Pasta La Vista

Final Project Technical Report

SE/COM S 3190 – Construction of User Interfaces
Spring 2025

Team Members:
Alec Ferchen - aferchen@iastate.edu
Mahri Antongiovanni - manton13@iastate.edu
May 11, 2025

1. Introduction

Hello, we are Alec Ferchen, a computer science major, and Mahri Antongiovanni, a cybersecurity engineering major. We are both sophomores here at Iowa State and have taken a few classes that will aid us in this project. Alec has taken classes such as 3090 (Software Development Practices) and 3210 (Computer Architecture and Machine Level Programming), which gives him prior knowledge on how to make a cohesive webpage. Mahri has taken a course called SE 186, where, as a team, you get to create a project of your choosing. Mahri's team created a quiz game similar to that of "Kahoot!" in a webpage format.. This prior experience will help her to create an appealing and functional web page for this project.

We chose option 2 for our project so that we can start fresh with the new tools we have and better implement what is needed for this project. For our project, we decided to create a website for a new and original Italian restaurant using the skills we have learned in the last half of this course, such as Node.js, React, and databases. The title of our restaurant is "Pasta la Vista," and we will work to offer a wide variety of dishes that can be ordered and paid for through the website. In addition to displaying the dishes, we will also have data about the customers and offer promotions and reward points for them to use at later purchases.

The goal for this project was to build on our preexisting skills to create a functional and cohesive website that users would enjoy interacting with. This was also our main motivation to create a website that was both easy to use for our audience and looked appealing to users as well. Some other goals and motivations we had while creating this course include keeping user data in profiles that can be accessed later, keeping products and dishes up to date and modifiable, and overall having a simplistic user-oriented design.

2. Project Description

The user flow of this website is very intuitive. The website starts off on the main browsing page, where the user can add items to their cart. After the user adds everything they want to the cart, they can review their cart and either proceed to payment or continue browsing. If the user chooses to proceed to payment, then the user will fill out a simple form and then receive a summary of their order once submitted. The user can also click on these tabs and their profile using the navigation bar at the top of the screen as another option for them to navigate. Major features for this project that are available to the user include:

- A browsing page that features all of the dishes we serve at the restaurant, including an image, price, calorie amount, and brief description.
- Two different kinds of searching functionality are used to display the items users are looking for.
- A way to pay for the dishes they wish to purchase.
- User login information is required to create an account with the restaurant and access data such as reward points.
- A navigation bar and numerous buttons for the user's convenience in interacting with the website.

There are three parts affected by CRUD operations. The first two are the products and dishes offered on the website and the admin accounts, which is through MongoDB, and the other is the user account information through SQL. For the actual products on the website, the read operation is

through the get function, which reads in all of the product information into the browsing page for the user to view products. The create is through the post function, which allows the owner to add dishes to the website. The update is through the put function, which allows the owner to access a product by its ID number and update its information. The delete is through the delete function, which allows the owner to remove products and dishes from the website. For the admin privileges you can create new admin accounts, update existing ones, as well as delete and view them.

The other CRUD operations affect the user profile data. The read functionality allows the owner to get all of the profiles and view their information, but it also allows a specific user to get their information based on their ID, so they can view their individual account. The create operation allows customers to create an account at the restaurant to see their information, such as reward points. The delete functionality allows the user to remove their account from the restaurant based on their user ID. The update functionality allows users to gain reward points with their purchases, as well as change aspects of their profile, like their birthday, username, password, and email, to name a few.

3. File and Folder Architecture

The basic structure of our code is split up into front and backend folders. Our backend is a simple db.js and index.js. In the DB, we have the route to a locally hosted database, which sends its data to the index file. Index uses this, and is where the actual methods for sending and receiving data are. In the frontend, we have an assets folder for all our images we used (the links to them), a components folder for all of our actual pages (like navigation bar and browse menu), and data folder for the initial menu items added in the first mini assignment

```
MN_7/
├── backend/
│   ├── db.js
│   └── index.js
└── frontend/
    └── initial_app/
        └── src/
            ├── main.jsx
            ├── app.jsx
            ├── assets/
            │   └── images
            ├── components/
            │   ├── about.jsx
            │   ├── browse.jsx
            │   ├── cart.jsx
            │   ├── navigation.jsx
            │   ├── payment.jsx
            │   └── ect
            └── data/
                └── menu_items.jsx
```

4. Code Explanation and Logic Flow

4.1. Frontend–Backend Communication

API requests are made in our project using Express. First, the server listens on port 8080, then CORS and Express are required, and finally, we have our endpoints for Mongo and SQL. Some of these are GET, POST, PUT, and DELETE requests. For example, to get users we send an endpoint to SQL with "/users/id" and get back the information on that specific ID's user

4.2. React Component Structure

We used states to set and use the cart, user, step, and order summary. Each step was initialized to carry the necessary props to its page, an example being the cart page gets the cart, setCart, and setStep props. All of this is initialized in our App.js.

4.3. Database Interaction

The Mongo database is used to store all of the product information which includes the items ID number, name of the dish, price, calorie amount, a brief description of what it is, an image, and what course it belongs to. This database is used when we need the products to show up on the screen such as when we needed to populate the browse part of the website. It is also used for the admin accounts so that not anyone can change the products or administrative accounts for the restaurant. When information from the database needs to be used it connects to the database over the local host and gets the information through a get method in the index.js file. The database is also used when information about the products is added, updated, or removed using CRUD operations.

Our SQL database is used to store user information. We have email, password, username, birthday, rewards points, and phone numbers for each user when they sign up. The user puts initial data in and has their reward points set to 0. They can then later add a phone number and change any part of their information, which is instantly uploaded to the database. The user also has access to delete their account if they so pleased. The only bit of data not controlled fully by the user is reward points of which they start with 0 and have no way of increasing unless buying from the restaurant. As of now there is no way to cash in rewards points but the system is still in place.

4.4. Code Snippets

Backend, getting user information

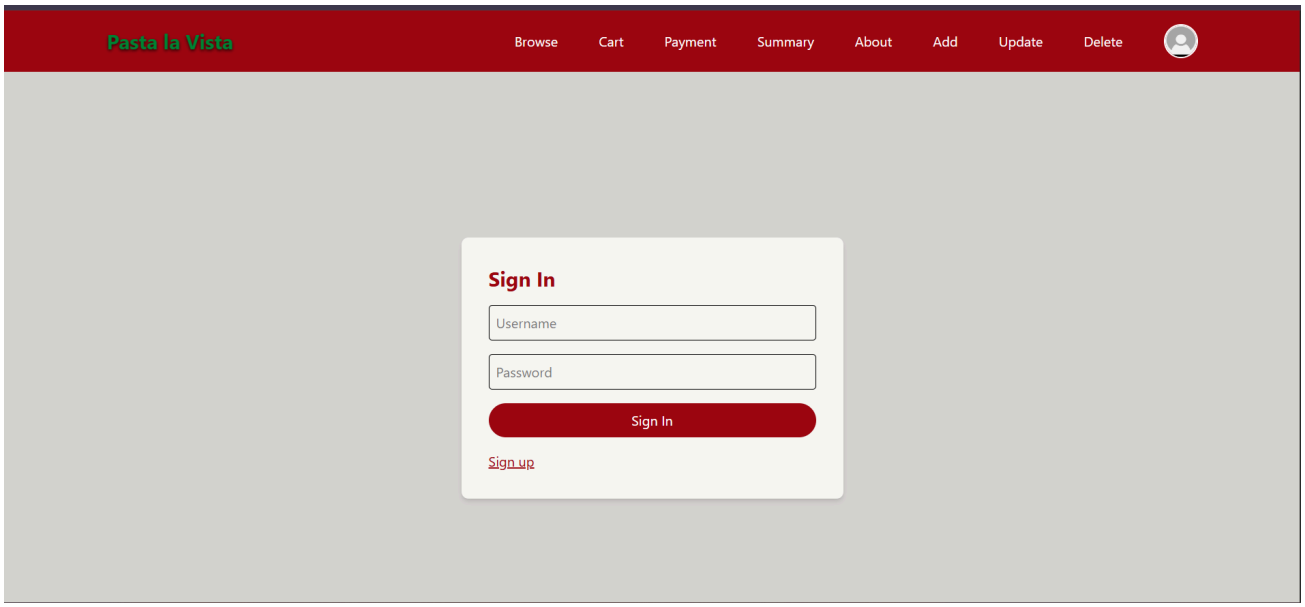
```
app.get("/users", async (req, res) => {
  try {
    const query = "SELECT * FROM pasta_users";
    const [result] = await db.query(query); // Execute the query and wait for the result
    console.log("Success in Reading MySQL");
    res.status(200).send(result); // Send the results as the response
  } catch (err) {
    // If an error occurs, catch it and send an appropriate error response
    console.error("Error in Reading MySQL :", err);
    res.status(500).send({ error: "An error occurred while fetching items." });
  }
});

app.get("/users/:id", async (req, res) => {
  try {
    // Read id from frontend
    const id = req.params.id;
    const query = "SELECT * FROM pasta_users WHERE id = ?";
    const [result] = await db.query(query, [id]); // Ensure to use array for parameters even if it's just one
    console.log("Success in Reading MySQL");
    res.status(200).send(result);
  } catch (err) {
    // If an error occurs, catch it and send an appropriate error response
    console.error("Error in Reading MySQL :", err);
    res.status(500).send({ error: "An error occurred while fetching items." });
  }
});
```

Navigation bar setup, switches steps based on button clicked

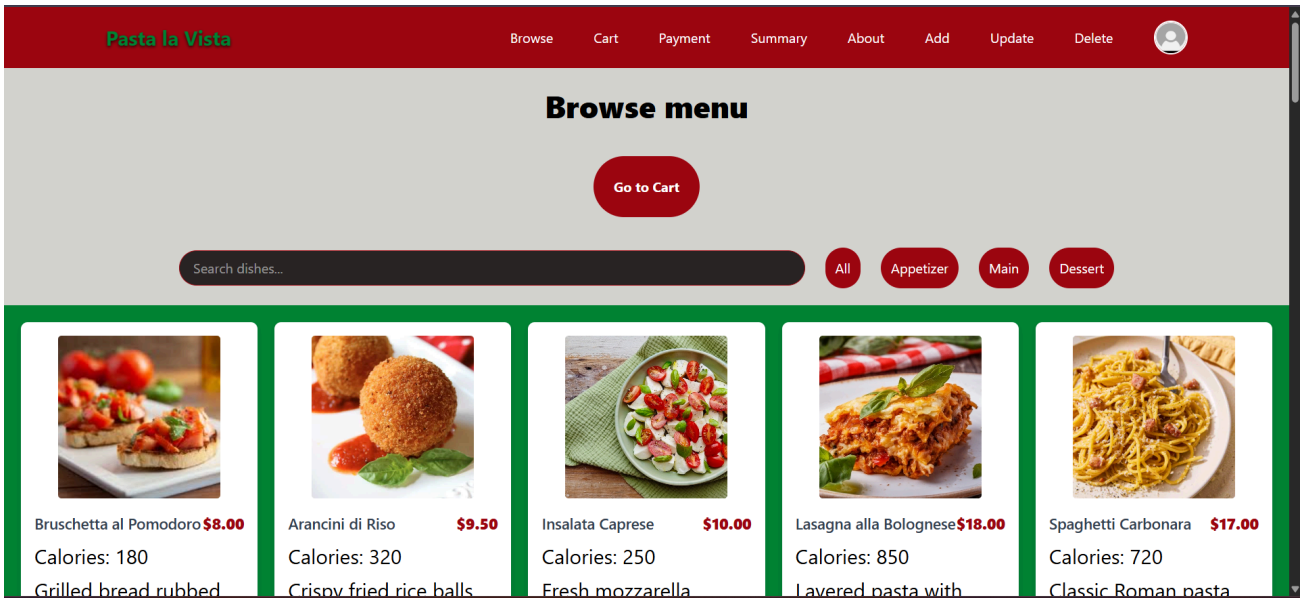
```
const Navigation = ({ setStep }) => {
  return (
    <nav className="bg-red-800 text-white p-4 shadow-md">
      <div className="flex justify-between items-center max-w-7xl mx-auto">
        <h1 className="text-2xl font-bold text-green-700 drop-shadow-[0_2px_2px_rgba(0,0,0,0.8)]">
          Pasta la Vista
        </h1>
        <div className="flex space-x-4">
          <button
            onClick={() => setStep("browse")}
            className="hover:bg-green-700 px-4 py-2 rounded-full transition-all"
          >
            Browse
          </button>
          <button
            onClick={() => setStep("cart")}
            className="hover:bg-green-700 px-4 py-2 rounded-full transition-all"
          >
            Cart
          </button>
          <button
            onClick={() => setStep("payment")}
            className="hover:bg-green-700 px-4 py-2 rounded-full transition-all"
          >
            Payment
          </button>
          <button
            onClick={() => setStep("summary")}
            className="hover:bg-green-700 px-4 py-2 rounded-full transition-all"
          >
            Summary
          </button>
          <button
            onClick={() => setStep("about")}
            className="hover:bg-green-700 px-4 py-2 rounded-full transition-all"
          >
            About
```

5. Web View Screenshots and Annotations

This image displays the sign in page for the user accounts. At the top you can see the navigation bar so that users can click on the tabs to go to separate aspects of the website. Further down the page is the sign in box for the user where the user enters the username and password to gain access to their account. If the user doesn't have an account already then they can sign up using the link below.



This is our browse menu featuring the navigation bar, search options, and product display. This page essentially displays all of the products and then ways to filter through them to find exactly what you are looking for. The user can search by typing the item in the search bar or by clicking the buttons to filter based on the course. The user can also click on the navigation bar tabs to navigate to different aspects of the website. When looking at the products the user can view their information and add them to the cart. Lastly the user can hit the go to cart menu to view what is in their cart.



with fresh tomatoes, basil, and olive oil.
Add to cart

and ragù, served with marinara sauce.
Add to cart

drizzled with balsamic glaze and extra virgin olive oil.
Add to cart

sauce, béchamel, and Parmesan cheese.
Add to cart

Pecorino Romano, and black pepper.
Add to cart

Risotto ai Funghi Porcini **$19.00**
Calories: 680
Creamy Arborio rice cooked with porcini mushrooms and Parmesan cheese.
Add to cart

Pollo alla Parmigiana **$20.00**
Calories: 790
Breaded chicken breast topped with marinara sauce and mozzarella, served with spaghetti.
Add to cart

Fettuccine Alfredo **$16.50**
Calories: 840
Rich pasta tossed in a creamy Parmesan and butter sauce.
Add to cart

Pizza Margherita **$15.00**
Calories: 700
Wood-fired pizza with tomato sauce, fresh mozzarella, and basil.
Add to cart

Osso Buco alla Milanese **$24.00**
Calories: 900
Slow-braised veal shank served with saffron risotto.
Add to cart

Add to cart

Tiramisu **$8.50**
Calories: 450
Classic layered dessert with espresso-soaked ladyfingers, mascarpone cream, and cocoa.
Add to cart

Cannoli Siciliani **$7.00**
Calories: 380
Crispy pastry shells filled with sweet ricotta cream and chocolate chips.
Add to cart

Panna Cotta **$7.50**
Calories: 300
Silky vanilla cream dessert served with mixed berry coulis.
Add to cart

Affogato al Caffè **$6.50**
Calories: 200
Vanilla gelato drowned in a shot of hot espresso.
Add to cart

Zuppa Toscana **$9.00**
Calories: 400
Hearty soup with Italian sausage, kale, potatoes, and cream.
Add to cart

Arancini di Riso **$9.50**

Calories: 320

Crispy fried rice balls filled with mozzarella and ragù, served with marinara sauce.

Add to cart

Carpaccio di Manzo **$10.50**

Calories: 300

Paper-thin slices of raw beef tenderloin, served with a garish.

Add to cart

Calamari Fritti **$12.00**

Calories: 450

Crispy fried Squid with lemon and a side of marinara sauce.

Add to cart

Melanzane alla Parmigiana **$9.50**

Calories: 420

Baked Eggplant with tomatoes and mozerella cheese topped with parmesean and basil.

Add to cart

Gnocchi al Pesto Genovese **$18.00**

Calories: 700

Potato gnocchi tossed in basil pesto sauce with pine nuts and parmesean cheese.

Add to cart

Tagliatelle al Ragu Bolognese **$18.50**

Calories: 750

Fettuccine pasta with beef and pork ragu sauce.

Add to cart

Saltimbocca alla Romana **$24.00**

Calories: 480

Veal Medallions topped with prosciutto, sage, and white wine sauce.

Add to cart

Branzino al Forno **$27.50**

Calories: 410

A whole roasted sea bass braised with lemon, garlic, and a variety of herbs.

Add to cart

Filetto di Manzo al Barolo **$29.50**

Calories: 590

Grilled beef tenderloin topped with a Barolo wine reduction.

Add to cart

Torta della Nonna **$8.00**

Calories: 390

Tart custard pie with pine nuts and powdered sugar.

Add to cart

Gelato Artigianale **$7.00**

Calories: 320

Torta Caprese **$9.50**

Calories: 420

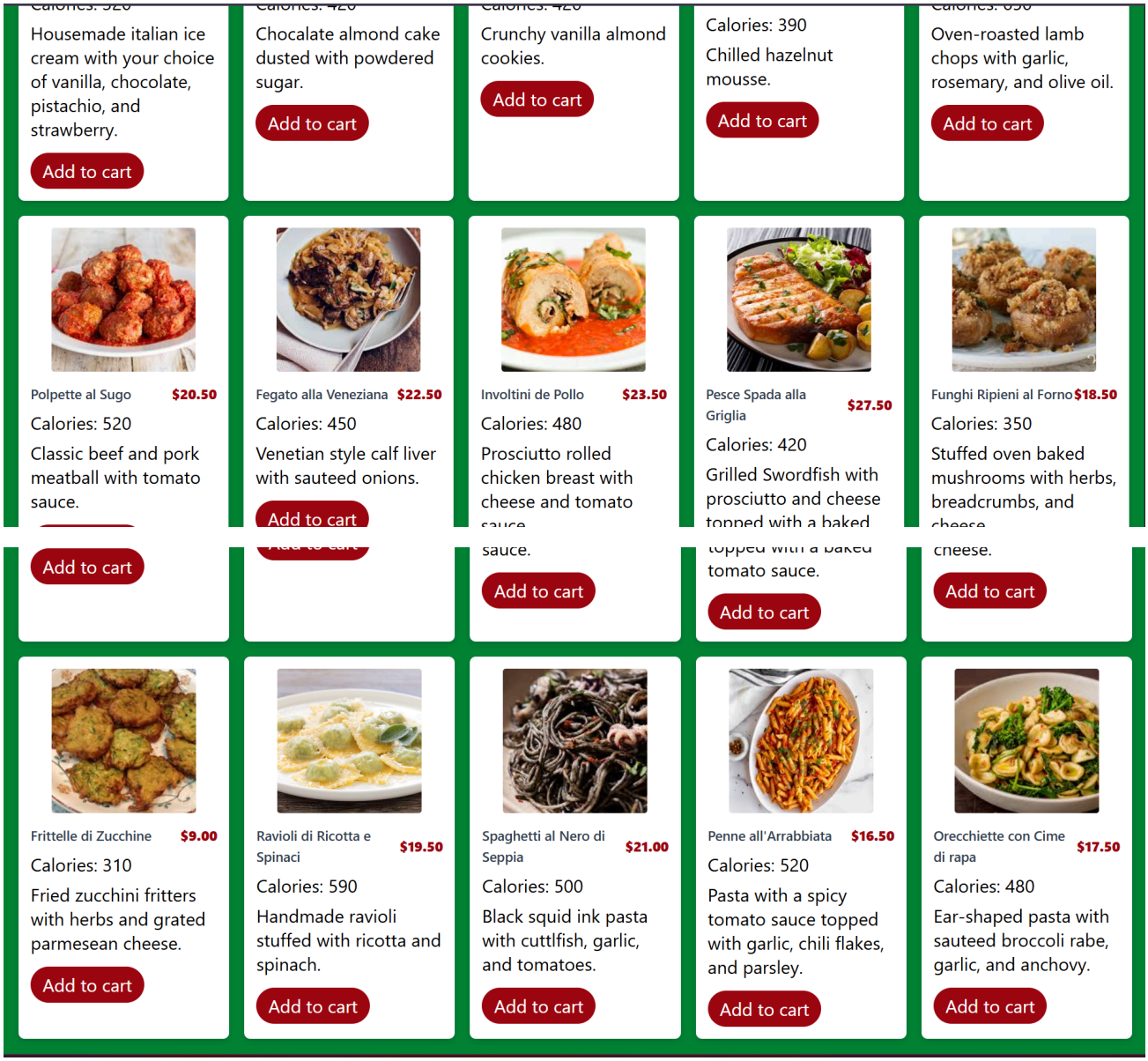
Biscotti di Prato **$9.00**

Calories: 420

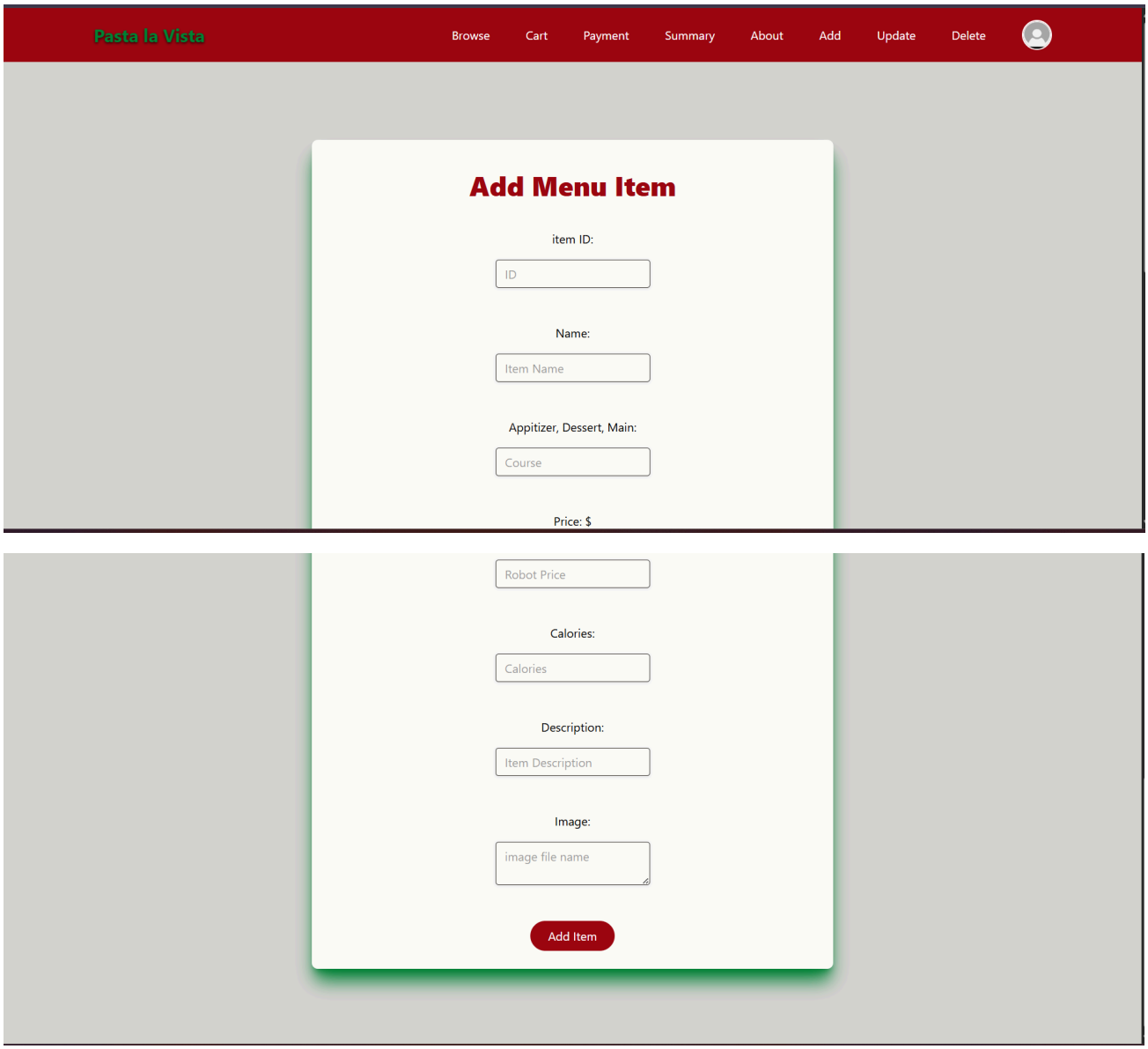
Semifreddo alle Nocciole **$8.50**

Angello al Forno **$26.50**

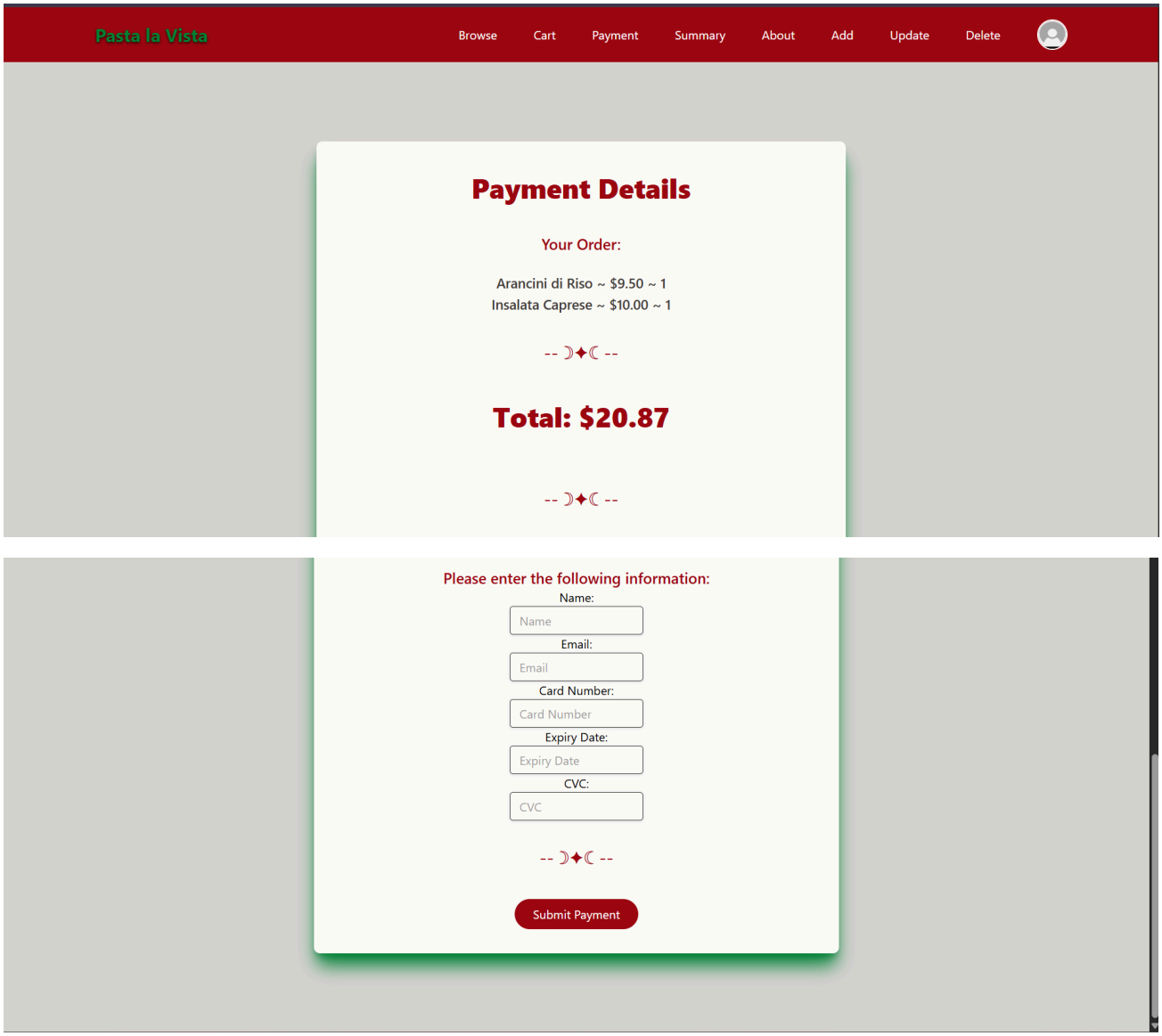
Calories: 650



This page allows for the addition of an item menu to the webpage. At the top there is a navigation bar which has been talked about previously. Below is the form to add an item to the menu where the user enters in the items ID, the name, course category, price, calories, description, and image for the additional item.



This is the payment details screen. At the top their is a navigation bar and below if the details of the users order. On this page the user can review what they ordered, see the total cost of their order, and submit a payment. To submit a payment the user will submit a form that collects their name, email, card number, expiry date, and CVC to complete the transaction.



6. Installation and Setup Instructions

To run the frontend all that is needed is running “sudo npm run dev” on our initial_app in the frontend folder. The backend for SQL and MongoDB is just running “node index.js” which sets it to run with SQL locally. However before running “node index.js” make sure that you are connected to MongoDB, and have installed express, mongodb, cors, and body-parser.

7. Contribution Overview

Alecs features:

- Pages: Browse, Cart, About, Navigation, UserInfo, User, PutUserInfo
- Specifics: Full CRUD for user information and reward points, navigation bar across all pages, real-time search bar, auto incrementing reward points for each dollar spent, easy updating for each part of a user's information, and a prompt telling you what your previous value was for the field youre changing using useEffect

Mahris features:

- Pages: Payment, Summary, Browse, AddAdmin, PutAdmin, DeleteAdmin, Admin, AdminInfo, AddProducts, PutProducts, DeleteProducts, Footer
- Specifics: Full CRUD for product and admin implementation, adding in data for products listed on menu, search functionality by course category, form creation for the payment of the user, creation of summary page, overall design and layout of all pages within the website, the website footer, and the ability to add, update, and delete products from the menu as well as admin accounts

8. Challenges Faced

1. Getting started with the backend was a difficult transition; my port wasn't working, my password needed to be reset, and I had my endpoint typed out wrong in my code. All of which slowed me down from actually getting to test and play with the backend stuff. Both of us had issues with getting our databases to interact accordingly with the rest of the website and to troubleshoot this we referenced and used discord to see if others were facing similar issues and looked back on previous class activities such as activities 13 and 14 ad a reference point to start incorporating databases into the project.
2. We ran into a problem of having too many buttons and pages to go to and didnt have a navigation bar like the midterm assignment so thinking of where to place the user profile, about page, and cart buttons was difficult until we made a navigation bar which solved all of those problems and was surprisingly easy to set up. This was done through looking at Tailwind's website and their “premade” navigation bars, seeing how they worked and using that logic and Tailwind structure to make our own that worked for our site.

9. Final Reflections

Mainly, we would do more backend work. It was hard to set up, but once it was up and running, changing things and getting data to and from it was extremely easy. We wish that had been set up from the beginning, so we could have planned out more to put in them to use in the project and built the project around the backend instead of the backend around the project. We would also be interested in how to make the website more secure. Most of the ways that we could create an account or gain admin access wouldn’t be the most accurate representation of how a secure website would implement this, so it would be fun to add more security to the site. We would also like to improve the overall look and aesthetics of the website. We used basic Tailwind and CSS but mainly focused on the features we were implementing in the project. There are definitely better-looking extensions out there that could be used to make our site look a lot more professional, but for now, it is functional and looks decent considering our main focus was on using what we learned in the second hald of this course. The last thing We would change is making a non-local backend and having all of our data on one backend server instead of two separate softwares on local devices. We would do this to make it so the site can fully be run by anyone and not just only ever be “halfway” run as the backend is locally on two separate devices. We would love to create something that could be shared with others easily because that is the goal of most sites.