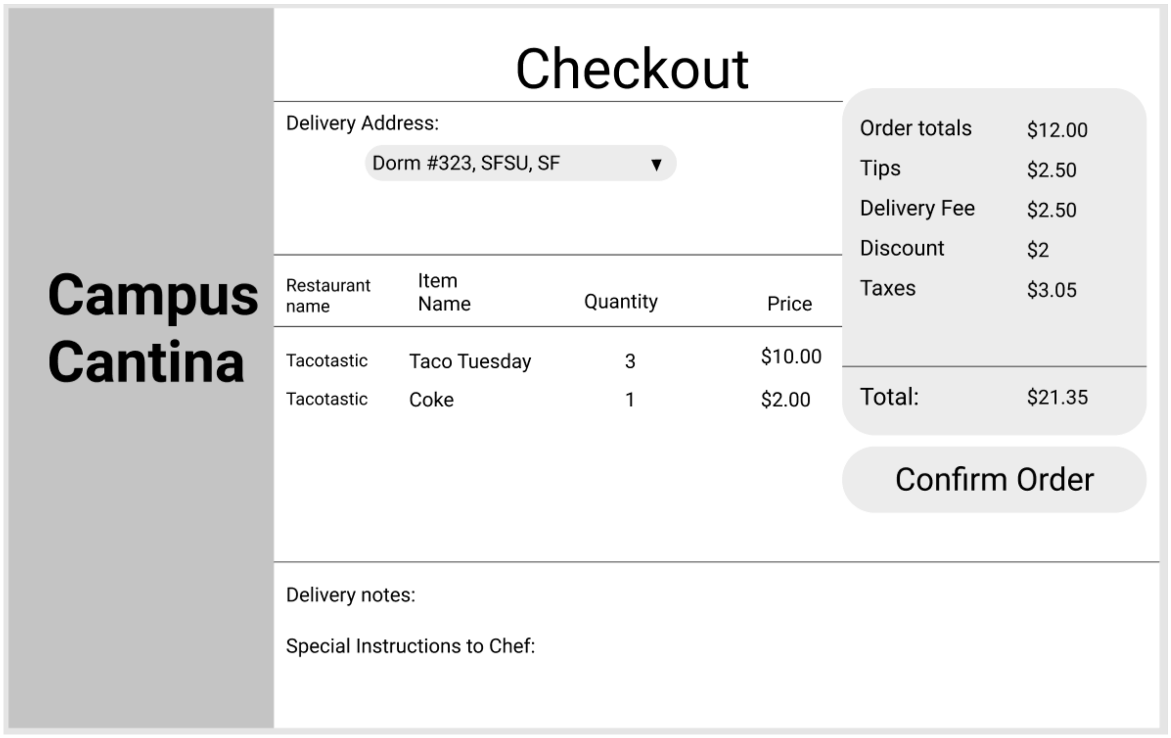
When she clicks on *Taco Tuesday Special,* she is prompted to sign up on the next screen. She enters her information, then clicks *Sign up.*





After signing up, Kim is able to add three tacos to her cart. She also adds a Coke and adjusts the tip for the driver. She clicks *Proceed to Checkout.*

On the Checkout Screen, she confirms her delivery address. She is able to see an itemized receipt of her order. She sends her order to the restaurant by clicking *Confirm Order.*



Campus Cantina

Checkout

Delivery Address:

Dorm #323, SFSU, SF ▾

Restaurant name	Item Name	Quantity	Price
Tacotastic	Taco Tuesday	3	$10.00
Tacotastic	Coke	1	$2.00

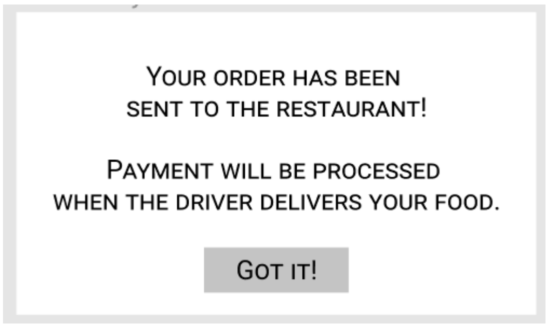
Order totals	$12.00
Tips	$2.50
Delivery Fee	$2.50
Discount	$2
Taxes	$3.05
Total:	$21.35

Confirm Order

Delivery notes:

Special Instructions to Chef:

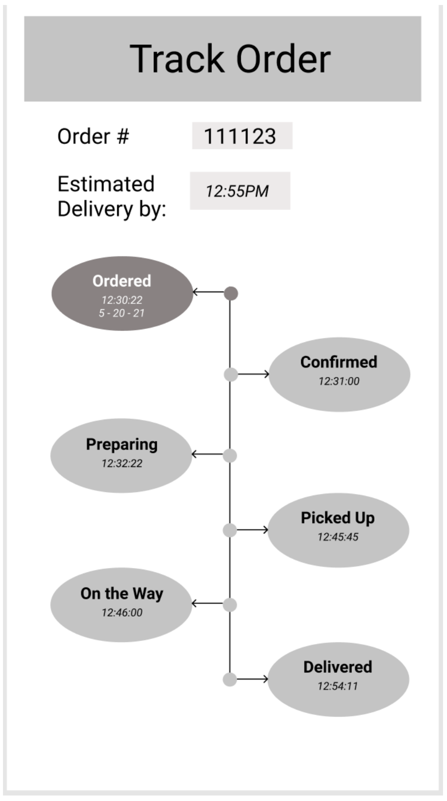
Once her order has been placed, a pop-up window will display on her screen reminding her to pay upon the driver's arrival.



YOUR ORDER HAS BEEN SENT TO THE RESTAURANT!

PAYMENT WILL BE PROCESSED WHEN THE DRIVER DELIVERS YOUR FOOD.

GOT IT!



Track Order

Order # 111123

Estimated Delivery by: 12:55PM

Ordered 12:30:22 5 - 20 - 21

Confirmed 12:31:00

Preparing 12:32:22

Picked Up 12:45:45

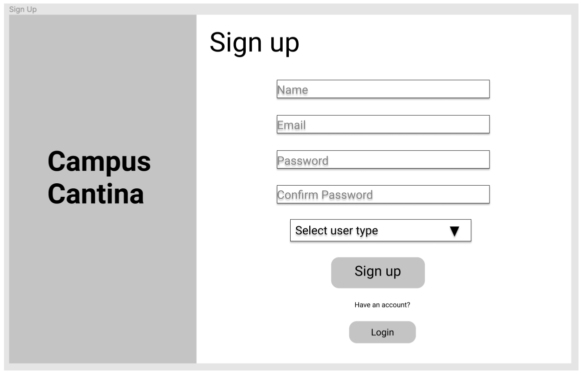
On the Way 12:46:00

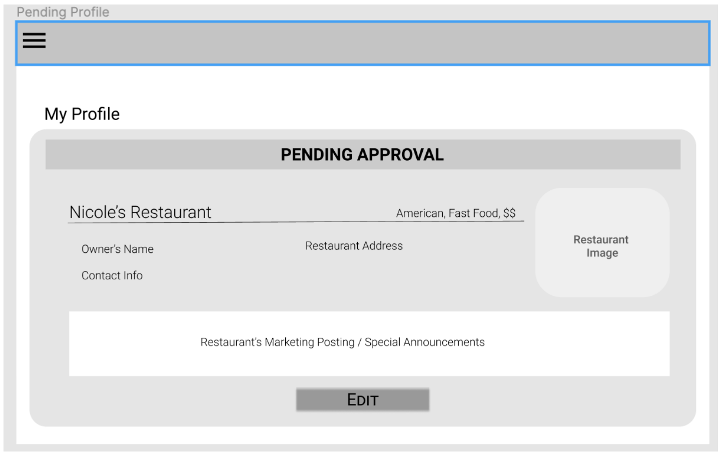
Delivered 12:54:11

After Kim clicks *Got It,* the following window will appear which easily helps her track her order. Once she's been alerted of the driver's arrival, she meets them outside her dorm to retrieve and pay for her food.

Storyboard 2: Registering Restaurant Account

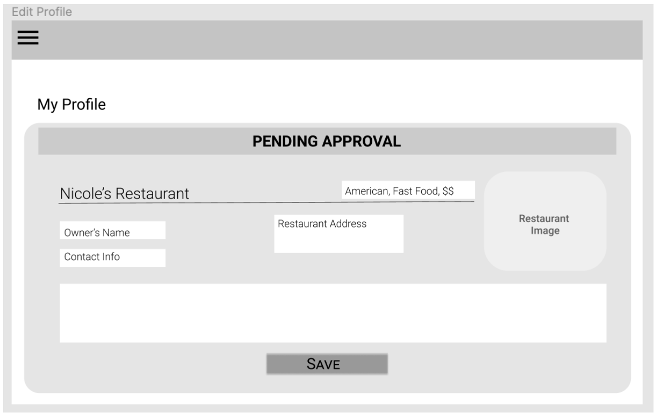
Nicole wants to register her restaurant in Campus Cantina. Nicole goes on to sign up and she creates an 'owner' type account.

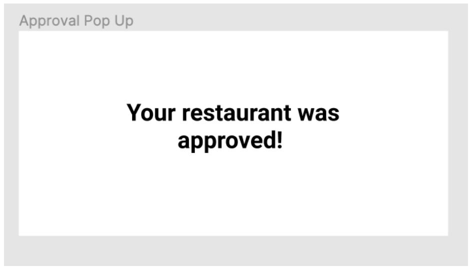




Nicole is directed to the 'pending profile' page. Nicole sees that her profile is still in pending status. Currently, Nicole can edit her restaurant's info and put an image for her restaurant page.

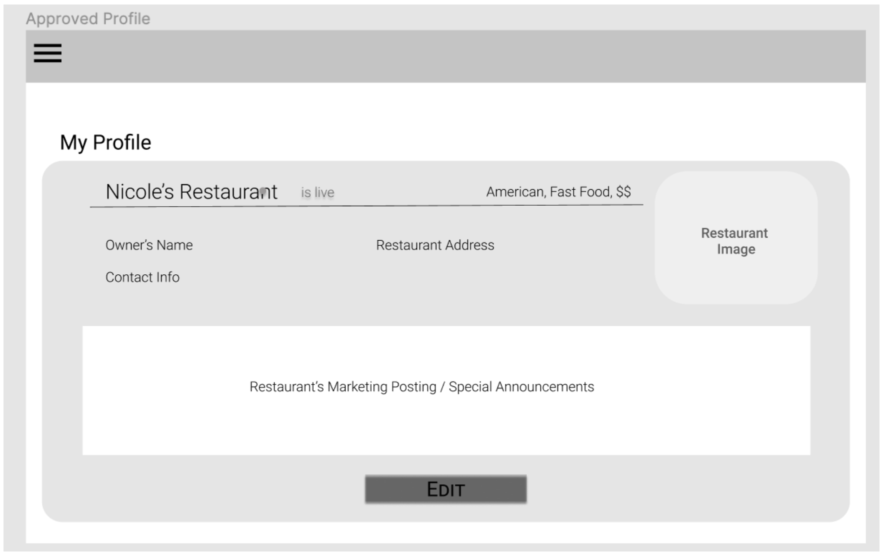
After entering Nicole's restaurant info, she is able to click the 'save' button and wait for the admin's approval of her restaurant.

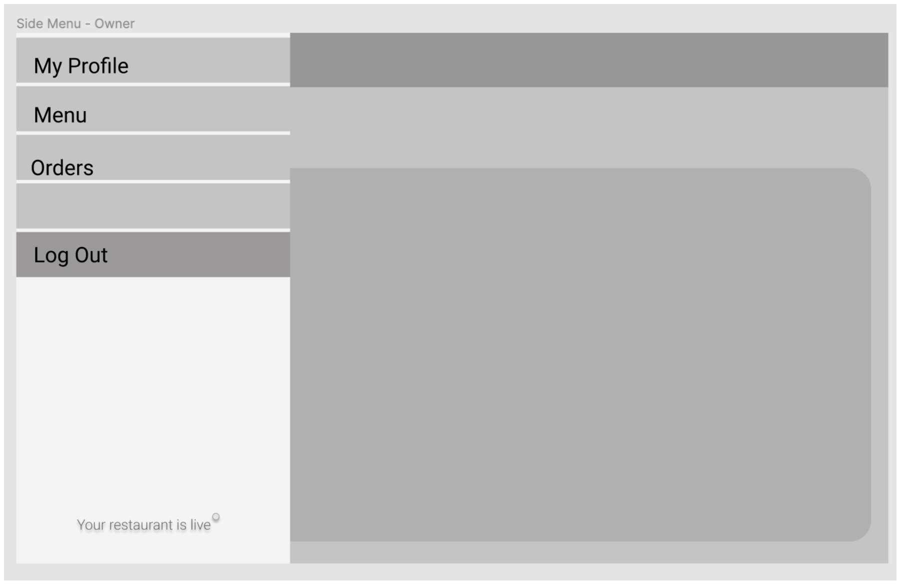




Next, Nicole receives a pop-up of the admin's approval.

After getting approved, Nicole is now officially partnered with Campus Cantina.





Next, Nicole went to the ‘owner’ menu page where she can view her restaurant profile, menu, orders, and the ability to log out.

Nicole decided to check out the ‘menu’ page where she has the ability to put up items from her restaurants. Nicole observed that she can put up item names, price and an

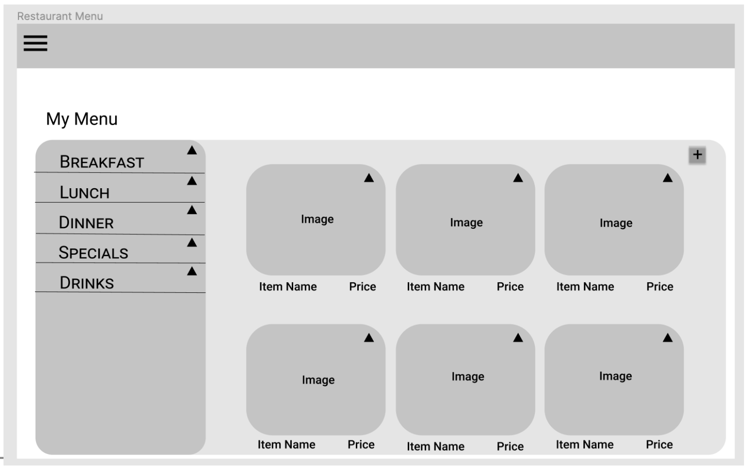
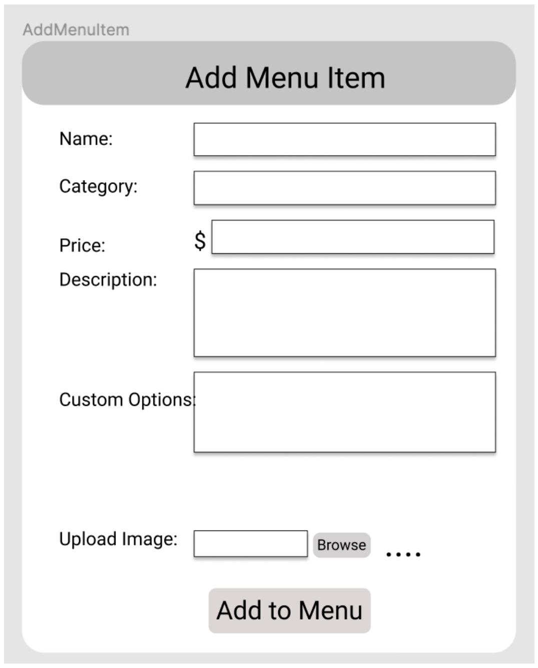


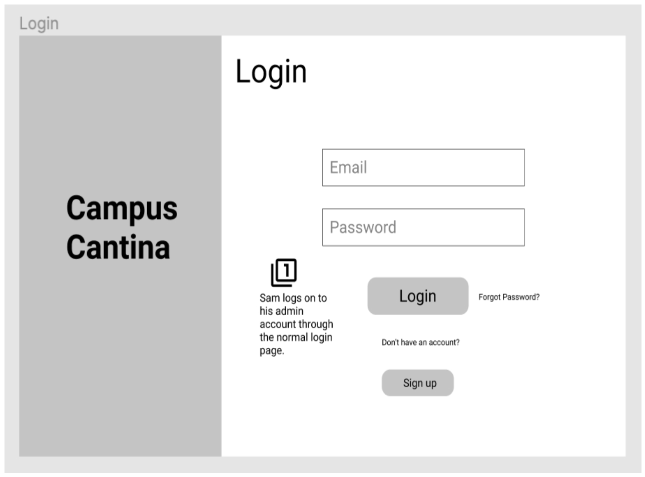
image for her menu item. She clicks the '+' button to add an item menu.



After clicking the '+' button to add an item menu, she enters the name, category, price, description, custom options and an image of her item. Finally, Nicole confirms the information she entered by clicking 'add to menu'.

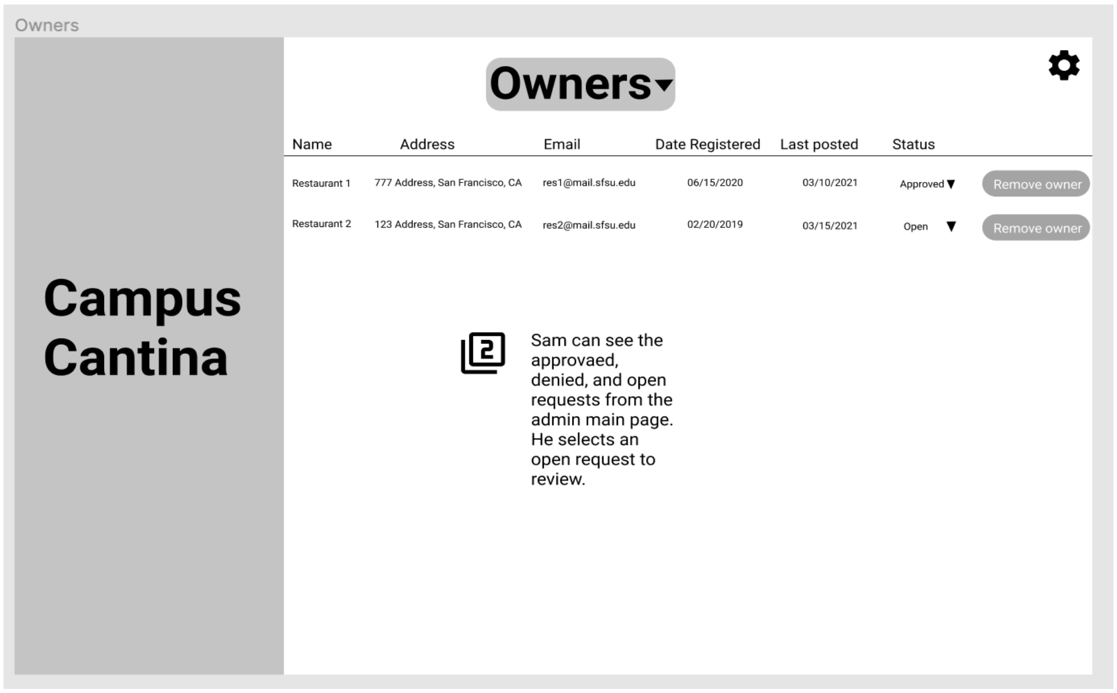
<u>Storyboard 3: Approving Restaurant Postings</u>

This storyboard and accompanying use case may not be implemented, as per CEO request. For demonstration, an 'Approved' column in our Restaurant table will be manually changed to 'True' from 'False' in the MySQL Workbench.



Sam arrives to work at CampusCantina and logs onto his admin account.

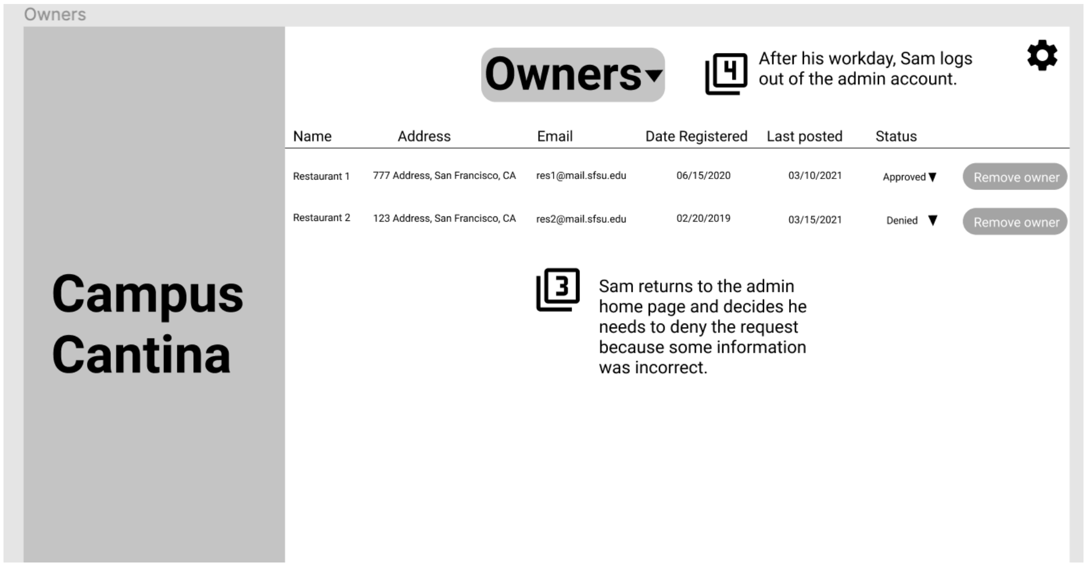
Sam sees a list of partnering requests from nearby restaurants. Sam can deny or approve the requests.



Name	Address	Email	Date Registered	Last posted	Status
Restaurant 1	777 Address, San Francisco, CA	res1@mail.sfsu.edu	06/15/2020	03/10/2021	Approved ▼
Restaurant 2	123 Address, San Francisco, CA	res2@mail.sfsu.edu	02/20/2019	03/15/2021	Open ▼

2 Sam can see the approvaed, denied, and open requests from the admin main page. He selects an open request to review.

Sam decides that he needs to deny the request for one of the restaurants because the second restaurant's information was incorrect.



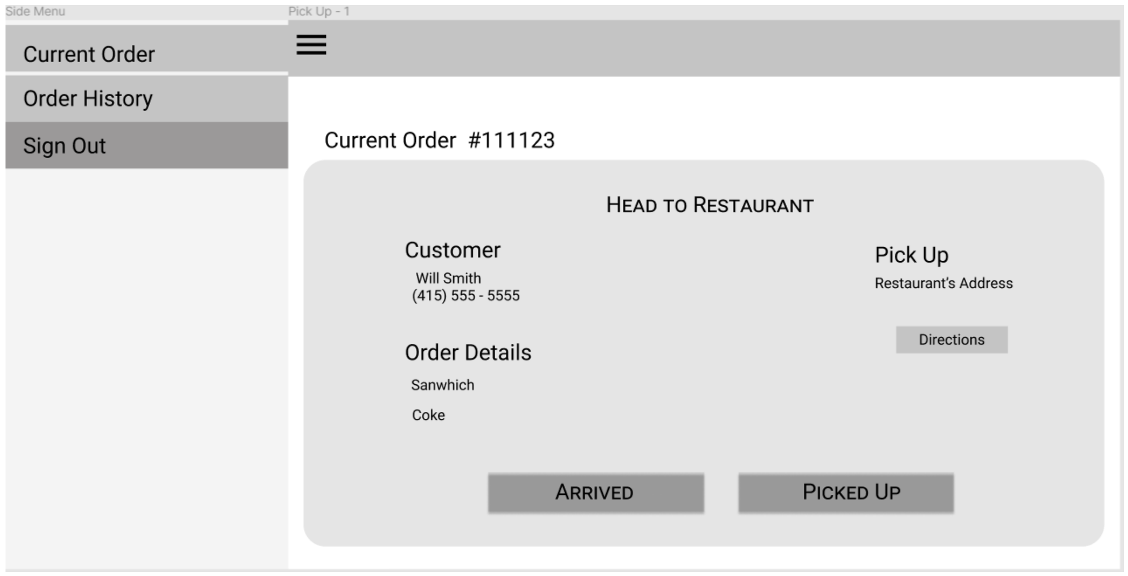
4 After his workday, Sam logs out of the admin account.

Name	Address	Email	Date Registered	Last posted	Status
Restaurant 1	777 Address, San Francisco, CA	res1@mail.sfsu.edu	06/15/2020	03/10/2021	Approved ▼
Restaurant 2	123 Address, San Francisco, CA	res2@mail.sfsu.edu	02/20/2019	03/15/2021	Denied ▼

3 Sam returns to the admin home page and decides he needs to deny the request because some information was incorrect.

Storyboard 4: Pickup & Drop-off Orders (Delivery Driver)

John starts his shift by logging onto Campus Cantina's website. He sees an order assigned to him on the home page.



Current Order #111123

HEAD TO RESTAURANT

Customer
Will Smith
(415) 555 - 5555

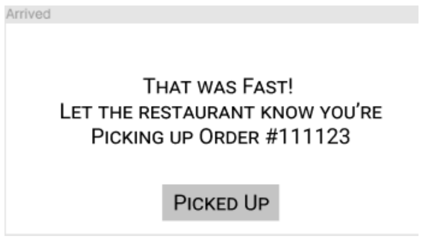
Pick Up
Restaurant's Address

Directions

Order Details
Sanwhich
Coke

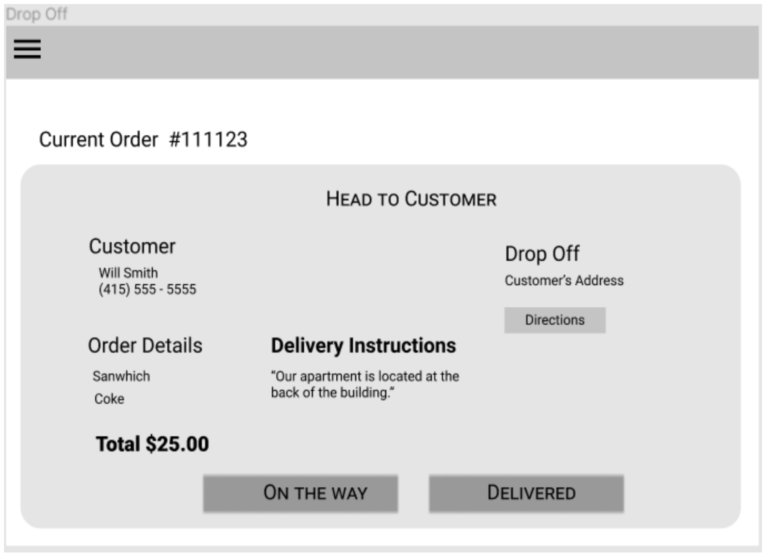
ARRIVED PICKED UP

He checks the order and hits on directions which opens a map used to navigate to the restaurant. Upon arrival, he notifies Campus Cantina by clicking on "Arrived". A small popup reminder is displayed to remind about the order number.



THAT WAS FAST!
LET THE RESTAURANT KNOW YOU'RE PICKING UP ORDER #111123

PICKED UP



Current Order #111123

HEAD TO CUSTOMER

Customer
Will Smith
(415) 555 - 5555

Drop Off
Customer's Address

Directions

Order Details
Sanwhich
Coke

Delivery Instructions
"Our apartment is located at the back of the building."

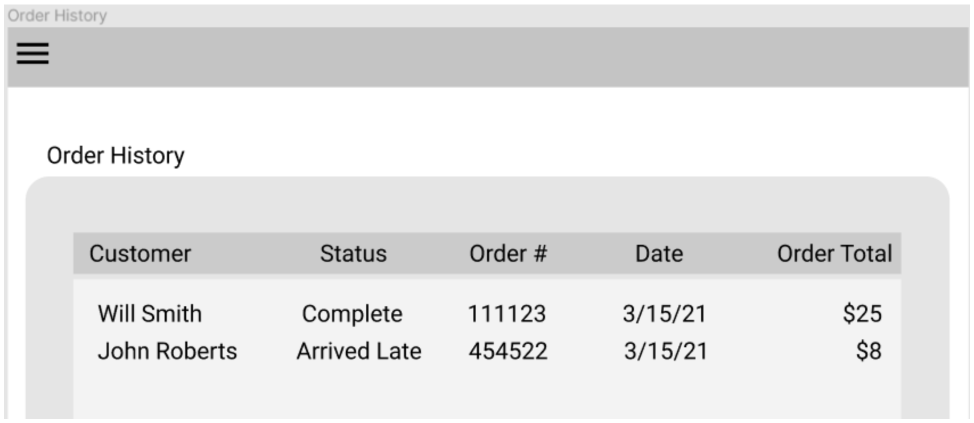
Total $25.00

ON THE WAY DELIVERED

He waits for the food to be cooked and he hits "Picked Up". The next screen would be displayed showing the customer's address with the Directions button. He hits "On the Way". On completing the delivery, Campus Cantina is notified by clicking on "Delivered".

Another order shows up on the homepage and he continues to work on the delivery.

John wants to check his previous orders and clicks on Order History on the menu.



Customer	Status	Order #	Date	Order Total
Will Smith	Complete	111123	3/15/21	$25
John Roberts	Arrived Late	454522	3/15/21	$8

5) High-level Architecture, Database Organization

DB organization:

For M2 we created a development/working database called 'campuscantina_test' on our EC2 server. The database will be accessed remotely by team members instead of each member creating/using local MySQL databases on their machines.

The development database currently has two tables:

1. Food_Cuisines
2. Restaurants

The Food_Cuisines table currently has two columns:

1. ID (of int type)
2. cuisine (varchar(255) type) connected to by foreign key from the Restaurant table.

The Restaurant table currently has 9 columns:

1) ID (of int type)
2) Name (of varchar(255) type)
3) cuisine (of varchar(255) type)
4) Tags (of varchar(255) type)
5) Price_Level (of varchar(4) type)
6) Address (of varchar(255) type)

7) Pic1 (of LONGBLOB type)
8) Lat (of float type)
9) Lng (of float type)

Media storage:

We have decided to use BLOBs for storing images related to restaurants and menus for the time being. In our Restaurants table, the 'Pic1' column is of LONGBLOB type.

For our Google Maps integration, we are currently storing restaurant latitude and longitude coordinates as float types in the 'Lat' and 'Lng' columns of the Restaurant table.

We also have an 'Address' column for street addresses which we may incorporate into Google Maps.

Search/filter architecture and implementation:

Our search relies on SQL %like%. It searches for matches to restaurant names, tags associated with each restaurant, and food cuisines. Users can use a drop-down list next to the search bar to select a specific cuisine or search directly in the search bar.

For example, a search for 'taco' will be matched to Mexican restaurants which have taco as one of their tags. An empty search currently returns all testing/development restaurants in our database. We will later update the search to include menu items.

Own APIs:

We are using Axios to make API calls in our frontend to a backend server that communicates with the MySQL database. Currently, we have an API call for '/api/cuisines' to populate from the DB the list of food cuisines, and a call for '/api/search' to take input and perform searches on the DB.

We will add to this list more API calls including calls for registering users, registering restaurant owners with restaurant information and pictures, and storing and retrieving orders.

6) Key Risks & Corresponding Solutions

Schedule risks:

- ❏ Teammates might have busy schedules during weekdays or have to work during weekends.
    - ❏ We make sure we meet at least a couple of times a week and we have not had any issues or trouble with our schedule.
- ❏ Power outage & Internet problems may cause members to miss meetings.
    - ❏ Noting down the minutes during the meeting and sending it over can help understand the tasks and discussion for members who were not able to join.

Technical risks:

- ❏ Technical unknowns at this time include properly implementing a login/logout system with persistent sessions and logged-in user information availability across the various sites in the application via either Passport.js, Express-session, or some alternative, and properly utilizing chat/messaging functionality via WebSocket or some alternative.
    - ❏ Resolving these issues involves research and reading into the documentation for the various libraries.
- ❏ Adding external libraries to the package can cause a discrepancy if all the team members are unaware of it.
    - ❏ Communicating about every major change helps easy collaboration.
- ❏ Everybody in the team has access to the remote git repository, hence anybody can push changes easily to any branch.
    - ❏ A better practice would be to raise a pull request and send it for review to either the backend lead or the frontend lead for all major commits.
    - ❏ Restricting access for the Master branch helps to manage multiple commits easily.
- ❏ Everybody is not familiar with the stack we are using in this project.
    - ❏ We have assigned the tasks according to the skills.
    - ❏ Made teams such that experienced one shall review and guide other members.

Teamwork risks:

- Working from home can hinder productivity.
    - It is important to hold regular meetings to keep members on track to meet deadlines.
- Miscommunication between teammates.
    - Everyone shall be given a chance to convey their opinions and feedback.
    - Making sure everybody contributes and members get help when required is necessary.

Legal/content risks:

- Aside from the copyright risks with using images obtained online, there might be some potential licensing risks with the various node packages and libraries we have used to date and might need to add in the future. The same applies for the Google Maps API we are using, and any other APIs we might need later.
    - Resolving this will require going through each library and API and reviewing its licensing agreement, and either continue using if free-use, giving credit if applicable, or finding alternatives.
- Currently, user data is stored directly in our database without any security features.
    - We shall consider securing the key user info, such as email and password, in a hash with an external library such as bcrypt.
- Our API call to the database is currently not performing any validation on user input, leaving our system vulnerable to SQL injection.
    - We will address this by adding a validation step before a query is sent to the DB.

7) Project Management

We split the team into frontend and backend sub-teams and assigned tasks for each. For Milestone 2, the backend team met separately to work on the vertical prototype, while the frontend team met separately to work on the Milestone 2 document and UI/UX mock-ups.

The Backend Lead shall assign tasks and target due dates for each member of the backend team and then merge the completed work together to be sent to our GitHub Master for review. The frontend lead shall assign tasks and due dates to each member of the frontend team and then coordinate with the Document Editor to combine the work.

We have been meeting twice a week for specific milestone delivery and also are communicating through the discord channel.

We are using Pull Requests in GitHub for code review. Moving forward, we plan on utilizing "GitHub Projects" to coordinate and assign tasks to team members. Tasks will be placed on the project board, assigned to a member, then marked as complete when done. We also plan on using the Issues feature in GitHub to fix coding bugs that may arise.