SW Engineering CSC 648-848 Spring 2023

Hungry Gator Application

May 23, 2023

Milestone 5

Team 5

Student Name	**Email**
Maliah Chin	mchin4@sfsu.edu
Hajime Miyazaki	hmiyazaki@sfsu.edu
Mario Leyva Moreno	mleyvamoreno@mail.sfsu.edu
Scott Nguyen	snguyen23@mail.sfsu.edu
Aneida Guadalupe Blanco Palacio	ablancopalacio@mail.sfsu.edu

URL: http://ec2-54-219-129-242.us-west-1.compute.amazonaws.com:3000

1) Product Summary: Hungry Gator

P1 List of Features:

1. Unregistered Users
    1.1 Unregistered users shall be able to make an account.
    1.2 Unregistered users shall be able to search/browse for a restaurant and its menu items.
    1.3 Unregistered users shall be able to view search results.
    1.4 Unregistered users shall be able to filter results by category.

2. SFSU Registered Users
    2.1 Registered users shall inherit all priority 1 unregistered user functionality.
    2.2 A registered user shall be associated with one account.
    2.3 A registered user shall be able to purchase food.
    2.4 A registered user shall be able to select drop off location on campus.
    2.5 A registered user shall be able to sort restaurants based on delivery time.
    2.6 A registered user shall be able to log in.

3. Admin
    3.1 Admin shall be required to approve all restaurant posts before they go live on the application.
    3.2 Admin shall be able to remove users.
    3.3 Admin shall be able to remove restaurants

4. Restaurant Owner
    4.1 A restaurant owner shall be able to register their restaurant on the app and wait up to 24 hours for admin approval.
    4.2 A restaurant owner shall be able to post a menu of their dishes on the app for users to see.
    4.3 A restaurant owner shall be able to register as a restaurant owner.
    4.4 A restaurant owner shall be able to log in.

5. Delivery Driver
    5.1 A delivery driver shall be required to register to apply for the job.
    5.2 A delivery driver shall be able to access their delivery details.

What Makes Our Product Unique:

Our product is an exclusive application provided for SFSU students, faculty, and staff. We are here to cater to the busy schedules of everyone in the SFSU community. As local students, we want to be able to provide a convenient service to our school. In our application, we provide special features such as search by delivery time and direct delivery to your on campus location. We provide flexible jobs, student discounts, and access to menus from local restaurants.

URL: http://ec2-54-219-129-242.us-west-1.compute.amazonaws.com:3000

2) Milestones 1-4 Documentation:

SW Engineering CSC 648-848 Spring 2023

Hungry Gator Application

Milestone 1

Team 5

Student Name	GitHub Username	Student Role
Maliah Chin	maliahc	Team Lead
Hajime Miyazaki	hmiyazaki95	Back End Lead
Mario Leyva Moreno	leyvaM	Front End Lead 1
Scott Nguyen	Saiber70	GitHub Master
Aneida Guadalupe Blanco Palacio	Blanco1220	Front End Lead 2

Team Lead Email: mchin4@sfsu.edu

History Table

Date Submitted	Date Revised
3/13/2023	5/5/23

1. Executive Summary:

College can be stressful for many students. As they are constantly on the move getting ready for the next class or studying all night for exams, they may not have the time to go out and buy groceries to cook, or buy food from their local restaurant. To make life easier for SFSU students and faculty, we have created the Hungry Gator application which is designed to deliver food anywhere on campus. This application is especially convenient for SFSU faculty and staff, who are constantly working in their offices, attending meetings, and teaching multiple courses with minimal breaks. Our application will ensure that food is delivered directly to them at a specific location on campus, so that they may not be inconvenienced to go out and buy food when dealing with time constraints. Another key advantage of our application compared to other food delivery apps, is that we are constantly providing discount opportunities and minimal delivery fees in order to make food more affordable for college students. Another big advantage headed towards the SFSU Community is quick delivery times for those who need a quick bite between classes. Students often have to incur various expenses due to high tuition costs and materials required for various courses, so it is our goal to make an already expensive service as affordable as possible to members of SFSU.

The Hungry Gator application provides an exclusive membership program to SFSU students, staff, and faculty, which will activate upon user registration. Members will receive exclusive discounts and reduced delivery fees on select restaurants, which will help make food more affordable and convenient for students and faculty alike. Our application also supports on-site delivery, a service that ensures food is delivered anywhere on campus by providing a digital map of the campus to delivery drivers. This will help drivers pinpoint an exact location on campus and ensure fast delivery times. This also helps the user track their orders and avoid communication issues with drivers about being unable to find the dropoff location, since drivers will be guided directly to your specific location on the map. Our application is also unique in terms of what is available since our goal is to get food delivered as quickly as possible to our users. The application will not show restaurants out of range and will prioritize the restaurants that are closest to the SFSU campus.

Our team is composed of individuals from various backgrounds who have come together to deliver a reliable service to our users. Our team is composed of a team lead who organizes tasks and provides design plans for the team to follow. We also have a github master who keeps track of our product's github page and ensures that all aspects of our application are up to date and ready for launch. The back-end developer is responsible for authenticating user accounts and managing our team's database where user data is safely secured. Finally, we have front-end leads who are responsible for providing a reliable user interface where users can easily interact with our application.

2. Personae and Main Use Cases:

Personas:

1. **Student: Jasmine**

About Jasmine

- Very Busy
- low budget
- loves to support local restaurants
- always looking for a good meal deal
- tech savvy

Goal and Scenarios:

- Decided she needs a quick and cheap meal. Jasmine wants to check if there is anything convenient nearby.
- If she finds something that meets her criteria, she would like to order it to be delivered as soon as possible to her direct location on campus.

2. **Professor: Michael**

About Michael

- very busy
- has limited time for waiting on meals
- has limited WWW skills
- always in different spots on campus
- likes to try new local restaurants

Goals:

- Decided he would like to order something for lunch in between classes. Michael wants to check if there is a quick bite that will meet his time constraints
- If he finds something that can be delivered to his direct location, he can save time and order the food ahead of time.

3. **Restaurant Owner: Tina**

About Tina

- new to the SFSU neighborhood
- has basic WWW skills
- looking for new business opportunities
- wants to support local students by offering student discounts
- has little patience for WWW sites

Goals:

- Tina wants a new way to advertise her business to her local community.
- Tina signs up for Hungry Gator to promote her business and provides student discounts.

4. **Admin: Kris**

Delivery Driver: Lucas

About Lucas

- needs a flexible job
- has good WWW skills
- likes to work on his own schedule
- directionally compromised

Goals:

- Lucas is looking for a job that will fit his class schedule. Wants to deliver but bad at directions
- Lucas chooses his own hours to work. He sees a map that can guide him to where he needs to make the delivery.

5. **Undergraduate Advisor/ Staff: Manuel**

- limited WWW skills
- not too patient with complicated WWW apps
- picky about which type of cuisine he likes
- likes to chose budget friendly items
- very busy

Goals:

- Manuel decided he wants Italian food for his short lunch break.
- If he uses the search bar to look up “italian” food, he will find a plethora of nearby italian restaurants available to deliver to his exact location

1. Student: Jasmine Order’s a Late Night Meal Deal

Jasmine is a **SFSU student** who has been busy all evening studying hard for her midterms. As the night progresses, Jasmine realizes that it’s time to eat but she fears she doesn’t have enough time to go pickup anything for dinner. Luckily, she heard about this new web application, Hungry Gators, that has been created specifically for SFSU students. Jasmine opens her laptop and enters the link for Hungry Gators. She then notices all of the appetizing pictures that inspire her to find a meal for the evening. She searches up her favorite cuisine, Italian, and looks through the results. She finds something yummy and even notices that there is a special student discount. Jasmine adds her food to her car and continues to check out. At checkout, she is then prompted to sign up using her SFSU email. Since she is a student, the chosen discount has been applied. She notices that there is a special function that allows for her food to get delivered directly to her dorm, so she enters her location. Jasmine then gets back to her books and waits for the speedy delivery.

2. Professor: Michael Finds a Convenient Lunch

Michael is an esteemed **professor** at SFSU. His days are full of meetings, new classes to teach, and hosting office hours to students in need. Because of his busy schedule, he finds it nearly impossible to have time to enjoy his meals. Recently, he has heard about this new web application that makes ordering food simple and quick called Hungry Gator. Michael opens his phone's browser and enters the link. He is greeted with pictures of delicious foods and drinks. He decides to order something fun and adds it to his cart. Michael uses his SFSU email to sign in at checkout. After signing up, he enters his location and notices the option to have it delivered directly to his office! Satisfied with the convenience of Hungry Gator, Michael checks out and waits for his order.

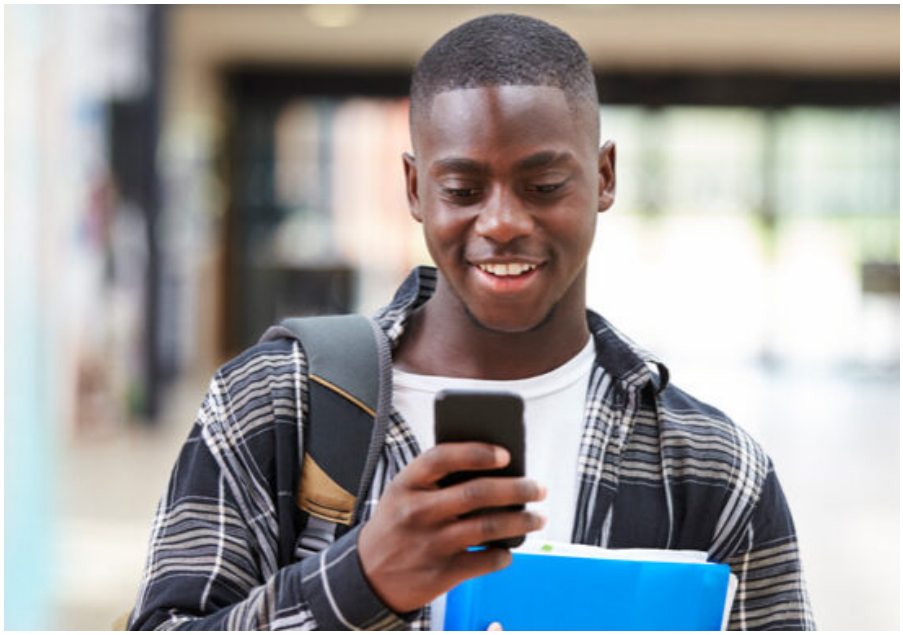
3. Restaurant Owner: Tina Promotes Her Local Restaurant

Tina is a local **restaurant owner** who works near SFSU. She has a cafe that has opened up recently and has been looking for a new opportunity to expand her business. A new web app has come across her radar that has peaked her interest, called the Hungry Gator. Tina's full schedule makes it difficult for her to learn new apps, but the simple layout of Hungry Gator offers her a quick way to grow her audience. She opens Hungry Gator and looks for the registration button for restaurants. She clicks on the button and is greeted with a form that asks for her basic information. Because Tina knows that the SFSU campus is bustling with business, she fills out the section for special offers aimed towards the students. She is also excited to upload photos that showcase her best snacks. Tina then completes the form and submits the request. A message that the form has been submitted and that it may take up to 24 hours for the form to be accepted appears and Tina acknowledges the information. Happy to have a chance to give back to students and create a new branch of business, Tina waits for her approval.

4. Admin: Kris Approves User’s Uploads To Be Displayed on the Application

Kris is the **admin** for Hungry Gator. His main goal is to observe and act upon requests handled on the application. Kris is the guy who acknowledges and approves the requests for photos uploaded, restaurants requesting to be featured on the site, and pending account information. Kris can also decline the requests if necessary. Kris is able to assist the customers and clients through email and has access to all the database information so he is able to review all of the data.

5. Delivery Driver: Lucas Finds a Flexible Job



Lucas is an SFSU student who wants a job that is flexible with his schedule as a student. Friends convinced him to apply for Hungry Gator. When starting his job, he finds the app easy to use and navigate. He signs in and various orders instantly pop up in his phone that he can decide to take. Being able to decide when to work gives him peace of mind about having time to do his assignments and attending classes. Besides being able to fit in his job at Hungry Gator in his busy schedule, Lucas feels safe making deliveries since the app gives him the choice of meeting up at a secure point on campus.

6. Undergraduate Advisor/ Staff: Manuel Orders a Quick Bite

Manuel is an undergraduate advisor and professor at SFSU. His responsibilities as an undergraduate advisor include meeting with students at specific times and assisting in the overall academic progress of students at SFSU. In addition to his staff responsibilities, he is also teaching several courses throughout the day. Due to his busy schedule, he finds it nearly impossible to enjoy his lunch and doesn't always prepare his own meals for the day. After hearing about the Hungry Gator application through one of his colleagues, Manuel decided to download the application and browse through the list of Indian restaurants. He was happy to find his favorite local restaurant and was even surprised by the option to have it delivered directly to his office. Manuel was happy with his purchase and was even given a membership discount after signing up with his SFSU email.

3. List of Main Data Items and Entities

Entity:

1. Unregistered Users
    - Searching/browsing for restaurants.
    - Able to register for an account using a valid SFSU email.
2. Registered Users
    - Ordering food and applying discount codes.
    - Selecting dropoff location on any purchase.
    - Editing user profile.
    - Associated with one SFSU email.
3. Admin
    - Manages the application's database.
    - Approves posts and assists users with registering.
4. Restaurant Owner (Restaurant)
    - Registering their restaurant on the site and posting images of their dishes.
5. Delivery
    - In charge of delivering food.

Main data items:

1. Restaurant
    - Contains the name, location, image of each menu item, food category and price of each dish.
2. Account
    - Account is associated with one user.
    - Contains a username, valid SFSU email, and password.
3. Discount or Promotion
    - Visible to any registered user.
    - Applicable to any order via a discount code.

4. Initial List of Functional Requirements

1. Unregistered Users
   1.1. Unregistered users shall be able to make an account.
   1.2. Unregistered users shall be required to make an account for ordering,
   1.3. Unregistered users shall be able to search/browse for a restaurant.
   1.4. Unregistered users shall be able to view search results and filter by category, price, and delivery time.
   1.5. Unregistered users shall have view only access to user ratings and reviews.

2. Registered Users
   2.1. A registered user shall be associated with one student/faculty account.
   2.2. A registered user shall be able to purchase food.
   2.3. A registered user shall be able to track their order.
   2.4. A registered user shall be able to see discount deals on select restaurants.
   2.5. A registered user shall be able to see their order history.
   2.6. A registered user shall be able to edit their user profile.
   2.7. A registered user shall be able to select drop off location on campus.
   2.8. A registered user shall be able to write reviews on restaurants and rate their food or delivery service.
   2.9 A registered user shall inherit all functionalities of unregistered Users

3. Admin
   3.1. Admin shall be responsible for verifying and authenticating user accounts.
   3.2. Admin shall be responsible for managing customer reviews and messages.
   3.3. Admin shall be responsible for storing encrypted user data in the database.
   3.4. Admin shall be responsible for providing feedback to users about incorrect account details, searching, and other features.
   3.5. Admin shall be responsible for retrieving requested user data for registered users.
   3.6 Admin shall approve all restaurant postings before they are displayed on application.

4. Discount or Promotions
   5.1. Student discounts shall be applied to students with a valid student ID
   5.2. All other discounts/promotions shall be applied to anyone with a verified SFSU email

5. Restaurant Owner
   6.1. A restaurant owner shall be able to register their restaurant on the app and wait up to 24 hours for admin approval.

6.2. A restaurant owner shall be able to post images of their dishes on the app for users to see visuals of their menus.

6. Delivery Driver
   7.1. A delivery driver shall be required to register to apply for the job.
   7.2. Upon employment, a delivery driver shall receive an ID verifying that he/she works as an independent contractor for the company.
   7.3 A delivery driver shall be able to receive delivery order with all the data and map of SFSU.

5. List of Non-Functional Requirements

1. Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0
2. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers
3. All or selected application functions shall render well on mobile devices
4. Data shall be stored in the database on the team's deployment server.
5. No more than 50 concurrent users shall be accessing the application at any time
6. Privacy of users shall be protected
7. The language used shall be English (no localization needed)
8. Application shall be very easy to use and intuitive
9. Application shall follow established architecture patterns
10. Application code and its repository shall be easy to inspect and maintain
11. Google analytics shall be used
12. No email clients shall be allowed. Interested users can only message to sellers via in-site messaging. One round of messaging (from user to seller) is enough for this application
13. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI.
14. Site security: basic best practices shall be applied (as covered in the class) for main data items
15. Media formats shall be standard as used in the market today
16. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development
17. The application UI (WWW and mobile) shall prominently display the following exact text on all pages "SFSU Software Engineering Project CSC 648-848, Spring 2023. For Demonstration Only" at the top of the WWW page nav bar. (Important so as to not confuse this with a real application).

6. Competitive Analysis:

Application	GrubHub	UberEats	PostMates	DoorDash	Hungry Gator
Search Filters	✓	✓	✓	✓	✓
User Registration	✓	✓	✓	✓	✓
Discount Programs	✓	✓	✓	✓	✓
User Reviews & Ratings	✓	✓	✓	✓	✓
SFSU Student/Faculty Pass	✕	✕	✕	✕	✓
On-campus delivery	✕	✕	✕	✕	✓
Search by delivery time	✕	✕	✕	✕	✓

✓ - Supported, ✕ - Not supported

Unlike our competitors, the Hungry Gator application is tailored to students at SFSU. This presents several unique features that are unavailable in current applications. Those features include an exclusive student membership pass that will allow students to enjoy exclusive discount deals and a $0 delivery fee on select restaurants. The pass will be free of charge and will activate upon user registration, and will expire after the student's expected graduation date. Students will be able to enjoy a variety of discount deals for free without having to worry about paying a monthly subscription fee like DashPass and Grubhub+. Another feature is on-site delivery, which will ensure that food is delivered to students anywhere on campus. Using a full map layout of the campus, drivers will be able to deliver food to students, whether at a specific building on campus, the library, or even student dorms. This application is also available to campus faculty and staff, who can enjoy the same benefits as students by signing up with their respective SFSU emails. Students and Faculty will have a reliable system that ensures fast delivery times and affordable food.

7. High-Level System Architecture and Technologies Used:

1. Cloud Server
    - Amazon Web Services (AWS)
2. Operating System
    - Ubuntu 18.04 Server
      For class CTO: Upgraded from OS version 16.04. Requesting the change since the previous version is no longer supported.
3. Web Server
    - NGINX v1.14.0
4. Database
    - MySQL v5.7.41
5. Server-Side Language
    - JavaScript: Node v16.19.1, npm v8.19.3
6. Supported Browsers
    - Google Chrome: v111.0.5563.64, v110.0.5481.177
    - FireFox: v110.0, v109.0.1
7. External API's
    - TBD
      Considering: Google maps API or Amazon Location Service

For our application, we will implement a fuzzy search functionality similar to Amazon. This means that our application will match the closest terms to what the user types in the search bar and not enforce a strict search that requires 100% accuracy. This will help deal with the issue of misspelled words which could potentially break our application if not handled correctly. We will have one long search bar with a filter button for filtering results. This will open a drop down menu that will include different filter options like specific food categories, pricing options, and delivery times. This ensures that regardless of what the user types in the search bar, our application will display the information relevant to that search.

8. Team 5 Roles:

Team Member Name:	**Team Member Role:**
Hajime Miyazaki	Back End Lead
Scott Nguyen	GitHub Master
Mario Leyva Moreno	Front End Lead 1
Aneida Guadalupe Blanco Palacio	Front End Lead 2
Maliah Chin	Team Lead

9. Checklist:

Team found a time slot to meet outside of the class **DONE**
Back end, Front end leads and Github master chosen **DONE**
Team ready and able to use the chosen back and front end frameworks and those who need to learn are working on learning and practicing **DONE**
Team lead ensured that all team members read the final M1 and agree/understand it before submission **DONE**
GitHub organized as discussed in class (e.g. master branch, development branch, folder for milestone documents etc.) **DONE**

NEW: Use of any GenAI tool like ChatGPT: say if you used ChatGPT or like and how and for what segment of Milestone 1 (brief paragraph). As per class policy: this is allowed as a help BUT you cannot copy and paste its output and claim it is your own text, you need to put it in quotes or modify it, and only for short sentences You also are responsible for accuracy of your submission, so any ChatGPT content needs to be checked by you. – We did not use ChatGPT for Milestone 1

MileStone 2

SW Engineering CSC 648-848 Spring 2023

Hungry Gator Application

March 20, 2023

Milestone 2

Team 5

Student Name	**Email**
Maliah Chin	mchin4@sfsu.edu
Hajime Miyazaki	hmiyazaki@sfsu.edu
Mario Leyva Moreno	mleyvamoreno@mail.sfsu.edu
Scott Nguyen	snguyen23@mail.sfsu.edu
Aneida Guadalupe Blanco Palacio	ablancopalacio@mail.sfsu.edu

History Table

Date Submitted	**Date Revised**
3/31/2023	5/5/23

1. Executive Summary:

College can be stressful for many students. As they are constantly on the move getting ready for the next class or studying all night for exams, they may not have the time to go out and buy groceries to cook, or buy food from their local restaurant. To make life easier for SFSU students, staff, and faculty, we have created the Hungry Gator application which is designed to deliver food anywhere on campus. This application is especially convenient for SFSU faculty and staff, who are constantly working in their offices, attending meetings, and teaching multiple courses with minimal breaks. Our application will ensure that food is delivered directly to them at a specific location on campus, so that they may not be inconvenienced to go out and buy food when dealing with time constraints. Another key advantage of our application compared to other food delivery apps, is that we are constantly providing discount opportunities and minimal delivery fees in order to make food more affordable for college students. Another big advantage headed towards the SFSU Community is quick delivery times for those who need a quick bite between classes. Students often have to incur various expenses due to high tuition costs and materials required for various courses, so it is our goal to make an already expensive service as affordable as possible to members of SFSU. In addition to these features, our users will also have quick access to some of the best local and big restaurant chains in the area, and avoid dealing with delivery delays by receiving live updates on delivery times, which will help the user report any issues in a timely manner.

The Hungry Gator application provides an exclusive membership program to SFSU students, faculty, and staff, which will activate upon user registration. Users will receive exclusive discounts and reduced delivery fees on select restaurants, which will help make food more affordable and convenient for our users. Our application also supports on-site delivery, a service that ensures food is delivered anywhere on campus by providing a digital map of the campus to delivery drivers. This will help drivers pinpoint an exact location on campus and ensure fast delivery times. This also helps the user track their orders and avoid communication issues with drivers about being unable to find the dropoff location, since drivers will be guided directly to your specific location on the map. Our application is also unique in terms of what is available since our goal is to get food delivered as quickly as possible to our users. The application will not show restaurants out of range and will prioritize the restaurants that are closest to the SFSU campus.

Our team is composed of individuals from various backgrounds who have come together to deliver a reliable service to our users. Our team is composed of a team lead who organizes tasks and provides design plans for the team to follow. We also have a github master who keeps track of our product's github page and ensures that all aspects of our application are up to date and ready for launch. The back-end developer is responsible for authenticating user accounts and managing our team's database where user data is safely secured. Finally, we have front-end leads

who are responsible for providing a reliable user interface where users can easily interact with our application.

2. List of Main Data Items and Entities:

Entity:

1. Unregistered Users
    - Searching/browsing for restaurants and food items.
    - Able to register for an account.
2. SFSU Registered Users
    - Ordering food and applying discount codes.
    - Selecting dropoff location on any purchase.
    - Editing user profile.
    - Associated with one SFSU email.
3. Admin
    - Manages the application's database.
    - Approves posts and assists users with registering.
4. Restaurant Owner (Restaurant)
    - Registering their restaurant on the site and posting menu items and related information.
5. Delivery Driver
    - In charge of delivering food.

Main data items:

1. Restaurant
    - Contains the restaurant name, restaurant location, restaurant thumbnail, delivery time to SFSU and restaurant category.

      1.a Restaurant Location
        - Contains the address, city, state, zip code, and country where the restaurant is located.
2. Food Dish
    - Contains the name, price, and image of each dish relevant to a particular restaurant.
    - A short description of the dish (optional).
3. Order
    - An order is associated with one user.
    - Contains the order name (person who made the order), the address of the drop-off location, and the status of each order.
    - Orders can be approved and canceled by delivery drivers.
4. Account

- Account is associated with one user.
- For students, staff, and faculty, each account contains the first and last name of the user, a valid SFSU email, and password.

5. Discount or Promotion
    - Visible to any registered user.
    - Applicable to any order via a discount code.
6. Delivery Information
    - Contains the delivery address and name of the user who placed the order.
7. Driver Information
    - Contains the driver's name, email, phone number, vehicle type, driver's license number, and password.

3. Functional Requirements - prioritized:

- **Priority 1:**
    1. Unregistered Users
        1.1 Unregistered users shall be able to make an account.
        1.2 Unregistered users shall be able to search/browse for a restaurant and its menu items.
        1.3 Unregistered users shall be able to view search results.
        1.4 Unregistered users shall be able to filter results by category.

    2. SFSU Registered Users
        2.1 Registered users shall inherit all priority 1 unregistered user functionality.
        2.2 A registered user shall be associated with one account.
        2.3 A registered user shall be able to purchase food.
        2.4 A registered user shall be able to track their order.
        2.5 A registered user shall be able to select drop off location on campus.
        2.6 A registered user shall be able to sort restaurants based on delivery time.
        2.7 A registered user shall be able to log in.

    3. Admin
        3.1 Admin shall be responsible for verifying user accounts.
        3.2 Admin shall be required to approve all restaurant posts before they go live on the application.
        3.3 Admin shall be responsible for providing live feedback to users about incorrect account details.
        3.4 Admin shall be able to remove users.
        3.5 Admin shall be able to remove restaurants

    4. Restaurant Owner

4.1 A restaurant owner shall be able to register their restaurant on the app and wait up to 24 hours for admin approval.
4.2 A restaurant owner shall be able to post a menu of their dishes on the app for users to see.
4.3 A restaurant owner shall be able to register as a restaurant owner.
4.4 A restaurant owner shall be able to log in.

5. Delivery Driver
   5.1 A delivery driver shall be required to register to apply for the job.
   5.2 A delivery driver shall be able to access their delivery details.

- **Priority 2:**
  1. Unregistered Users
     1.1 An unregistered user shall have access to user ratings and reviews (read-only).

  2. Registered Users
     2.1 A registered user shall inherit all priority 2 unregistered user functionality.
     2.2 A registered user shall be able to see discount deals on select restaurants.
     2.3 A registered user shall be able to write reviews on restaurants.
     2.4 A registered user shall be able to rate their food or delivery service.

  3. Admin
     3.1 The admin shall be responsible for managing customer reviews and messages.

  4. Discounts or Promotions
     5.1 Student discounts shall be applied to students with a valid student ID.
     5.2 All other discounts/promotions shall be applied to anyone with a verified SFSU email.

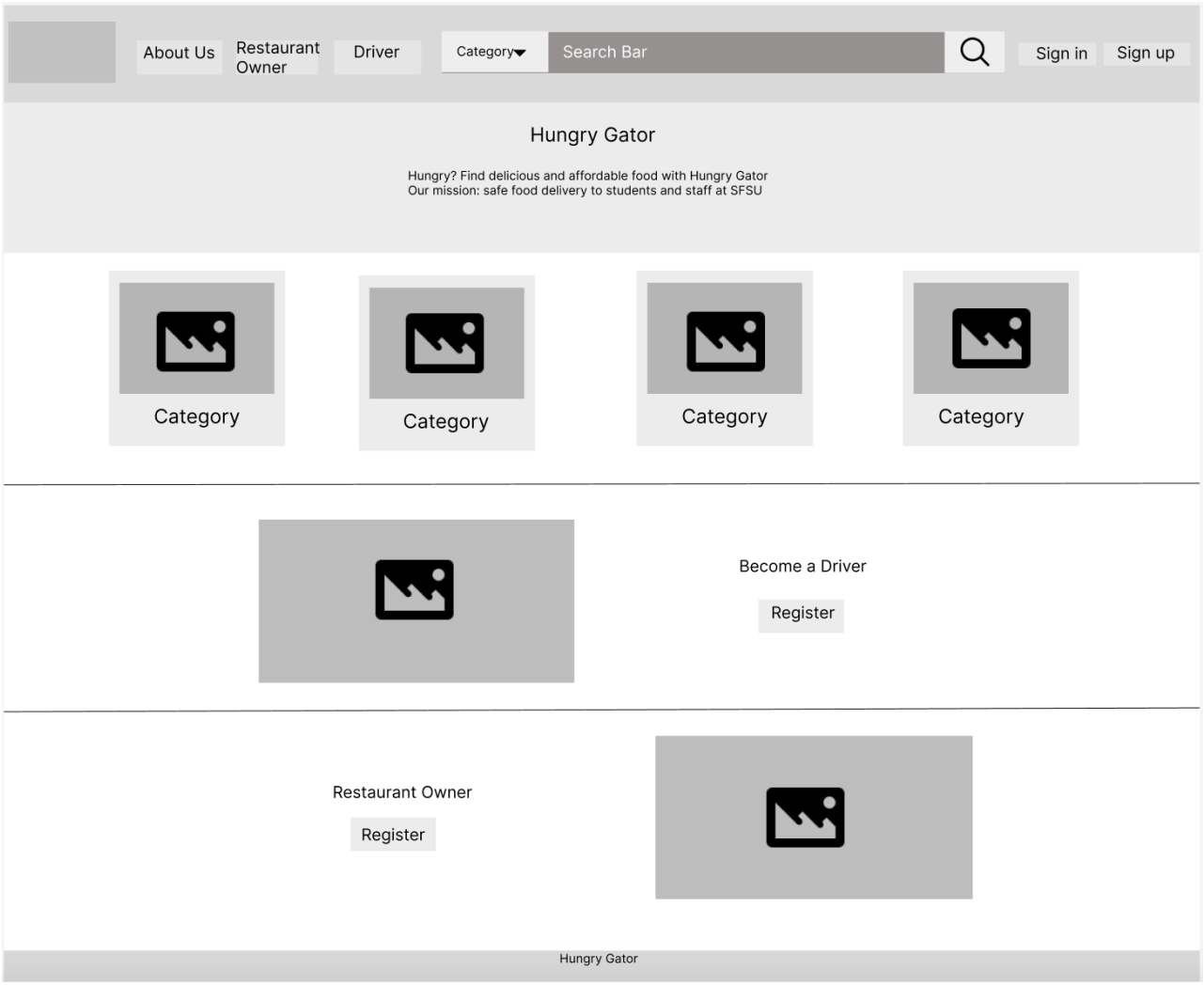
- **Priority 3:**
  1. Registered Users
     1.1 A registered user shall be able to see their order history.
     1.2 A registered user shall be able to edit their user profile.
     1.3 Registered users shall be able to add money to their gator cards for purchasing (conceptual idea).

4. UI Storyboards for each main use case:

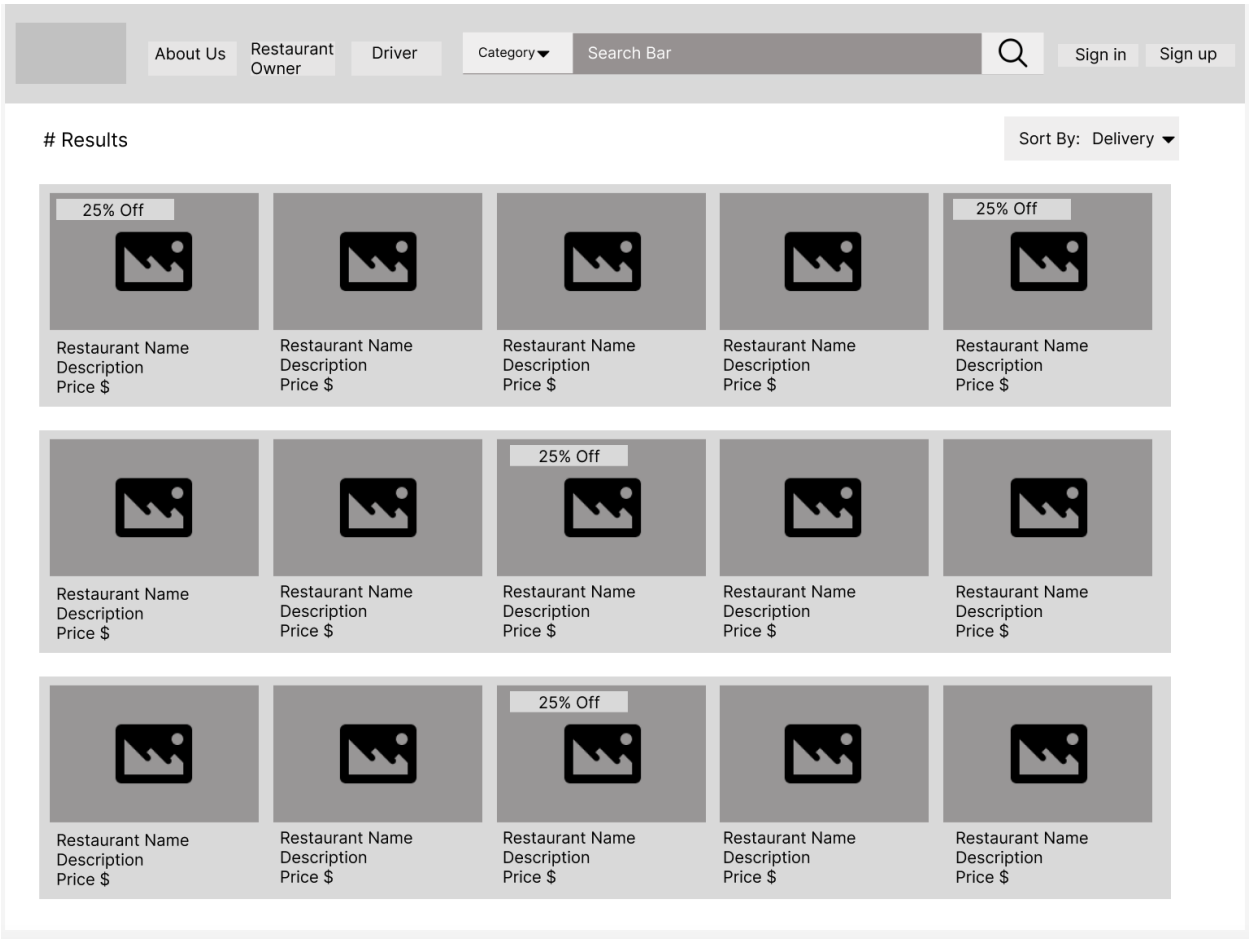
1. Student's Late Night Cravings, Satisfied.

Jasmine is a SFSU student who has been busy all evening studying hard for her midterms. As the night progresses, Jasmine realizes that it's time to eat but she fears she doesn't have enough time to go pickup anything for dinner. Luckily, she heard about this new web application, Hungry Gators, that has been created specifically for SFSU students. **(1)** Jasmine opens her laptop and enters the link for Hungry Gators. **(2)** She browses through the categories noticing all of the appetizing pictures that inspire her to find a meal for the evening. Since she is on a budget, she sorts by price and browses through the list of Italian restaurants. She finds something yummy and even notices that there is a special student discount. **(3)** Jasmine adds her food to her order and continues to check out. **(4)** Prior to placing her order, she is taken to the sign up page where she is prompted to sign up using her SFSU email. **(5)** Since she is a student, the chosen discount has been applied. She notices that there is a special function that allows for her food to get delivered directly to her dorm, so she enters her location. Jasmine then gets back to her books and waits for the speedy delivery.

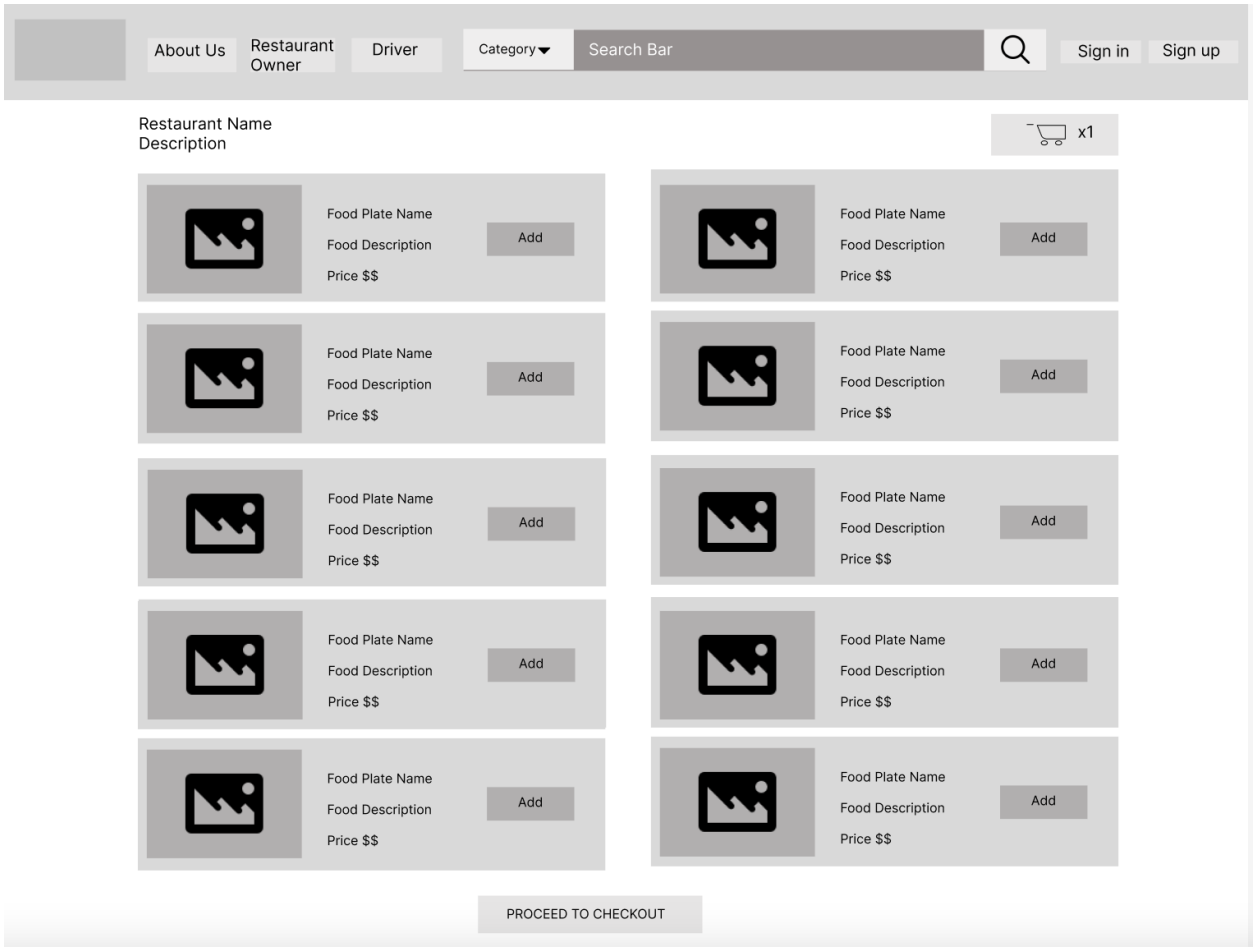
(1)


About Us
Restaurant Owner
Driver
Category
Search Bar
Sign in
Sign up
Hungry Gator
Hungry? Find delicious and affordable food with Hungry Gator
Our mission: safe food delivery to students and staff at SFSU
Category
Category
Category
Category
Become a Driver
Register
Restaurant Owner
Register
Hungry Gator

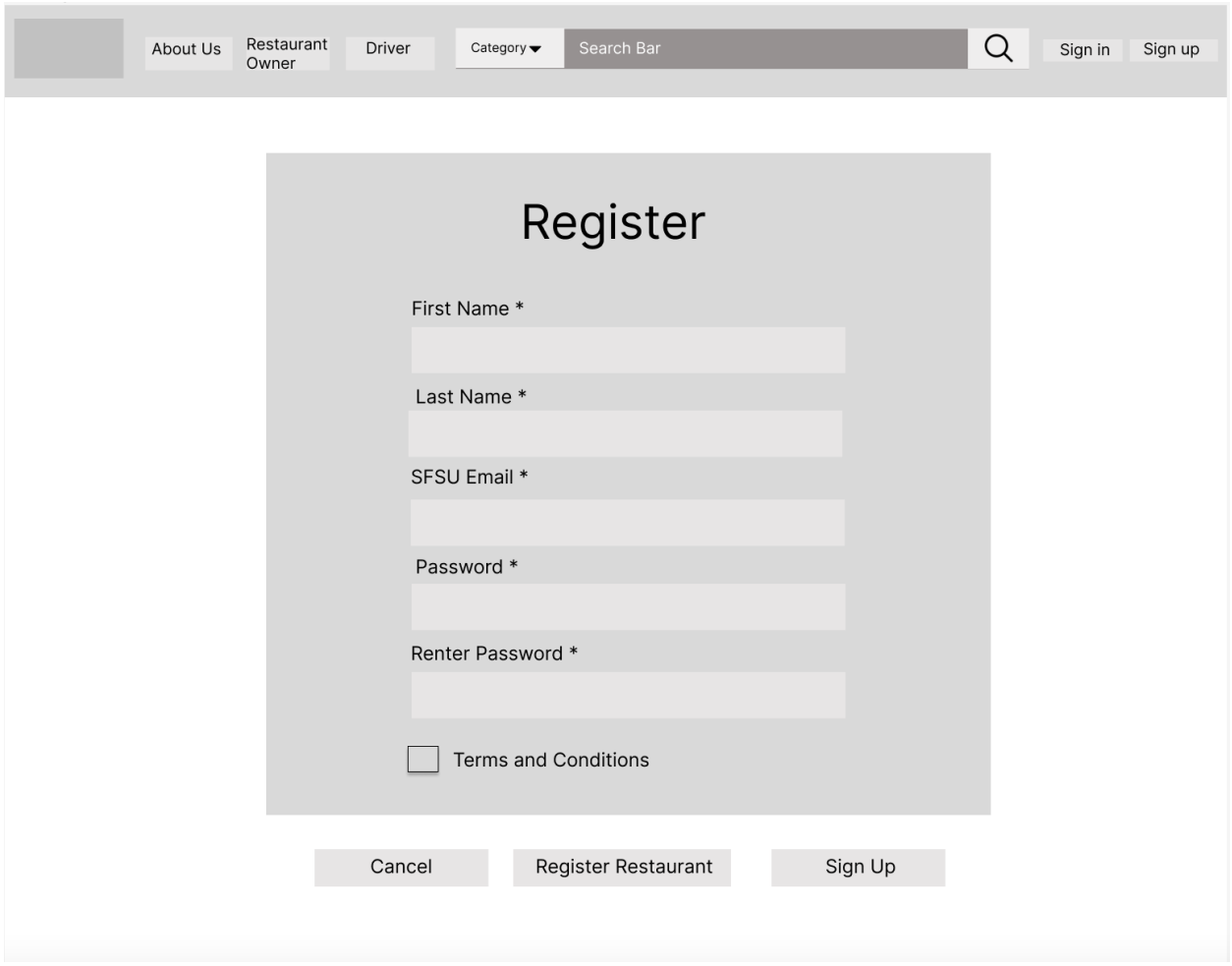
(2)


About Us
Restaurant Owner
Driver
Category
Search Bar
Sign in
Sign up
Results
Sort By: Delivery
25% Off
Restaurant Name
Description
Price $
25% Off
Restaurant Name
Description
Price $
Restaurant Name
Description
Price $
Restaurant Name
Description
Price $
25% Off
Restaurant Name
Description
Price $
Restaurant Name
Description
Price $
Restaurant Name
Description
Price $
25% Off
Restaurant Name
Description
Price $
Restaurant Name
Description
Price $
Restaurant Name
Description
Price $
Restaurant Name
Description
Price $
25% Off
Restaurant Name
Description
Price $
Restaurant Name
Description
Price $
Restaurant Name
Description
Price $
Restaurant Name
Description
Price $
Restaurant Name
Description
Price $

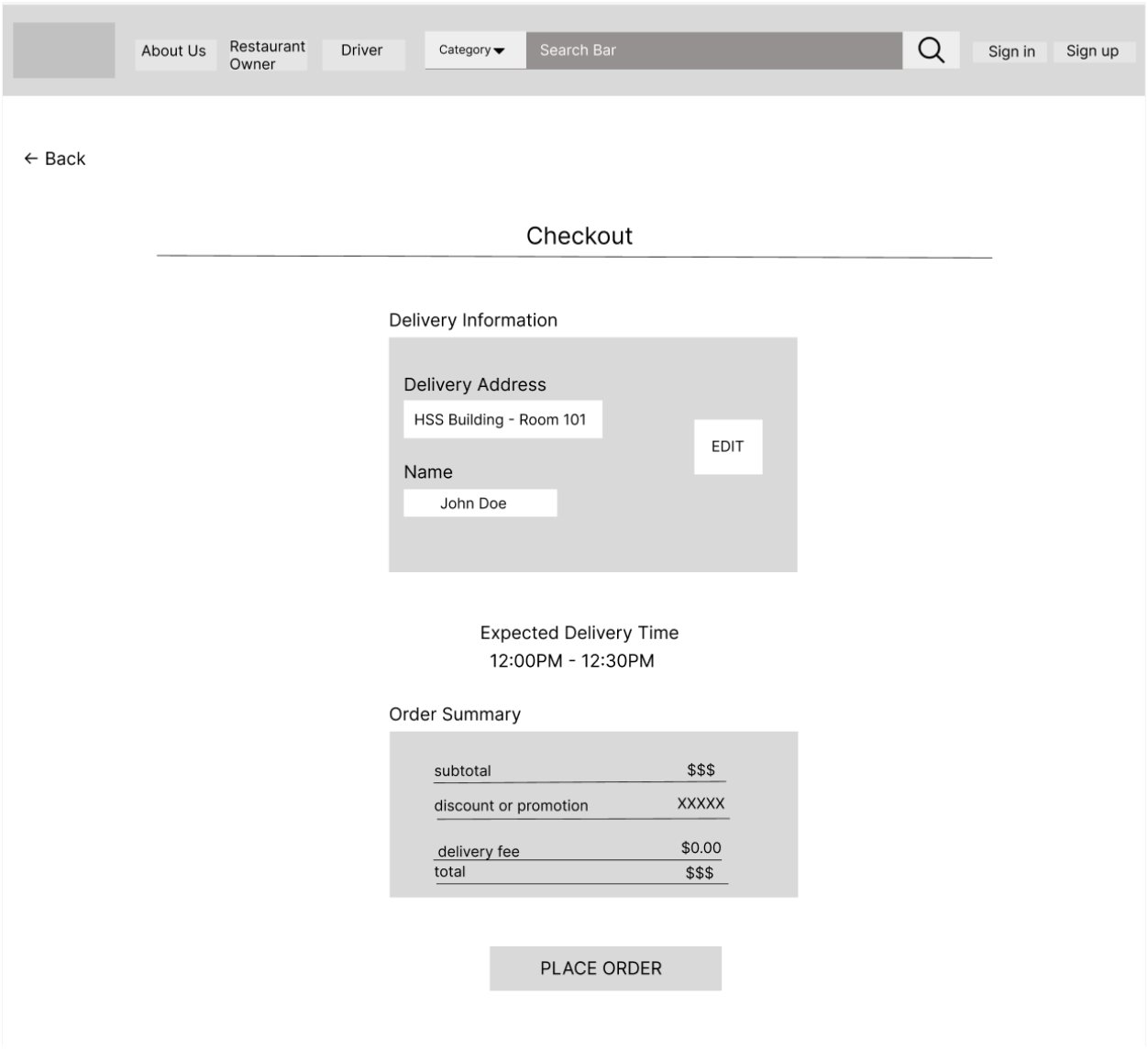
(3)


About Us
Restaurant Owner
Driver
Category
Search Bar
Sign in
Sign up
Restaurant Name
Description
x1
Food Plate Name
Food Description
Price $$
Add
PROCEED TO CHECKOUT

(4)


About Us
Restaurant Owner
Driver
Category
Search Bar
Sign in
Sign up
Register
First Name *
Last Name *
SFSU Email *
Password *
Renter Password *
Terms and Conditions
Cancel
Register Restaurant
Sign Up

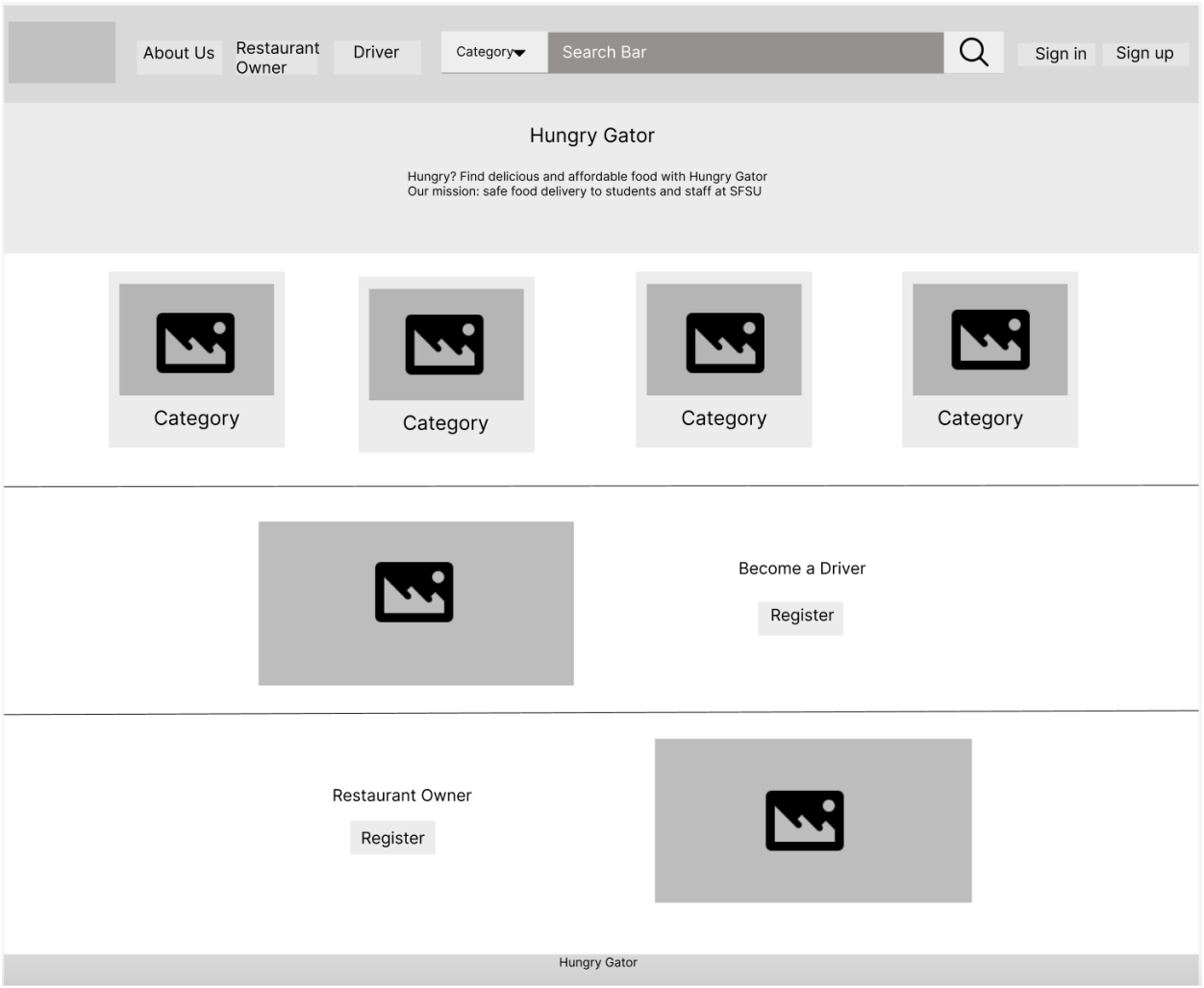
(5)


About Us
Restaurant Owner
Driver
Category
Search Bar
Sign in
Sign up
← Back
Checkout
Delivery Information
Delivery Address
HSS Building - Room 101
EDIT
Name
John Doe
Expected Delivery Time
12:00PM - 12:30PM
Order Summary
subtotal $$$
discount or promotion XXXXX
delivery fee $0.00
total $$$
PLACE ORDER

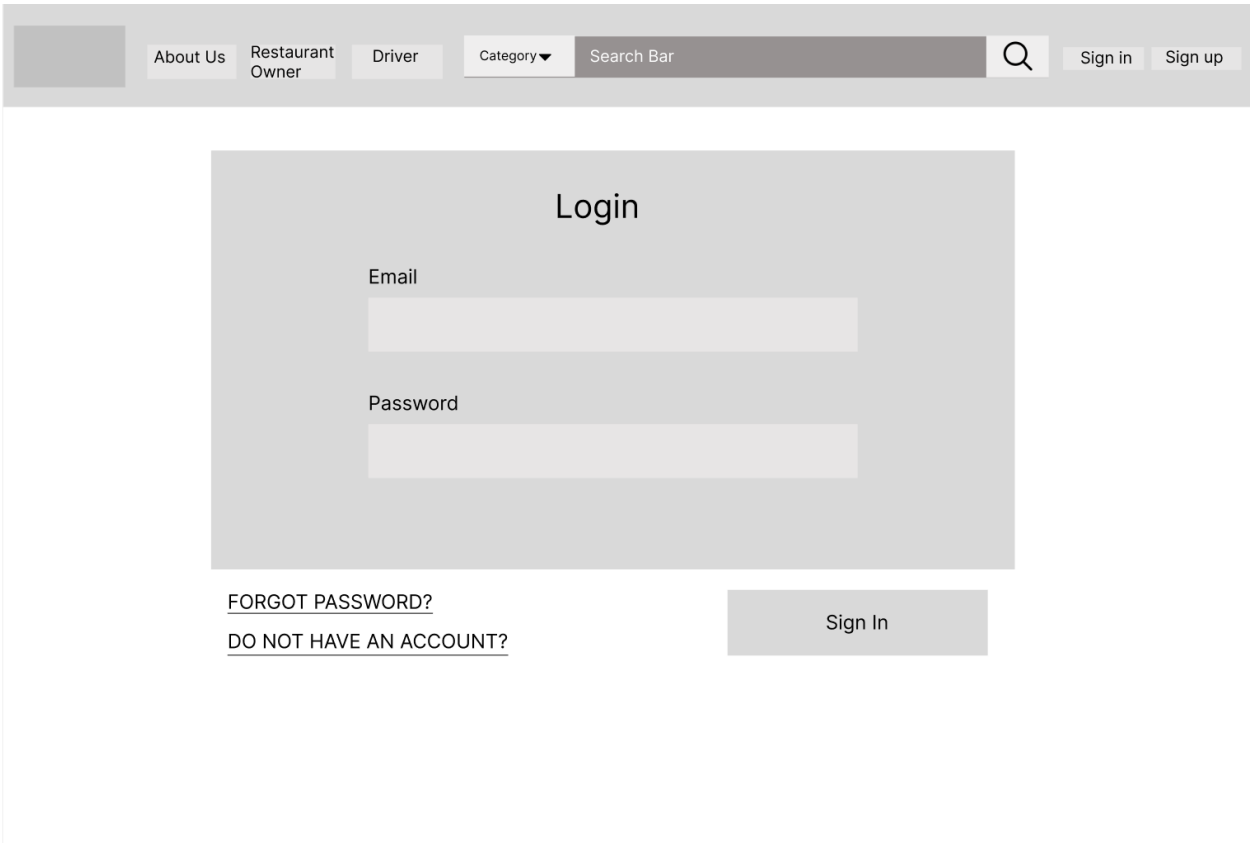
2. Professor/Staff Orders a Quick Bite:

Manuel is an undergraduate advisor and professor at SFSU. His responsibilities as an undergraduate advisor include meeting with students at specific times and assisting in the overall academic progress of students at SFSU. In addition to his staff responsibilities, he is also teaching several courses throughout the day. Due to his busy schedule, he finds it nearly impossible to enjoy his lunch and doesn't always prepare his own meals for the day. After hearing about the Hungry Gator application through one of his colleagues, Manuel decided to download the application and create an account. Hesitant to use the application for the first time despite having an account, he decided to give it a try due to a busy afternoon where he had very little time to go out and buy food himself. **(1)** He signed in using the login page and **(2)** browsed through the list of Indian restaurants. Due to time constraints, he sorted the results by delivery time and was delighted to find his favorite local Indian restaurant. **(3)** He finds his favorite dish and proceeds to the checkout page. **(4)** At checkout, he fills out the delivery information and was even surprised by the option to have it delivered directly to his office. Manuel was happy with his purchase and was even given a membership discount for being registered with an SFSU email.

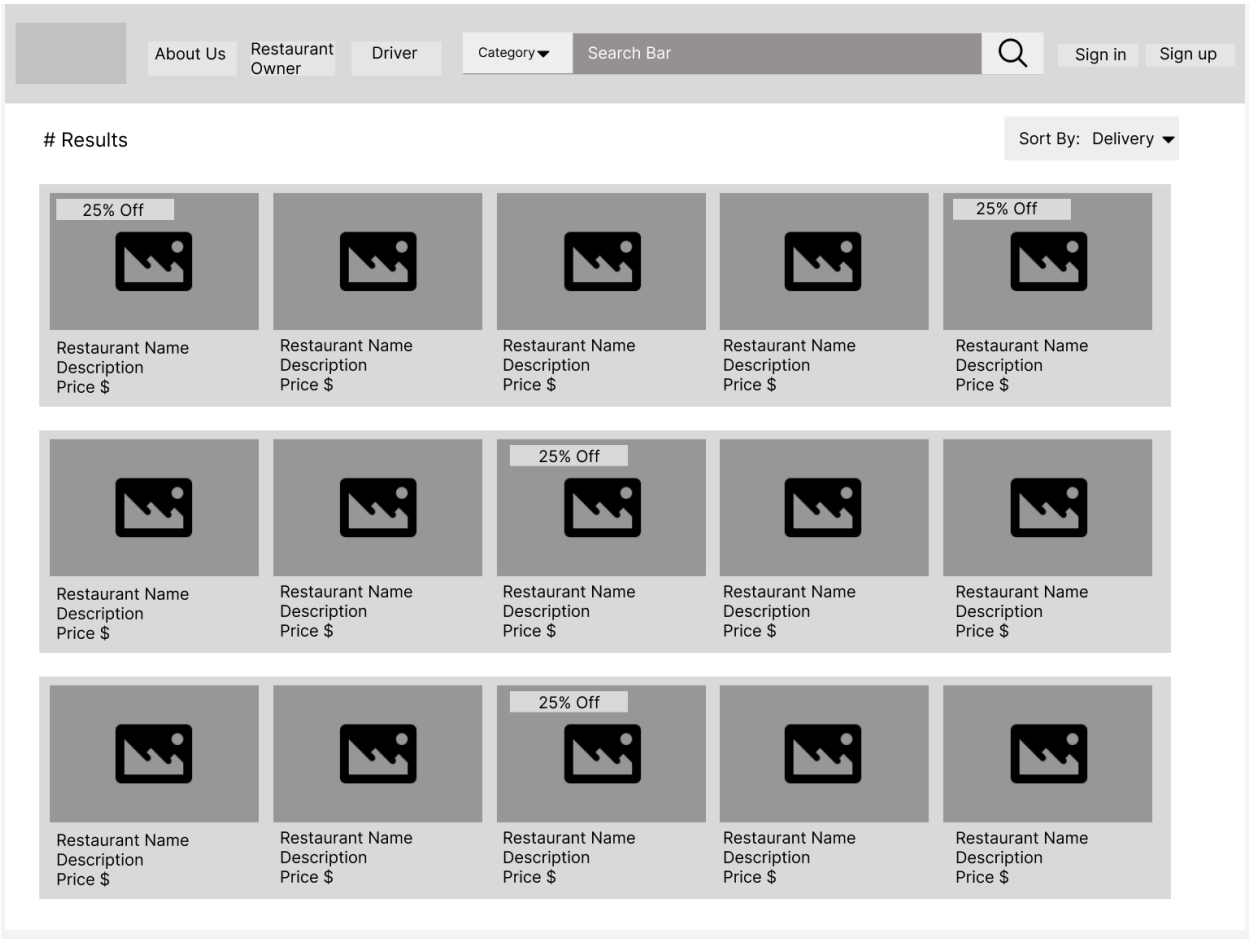
(1)


About Us
Restaurant Owner
Driver
Category
Search Bar
Sign in
Sign up
Hungry Gator
Hungry? Find delicious and affordable food with Hungry Gator
Our mission: safe food delivery to students and staff at SFSU
Category
Category
Category
Category
Become a Driver
Register
Restaurant Owner
Register
Hungry Gator

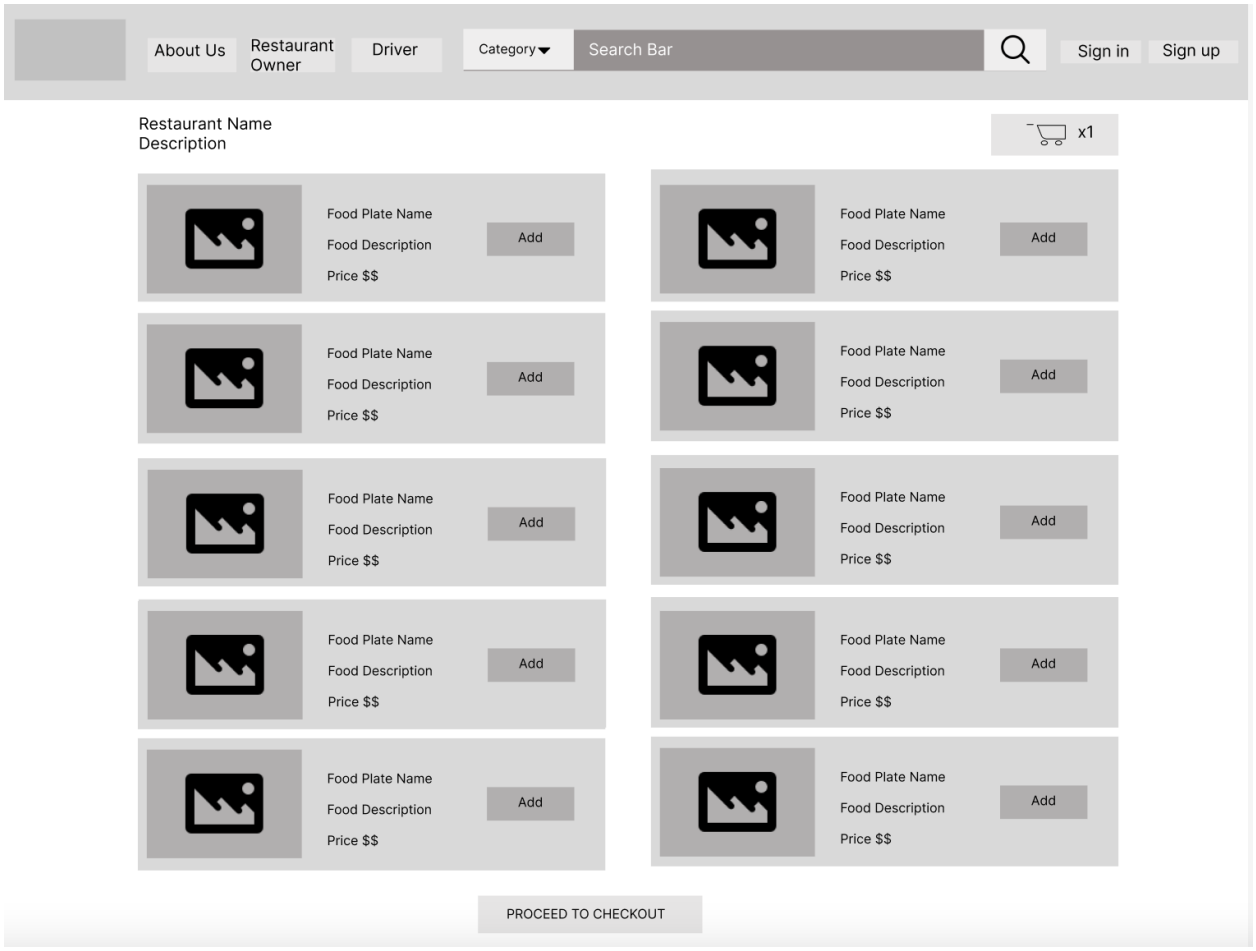
(2)


About Us
Restaurant Owner
Driver
Category
Search Bar
Sign in
Sign up
Login
Email
Password
FORGOT PASSWORD?
DO NOT HAVE AN ACCOUNT?
Sign In

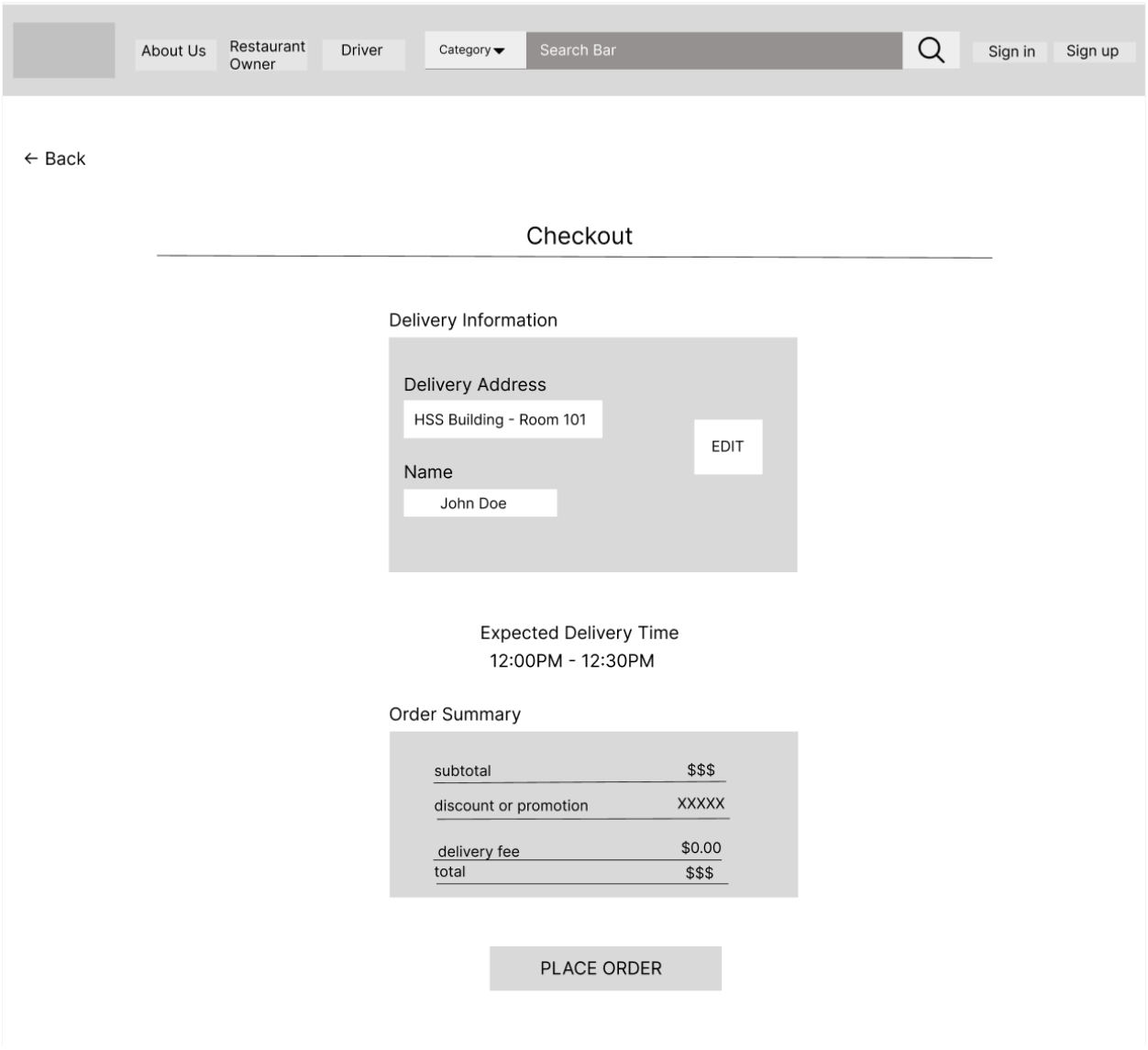
(3)


About Us
Restaurant Owner
Driver
Category
Search Bar
Sign in
Sign up
Results
Sort By: Delivery
25% Off
Restaurant Name
Description
Price $

(4)


About Us
Restaurant Owner
Driver
Category
Search Bar
Sign in
Sign up
Restaurant Name
Description
x1
Food Plate Name
Food Description
Price $$
Add
PROCEED TO CHECKOUT

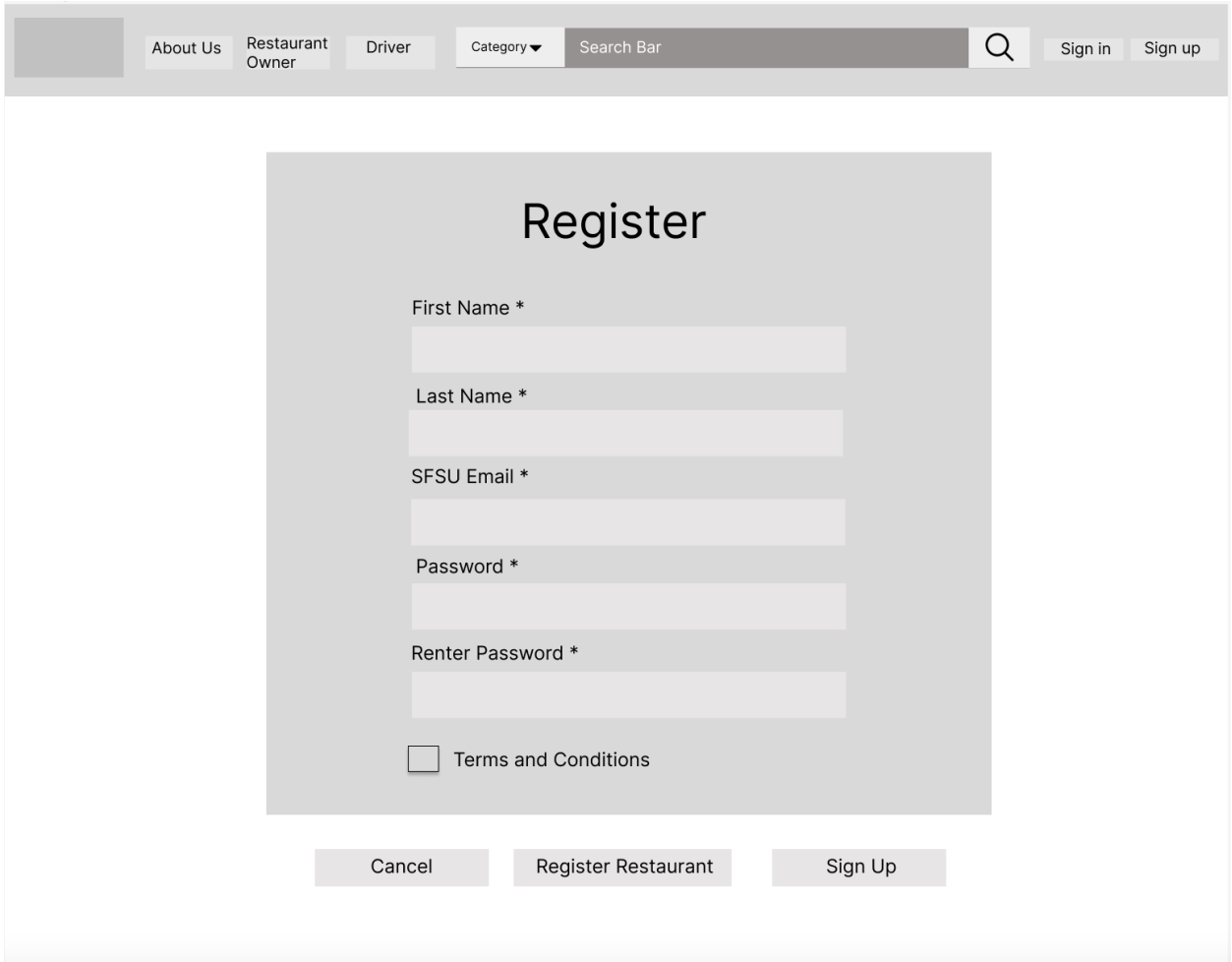
(5)


About Us
Restaurant Owner
Driver
Category
Search Bar
Sign in
Sign up
← Back
Checkout
Delivery Information
Delivery Address
HSS Building - Room 101
EDIT
Name
John Doe
Expected Delivery Time
12:00PM - 12:30PM
Order Summary
subtotal $$$
discount or promotion XXXXX
delivery fee $0.00
total $$$
PLACE ORDER

3. Restaurant Owner Obtains New Marketing Idea:

Tina is a local restaurant owner who works near SFSU. She has a cafe that has opened up recently and has been looking for a new opportunity to expand her business. A new web app has come across her radar that has peaked her interest, called the Hungry Gator. Tina's full schedule makes it difficult for her to learn new apps, but the simple layout of Hungry Gator offers her a quick way to grow her audience. **(1)** She opens Hungry Gator and looks for the registration button for restaurants. **(2)** She clicks on the button and is greeted with a form that asks for her basic information. Because Tina knows that the SFSU campus is bustling with business, she fills out the section for special offers aimed towards the students. She is also excited to upload photos that showcase her best snacks. Tina then completes the form and submits the request. A message that the form has been submitted and that it may take up to 24 hours for the form to be accepted appears and Tina acknowledges the information. Happy to have a chance to give back to students and create a new branch of business, Tina waits for her approval.

(1)



(2)

Restaurant Details

Restaurant Name *

Restaurant Name

Restaurant Address *

Restaurant Street Address

Street Address*

City*

San Francisco

State/Region*

CA

Country*

U.S.

Zip Code*

90000

Special Offers Checklist (optional)

Free Delivery

Weekly Discounts and Promotions

Restaurant Thumbnail *

Upload Image:

BROWSE

Requests submitted will be approved within 24 hours*

Menu

Category

Food Plate Name

Food Description

Price $$

Edit

Add

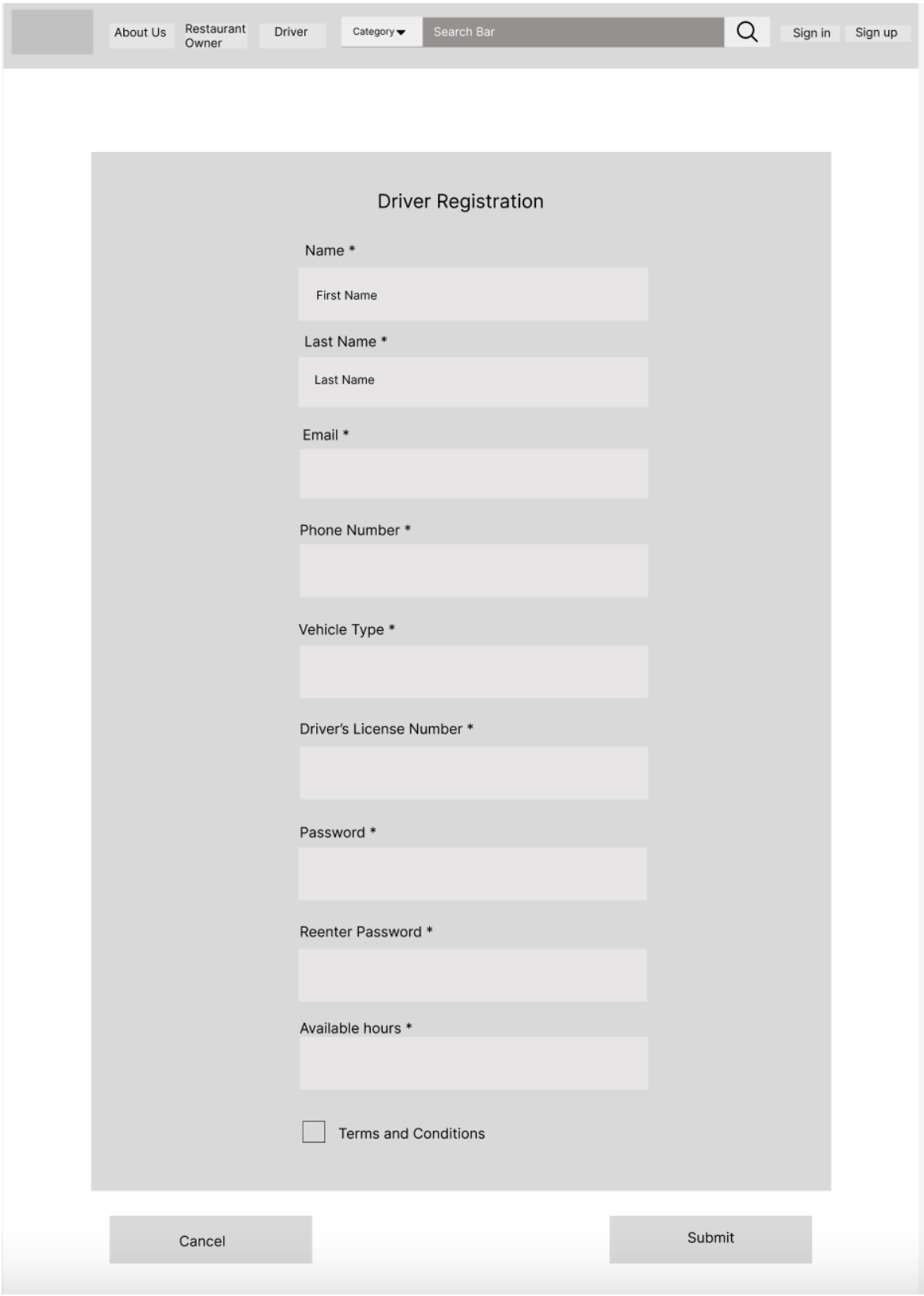
CANCEL

SUBMIT REQUEST

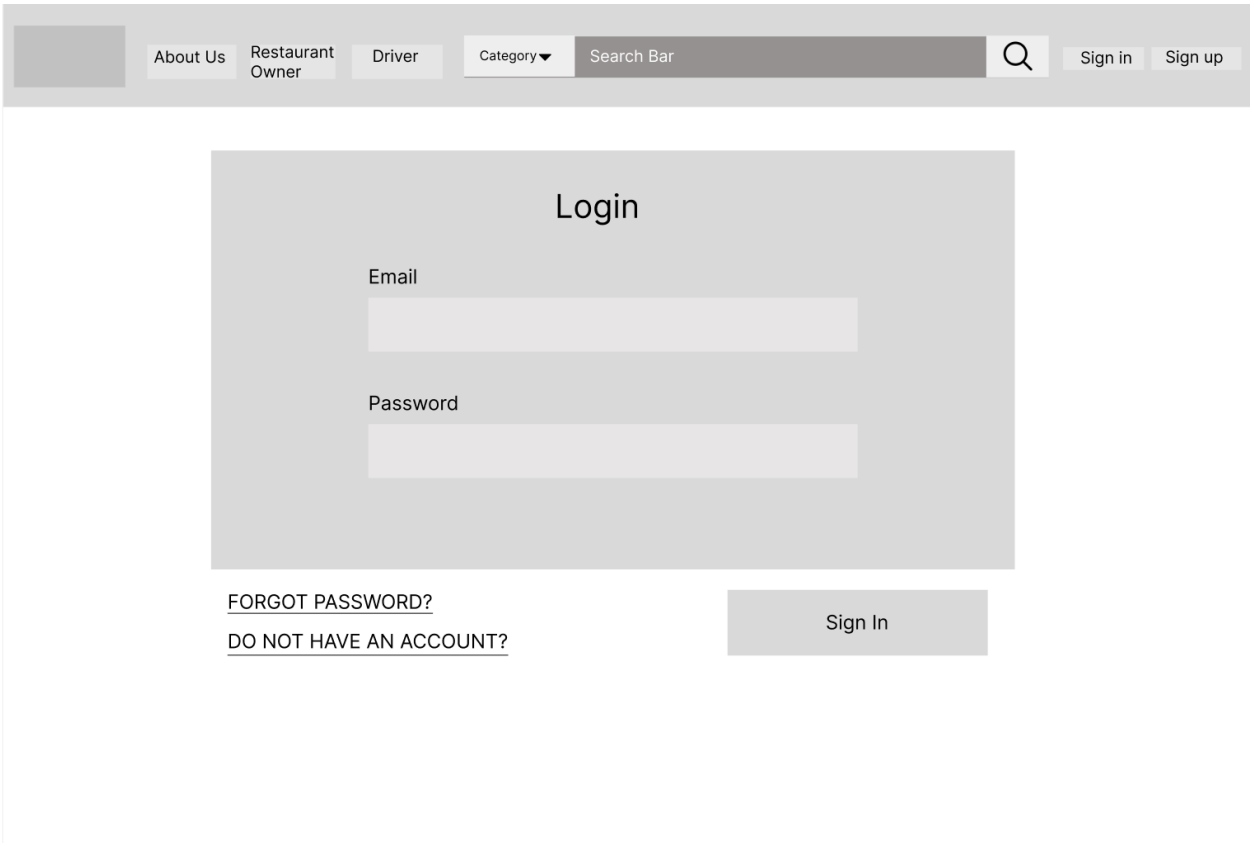
4. Delivery Driver Gets Orders:

Lucas is an SFSU student who wants a job that is flexible with his schedule as a student. Friends convinced him to apply for Hungry Gator. He opens a browser in his laptop and enters the link for the website. **(1)** He finds the sign up page for delivery drivers and begins filling out the application. After being approved for the job, **(2)** he signs in. **(3)** He is then shown various orders that instantly pop up in his phone and he can decide which one to take. He notices that there is a map provided for on campus deliveries in the order information, which will prevent him from getting lost. Being able to decide when to work gives him peace of mind about having time to do his assignments and attending classes. Besides being able to fit in his job at Hungry Gator in his busy schedule, Lucas feels safe making deliveries since the app gives him the choice of meeting up at a secure point on campus.

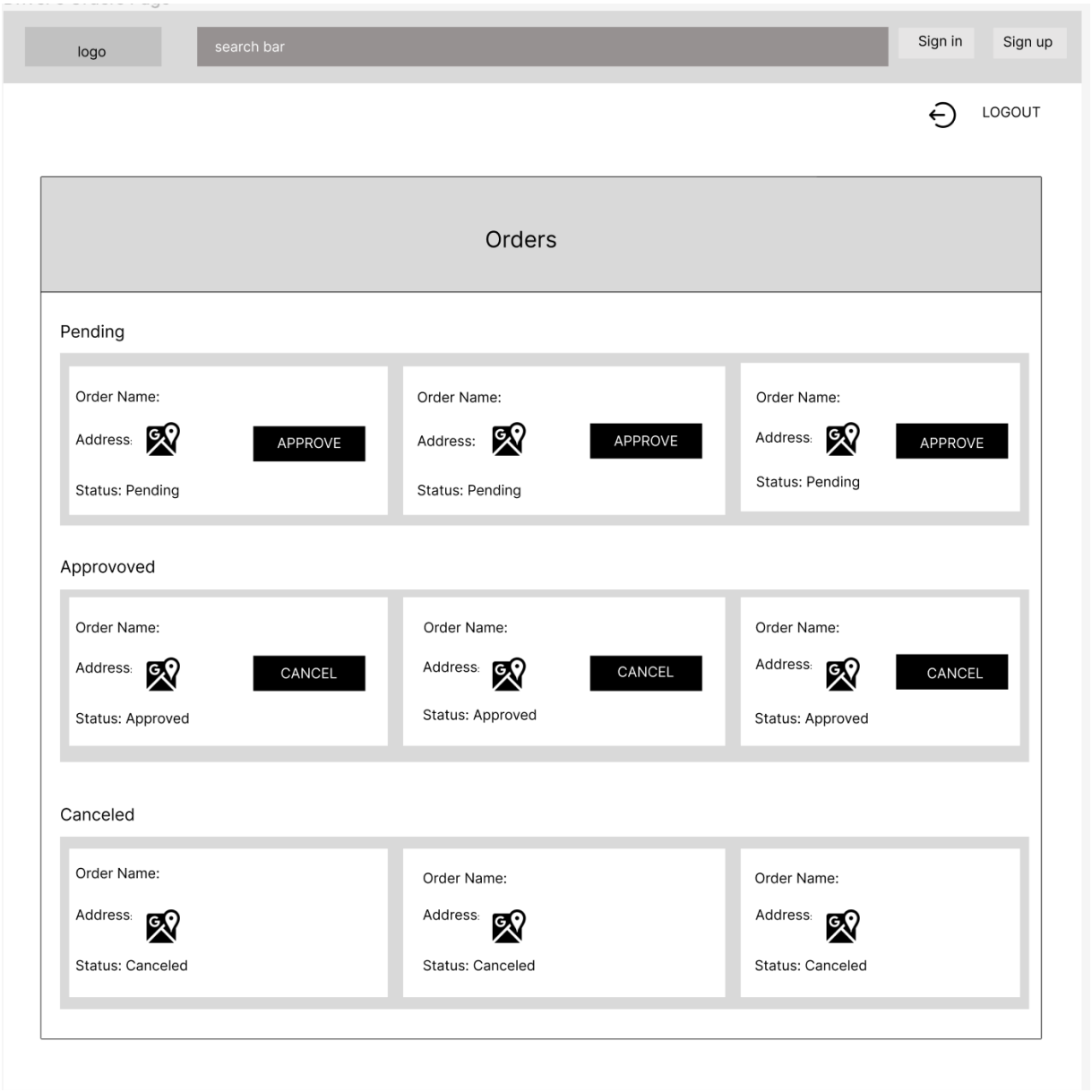
(1)


About Us
Restaurant Owner
Driver
Category
Search Bar
Sign in
Sign up
Driver Registration
Name *
First Name
Last Name *
Last Name
Email *
Phone Number *
Vehicle Type *
Driver's License Number *
Password *
Reenter Password *
Available hours *
Terms and Conditions
Cancel
Submit

(2)


About Us
Restaurant Owner
Driver
Category
Search Bar
Sign in
Sign up
Login
Email
Password
FORGOT PASSWORD?
DO NOT HAVE AN ACCOUNT?
Sign In

(3)


logo
search bar
Sign in
Sign up
LOGOUT
Orders
Pending
Order Name:
Address:
APPROVE
Status: Pending
Order Name:
Address:
APPROVE
Status: Pending
Order Name:
Address:
APPROVE
Status: Pending
Approvoved
Order Name:
Address:
CANCEL
Status: Approved
Order Name:
Address:
CANCEL
Status: Approved
Order Name:
Address:
CANCEL
Status: Approved
Canceled
Order Name:
Address:
Status: Canceled
Order Name:
Address:
Status: Canceled
Order Name:
Address:
Status: Canceled

5. High level Architecture, Database Organization summary only:

- DB Organization:
    1. Account
        - Account ID
        - First Name
        - Last Name
        - Email
        - Password
    2. Restaurant
        - Restaurant ID
        - Restaurant Name
        - Restaurant Address
        - Restaurant Thumbnail (pointer to BLOB)
        - Restaurant Category
        - Restaurant Estimated Delivery Time
        - Restaurant Estimated Price Range
    3. Food Dish
        - Food Dish ID
        - Food Name
        - Food Price
        - Food Image (pointer to BLOB)
        - Food Description (optional)
    4. Order
        - Order ID
        - Delivery Address
        - Order Status
    5. Driver Info
        - Account ID
        - Name
        - Email
        - Phone Number
        - Vehicle Type
        - License ID
        - Password
    6. Media
        - Media ID
        - Title
        - Category

- Media Storage:
  We will store any needed images, video, and audio in DB BLOBS. This may prove to be the best option for storing large chunks of data that we can store and reference within the database when needed. For example, we may want to include images of restaurants and food dishes, but we cannot include the raw data in the tables. Therefore, we can reference a media object that contains a pointer to the BLOB in order to view the requested data. For media we will use common formats like MP4, which can handle both video and audio types, and PNG for images. For GPS, we will use the most common formats like Decimal Degrees (DD) to keep track of the coordinates.

- Search/filter architecture and implementation:
  For searching, we will use SQL to organize items for the user using a sorting algorithm that will show results based on how recently each item was created in the database. This is easier to maintain and understand since we know the behavior of search results and how the results will be displayed to the user. In order to implement this, we need to reference one or more columns from the Restaurant table to sort these items based on the user search. We will do this using the LIKE operator in MySQL to match items to a specific set of characters that form a pattern for searching. For example, if the user is interested in searching for a specific category of restaurants like Italian, we may want to use the restaurant category and the LIKE operator to select the items in the restaurant table that match the term ‘Italian%’. This will return all relevant results and can be used in tandem with other SQL to sort/filter items in a specific way (i.e., by price or delivery time).

- Some non-trivial algorithms to implement may include a rating/ranking system where the user is able to sort results by rating.

6. Risks:

Skill risks: Team has limited knowledge of Bootstrap and BLOBs, will outsource to appropriate resources to acquire the skill.

Team is new to API for google maps, outsourcing to appropriate resources and learning how to use API's will help us acquire the skill.

Schedule risks: Team is made of students with heavy schedules which can compromise lower priority tasks due to approaching finals season. In order to resolve the issue, we may have to reduce our scope.

Technical risks: Virtual machine and connections to server has proven to be a difficult task, messaging the CTO and understanding the problem will help us solve this risk.

Teamwork risks: Team needs to have more frequent meetings to discuss project goals and any issues that may arise to resolve them in a timely manner. To resolve this, we will use our preferred method of communication to set up meetings that work for all team members.

7. Project Management:

In the milestones accomplished so far, including milestone 2, the group had a set list of responsibilities that corresponded to their role in the team. If an issue were to surface, our communication to the appropriate outlet has proven to be a great source of resolving the problems. For further revisions of the project as a whole as well as this document, we plan to keep this milestone as a living document using the revisions table as a log of our edits. This document will serve as a template and resource to look back on, in order to keep track of our priorities and our progress. Our next step is to work on our Vertical SW Prototype. As we move forward, we plan to use Bootstrap, study the google map API's, and learn about our database to server connections. To ensure the team is on the same page, we will attend weekly meetings. We also plan to have a day to do a code walkthrough where each end of the SW development team explains their code and how to use it. To reinforce our accessibility of the product, we also plan to have members of the team test the important functions. This allows us to have a fresh set of eyes on what we can use to improve user experience. By expanding our skill set and implanting the lessons learned in class, we will be able to efficiently manage the upcoming milestones.

MileStone 3

SW Engineering CSC 648-848 Spring 2023

Hungry Gator Application

April 27, 2023

Milestone 3
Team 5

Student Name	**Email**
Maliah Chin	mchin4@sfsu.edu
Hajime Miyazaki	hmiyazaki@sfsu.edu
Mario Leyva Moreno	mleyvamoreno@mail.sfsu.edu
Scott Nguyen	snguyen23@mail.sfsu.edu
Aneida Guadalupe Blanco Palacio	ablancopalacio@mail.sfsu.edu

History Table

Date Submitted	**Date Revised**
4/27/2023	5/21/23

Milestone 3: Summary

Team Number: 05
Meeting Date: 4/26/30 1800

Our Priority 1 tasks have been reduced to meet the minimal viable product for release on the final day of class. This document contains a summary of feedback as well as what our current priorities are. Below is a list of what we need to do in order to meet our goals.

UI PAGE FEEDBACK:

Restaurant Details page:
Make the text field smaller for descriptions.
One column for all forms and text entries
Enter time to deliver, should be selectable
Add a category pull down to chose instead of typing in the category (to avoid typos)
*** = mandatory field (include this message on the top portion of the forms pages)

Drivers:
Show my orders -> goes to order page where they can pick which order to deliver

Checkout pages:
Create and include new pages for checkout functions
Requirements:
Order, total price, check out
*Use stacks for responsive design!
*Use flags instead of a cart functions to get the user's order requests

CODE REVIEW FEEDBACK:

Include headers in each file
Create a persistent search bar/ typing search
Create the order/checkout function for registered users (Use flags for ordering — avoid cart)

GITHUB FEEDBACK:

Create more detailed comments , a little more depth, have the comments be more specific

DATABASE FEEDBACK:

Create a way for Admin to approve/deny restaurant display requests: Admin on workbench, create a new column that sets the display as false automatically and when the admin changes it to true, we can display the restaurant
Have unregistered users create accounts: Allow for users to be entered and saved into the database
Pull down for restaurant categories on restaurant registration: On the registration form, be sure to have the restaurant chose their category from a **drop down menu** so we can make it more consistent with database
Delivery time selection: Add time to deliver for each restaurant on restaurant registration form
password should be encrypted
Restaurant key is foreign key to category : keep up with the consistency of the database, all of the keys should stem from one place
Fix path names in database for the images in Restaurant table (architecture slides on path names relative paths instead of absolute/ protect repo as one of the assets)

FRONT END FEEDBACK:

Restaurant owners to post food items/descriptions/photos: update menus upload form
Fix the format of the nav bar on the home screen + about us pages! (picture includes below of mockup)
Make the logo bigger on home screen : Allows for the user to see it better
Pull down for restaurant categories on restaurant registration so we can make it more consistent with database → for restaurant registration (instead of typing the category)
Include time for delivery on the time cards : Our most sought after feature is the delivery time so if possible we can sort the time cards based on delivery time
Repeat name of restaurant on menu page : When we display the menu, be sure to include which restaurant the menu is coming from
If there is no results for the users search include options: if there are 0 results for user search, ask them “Try other options” and display other menus!
** during display change 2 results found to 2 results found out of (total restaurants)

FEEDBACK ON TEAMWORK/RISK MANAGEMENT:

Teamwork is improving. The team is more cohesive and mostly responsive.

Risk Management:
Technical skills: Database skills are improving, but still minor issues prevail and may impact projects admin requirement functionality. To solve this issue, we are reaching out to CTO and other sources.

We have done an architecture review to check that developers adhere to MVC pattern, coding style, and minimal agreed documentation.

P1 List of Features:

1. Unregistered Users
    1.1 Unregistered users shall be able to make an account.
    1.2 Unregistered users shall be able to search/browse for a restaurant and its menu items.
    1.3 Unregistered users shall be able to view search results.
    1.4 Unregistered users shall be able to filter results by category.

2. SFSU Registered Users
    2.1 Registered users shall inherit all priority 1 unregistered user functionality.
    2.2 A registered user shall be associated with one account.
    2.3 A registered user shall be able to purchase food.
    2.4 A registered user shall be able to select drop off location on campus.
    2.5 A registered user shall be able to sort restaurants based on delivery time.
    2.6 A registered user shall be able to log in.

3. Admin
    3.1 Admin shall be required to approve all restaurant posts before they go live on the application.
    3.2 Admin shall be able to remove users.
    3.3 Admin shall be able to remove restaurants

4. Restaurant Owner
    4.1 A restaurant owner shall be able to register their restaurant on the app and wait up to 24 hours for admin approval.
    4.2 A restaurant owner shall be able to post a menu of their dishes on the app for users to see.

4.3 A restaurant owner shall be able to register as a restaurant owner.
4.4 A restaurant owner shall be able to log in.

5. Delivery Driver
    5.1 A delivery driver shall be required to register to apply for the job.
    5.2 A delivery driver shall be able to access their delivery details.

MileStone 4

SW Engineering CSC 648-848 Spring 2023

Hungry Gator Application

May 21, 2023

Milestone 4

Team 5

Student Name	Email
Maliah Chin	mchin4@sfsu.edu
Hajime Miyazaki	hmiyazaki@sfsu.edu
Mario Leyva Moreno	mleyvamoreno@mail.sfsu.edu
Scott Nguyen	snguyen23@mail.sfsu.edu
Aneida Guadalupe Blanco Palacio	ablancopalacio@mail.sfsu.edu

History Table

Date Submitted	Date Revised
5/21/2023	

1. Product Summary:

The Hungry Gator Application

The Hungry Gator application provides an exclusive membership program to SFSU students, faculty, and staff, which will activate upon user registration. Users will receive exclusive discounts and reduced delivery fees on select restaurants, which will help make food more affordable and convenient for our users. Our application also supports on-site delivery, a service that ensures food is delivered anywhere on campus by providing a digital map of the campus to delivery drivers. This will help drivers pinpoint an exact location on campus and ensure fast delivery times. This also helps the user track their orders and avoid communication issues with drivers about being unable to find the dropoff location, since drivers will be guided directly to your specific location on a featured map of the SFSU Campus. Our users will also have quick access to some of the best local and big restaurant chains in the area. Our application is unique in terms of what is available since our goal is to get food delivered as quickly and conveniently as possible to our users at an affordable price.

P1 List of Features:

1. Unregistered Users
    - 1.1 Unregistered users shall be able to make an account.
    - 1.2 Unregistered users shall be able to search/browse for a restaurant and its menu items.
    - 1.3 Unregistered users shall be able to view search results.
    - 1.4 Unregistered users shall be able to filter results by category.

2. SFSU Registered Users
    - 2.1 Registered users shall inherit all priority 1 unregistered user functionality.
    - 2.2 A registered user shall be associated with one account.
    - 2.3 A registered user shall be able to purchase food.
    - 2.4 A registered user shall be able to select drop off location on campus.
    - 2.5 A registered user shall be able to sort restaurants based on delivery time.
    - 2.6 A registered user shall be able to log in.

3. Admin
    - 3.1 Admin shall be required to approve all restaurant posts before they go live on the application.
    - 3.2 Admin shall be able to remove users.

3.3 Admin shall be able to remove restaurants

4. Restaurant Owner
    4.1 A restaurant owner shall be able to register their restaurant on the app and wait up to 24 hours for admin approval.
    4.2 A restaurant owner shall be able to post a menu of their dishes on the app for users to see.
    4.3 A restaurant owner shall be able to register as a restaurant owner.
    4.4 A restaurant owner shall be able to log in.

5. Delivery Driver
    5.1 A delivery driver shall be required to register to apply for the job.
    5.2 A delivery driver shall be able to access their delivery details.

URL:
http://ec2-54-219-129-242.us-west-1.compute.amazonaws.com:3000

2. Usability Test Plan: Search Function

Test Objective: In this test, we will be prompting the user to test the functionality of the search function on our application. In order to ensure that our search bar contains accurate results and is easy to use, we have decided to create a usability test plan. Our goal is to get the user to demonstrate the effectiveness, efficiency and overall satisfaction of this feature. Then we will implement their feedback to create a better experience.

Test Background and Setup:
System Setup: We are using AWS servers to host our application. On our end, we would run the server and retrieve our url. The user would then run the URL in their browser.

Starting Point: The starting point of this test would be the home page of our application. They would begin at the home page and find their way to our search function and to their results.
Intender Users: The intended users vary from age to age, but overall our application is meant for the entire SFSU community.
URL: ec2-54-219-129-242.us-west-1.compute.amazonaws.com
What We Are Measuring: We will be measuring efficiency, effectiveness and overall user satisfaction with the search function operations and placement.

Usability Task Description:
The task provided is described below with the intention of testing our search function to find nearby restaurants.

The Task:
"Search for Italian food"
Application State:
Home page, search bar empty
Successful completion criteria:
User is presented with Italian Restaurants as the result of inputting category in search bar
Benchmark:
completed in 45 seconds

Plan for Evaluation of Effectiveness:
To measure our user's effectiveness when performing the task of searching for a category of food, we plan to use the successful completion criteria section of our table. The table is provided under the Usability Task Description and contains certain categories to serve as a baseline of what the testers are looking for to assess whether or not the task is completed. In this case, the successful completion criteria provides the output expected if our search function is to work as intended. Given that the usr reports back with the state of their task and the output they were able to achieve, we would compare that and the successful completion criteria and rate whether or not

the task was completed successfully. If there are enough results that match the successful completion criteria, we would say that the task given scored high in effectiveness.

Plan for Evaluation of Efficiency:

To measure the efficiency of the task, we will be using a similar method as our evaluation of effectiveness. In our Task Description Section, we provide a Benchmark criteria in our table that is measured in seconds/minutes. In order to see if our application's usability is efficient, we set our standard to 45 seconds to find the search bar, look up the specific category, and retrieve the results. We plan to give about a 5 second padding for slow typers and maybe users who are not as tech savvy because our application is meant to reach a broader range audience.

Plan for Evaluation of User Satisfaction:

We will be using the Likert Scale to test and measure the user's satisfaction with this function. Below we have created our own questions based off of the Likert Scale requirements. We plan to pass out the questionnaires when the user is done with the task.

*Mark the box you feel is most accurate

	Strongly Disagree	**Disagree**	**Neutral**	**Agree**	**Strongly Agree**
Searching for restaurants is efficient					
In real application, you would use the search function					
The process of using the search function is easy and intuitive					

3. QA Test Plan: Search Function

Test objectives: Test is to check if SW performs to specs. This test gives instructions to testers. These instructions are prespecified on what actions to test and the result of what those actions bring. In this case it would be testing the search field and category field. By giving instructions to testers we are able to check functionality of the search function whether the results match and reporting pass or fail of function. The pass and fail are necessary to find bugs within our search function.

HW and SW setup:

System Setup: We are using AWS servers to host our application. On our end, we would run the server and retrieve our url. The user would then run the URL in their browser.

URL: ec2-54-219-129-242.us-west-1.compute.amazonaws.com

Feature to be tested: search field and category field

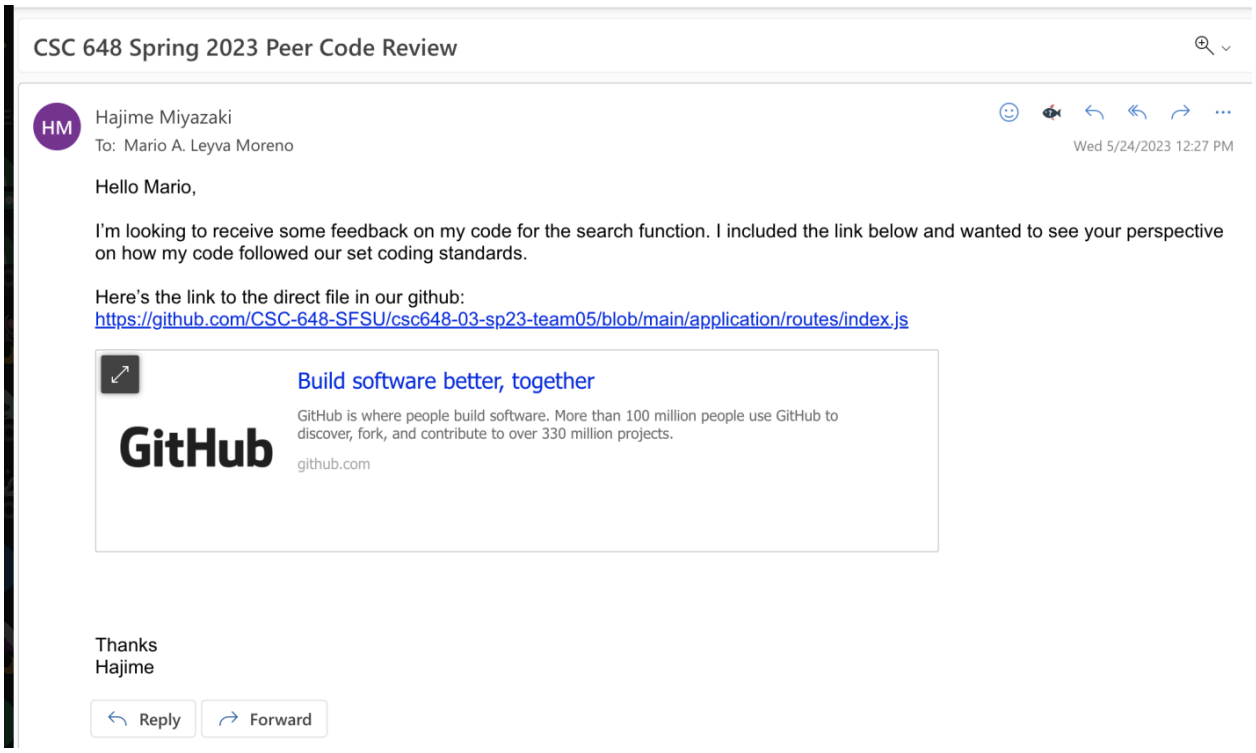
QA Test plan:

Test #	Title	Description	Input	Expected correct output	Server 1 Results (Pass/Fail)	Server 2 Results (Pass/Fail)
1	Search category	Test % like In search for name field	Enter “Italian” In search field	Check that you get 1 results, all containing string “italian” in name field	PASS	PASS
2	Category	CATEGORY Testing category selection field	Select “Asian” in category field	Get 2 results, all containing string “Asian” in name field	PASS	PASS
3	Auto suggestion	Test % like Testing auto suggestion	Enter “blue” in search	Get 7 results, non containing string “blue” in	FAIL	FAIL

			field	name field		
4	Max input	Test % like Testing max range of search	Enter “aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa” in search field	Get 7 results	PASS	PASS
5	Matching food and category	CATEGORY, and Test % like Testing search field and category field	Enter “tacos” in search field , Select “Mexican “ in category field	Get 2 results, all containing “tacos” or “Mexican” in name field	PASS	PASS
6	Stays in search field	Test % like Testing search field	Enter “hamburger” In search field	Search field contains “hamburger”	PASS	PASS
7	Non alphabetical input	Test % like Testing non alphabetical characters	Enter “*#+” In the search field	Get 7 results	FAIL	FAIL
8	Category selection stays	CATEGORY Testing search field	Select “American” in category field	Category field contains “American”	FAIL	FAIL

4. Peer Code Review: Hajime and Mario

Here are our screenshots of a portion of how we implemented code review.



M

Mario A. Leyva Moreno

To: Hajime Miyazaki

Wed 5/24/2023 2:27 PM

Search - Peer Review.pdf
456 KB

Hey Hajime,

After reviewing the code, I have a summary of how to modify your code to fit the coding standards a bit more. I would suggest commenting on the purpose of the code more often. I noticed you had the headers on the file which was nice to see. The variable names were also good and read well as plain English. I would suggest making your github comments more descriptive on what exactly was done in the commit. Also, the code could be better optimized for performance and clarity. Other than that, the code looks good! Here is the google doc containing my feedback:

Thanks, Mario

Reply Forward

Peer review on search functionality:

Header Comments:

- Good job on adding header comments, but ensure that it is relevant to the code that you are writing. Current header describes mysql database connection setup, but the code is related to the application's search functionality.

```
/*************************************************************
* Author: Mario Leyva Moreno and Hajime Miyazaki
*
* File: database.js
*
* Description: The purpose of this file is to set up a connection to our mysql database.
*
*************************************************************/
```

Naming Conventions:

- Excellent naming conventions. Variables are clear and in plain English so as not to cause confusion. The names clearly specify which values are used in the search.

```
75   function search(req, res, next) {
76       let searchTerm = req.query.search;
77       let category = req.query.category;
78       let filter = req.query.filter;
79       let query = 'SELECT * FROM Restaurant';
80
```

Additional comments on coding style and comments:

- Need to add descriptive comments on what the code is doing. Describe what information the query is retrieving and how these values are queried and stored in the .then block. Describe how errors would be handled in case of incorrect search information.

- Code is not very optimized. Avoid repetition of code and hardcoding by optimizing this code to execute the query only after all the information has been retrieved.

```
81      if (searchTerm != '' && category != '') {
82          query = `SELECT * FROM Restaurant WHERE restaurant_category = ? AND restaurant_name LIKE ?`;
83          db.execute(query, [category, `%${searchTerm}%`])
84              .then(([result, fields]) => {
85                  req.searchResult = result;
86                  req.searchTerm = searchTerm;
87                  req.category = category;
88                  next();
89              })
90              .catch(err => {
91                  req.searchResult = [];
92                  req.searchTerm = '';
93                  req.category = '';
94                  console.error(err);
95                  next();
96              });
97      } else if (searchTerm != '' && category == '') {
98          query = `SELECT * FROM Restaurant WHERE restaurant_name LIKE ?`;
99          db.execute(query, [`%${searchTerm}%`])
100             .then(([result, fields]) => {
101                 req.searchResult = result;
102                 req.searchTerm = searchTerm;
103                 req.category = category;
104                 next();
105             })
106             .catch(err => {
107                 req.searchResult = [];
108                 req.searchTerm = '';
109                 req.category = '';
110                 console.error(err);
111                 next();
```

- As mentioned previously, avoid repetition of code for better optimization and add comments so others can follow along with what the code is doing.

```
113         } else if (searchTerm == '' && category != '') {
114             query = `SELECT * FROM Restaurant WHERE restaurant_category = ?`;
115             db.execute(query, [category])
116                 .then(([result, fields]) => {
117                     req.searchResult = result;
118                     req.searchTerm = searchTerm;
119                     req.category = category;
120                     next();
121                 })
122                 .catch(err => {
123                     req.searchResult = [];
124                     req.searchTerm = '';
125                     req.category = '';
126                     console.error(err);
127                     next();
128                 });
129         } else {
130             db.execute(query)
131                 .then(([result, fields]) => {
132                     req.searchResult = result;
133                     req.searchTerm = searchTerm;
134                     req.category = category;
135                     next();
136                 })
137                 .catch(err => {
138                     req.searchResult = [];
139                     req.searchTerm = '';
140                     req.category = '';
141                     console.error(err);
142                     next();
143                 });
144         }
145     }
```

- Overall, the search works mostly as expected and correctly returns information based on the queries.

5. Self-Check on Best Practices for Security

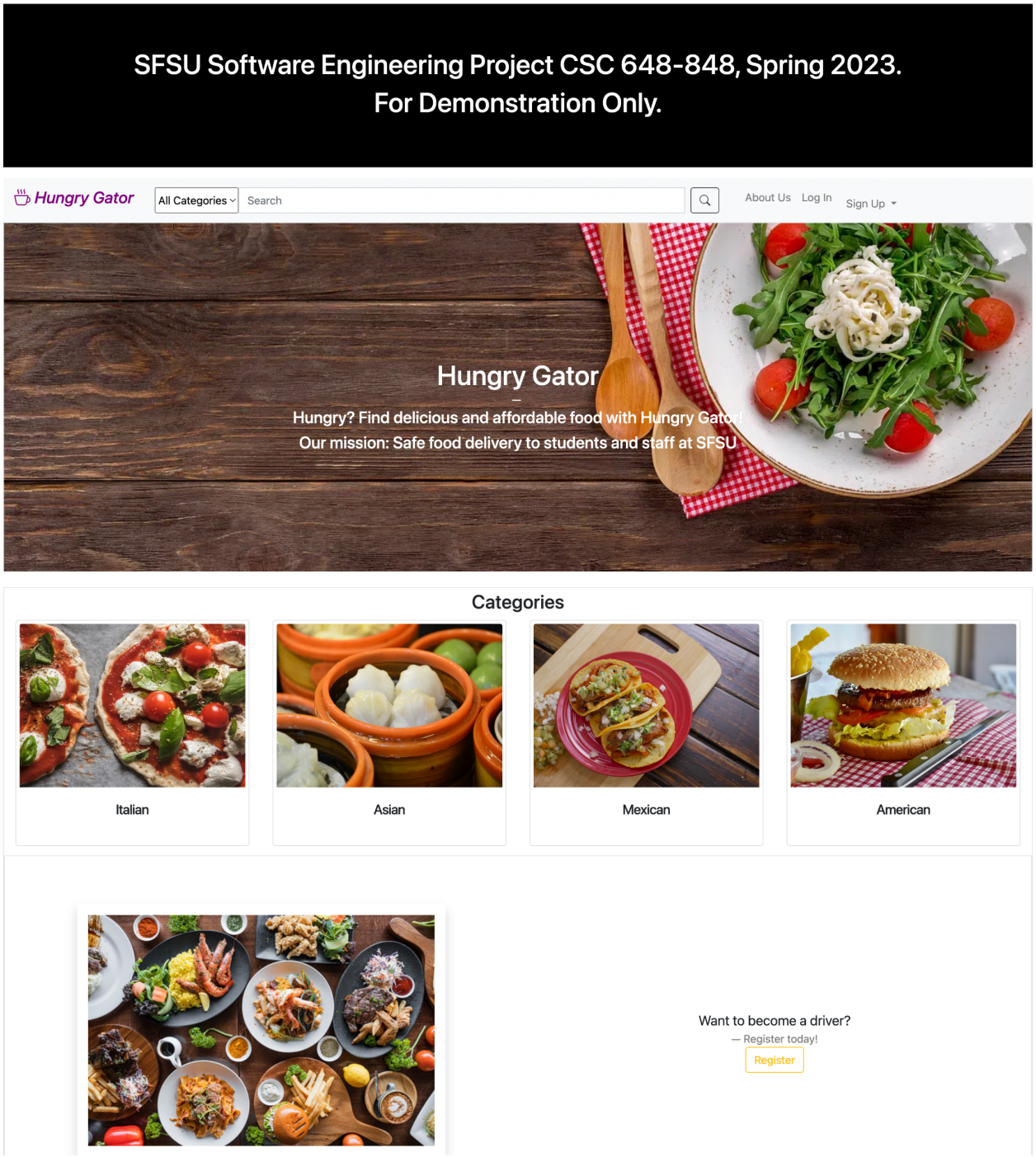
- The types of major assets we are protecting include user data for registration such as first name, last name, email, address, state, zip code. Depending on the user, if they are an SFSU user then the SFSU email is protected
- Authentication data such as usernames and passwords
- Passwords in the database are encrypted
- Input data validation includes:
  SFSU student email registration to have “sfsu.edu” in their email
  Agreeing to the terms and conditions
  Passwords to contain the following: 1 number, 1 uppercase letter, and 1 special character (- + ! @ # $ ^ & *)
- For driver registration, register their vehicle, valid CA ID, working hours
- For restaurant registration, register their city, ZIP code, image of the restaurant, etc.

Assets to be protected	**Types of possible/expected atks**	**Strategy to protect the asset**
User passwords	Unauthorized access, malware	passwords in the database are encrypted
Authentication data	Unauthorized access	server side validation to include
Server	Data breach, SQL injection	limit on the search bar to 40 alpha-numeric characters

6. Self-Check on Adherence to Original Non-Functional Specs

1. Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0 **DONE**
2. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers **DONE**
3. All or selected application functions shall render well on mobile devices **ON TRACK, works okay**
4. Data shall be stored in the database on the team's deployment server. **DONE**
5. No more than 50 concurrent users shall be accessing the application at any time **DONE**
6. Privacy of users shall be protected**DONE**
7. The language used shall be English (no localization needed) **DONE**
8. Application shall be very easy to use and intuitive **DONE**
9. Application shall follow established architecture patterns **DONE**
10. Application code and its repository shall be easy to inspect and maintain **DONE**
11. Google analytics shall be used ISSUE: did not do it
12. No email clients shall be allowed. Interested users can only message to sellers via in-site messaging. One round of messaging (from user to seller) is enough for this application **DONE**
13. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI. **DONE**
14. Site security: basic best practices shall be applied (as covered in the class) for main data items **DONE**
15. Media formats shall be standard as used in the market today **DONE**
16. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development **DONE**
17. The application UI (WWW and mobile) shall prominently display the following exact text on all pages "SFSU Software Engineering Project CSC 648-848, Spring 2023. For Demonstration Only" at the top of the WWW page nav bar. (Important so as to not confuse this with a real application). **DONE**

4) Product ScreenShots



Restaurant owner?

— Register your restaurant today!

Register

Hungry Gator

SFSU Software Engineering Project CSC 648-848, Spring 2023.
For Demonstration Only.

Hungry Gator | All Categories | Search | About Us | Log In | Sign Up

Login

User Role:

Username

Username

Password

Password

Forgot password?

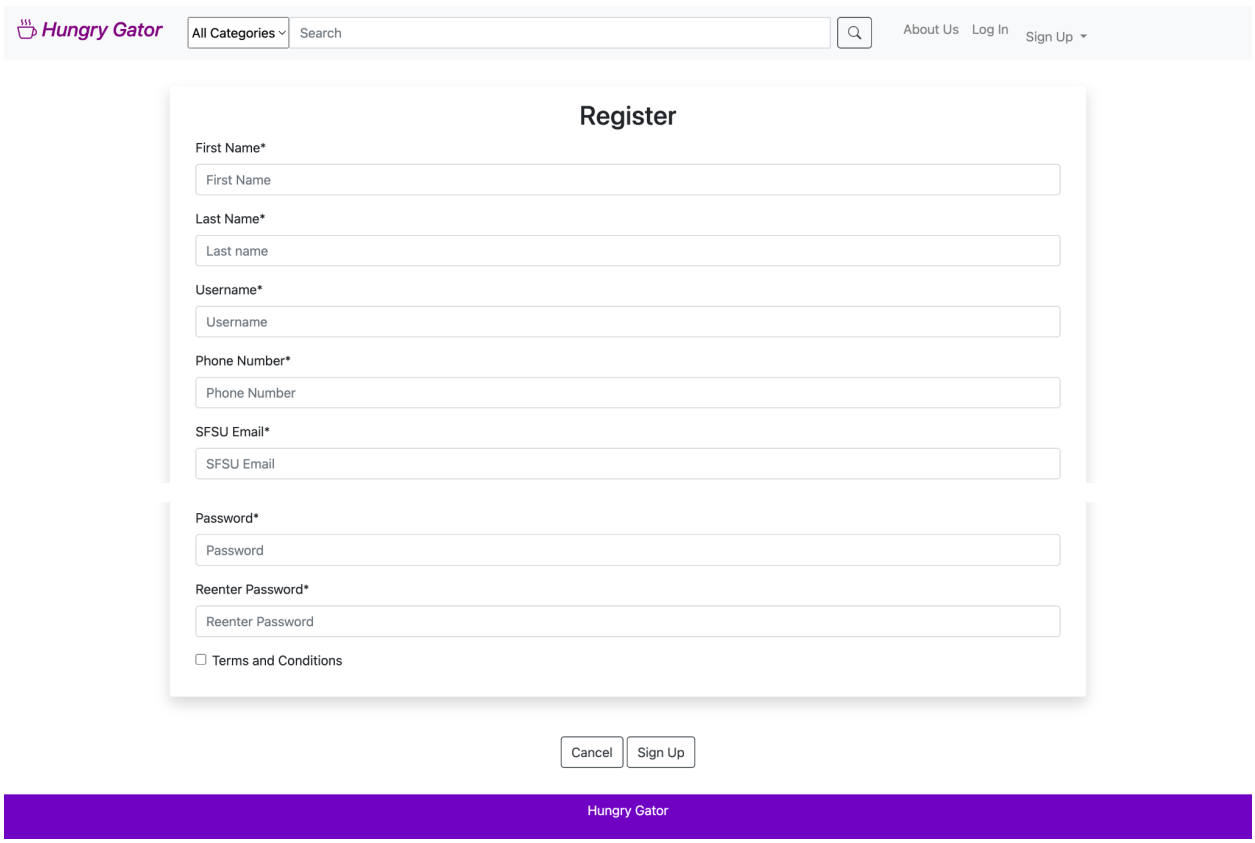
Don't have an account?

- Register here for SFSU users!
- Register here for delivery drivers!
- Restaurant Owners? Register here!

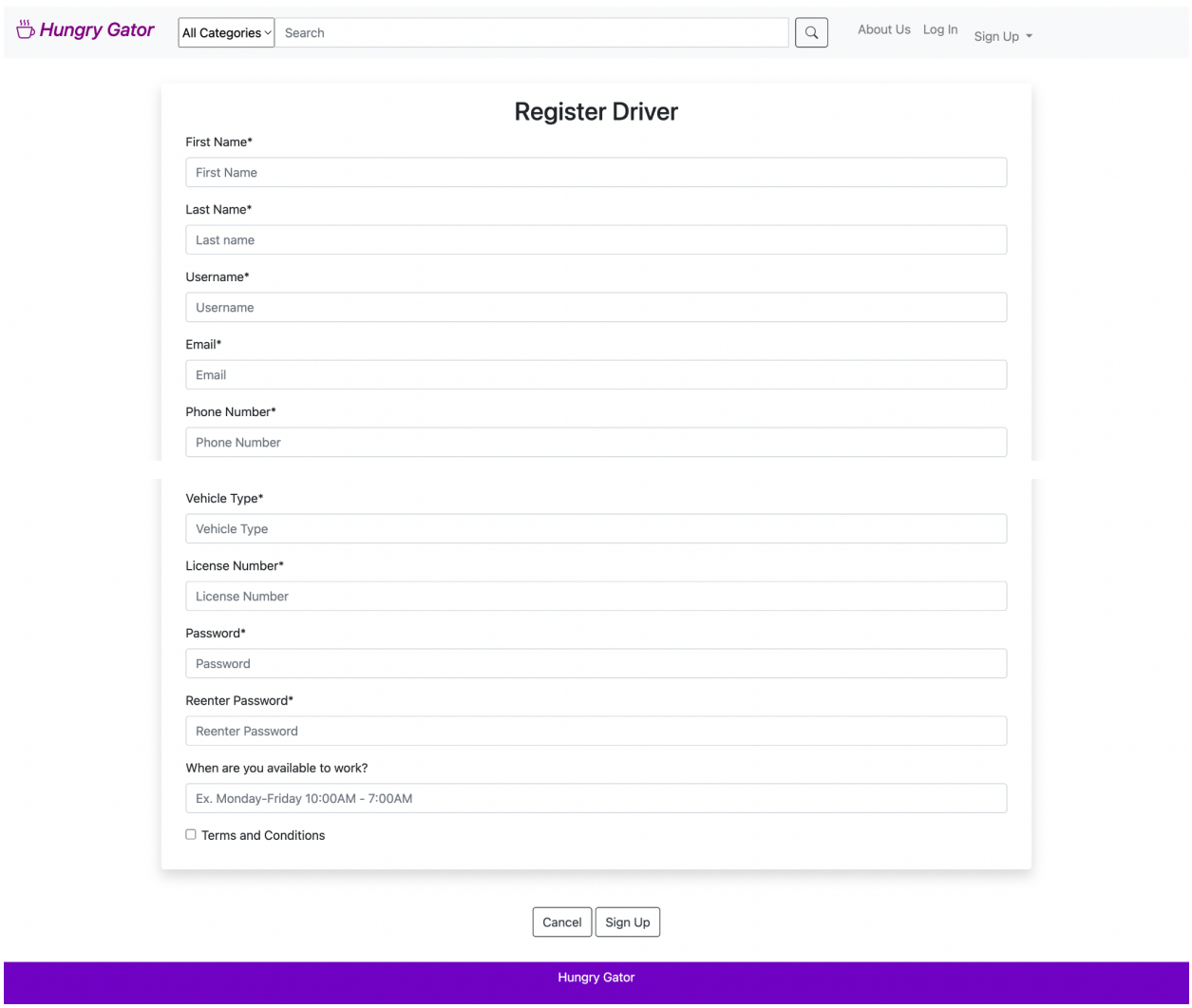
Sign In

Hungry Gator

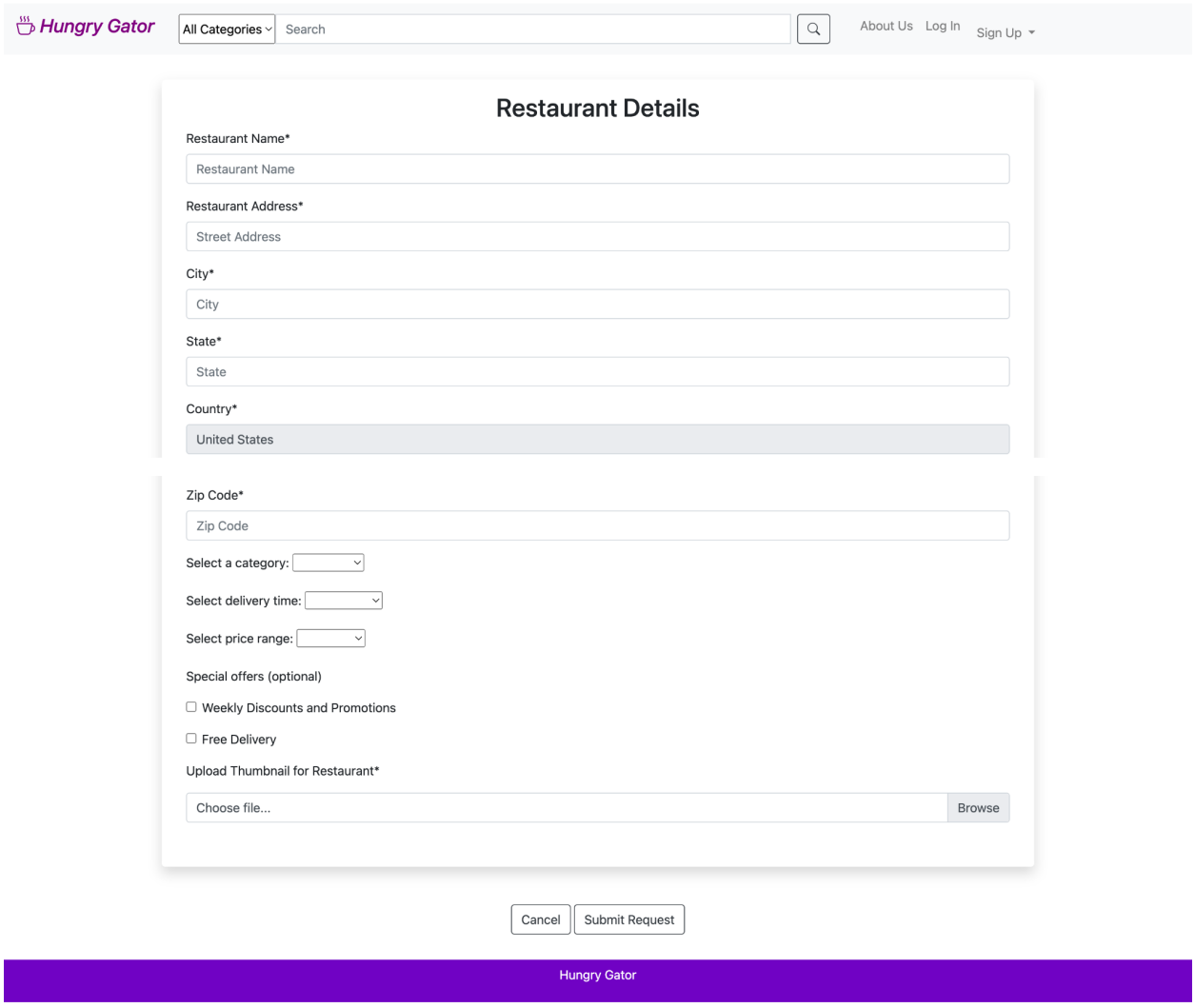
SFSU Software Engineering Project CSC 648-848, Spring 2023.
For Demonstration Only.

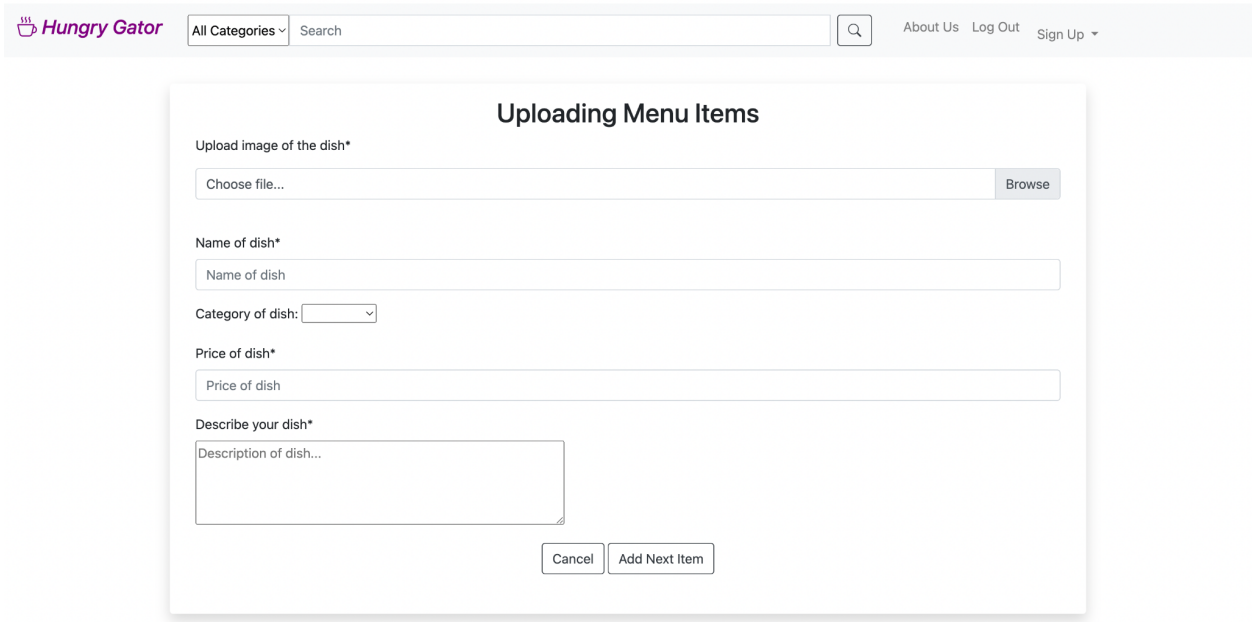


SFSU Software Engineering Project CSC 648-848, Spring 2023.
For Demonstration Only.

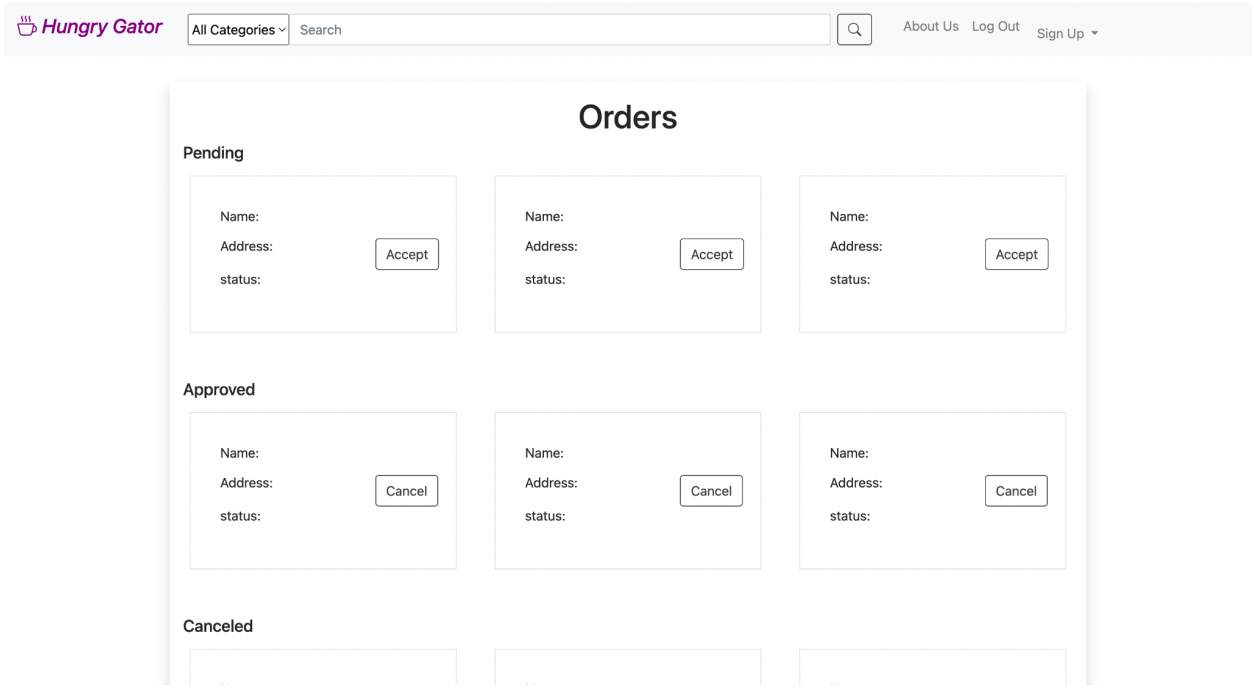


SFSU Software Engineering Project CSC 648-848, Spring 2023.
For Demonstration Only.





Hungry Gator



Canceled

Name:
Address:
status:

Name:
Address:
status:

Name:
Address:
status:

Hungry Gator

SFSU Software Engineering Project CSC 648-848, Spring 2023.
For Demonstration Only.

Hungry Gator | All Categories | Search | About Us | Log In | Sign Up

7 results found

Sort By:

Marugame Udon

Address: 3251 20th Ave, Space 184, San Francisco, CA 94132

Price: $20-$30

Expected Delivery Time: 0000-00-00 00:00:00

Category: Asian

Shake Shack

Address: 3251 20th Ave San Francisco, CA 94132

Price: $10-$20

Expected Delivery Time: 0000-00-00 00:00:00

Category: American

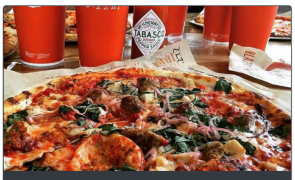
Chipotle

Address: 3251 20th Ave San Francisco, CA 94132

Price: $20-$30

Expected Delivery Time: 0000-00-00 00:00:00

Category: Mexican



Blaze Pizza

Address: 3251 20th Ave San Francisco, CA 94132

Price: $20-$30

Expected Delivery Time: 0000-00-00 00:00:00

Category: Italian

Panda Express

Address: 3251 20th Ave San Francisco, CA 94132

Price: $10-$20

Expected Delivery Time: 0000-00-00 00:00:00

Category: Asian

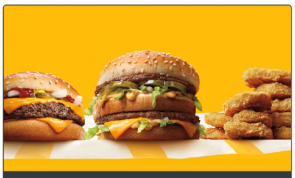
Cadillac Bar & Grill

Address: 44 9th St, San Francisco, CA 94103

Price: $20-$30

Expected Delivery Time: 0000-00-00 00:00:00

Category: Mexican



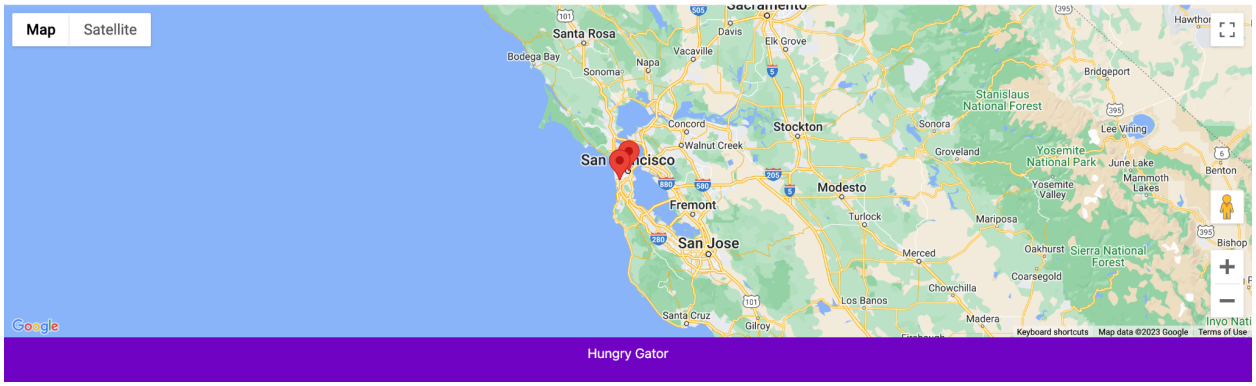
McDonald's

Address: 255 Winston Dr San Francisco, CA 94132

Price: <$10

Expected Delivery Time: 0000-00-00 00:00:00

Category: American



Hungry Gator

SFSU Software Engineering Project CSC 648-848, Spring 2023.

For Demonstration Only.

Hungry Gator

Name of Restaurant

description

Proceed To Checkout

Name of Plate

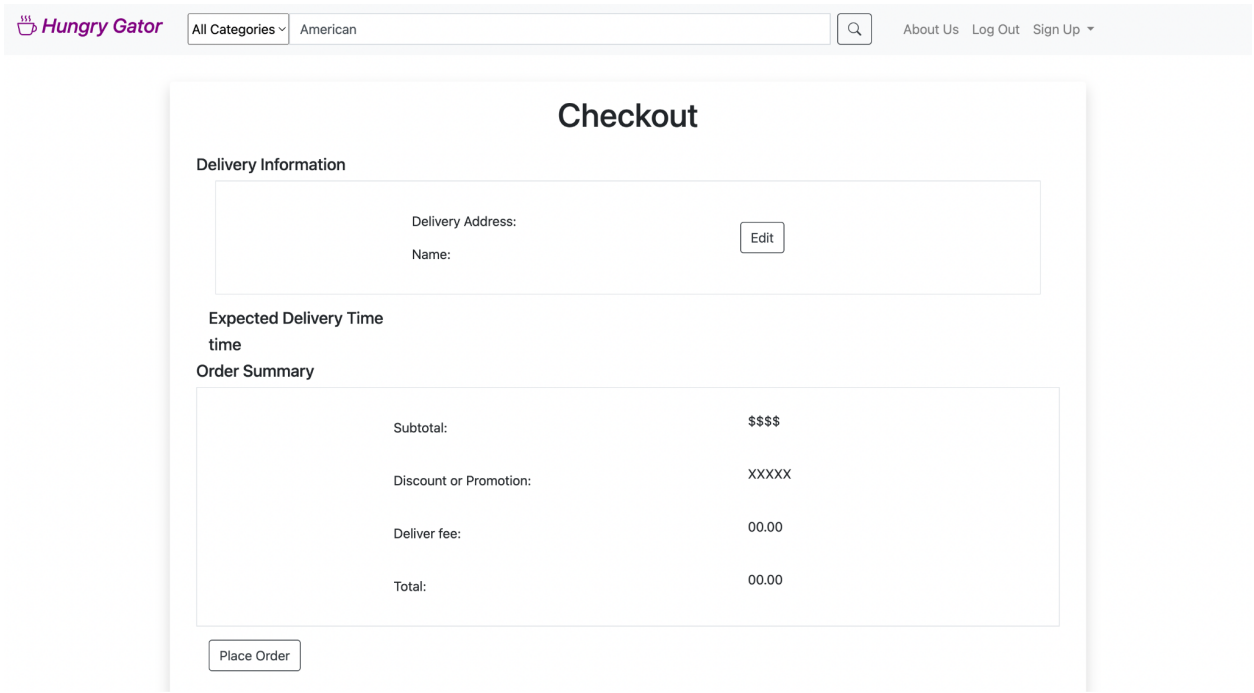
Description of dish

Price

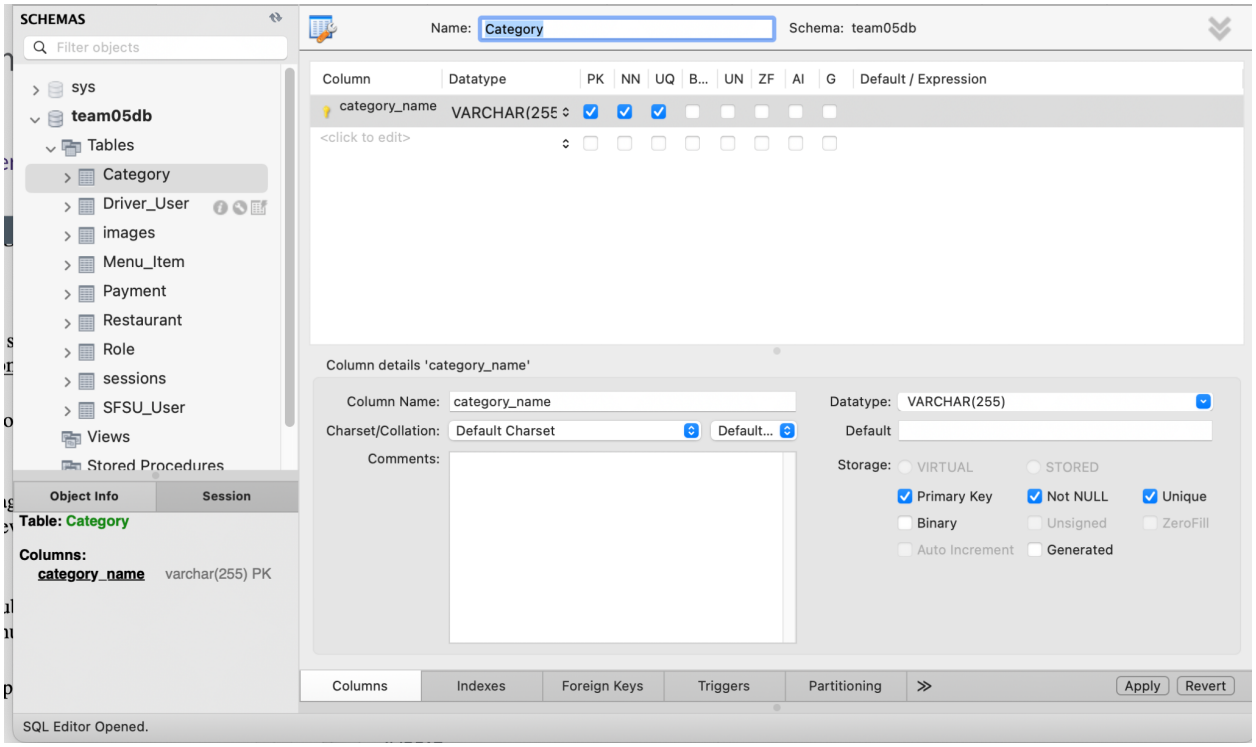
Price

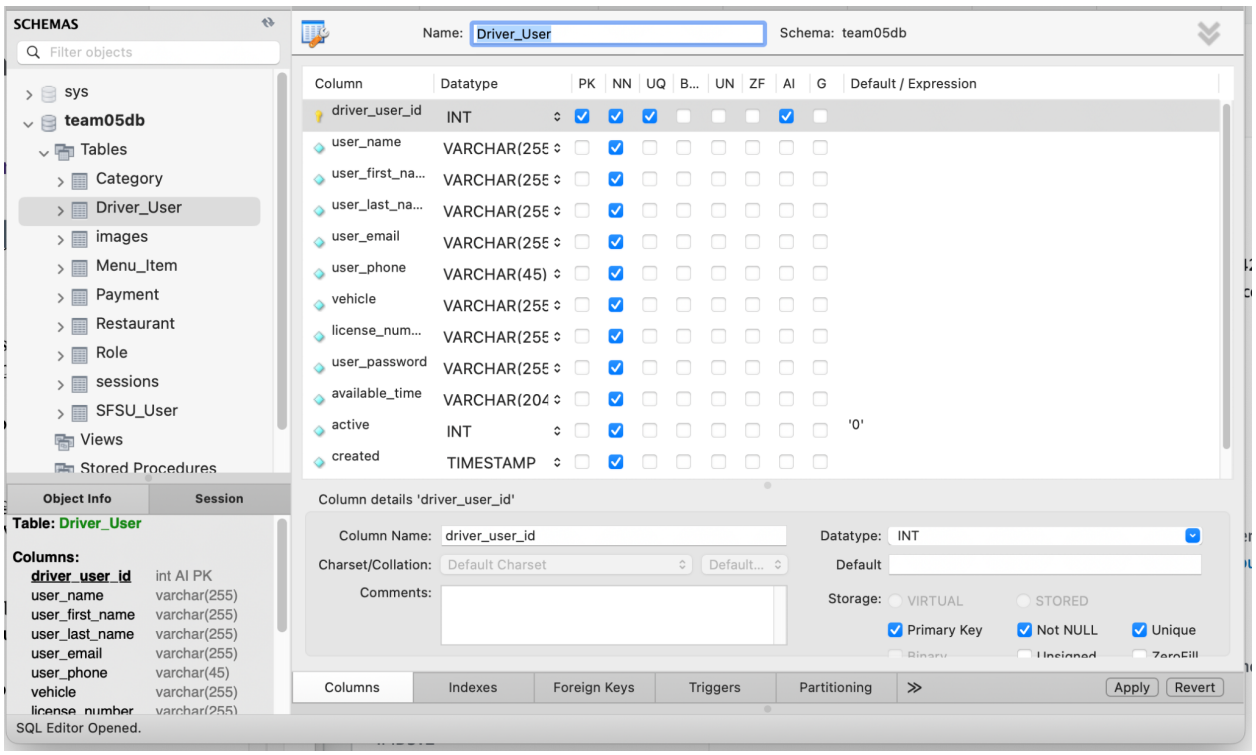
Add

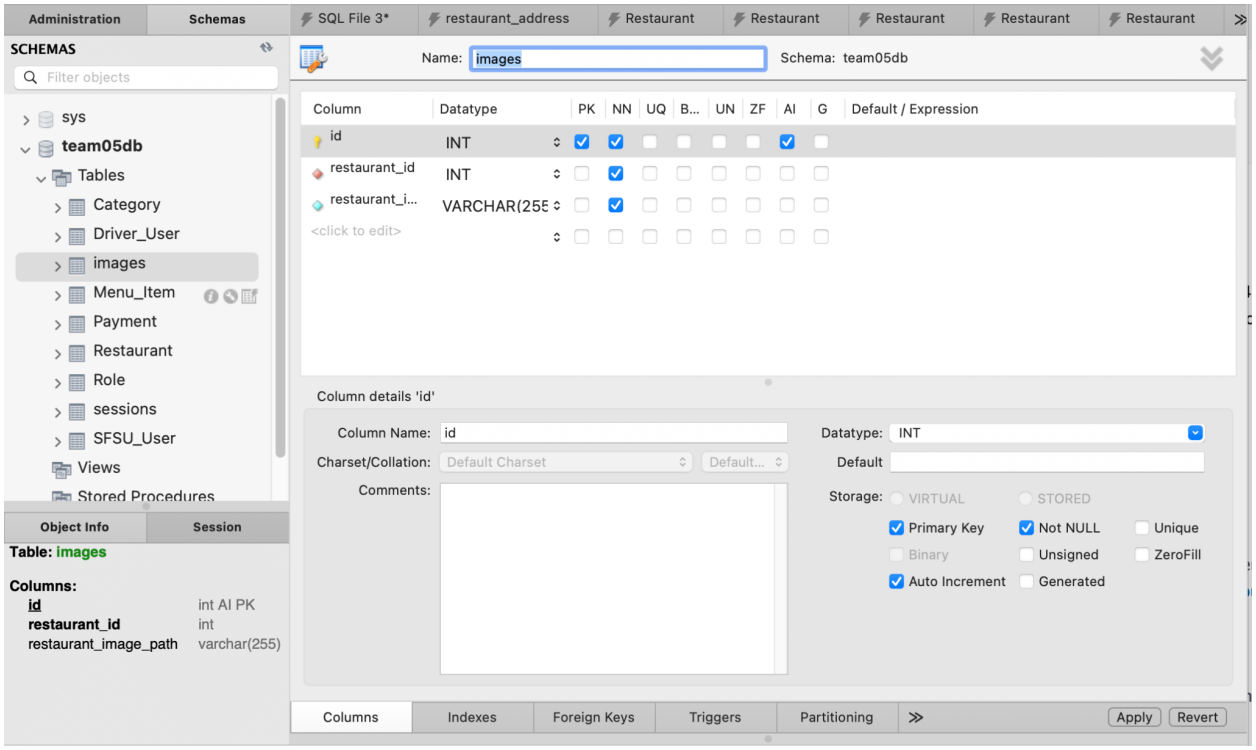
Hungry Gator

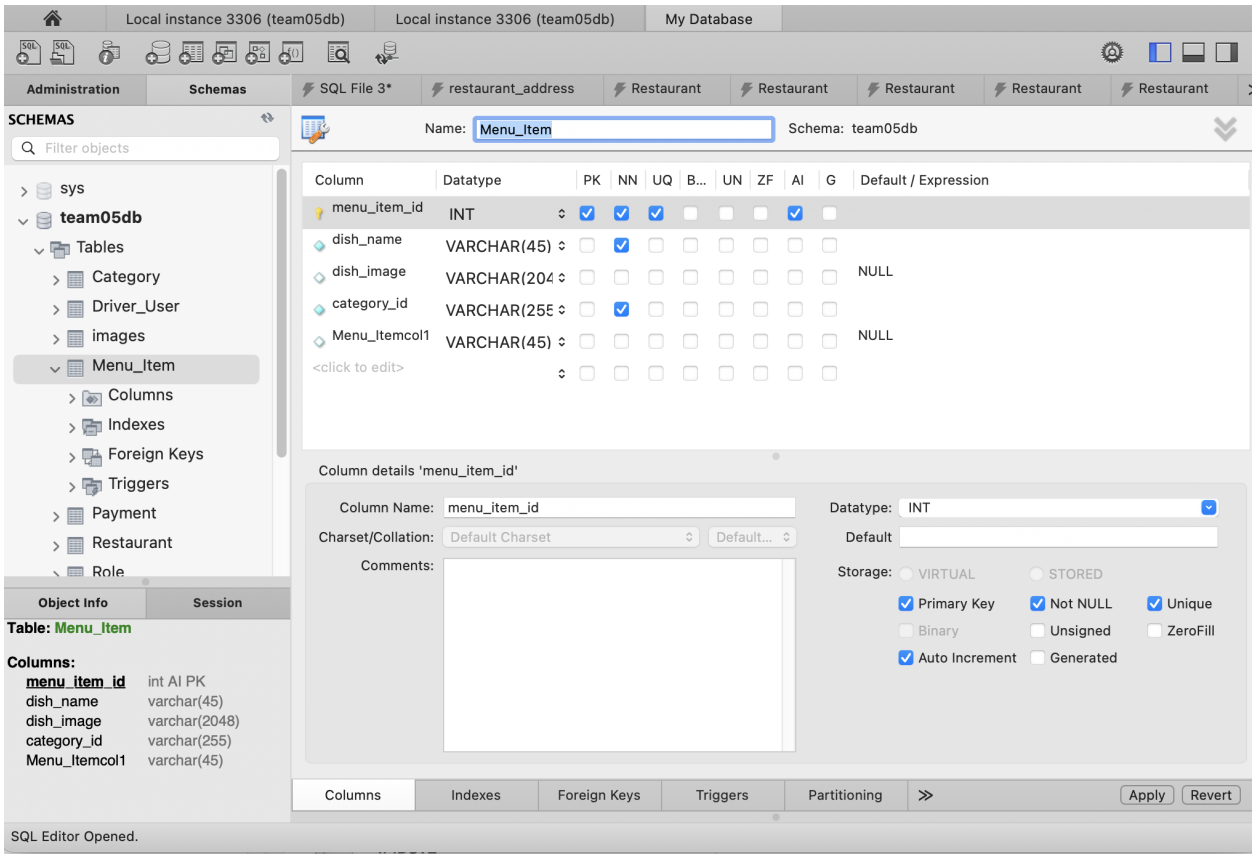

Hungry Gator
All Categories
American
About Us
Log Out
Sign Up
Checkout
Delivery Information
Delivery Address:
Name:
Edit
Expected Delivery Time
time
Order Summary
Subtotal:
$$$$
Discount or Promotion:
XXXXX
Deliver fee:
00.00
Total:
00.00
Place Order

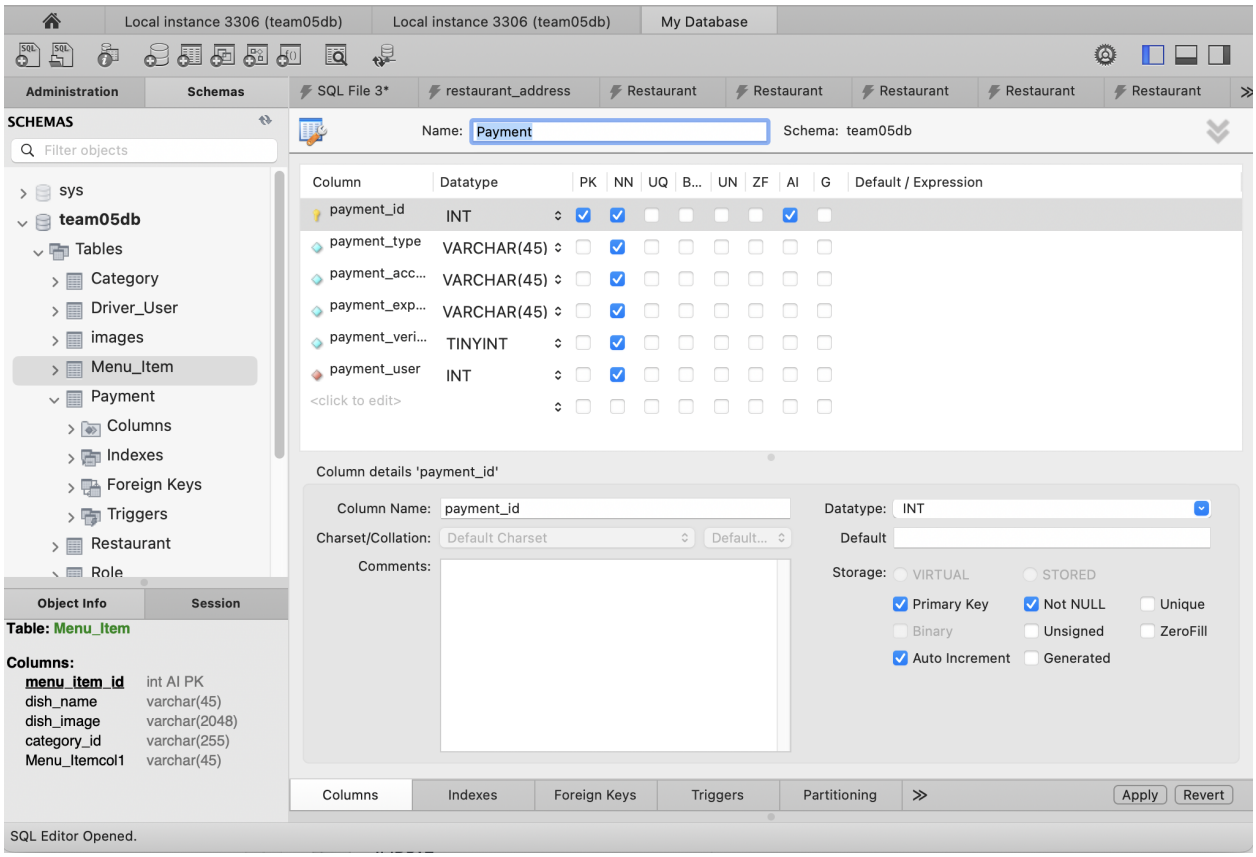
5) Database Organization

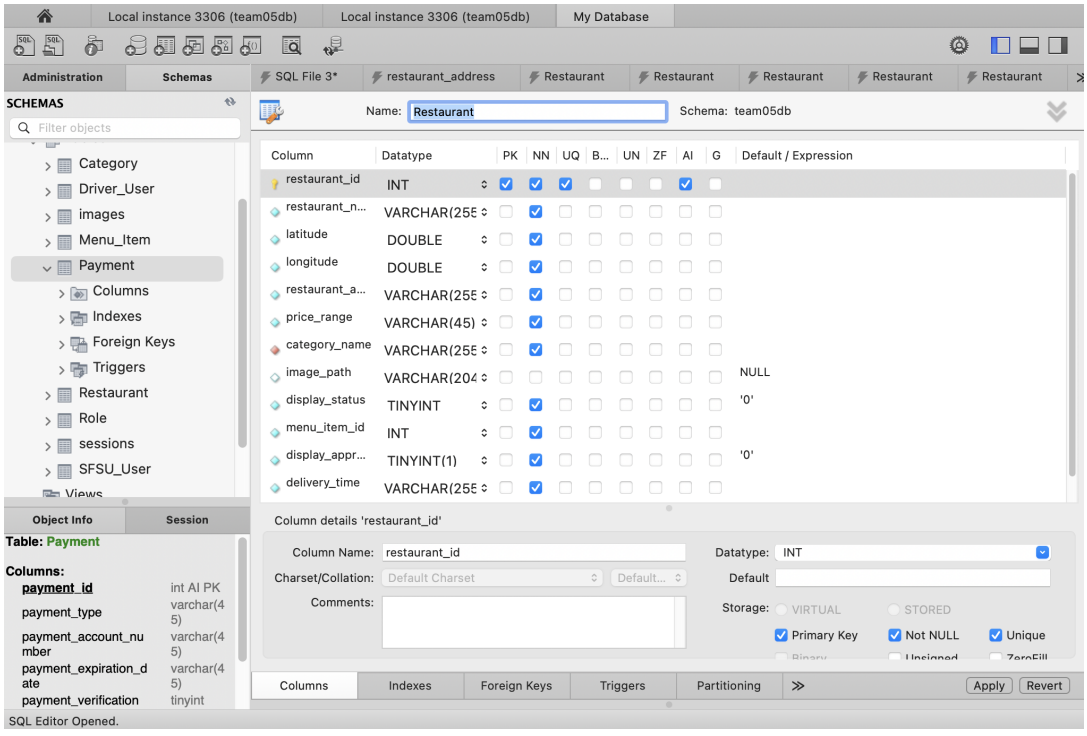


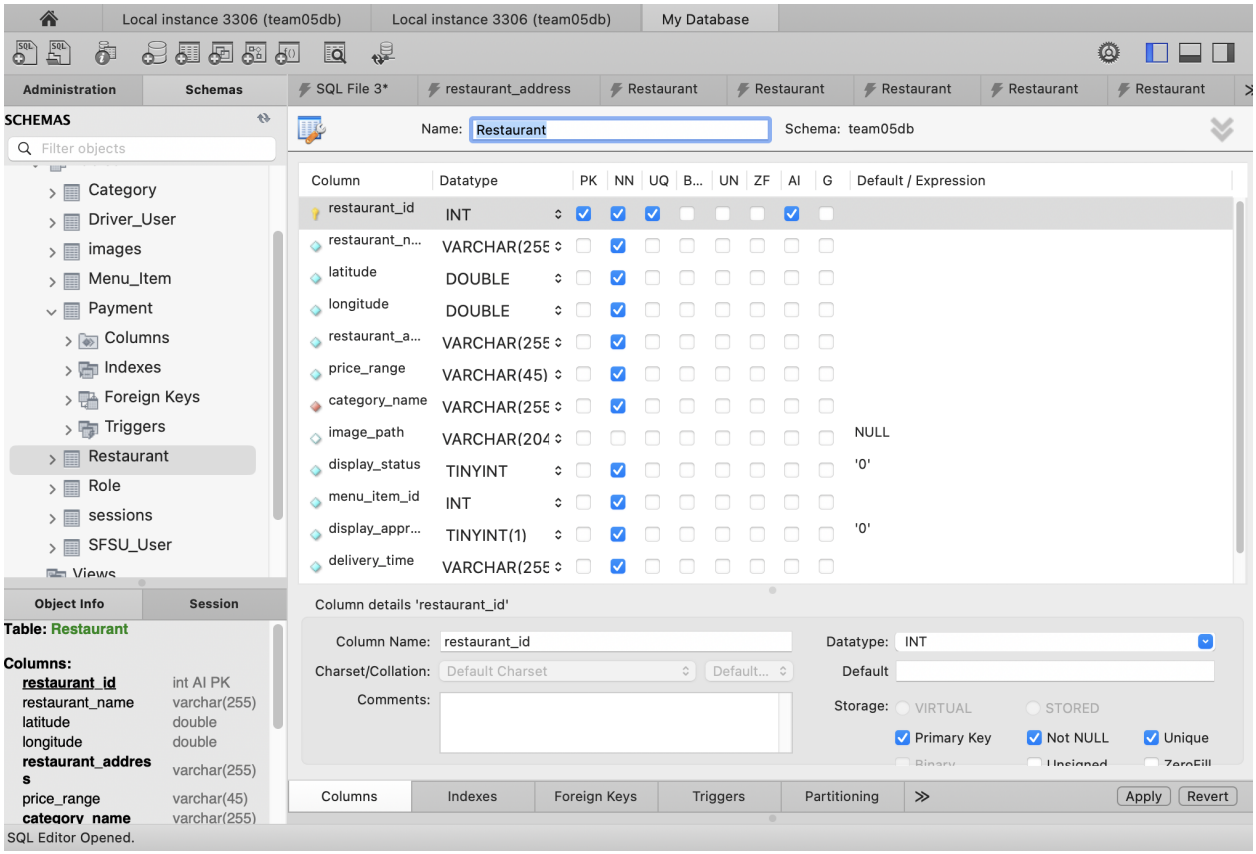


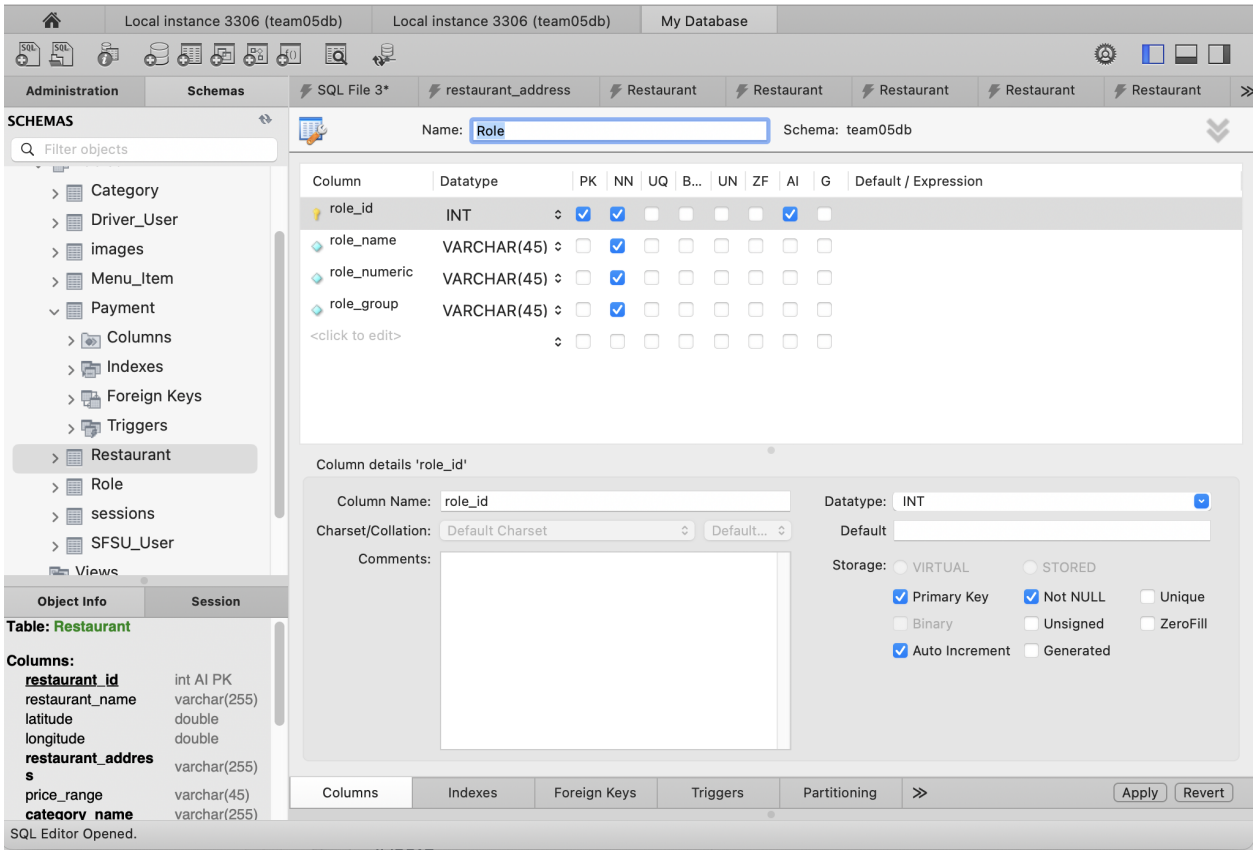

Administration
Schemas
SQL File 3*
restaurant_address
Restaurant
SCHEMAS
Filter objects
sys
team05db
Tables
Category
Driver_User
images
Menu_Item
Payment
Restaurant
Role
sessions
SFSU_User
Views
Stored Procedures
Name: images
Schema: team05db
Column
Datatype
PK
NN
UQ
B...
UN
ZF
AI
G
Default / Expression
id
INT
restaurant_id
INT
restaurant_i...
VARCHAR(255
<click to edit>
Column details 'id'
Column Name: id
Charset/Collation: Default Charset
Default...
Comments:
Datatype: INT
Default
Storage: VIRTUAL STORED
Primary Key
Not NULL
Unique
Binary
Unsigned
ZeroFill
Auto Increment
Generated
Object Info
Session
Table: images
Columns:
id int AI PK
restaurant_id int
restaurant_image_path varchar(255)
Columns
Indexes
Foreign Keys
Triggers
Partitioning
Apply
Revert

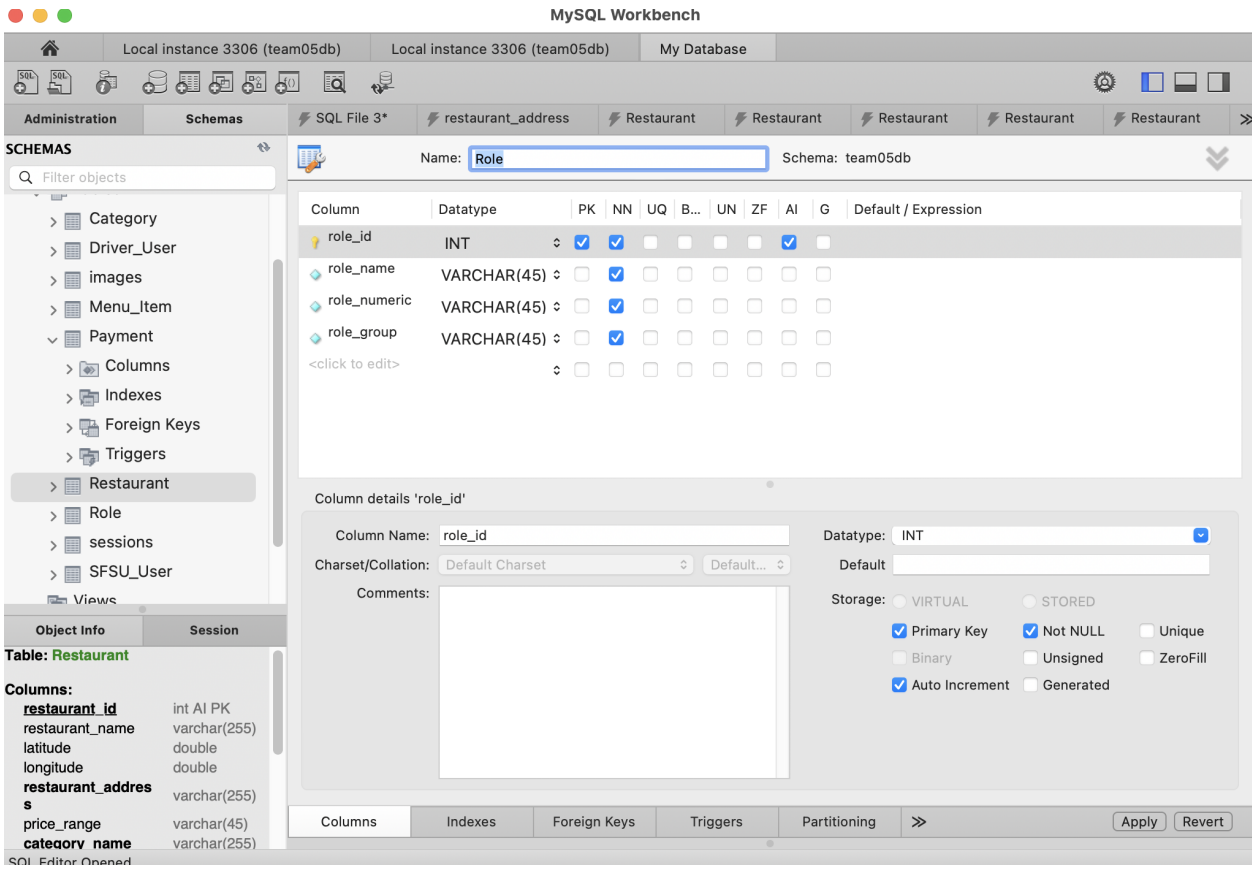

Local instance 3306 (team05db)
Local instance 3306 (team05db)
My Database
Administration
Schemas
SQL File 3*
restaurant_address
Restaurant
SCHEMAS
Filter objects
sys
team05db
Tables
Category
Driver_User
images
Menu_Item
Columns
Indexes
Foreign Keys
Triggers
Payment
Restaurant
Role
Name: Menu_Item
Schema: team05db
Column
Datatype
PK
NN
UQ
B...
UN
ZF
AI
G
Default / Expression
menu_item_id
INT
dish_name
VARCHAR(45)
dish_image
VARCHAR(204
NULL
category_id
VARCHAR(255
Menu_Itemcol1
VARCHAR(45)
NULL
<click to edit>
Column details 'menu_item_id'
Column Name: menu_item_id
Charset/Collation: Default Charset
Default...
Comments:
Datatype: INT
Default
Storage: VIRTUAL STORED
Primary Key
Not NULL
Unique
Binary
Unsigned
ZeroFill
Auto Increment
Generated
Object Info
Session
Table: Menu_Item
Columns:
menu_item_id int AI PK
dish_name varchar(45)
dish_image varchar(2048)
category_id varchar(255)
Menu_Itemcol1 varchar(45)
Columns
Indexes
Foreign Keys
Triggers
Partitioning
Apply
Revert
SQL Editor Opened.

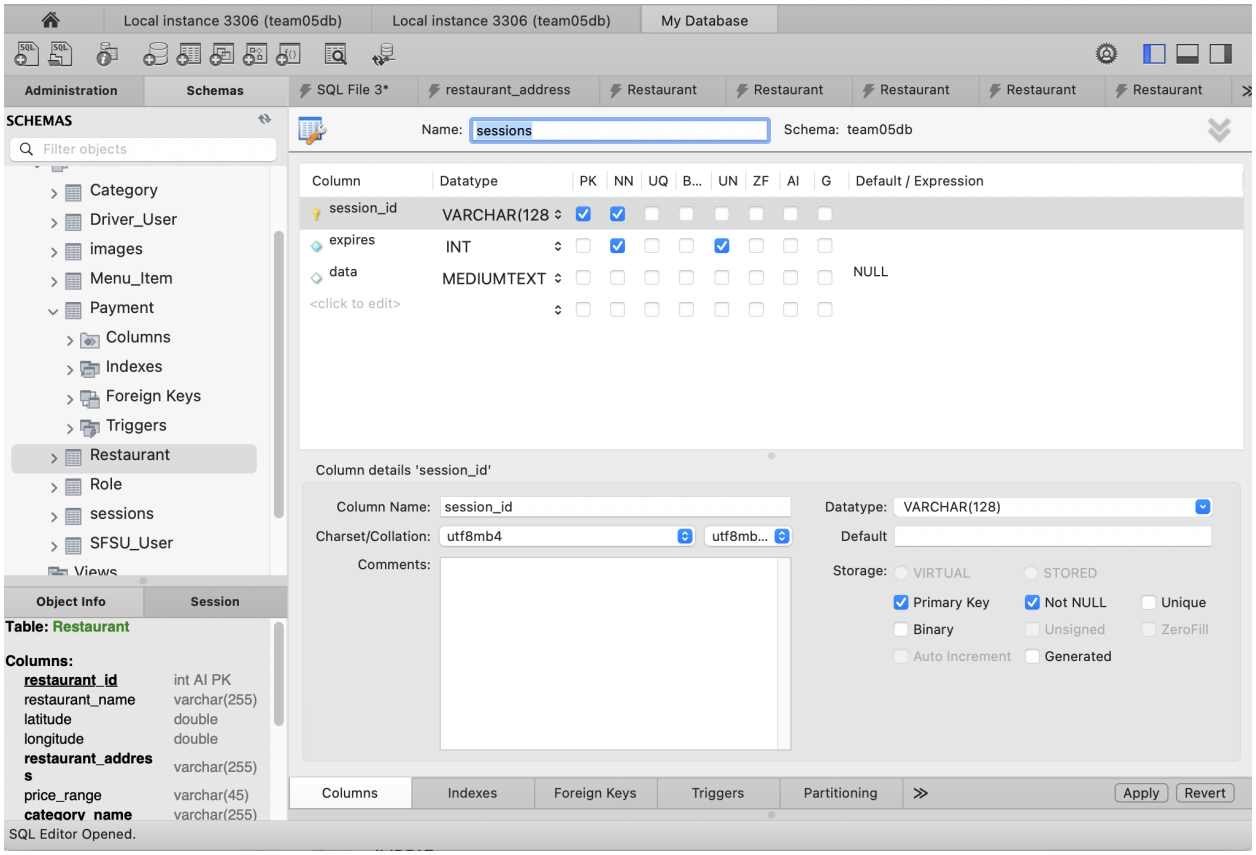

Local instance 3306 (team05db)
Local instance 3306 (team05db)
My Database
Administration
Schemas
SQL File 3*
restaurant_address
Restaurant
SCHEMAS
Filter objects
sys
team05db
Tables
Category
Driver_User
images
Menu_Item
Payment
Columns
Indexes
Foreign Keys
Triggers
Restaurant
Role
Name: Payment
Schema: team05db
Column
Datatype
PK
NN
UQ
B...
UN
ZF
AI
G
Default / Expression
payment_id
INT
payment_type
VARCHAR(45)
payment_acc...
payment_exp...
payment_veri...
TINYINT
payment_user
<click to edit>
Column details 'payment_id'
Column Name: payment_id
Charset/Collation: Default Charset
Comments:
Datatype: INT
Default
Storage: VIRTUAL STORED
Primary Key
Not NULL
Unique
Binary
Unsigned
ZeroFill
Auto Increment
Generated
Columns
Indexes
Foreign Keys
Triggers
Partitioning
Apply
Revert
Object Info
Session
Table: Menu_Item
Columns:
menu_item_id int AI PK
dish_name varchar(45)
dish_image varchar(2048)
category_id varchar(255)
Menu_Itemcol1 varchar(45)
SQL Editor Opened.

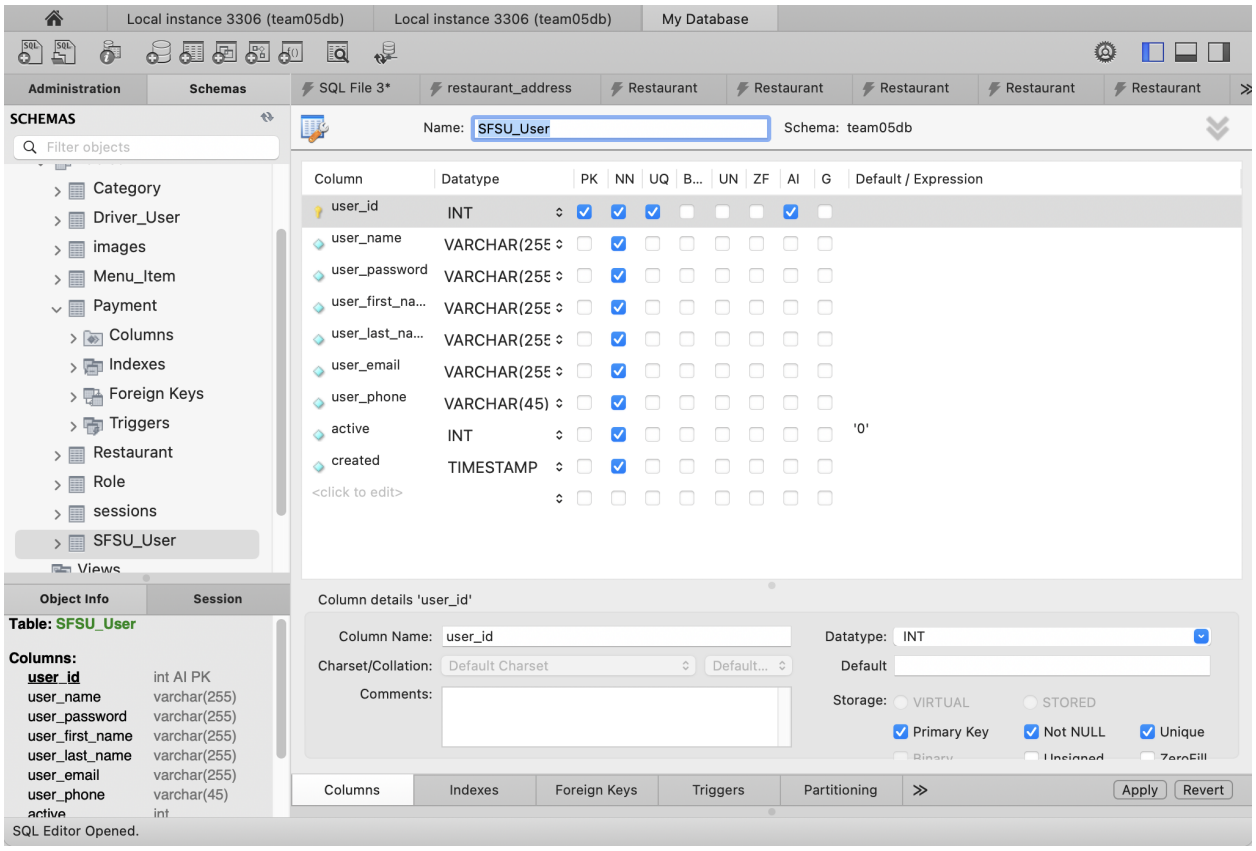

Local instance 3306 (team05db)
Local instance 3306 (team05db)
My Database
Administration
Schemas
SQL File 3*
restaurant_address
Restaurant
SCHEMAS
Filter objects
Category
Driver_User
images
Menu_Item
Payment
Columns
Indexes
Foreign Keys
Triggers
Restaurant
Role
sessions
SFSU_User
Views
Name: Restaurant
Schema: team05db
Column
Datatype
PK
NN
UQ
B...
UN
ZF
AI
G
Default / Expression
restaurant_id
INT
restaurant_n...
latitude
DOUBLE
longitude
DOUBLE
restaurant_a...
price_range
VARCHAR(45)
category_name
image_path
NULL
display_status
TINYINT
'0'
menu_item_id
INT
display_appr...
TINYINT(1)
'0'
delivery_time
Column details 'restaurant_id'
Column Name: restaurant_id
Charset/Collation: Default Charset
Comments:
Datatype: INT
Default
Storage: VIRTUAL STORED
Primary Key
Not NULL
Unique
Columns
Indexes
Foreign Keys
Triggers
Partitioning
Apply
Revert
Object Info
Session
Table: Payment
Columns:
payment_id int AI PK
payment_type varchar(45)
payment_account_number varchar(45)
payment_expiration_date varchar(45)
payment_verification tinyint
SQL Editor Opened.


Local instance 3306 (team05db)
Local instance 3306 (team05db)
My Database
Administration
Schemas
SQL File 3*
restaurant_address
Restaurant
Restaurant
Restaurant
Restaurant
Restaurant
SCHEMAS
Filter objects
Category
Driver_User
images
Menu_Item
Payment
Columns
Indexes
Foreign Keys
Triggers
Restaurant
Role
sessions
SFSU_User
Views
Name: Restaurant
Schema: team05db
Column
Datatype
PK
NN
UQ
B...
UN
ZF
AI
G
Default / Expression
restaurant_id
INT
restaurant_n...
VARCHAR(255
latitude
DOUBLE
longitude
DOUBLE
restaurant_a...
VARCHAR(255
price_range
VARCHAR(45)
category_name
VARCHAR(255
image_path
VARCHAR(204
NULL
display_status
TINYINT
'0'
menu_item_id
INT
display_appr...
TINYINT(1)
'0'
delivery_time
VARCHAR(255
Object Info
Session
Table: Restaurant
Columns:
restaurant_id int AI PK
restaurant_name varchar(255)
latitude double
longitude double
restaurant_address varchar(255)
price_range varchar(45)
category_name varchar(255)
Column details 'restaurant_id'
Column Name: restaurant_id
Charset/Collation: Default Charset
Default...
Comments:
Datatype: INT
Default
Storage: VIRTUAL STORED
Primary Key
Not NULL
Unique
Columns
Indexes
Foreign Keys
Triggers
Partitioning
Apply
Revert
SQL Editor Opened.


Local instance 3306 (team05db)
Local instance 3306 (team05db)
My Database
Administration
Schemas
SQL File 3*
restaurant_address
Restaurant
Restaurant
Restaurant
Restaurant
Restaurant
SCHEMAS
Filter objects
Category
Driver_User
images
Menu_Item
Payment
Columns
Indexes
Foreign Keys
Triggers
Restaurant
Role
sessions
SFSU_User
Views
Name: Role
Schema: team05db
Column
Datatype
PK
NN
UQ
B...
UN
ZF
AI
G
Default / Expression
role_id
INT
role_name
VARCHAR(45)
role_numeric
VARCHAR(45)
role_group
VARCHAR(45)
<click to edit>
Column details 'role_id'
Column Name: role_id
Charset/Collation: Default Charset
Default...
Comments:
Datatype: INT
Default
Storage: VIRTUAL STORED
Primary Key
Not NULL
Unique
Binary
Unsigned
ZeroFill
Auto Increment
Generated
Columns
Indexes
Foreign Keys
Triggers
Partitioning
Apply
Revert
Object Info
Session
Table: Restaurant
Columns:
restaurant_id int AI PK
restaurant_name varchar(255)
latitude double
longitude double
restaurant_address varchar(255)
price_range varchar(45)
category_name varchar(255)
SQL Editor Opened.


MySQL Workbench
Local instance 3306 (team05db)
Local instance 3306 (team05db)
My Database
Administration
Schemas
SQL File 3*
restaurant_address
Restaurant
Restaurant
Restaurant
Restaurant
Restaurant
SCHEMAS
Filter objects
Category
Driver_User
images
Menu_Item
Payment
Columns
Indexes
Foreign Keys
Triggers
Restaurant
Role
sessions
SFSU_User
Views
Name: Role
Schema: team05db
Column
Datatype
PK
NN
UQ
B...
UN
ZF
AI
G
Default / Expression
role_id
INT
role_name
VARCHAR(45)
role_numeric
VARCHAR(45)
role_group
VARCHAR(45)
<click to edit>
Column details 'role_id'
Column Name: role_id
Charset/Collation: Default Charset
Default...
Comments:
Datatype: INT
Default
Storage: VIRTUAL STORED
Primary Key
Not NULL
Unique
Binary
Unsigned
ZeroFill
Auto Increment
Generated
Columns
Indexes
Foreign Keys
Triggers
Partitioning
Apply
Revert
Object Info
Session
Table: Restaurant
Columns:
restaurant_id int AI PK
restaurant_name varchar(255)
latitude double
longitude double
restaurant_addres s varchar(255)
price_range varchar(45)
category_name varchar(255)
SQL Editor Opened


Local instance 3306 (team05db)
Local instance 3306 (team05db)
My Database
Administration
Schemas
SQL File 3*
restaurant_address
Restaurant
SCHEMAS
Filter objects
Category
Driver_User
images
Menu_Item
Payment
Columns
Indexes
Foreign Keys
Triggers
Restaurant
Role
sessions
SFSU_User
Views
Name: sessions
Schema: team05db
Column
Datatype
PK
NN
UQ
B...
UN
ZF
AI
G
Default / Expression
session_id
VARCHAR(128
expires
INT
data
MEDIUMTEXT
NULL
<click to edit>
Column details 'session_id'
Column Name: session_id
Charset/Collation: utf8mb4
utf8mb...
Comments:
Datatype: VARCHAR(128)
Default
Storage: VIRTUAL STORED
Primary Key
Not NULL
Unique
Binary
Unsigned
ZeroFill
Auto Increment
Generated
Columns
Indexes
Foreign Keys
Triggers
Partitioning
Apply
Revert
Object Info
Session
Table: Restaurant
Columns:
restaurant_id int AI PK
restaurant_name varchar(255)
latitude double
longitude double
restaurant_addres s varchar(255)
price_range varchar(45)
SQL Editor Opened.

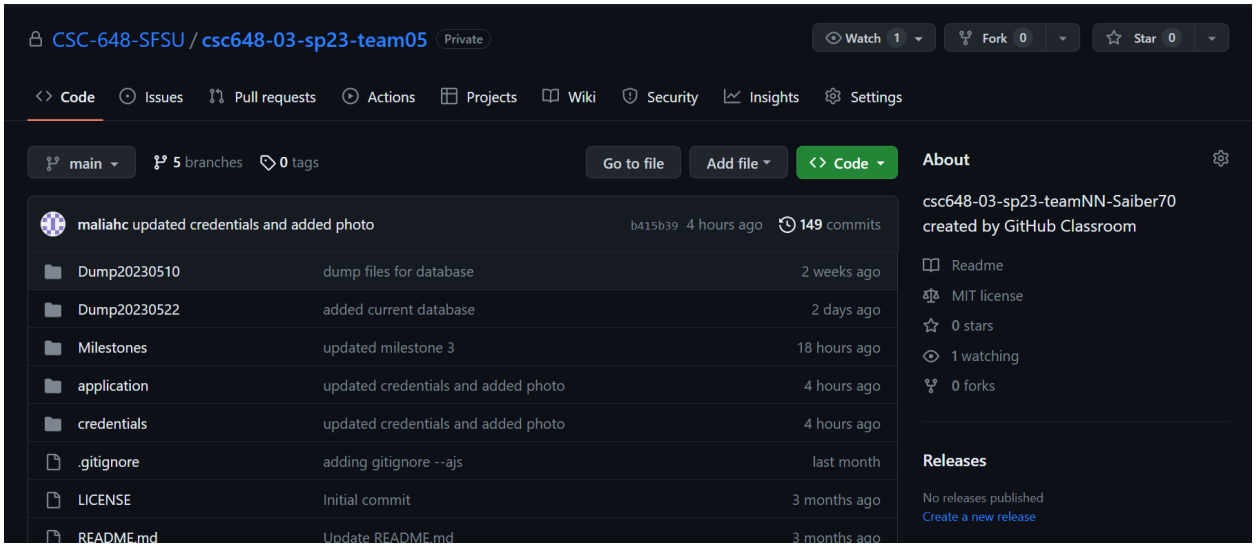

Local instance 3306 (team05db)
Local instance 3306 (team05db)
My Database
Administration
Schemas
SQL File 3*
restaurant_address
Restaurant
SCHEMAS
Filter objects
Category
Driver_User
images
Menu_Item
Payment
Columns
Indexes
Foreign Keys
Triggers
Restaurant
Role
sessions
SFSU_User
Views
Name: SFSU_User
Schema: team05db
Column
Datatype
PK
NN
UQ
B...
UN
ZF
AI
G
Default / Expression
user_id
INT
user_name
VARCHAR(255
user_password
VARCHAR(255
user_first_na...
VARCHAR(255
user_last_na...
VARCHAR(255
user_email
VARCHAR(255
user_phone
VARCHAR(45)
active
INT
'0'
created
TIMESTAMP
<click to edit>
Column details 'user_id'
Column Name: user_id
Charset/Collation: Default Charset
Default...
Comments:
Datatype: INT
Default
Storage: VIRTUAL STORED
Primary Key
Not NULL
Unique
Columns
Indexes
Foreign Keys
Triggers
Partitioning
Apply
Revert
Object Info
Session
Table: SFSU_User
Columns:
user_id int AI PK
user_name varchar(255)
user_password varchar(255)
user_first_name varchar(255)
user_last_name varchar(255)
user_email varchar(255)
user_phone varchar(45)
SQL Editor Opened.

6) GitHub Organizations

List of Main Branches:

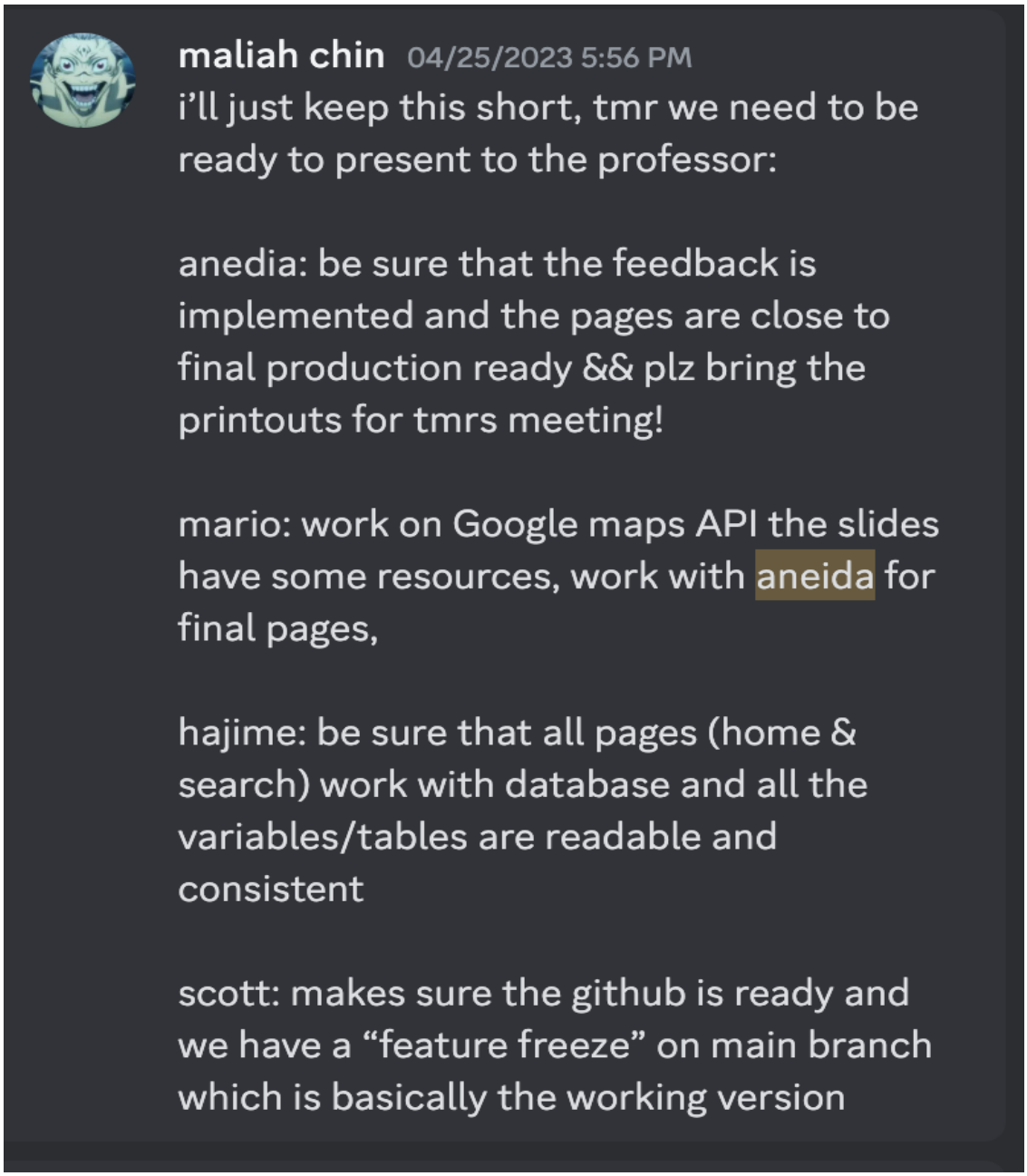
main, m0, m2, m3, m4

All team members have access to the main branch



7) Project management

One of the ways we decided to work with was Discord, we also had in-person and online meetings. We also used email as a way to connect and discuss our project plan.



MC

Maliah Chin

To: Dragutin Petkovic; Anthony John Souza; Scott Nguyen; Mario A. Leyva Moreno; Aneida Guadalupe Blanco Palacio; Hajime Miyazaki

Tue 4/11/2023 7:48 PM

Hello All,

After meeting with Professor Souza for assistance with Milestone 2 Vertical Prototype, we found that we need to reconvene and reorganize our project.

First and foremost, we noticed that our coding structure needs to be refactored. In order to reduce our scope, we needed to remove Nginx due to issues with processing some of our data. We also had to create a new instance of the server and reconnect all of our code, dependencies, and configurations. We found that during the refactoring, we need to reassess which structure we would like to use as a team. Please respond to this email by midnight whether you would like to use **ejs** or **hbs** as framework for our pages.

Since it's crunch time, I have decided to organize our tasks and assign each member what we need to accomplish/resolve to stay on track with our schedule. Please feel free to contact me if you have any questions!

Mario - To reference some of the suggestions we discussed today at office hours, I've included a couple of links and some of the commands used.

-- pm2 installation and usage
https://www.digitalocean.com/community/tutorials/how-to-use-pm2-to-setup-a-node-js-production-environment-on-an-ubuntu-vps

-- cp *html ../..
* to copy all of our html files over to their new destination
-- rm application
** to remove our application folder (after w)
-- npm express-generator --hbs (or --ejs) application
https://expressjs.com/en/starter/generator.html
** to create a sample application that will show us a more organized framework

Aneida - To ensure that we stay on track for the next deadline of Milestone 3, review the Milestone 3 Horizontal Prototype description in the file on Canvas. Then, begin creating functional UI screens, I suggest starting with the home page so we can have a general theme to follow for the remaining screens.

Hajime - During the process of creating a new instance, we have some new information about the database so please contact Mario for an update. When our search function calls fetch, we've noticed some issues with data processing that you need to look into. Also, look into routing using express so we can start connecting the UI screens to our backend and also the pages we have now to each other.
I included a link that might be helpful for finding the problem:

https://www.digitalocean.com/community/tutorials/how-to-use-the-javascript-fetch-api-to-get-data
https://www.digitalocean.com/community/tutorials/nodejs-express-routing

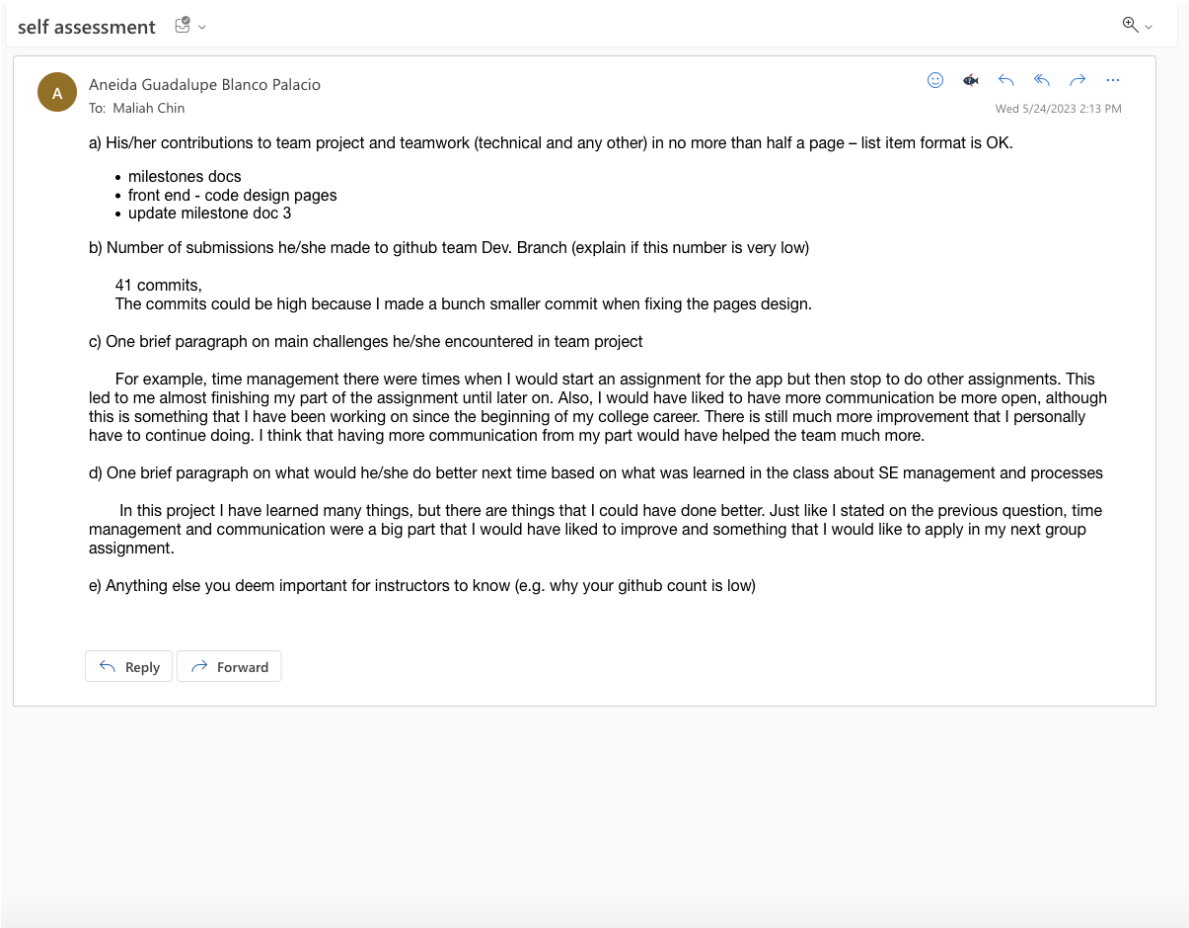
Scott - One of our biggest issues right now is routing and linking our pages together to the backend. I included a link to help you learn more about our issue. Please contact the backend lead to bring you up to date with our code, as well as how to collaborate on this problem.
https://www.digitalocean.com/community/tutorials/nodejs-express-routing

As a reminder, our grade is based on teamwork, and I'd like for us to all be on the same page for the future of this project!

Thank you,
Maliah

8) TeamWork Evaluation

Aneida



self assessment

Aneida Guadalupe Blanco Palacio
To: Maliah Chin
Wed 5/24/2023 2:13 PM

a) His/her contributions to team project and teamwork (technical and any other) in no more than half a page – list item format is OK.

- milestones docs
- front end - code design pages
- update milestone doc 3

b) Number of submissions he/she made to github team Dev. Branch (explain if this number is very low)

41 commits,
The commits could be high because I made a bunch smaller commit when fixing the pages design.

c) One brief paragraph on main challenges he/she encountered in team project

For example, time management there were times when I would start an assignment for the app but then stop to do other assignments. This led to me almost finishing my part of the assignment until later on. Also, I would have liked to have more communication be more open, although this is something that I have been working on since the beginning of my college career. There is still much more improvement that I personally have to continue doing. I think that having more communication from my part would have helped the team much more.

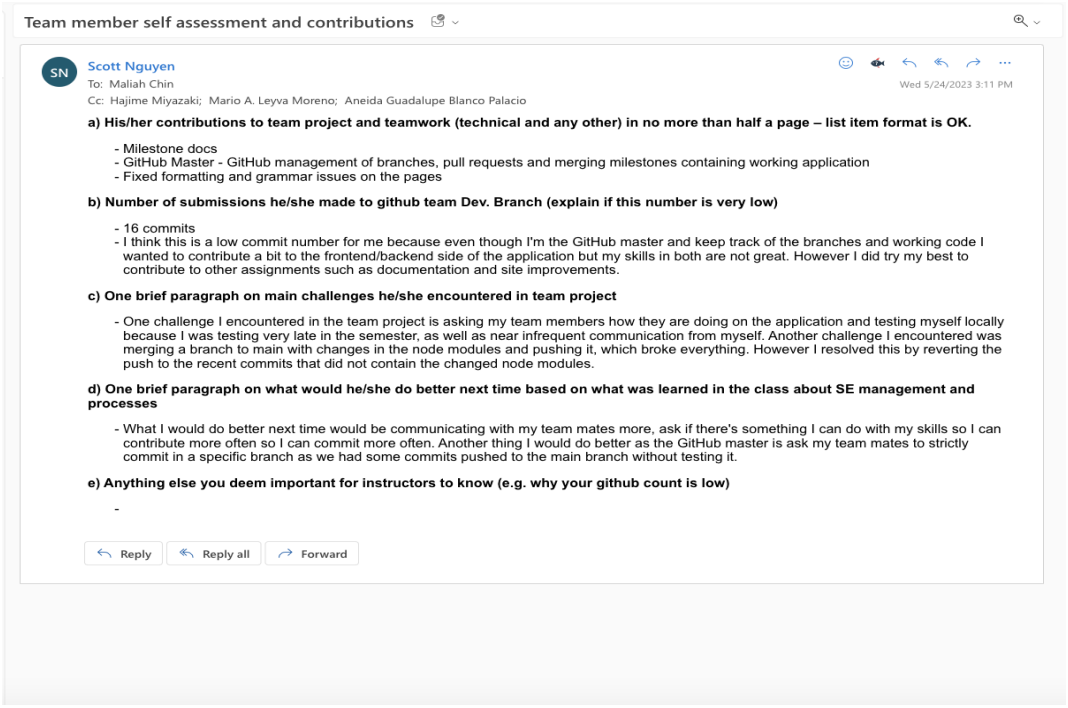
d) One brief paragraph on what would he/she do better next time based on what was learned in the class about SE management and processes

In this project I have learned many things, but there are things that I could have done better. Just like I stated on the previous question, time management and communication were a big part that I would have liked to improve and something that I would like to apply in my next group assignment.

e) Anything else you deem important for instructors to know (e.g. why your github count is low)

Reply Forward

Scott



Team member self assessment and contributions

SN **Scott Nguyen**

To: Maliah Chin

Cc: Hajime Miyazaki; Mario A. Leyva Moreno; Aneida Guadalupe Blanco Palacio

Wed 5/24/2023 3:11 PM

a) His/her contributions to team project and teamwork (technical and any other) in no more than half a page – list item format is OK.

- Milestone docs
- GitHub Master - GitHub management of branches, pull requests and merging milestones containing working application
- Fixed formatting and grammar issues on the pages

b) Number of submissions he/she made to github team Dev. Branch (explain if this number is very low)

- 16 commits
- I think this is a low commit number for me because even though I'm the GitHub master and keep track of the branches and working code I wanted to contribute a bit to the frontend/backend side of the application but my skills in both are not great. However I did try my best to contribute to other assignments such as documentation and site improvements.

c) One brief paragraph on main challenges he/she encountered in team project

- One challenge I encountered in the team project is asking my team members how they are doing on the application and testing myself locally because I was testing very late in the semester, as well as near infrequent communication from myself. Another challenge I encountered was merging a branch to main with changes in the node modules and pushing it, which broke everything. However I resolved this by reverting the push to the recent commits that did not contain the changed node modules.

d) One brief paragraph on what would he/she do better next time based on what was learned in the class about SE management and processes

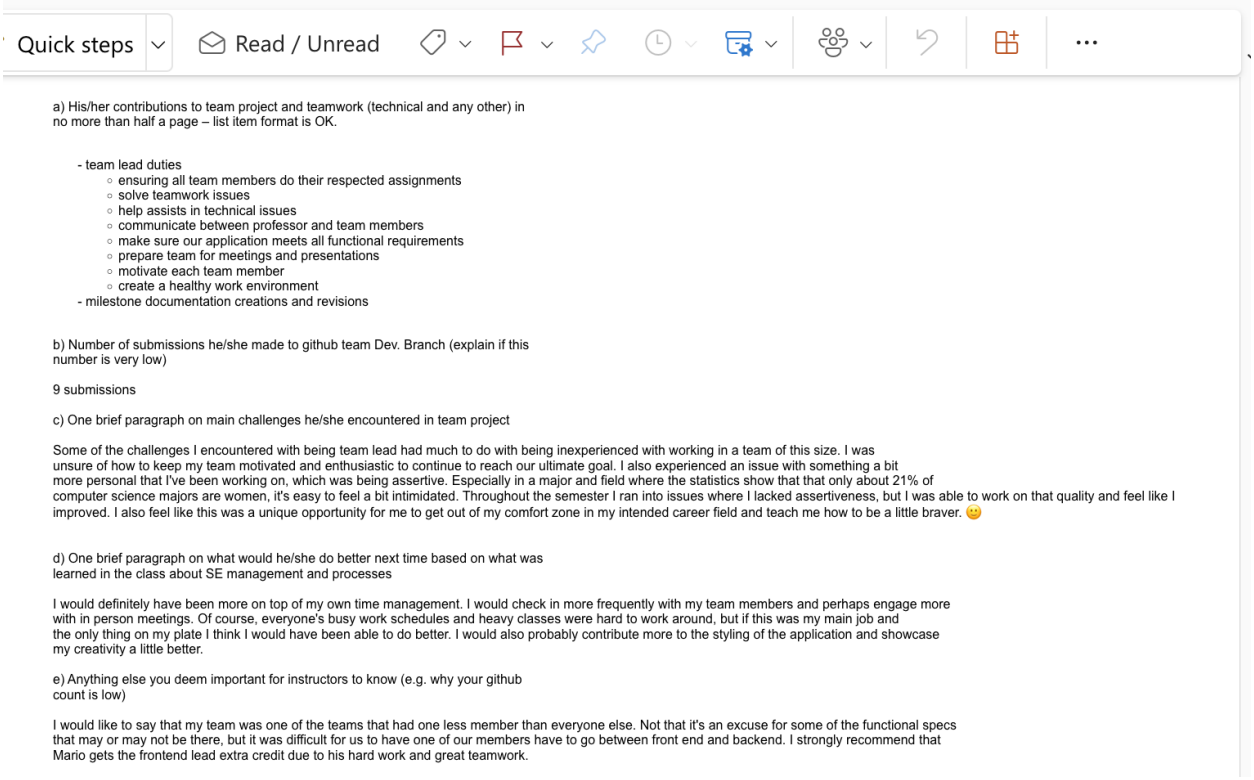
- What I would do better next time would be communicating with my team mates more, ask if there's something I can do with my skills so I can contribute more often so I can commit more often. Another thing I would do better as the GitHub master is ask my team mates to strictly commit in a specific branch as we had some commits pushed to the main branch without testing it.

e) Anything else you deem important for instructors to know (e.g. why your github count is low)

-

Reply Reply all Forward

Maliah:



Quick steps Read / Unread

a) His/her contributions to team project and teamwork (technical and any other) in no more than half a page – list item format is OK.

- team lead duties
  - ensuring all team members do their respected assignments
  - solve teamwork issues
  - help assists in technical issues
  - communicate between professor and team members
  - make sure our application meets all functional requirements
  - prepare team for meetings and presentations
  - motivate each team member
  - create a healthy work environment
- milestone documentation creations and revisions

b) Number of submissions he/she made to github team Dev. Branch (explain if this number is very low)

9 submissions

c) One brief paragraph on main challenges he/she encountered in team project

Some of the challenges I encountered with being team lead had much to do with being inexperienced with working in a team of this size. I was unsure of how to keep my team motivated and enthusiastic to continue to reach our ultimate goal. I also experienced an issue with something a bit more personal that I've been working on, which was being assertive. Especially in a major and field where the statistics show that that only about 21% of computer science majors are women, it's easy to feel a bit intimidated. Throughout the semester I ran into issues where I lacked assertiveness, but I was able to work on that quality and feel like I improved. I also feel like this was a unique opportunity for me to get out of my comfort zone in my intended career field and teach me how to be a little braver. 🙂

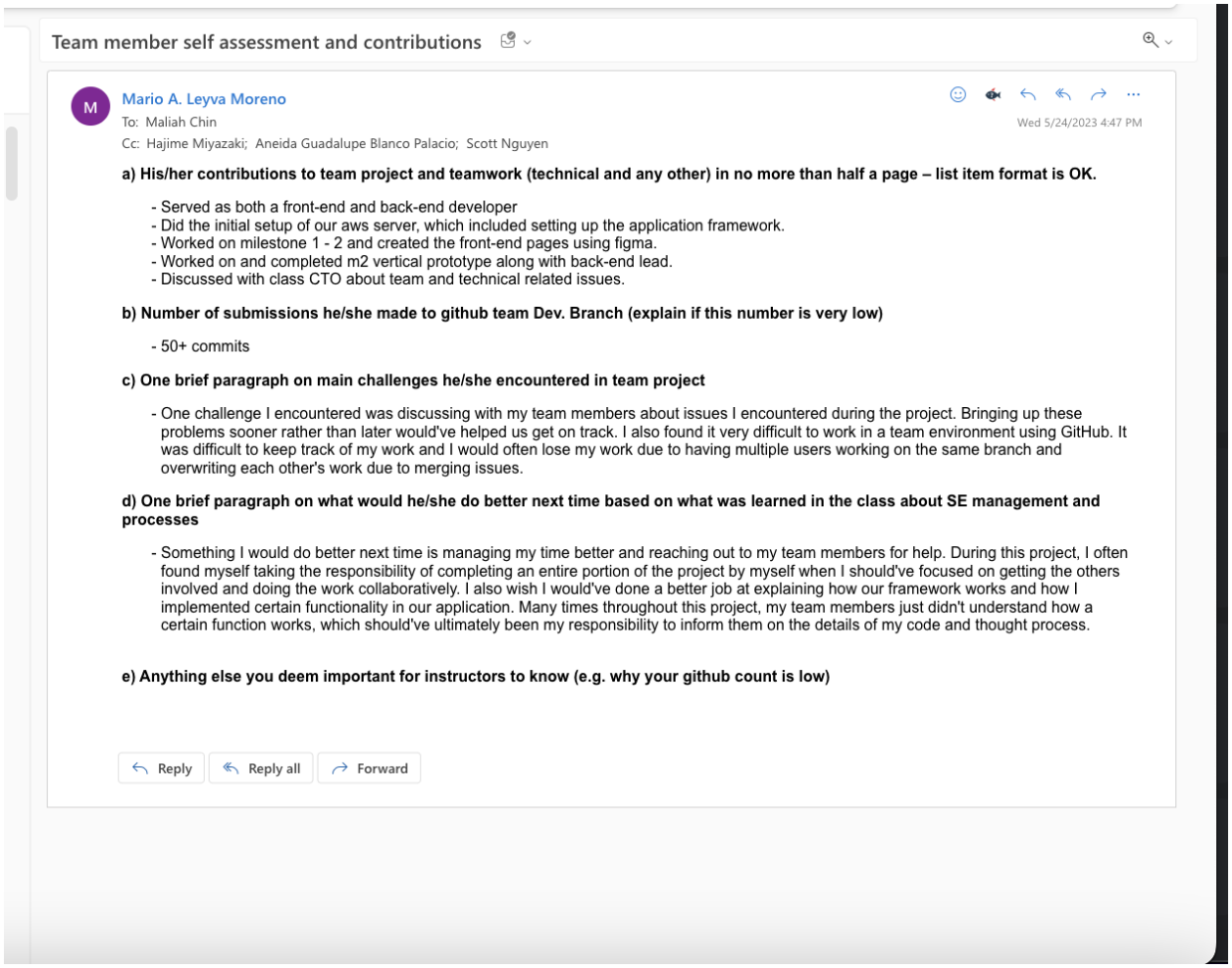
d) One brief paragraph on what would he/she do better next time based on what was learned in the class about SE management and processes

I would definitely have been more on top of my own time management. I would check in more frequently with my team members and perhaps engage more with in person meetings. Of course, everyone's busy work schedules and heavy classes were hard to work around, but if this was my main job and the only thing on my plate I think I would have been able to do better. I would also probably contribute more to the styling of the application and showcase my creativity a little better.

e) Anything else you deem important for instructors to know (e.g. why your github count is low)

I would like to say that my team was one of the teams that had one less member than everyone else. Not that it's an excuse for some of the functional specs that may or may not be there, but it was difficult for us to have one of our members have to go between front end and backend. I strongly recommend that Mario gets the frontend lead extra credit due to his hard work and great teamwork.

Mario:



Team member self assessment and contributions

Mario A. Leyva Moreno

To: Maliah Chin

Cc: Hajime Miyazaki; Aneida Guadalupe Blanco Palacio; Scott Nguyen

Wed 5/24/2023 4:47 PM

a) His/her contributions to team project and teamwork (technical and any other) in no more than half a page – list item format is OK.

- Served as both a front-end and back-end developer
- Did the initial setup of our aws server, which included setting up the application framework.
- Worked on milestone 1 - 2 and created the front-end pages using figma.
- Worked on and completed m2 vertical prototype along with back-end lead.
- Discussed with class CTO about team and technical related issues.

b) Number of submissions he/she made to github team Dev. Branch (explain if this number is very low)

- 50+ commits

c) One brief paragraph on main challenges he/she encountered in team project

- One challenge I encountered was discussing with my team members about issues I encountered during the project. Bringing up these problems sooner rather than later would've helped us get on track. I also found it very difficult to work in a team environment using GitHub. It was difficult to keep track of my work and I would often lose my work due to having multiple users working on the same branch and overwriting each other's work due to merging issues.

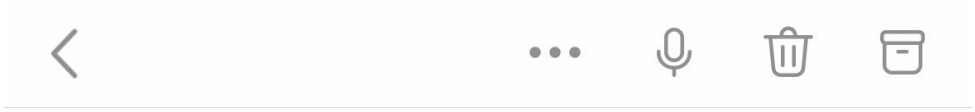
d) One brief paragraph on what would he/she do better next time based on what was learned in the class about SE management and processes

- Something I would do better next time is managing my time better and reaching out to my team members for help. During this project, I often found myself taking the responsibility of completing an entire portion of the project by myself when I should've focused on getting the others involved and doing the work collaboratively. I also wish I would've done a better job at explaining how our framework works and how I implemented certain functionality in our application. Many times throughout this project, my team members just didn't understand how a certain function works, which should've ultimately been my responsibility to inform them on the details of my code and thought process.

e) Anything else you deem important for instructors to know (e.g. why your github count is low)

Reply Reply all Forward

Hajime:



a) His/her contributions to team project and teamwork (technical and any other) in
no more than half a page – list item format is OK.

I spent a lot of time into completing the application development in this semester. As a back-end lead, I made registrations and some search functionality in this project.

b) Number of submissions he/she made to github team Dev. Branch (explain if this
number is very low)
I committed 16 times at least but I feel like this isn't a good number of commits as a backend.

c) One brief paragraph on main challenges he/she encountered in team project
main challenge I encountered in team was that some code I made didn't work with other functionality that other backend lead created for the different purposes. some time we had a problem with communicating with team members because of the different school schedule.