

## **SW Engineering CSC648/848 Spring 2021**

**Project title: “Campus Cantina”**

**Milestone 2**

**Team 04**

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

### **Revisions History:**

<b>Version Number</b>	<b>Date Created/Revised</b>	<b>Date Submitted</b>	<b>Instructor Comments</b>
Version 1	March 14 2021	March 19 2021	March 20 2021
Version 2	March 21 2021	March 26 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. SFSU Customers

**1.1 Definition:** SFSU customers are the main users of the app and they 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.
- 1.2.6 **Email:** Email used for registration and login.
- 1.2.7 **Password:** Password used for login.
- 1.2.8 **Status:** Active/Inactive status of the user (can be changed by Admin).

### 2. Driver

**2.1 Definition:** We shall store information on the drivers on the platform for registration, login, and assigning orders.

#### 2.2 Sub-items:

- 2.2.1 **ID:** A unique ID for the driver for the database.
- 2.2.2 **Name:** The full name of the driver.
- 2.2.3 **Phone:** A contact phone number for the driver.

2.2.4 **Email:** The driver's email used for login.

2.2.5 **Password:** A password used for login.

2.2.6 **Restaurant:** The name of the restaurant chosen by the driver to work for.

### 3. Restaurant Owner

#### 3.1 Restaurant

3.1.1 **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.

##### 3.1.2 Sub-items:

3.1.2.1 **ID:** A unique ID for the restaurant for the database.

3.1.2.2 **Restaurant Name:** The name of the restaurant.

3.1.2.3 **Address:** Street address of the restaurant.

3.1.2.4 **Phone:** Contact phone number.

3.1.2.5 **Cuisine:** Cuisine type, such as Italian, Indian, Mexican, etc.

3.1.2.6 **Tags:** Tags for the restaurant, such as type and menu item names.

3.1.2.7 **Small\_Pic:** Small pic for the restaurant to be displayed in searches.

3.1.2.8 **Large\_Pic:** A larger pic to be displayed on the restaurant page.

3.1.2.9 **Price level:** Cost of food, from one \$ to four \$\$\$\$.

3.1.2.10 **Lat:** The latitude coordinate of the restaurant for GoogleMaps.

3.1.2.11 **Lng:** The longitude coordinate of the restaurant for GoogleMaps.

3.1.2.12 **Approved:** The approval status of the restaurant. Default is false.

#### 3.2 Restaurant Owner Profile

3.2.1 **Definition:** We shall store restaurant owner information for restaurant owners to register and login, including email, password, and contact info.

##### 3.2.2 Sub-items:

3.2.2.1 **ID:** A unique ID for the restaurant owner for the database

3.2.2.2 **Name:** Name of the restaurant owner

3.2.2.3 **Phone:** Contact phone for the restaurant owner

3.2.2.4 **Email:** Restaurant owner email used for registration and login.

3.2.2.5 **Password:** Password used for login

#### 3.3 Restaurant Menu Items

3.3.1 **Definition:** We shall store each menu item available at the restaurant along with its description, price, and pictures, provided by the restaurant owner.

**3.3.2 Sub-items:**

- 3.3.2.1 **ID:** A unique ID for the menu item.
- 3.3.2.2 **Restaurant ID:** The ID for the restaurant the item belongs to.
- 3.3.2.3 **Name:** The name of the menu item.
- 3.3.2.4 **Description:** A short description of the menu item.
- 3.3.2.5 **Price:** The price of the item.
- 3.3.2.6 **Custom Options:** Customizations for the menu item.

**4. Food Cuisines**

4.1 **Definition:** A table in the database used to contain all the cuisine types available on the app, used to populate cuisine drop-down lists

**4.2 Sub-items:**

- 4.2.1 **ID:** A unique ID for the cuisine type for the database
- 4.2.3 **Cuisine:** The name of the cuisine type, such as Indian, Mexican, etc.

**5. Orders**

5.1 **Definition:** We shall store order information for each order a registered user (SFSU customer) makes, to be seen by the restaurant owners, drivers, and in the user order history.

**5.2 Sub-items:**

- 5.2.1 **ID:** A unique ID for each order for the database.
- 5.2.2 **Restaurant ID:** The ID of the restaurant the order was placed from.
- 5.2.3 **Restaurant Name:** The name of the restaurant the order was placed from.
- 5.2.4 **Restaurant Address:** The address of the restaurant, for drivers or pickup.
- 5.2.5 **User ID:** The ID of the user that placed the order.
- 5.2.6 **User Name:** The name of the user that placed the order.
- 5.2.7 **Delivery Location:** The address to where the order shall be delivered.
- 5.2.8 **Order Contents:** The menu items added to the order.
- 5.2.9 **Tip:** The tip added by the user for the driver.
- 5.2.10 **Delivery Fee:** The fee charged to the user for the delivery.
- 5.2.11 **Service Fee:** The service fee charged to the user to process the order.
- 5.2.12 **Total:** The total amount for the order charged to the user.
- 5.2.13 **Delivery ETA:** The estimated time of arrival for the delivery.
- 5.2.14 **Delivery Instructions:** The instructions for the driver on how to deliver.

### 3) Functional Requirements - Prioritized

#### Priority 1:

##### Unregistered Users / Guest:

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.

##### Registered Users (SFSU Customers):

5. Registered Users have the same privileges as the unregistered users and the below additional ones.
6. SFSU faculty, students and staff shall register for seeking the delivery service of 'Campus Cantina' and shall login with their credentials.
7. Registered Users shall either request for delivery on the campus by providing their location or pick up from the restaurant directly.

##### Restaurant Owners:

8. Restaurant Owners shall be able to register to advertise their services.
9. Restaurant Owners shall request approval before their restaurants can go live.
10. Restaurant Owners shall be able to add menu items along with photos and prices.

##### Driver:

11. Driver shall be able to login and check the orders assigned to him/her.
12. Driver shall be able to see customer order items.
13. Driver shall be able to see the customer drop-off location on the SFSU campus map.

**Administrator:**

14. The site Administrator shall be required to approve the restaurants before going live on the site.
15. The site Administrator shall be able to delete or remove inappropriate users, drivers, owners.

**Priority 2:****Unregistered Users / Guest:**

16. Campus/Local Favorites shall be displayed on the Home page.

**Registered Users (SFSU Customers):**

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. Personal favorites shall be displayed on the Home screen.

**Priority 3:****Unregistered Users / Guest:**

20. Unregistered Users shall be able to enter a chat to request help.

**Registered Users (SFSU Customers):**

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. Registered Users shall be able to update their profile and can have maximum 3 recent addresses stored for easy checkout. (Delivery in SFSU campus)

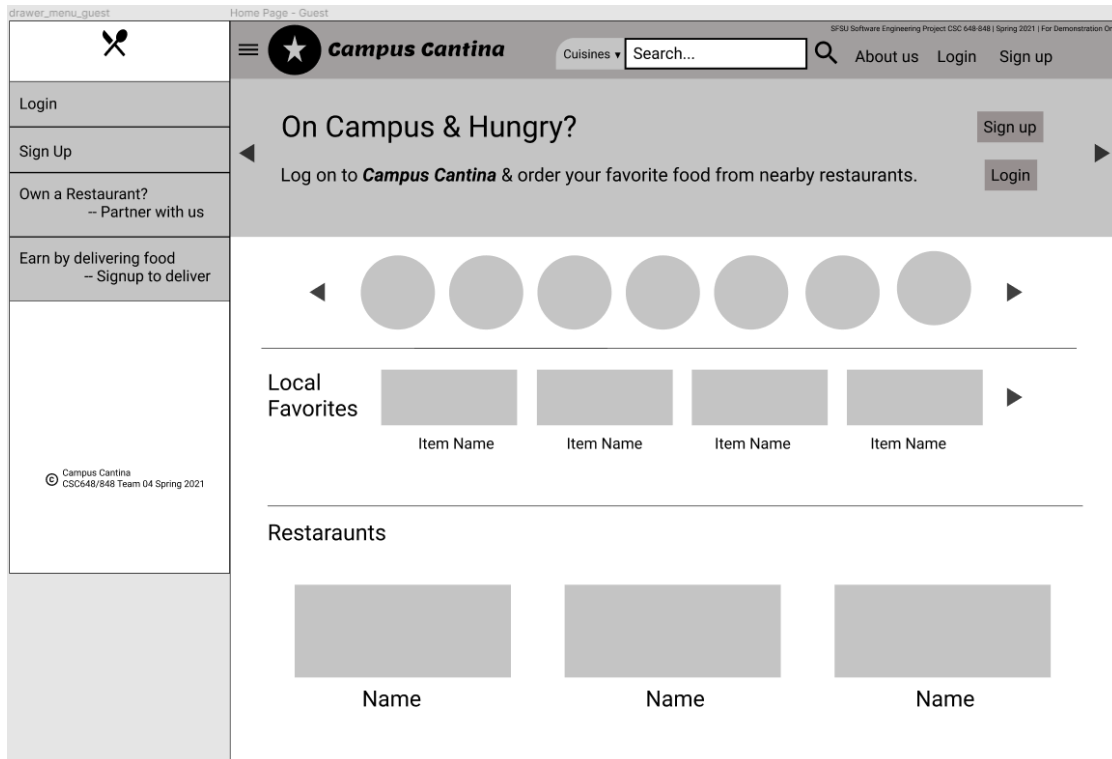
**Driver:**

24. Driver shall be able to chat with customers.
25. Drivers shall be able to share their location after signing in so that an order can be assigned to him/her.

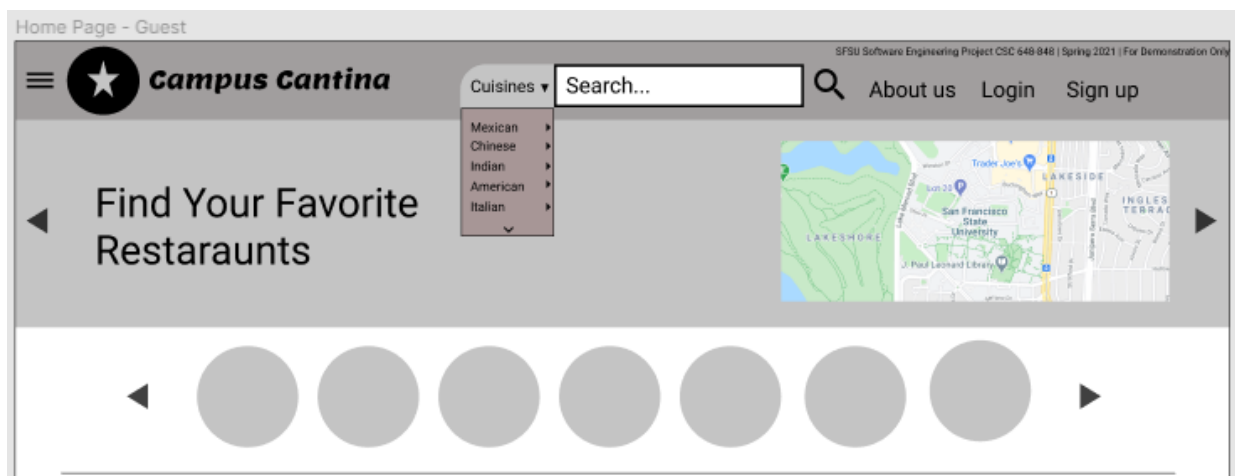
## 4) UI Mockups and Storyboards

### Storyboard 1: Ordering Food

Kim gets on Campus Cantina to order some food.

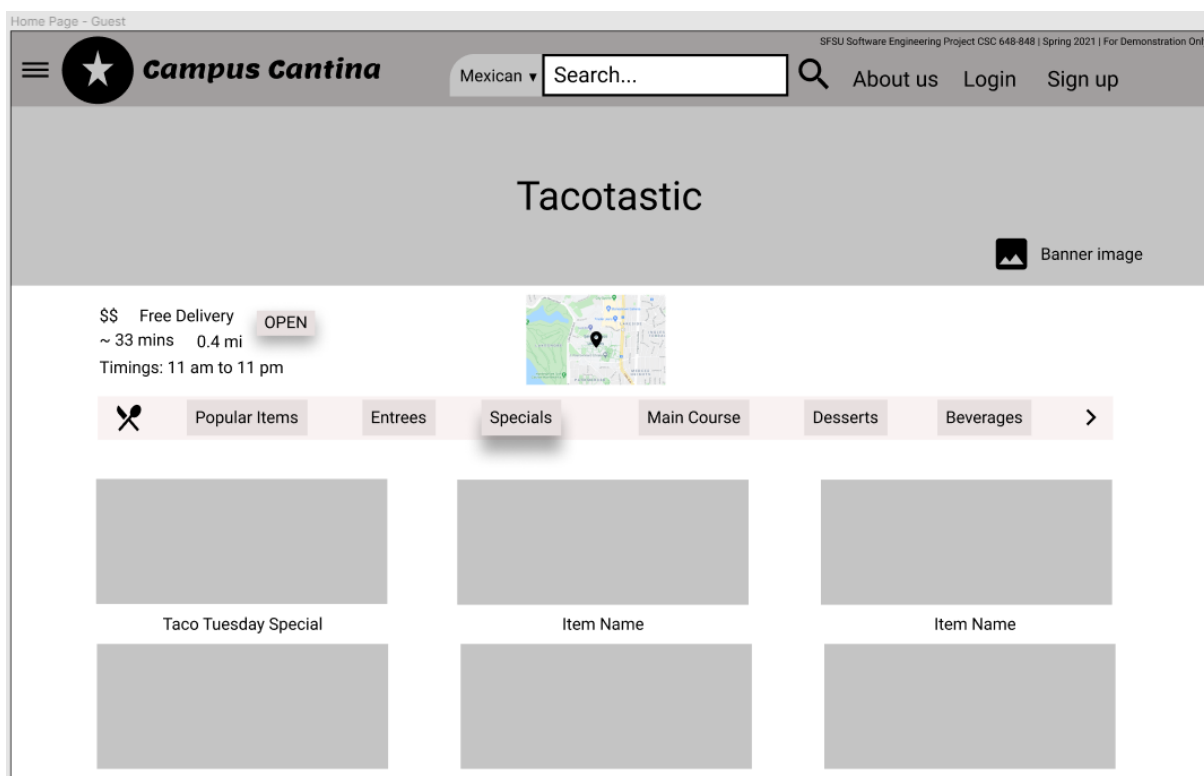
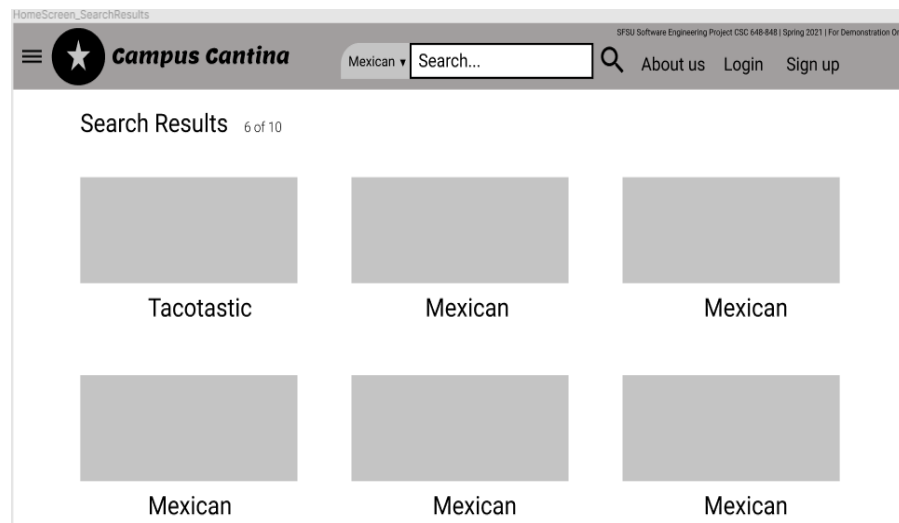


Since she's craving tacos, she selects *Mexican* from the dropdown in the search bar (located at the top of the screen).



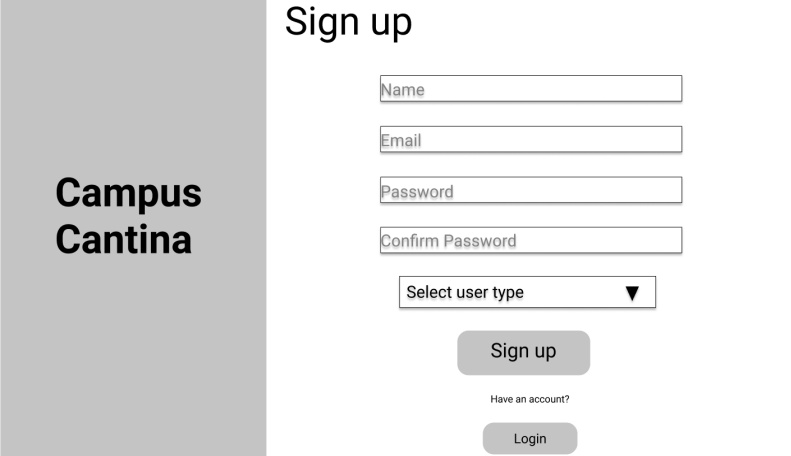


She is shown a list of Mexican restaurants in her area.



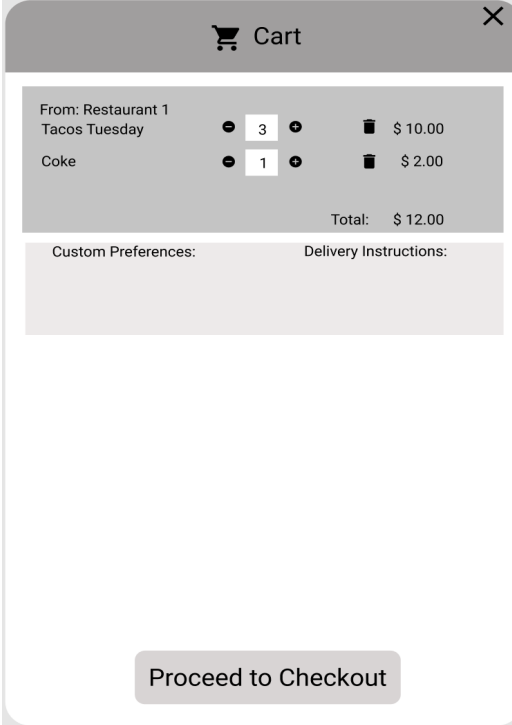
She settles on Tacotastic. As she browses through the categories, she finds a Taco Tuesday deal in the Specials section.

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*.



The image shows a mobile app screen for 'Campus Cantina' with a 'Sign up' form. The form includes input fields for Name, Email, Password, and Confirm Password. Below these is a dropdown menu labeled 'Select user type' with a downward arrow. At the bottom of the form are two buttons: 'Sign up' and 'Login'. A link 'Have an account?' is positioned between the two buttons. The background is a solid light gray.

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*.



The image shows a mobile app screen for the 'Cart' in the 'Campus Cantina' app. The cart contains two items: 'Tacos Tuesday' (3 items, \$10.00) and 'Coke' (1 item, \$2.00). The total is \$12.00. Below the items are two sections: 'Custom Preferences:' and 'Delivery Instructions:'. At the bottom is a button labeled 'Proceed to Checkout'.

Item	Quantity	Price
From: Restaurant 1 Tacos Tuesday	3	\$ 10.00
Coke	1	\$ 2.00
<b>Total:</b>		<b>\$ 12.00</b>

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.

Kim will be alerted of the driver's arrival, she meets them outside her dorm to retrieve and pay for her food.

On the way

**YOUR ORDER HAS BEEN SENT TO THE RESTAURANT!**

**PLEASE HANDOVER THE PAYMENT TO THE DRIVER AFTER RECEIVING YOUR ORDER**

GOT IT!

Kim shall be able to check the placed order in the My Orders section on the home page Navbar now that she is logged in.

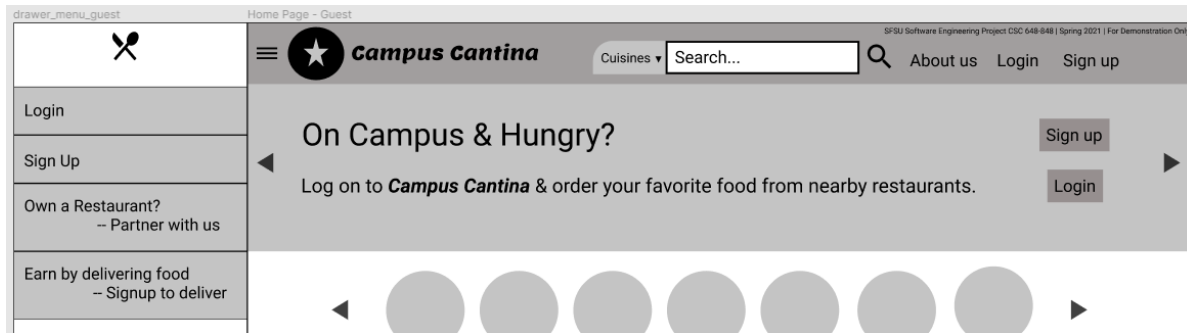
drawer\_menu\_RegUser Home Page - RegUser

  
View Profile  
Sign Out

**Campus Cantina**  
Cuisines Search... About us My Orders Sign Out

### Storyboard 2: Registering Restaurant Account

Nicole wants to register her restaurant in Campus Cantina. Nicole clicks on “*Own a Restaurant* -- *Partner with us*” link in the Nav bar menu and she is asked to Sign Up.



The screenshot shows the "Restaurant Owner Sign up" form. On the left is the Campus Cantina logo and the text "Own a Restaurant? Sign up & Partner with us". The form fields are: Restaurant Name, Restaurant Address, Restaurant Contact Number, Owner Name, Owner Email, Password, and Confirm Password. There is a "Sign up" button and a "Login" button with the text "Have an account?". The footer includes the Campus Cantina logo and the text "©2020-2021 Team 04 Spring 2021".

After, Nicole was able to sign in using her new registered owner account.

The screenshot shows the "Restaurant Owner Login" form. On the left is the Campus Cantina logo and the text "Own a Restaurant? Sign up & Partner with us". The form fields are: Email and Password. There is a "Login" button and a "Forgot Password?" link. Below the login button is the text "Don't have an account?" and a "Sign up" button. The footer includes the Campus Cantina logo and the text "©2020-2021 Team 04 Spring 2021".

Next, Nicole is directed to the 'pending profile' page. Nicole sees that her profile is still in pending status and will be covered by an admin within 24 hours. Currently, Nicole can edit her restaurant's info and put an image.

ownerprofile

My Profile

**PENDING ADMIN APPROVAL WITHIN 24 HOURS**

Nicole's Restaurant American, Fast Food, \$\$

Owner's Name:

Contact Info:

Restaurant Address:

Restaurant Image

Restaurant's Marketing Posting / Special Announcements

EDIT

edit\_ownerprofile

My Profile

**PENDING ADMIN APPROVAL WITHIN 24 HOURS**

Nicole's Restaurant American, Fast Food, \$\$

Owner's Name:

Contact Info:

Restaurant Address:

Restaurant Image

Restaurant's Marketing Posting / Special Announcements

CANCEL SAVE

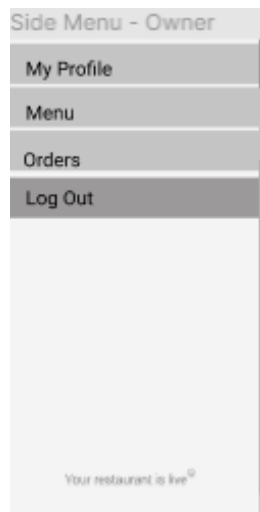
After entering Nicole's restaurant info, she is able to click the 'save' button and wait for the admin's approval of her restaurant.

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

Approval Pop Up

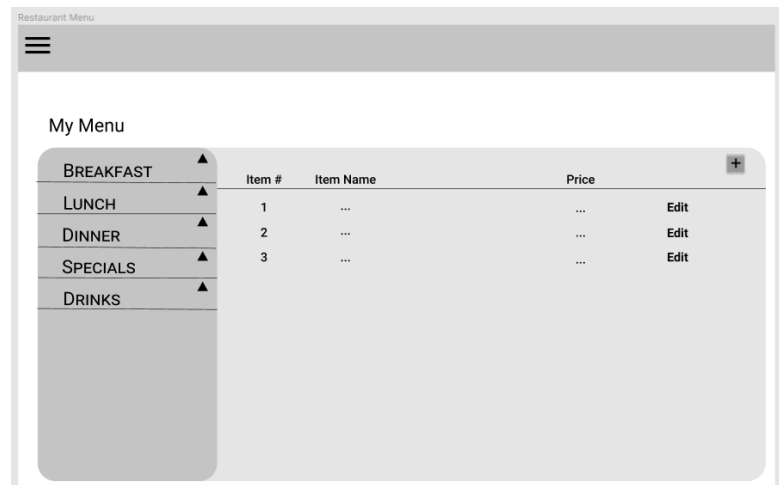
**Your restaurant was approved!**

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.

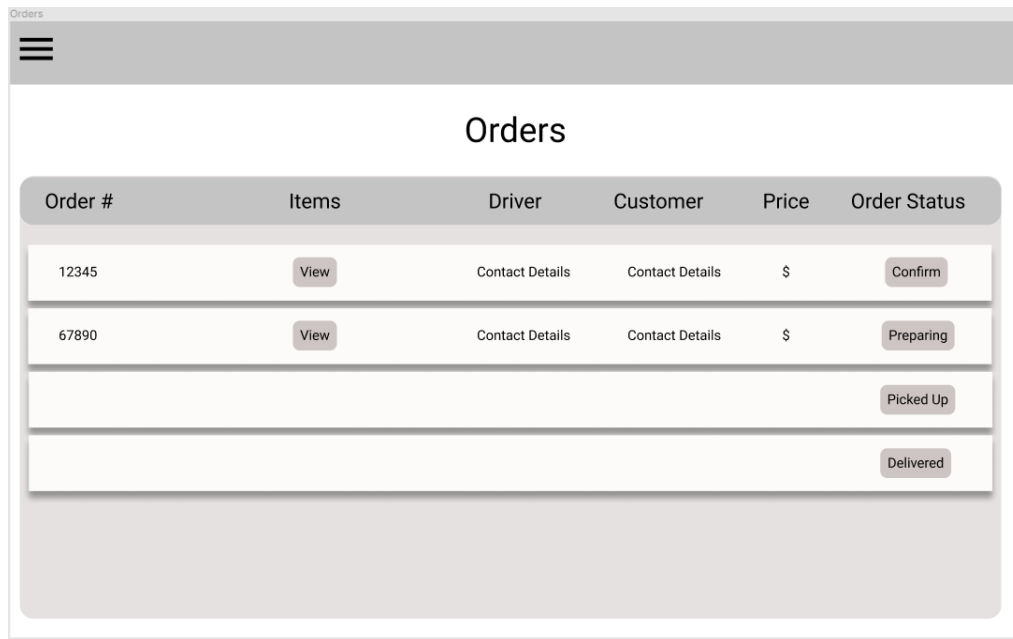
Nicole decided to check out the 'menu' page where she has the ability to put up items from her restaurants. Nicole observed that she can put up item names and prices for her item menu. She clicks the '+' button to add an item menu.



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

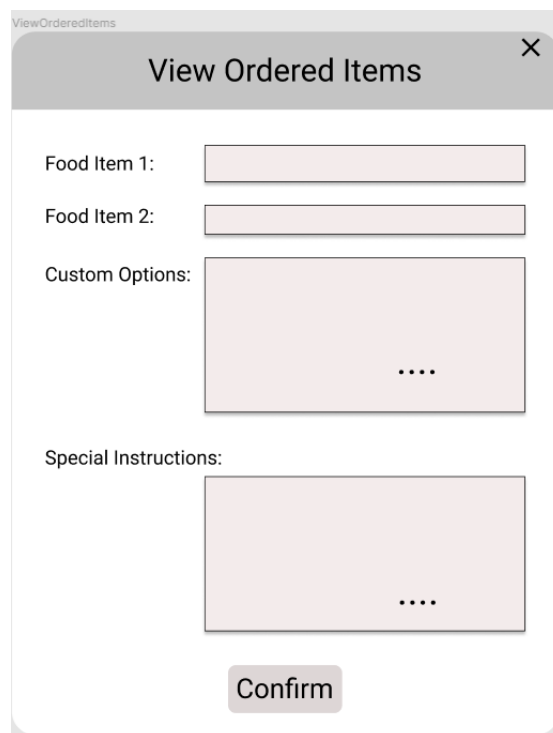
A form titled "Add Menu Item" with a light gray background. It contains five input fields: "Name:", "Category:", "Price:" (with a "\$" symbol), "Description:", and "Custom Options:". At the bottom of the form are two buttons: "CANCEL" and "SAVE".

Nicole shall be able to check their orders by clicking on Orders from the Menu options.



Order #	Items	Driver	Customer	Price	Order Status
12345	<a href="#">View</a>	Contact Details	Contact Details	\$	<a href="#">Confirm</a>
67890	<a href="#">View</a>	Contact Details	Contact Details	\$	<a href="#">Preparing</a>
					<a href="#">Picked Up</a>
					<a href="#">Delivered</a>

Nicole views the items ordered and confirms it. After the food is prepared, it is handed over to the assigned driver.



### View Ordered Items

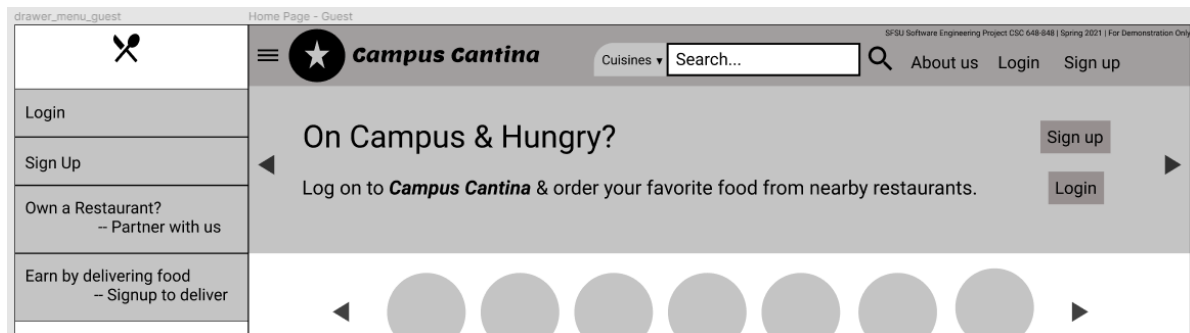
Food Item 1:

Food Item 2:

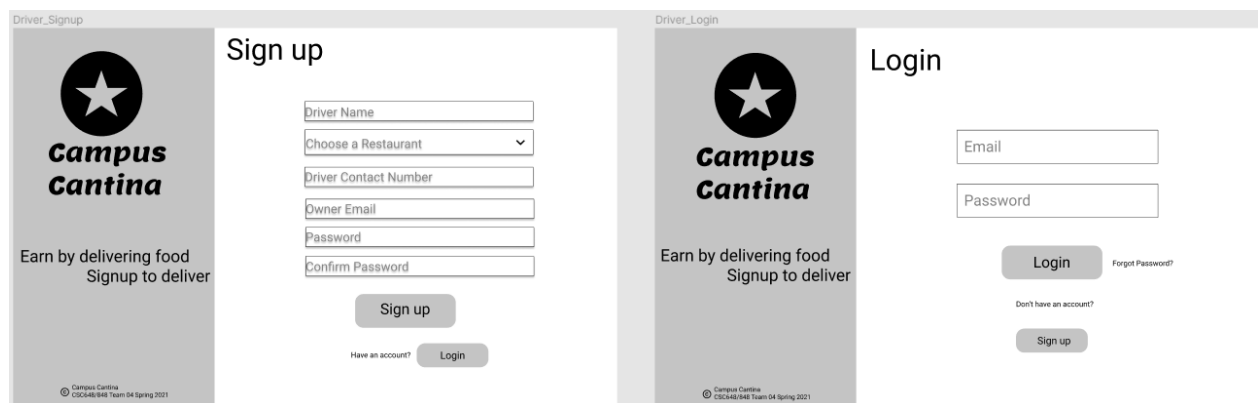
Custom Options:

Special Instructions:

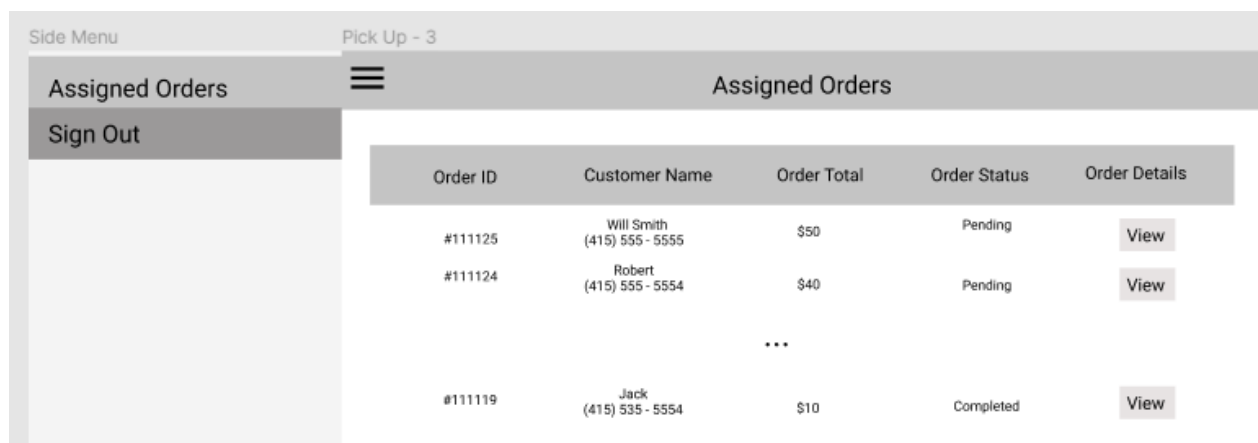
[Confirm](#)

Storyboard 3: Pickup & Drop-off Orders (Delivery Driver)

John wants to sign-up and work for Campus Cantina. He navigates to the driver signup page by clicking on “Earn by Delivering Food” on the home page side menu.



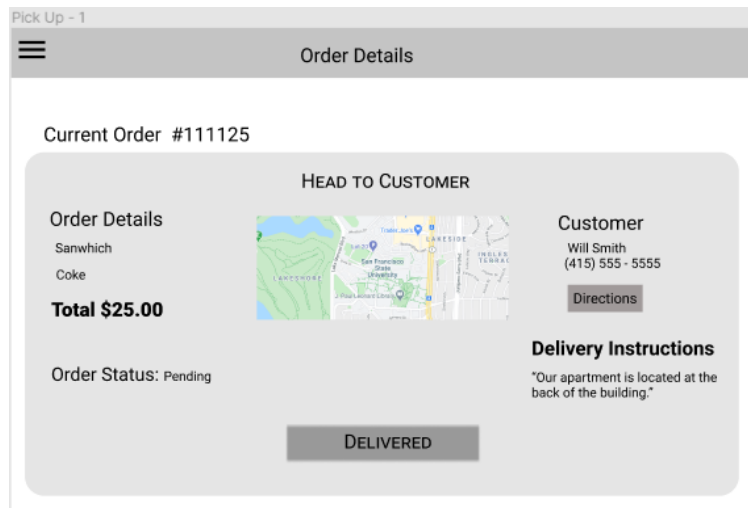
He chooses a restaurant to work for in the signup form. John starts his shift by logging onto Campus Cantina’s website.



On the Assigned Orders screen, he picks the order for delivery by clicking on View.



Order Details screen is displayed. He checks the order and picks up the order from the restaurant and hits on directions which opens a map used to navigate to the customer's location. On completing the delivery, the driver clicks on the Delivered button and is navigated to the home page. John selects another order from the "Assigned Orders" section.



## 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 shall have the following tables:

1. SFSU\_Customers
2. Restaurant\_Owners
3. Drivers
4. Food\_Cuisines
5. Restaurants
6. Menu\_Items
7. Orders

The SFSU\_Customers table shall have 8 columns:

- 1) ID (of int type)
- 2) Name (of varchar(255) type)

- 3) Address (of varchar(255) type)
- 4) Type (of varchar(255) type)
- 5) Phone (of int type)
- 6) Email (of varchar(255) type)
- 7) Password (of varchar(255) type)
- 8) Status (of varchar(255) type)

The Resutauant\_Owners table shall have 5 columns:

- 1) ID (of int type)
- 2) Name (of varchar(255) type)
- 3) Phone (of int type)
- 4) Email of (of varchar(255) type)
- 5) Password (of varchar(255) type)

The Drivers table shall have 6 columns:

- 1) ID (of int type)
- 2) Name (of varchar(255) type)
- 3) Phone (of int type)
- 4) Email (of varchar(255) type)
- 5) Password (of varchar(255) type)
- 6) Restaurant (of varchar(255) type)

The Food\_Cuisines table shall have 2 columns:

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

The Restaurants table shall have 12 columns:

- 1) ID (of int type)
- 2) Name (of varchar(255) type)

- 3) Address (of varchar(255) type)
- 4) Phone (of int type)
- 5) Cuisine (of varchar(255) type)
- 6) Tags (of varchar(255) type)
- 7) Small\_Pic (of LONGBLOB type)
- 8) Large\_Pic (of LONGBLOB type)
- 9) Price\_Level (of varchar(4) type)
- 10) Lat (of float type)
- 11) Lng (of float type)
- 12) Approved (of boolean type)

The Menu\_Items table shall have 6 columns:

- 1) ID (of int type)
- 2) Restaurant ID (of int type)
- 3) Name (of varchar(255) type)
- 4) Description (of varchar(255) type)
- 5) Price of (int type)
- 6) Custom Options (of varchar(255) type)

The Orders table shall have 14 columns:

- 1) ID (of int type)
- 2) Restaurant ID (of int type)
- 3) Restaurant Name (of varchar(255) type)
- 4) Restaurant Address (of varchar(255) type)
- 5) User ID (of int type)
- 6) User Name (of varchar(255) type)
- 7) Delivery Location (of varchar(255) type)
- 8) Order Contents (of varchar(255) type)
- 9) Tip (of int type)
- 10) Delivery fee (of int type)
- 11) Service fee (of int type)

- 12) Total of (int type)
- 13) Delivery ETA (of varchar(255) type)
- 14) Delivery Instructions (of varchar(255) type)

***Media storage:***

We have decided to use BLOBs for storing images related to restaurants and menus. In our Restaurants table, the 'Small\_Pic' and 'Large\_Pic' columns are of LONGBLOB type, as is the 'Picture' column in our Menu\_Items table.

For our Google Maps integration, we are 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, menu items, tags associated with each restaurant, and food cuisine types. Users can use a drop-down list next to the search bar to select a specific cuisine and/or search directly in the search bar. For example, selecting 'Mexican' in the drop-down menu and hitting search will return all Mexican restaurants in the DB. A search can also be done on menu items or tags for each restaurant, for example a search for 'taco' will be matched to restaurants which have taco as one of their menu items or tags. A search for a cuisine without selecting anything in the drop-down menu will return all restaurants of that type of cuisine. An empty search with no cuisine selected returns all testing/development restaurants in our database.

***Own APIs:***

We are using Axios to make API calls in our frontend to a backend server that communicates with the MySQL database. 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.

Other API calls shall include calls for registering users, registering restaurant owners with restaurant information and pictures, and storing and retrieving orders.

## 6) Key Risks & Corresponding Solutions

### Schedule risks:

- ❑ Delivery on Schedule.
  - ❑ We shall focus mainly on P1 list scope to deliver a minimal good viable product.
- ❑ 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:

- ❑ 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.
  - ❑ We also want to stay organized and regularly keep track of the progress by using github projects as a project management tool.

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.