Dan’s Frappuccino Paradise

Group 8

John Belnap, Master of Documentation

Caden Harris, King of the Burndown Charts

Kollin Murphy, Project Manager

Trenton Peters, Senior Software Engineer

Kollin Murphy

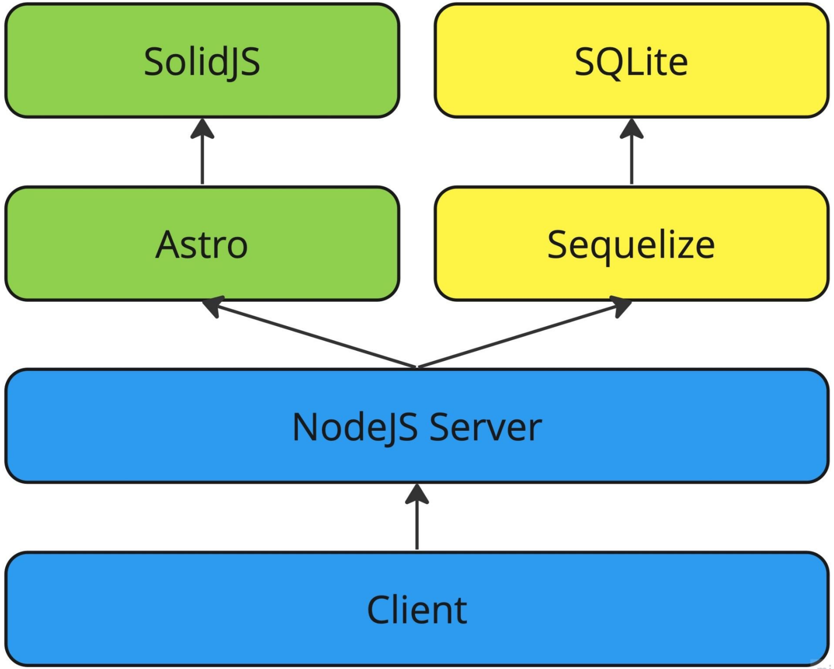
Project Overview

Project Architecture

In the early stages of the project, we proposed several different types of architecture. These included a Django full-stack application, a React frontend & ExpressJS backend, and Astro with a SolidJS frontend.

Due to a lack of confidence in our abilities to create a clean project using Django, we decided to use Astro. Astro is modern site builder that allows developers the freedom to use any front-end UI library. We decided to pair it with SolidJS due to its simplicity, reliability, and performance.

Furthermore, one of our team members had significant experience using Astro and was confident that it had a small enough learning curve to use for this project.



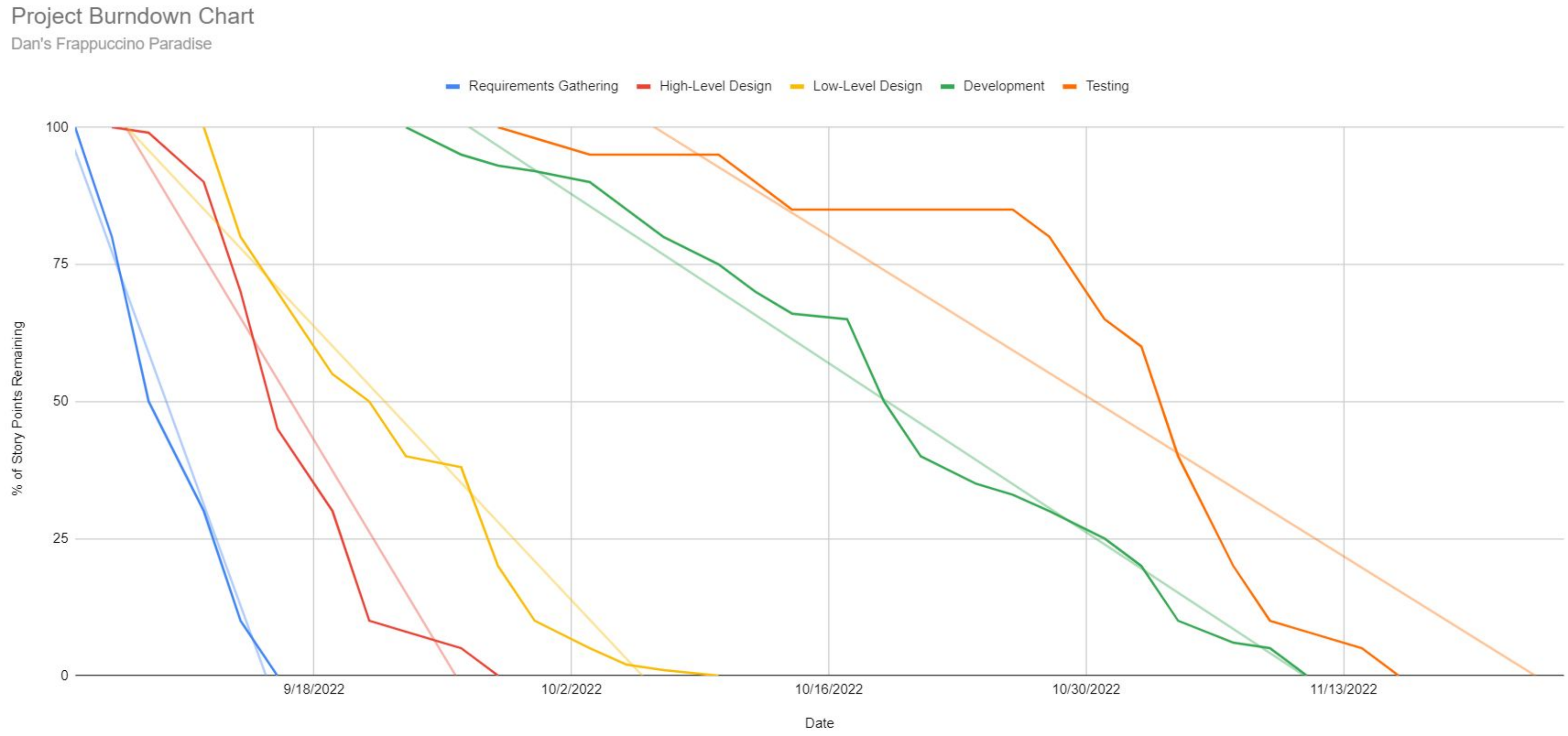
Project Design

To design our project, we first wrote up a complete requirements document. With our requirements in hand, we were able to design a low-fidelity prototype as a group. We used brainstorming techniques where every idea has merit to generate as many ideas as we possibly could. We explored some of those ideas as we drew out wireframe sketches, and discarded others that weren't as promising.

At the end of our design meeting, we had a 12-page design document with sketches of each page of the website. This served as a starting point for the rest of our project. We frequently referred back to it throughout the development process.

Agile Practices

- Planning Poker
  - This allowed us to express our expectations of the duration of various tasks without any anchoring of the estimates. This helped in dividing tasks between team members and assigning story points.
- Daily Standup Documents
  - These regular documents helped us to stay focused and to follow up on previous tasks. It made sure that we were able to account for any potential loose ends during the development process.
- Sprint Planning & Retrospective Meetings
  - During our sprint planning meetings, we were able to look at the requirements document and create tasks to accomplish specific requirements.
  - During our retrospective meetings, we were able to evaluate the direction the project was heading and ensure that the requirements were being successfully met.



Burndown Chart

Team Velocity

- Sprint 1 (9/5 - 9/16): 34 points per sprint
- Sprint 2 (9/19 - 9/30): 46 points per sprint
- Sprint 3 (10/3 - 10/14): 23 points per sprint
- Sprint 4 (10/17 - 10/28): 24 points per sprint
- Sprint 5 (10/31 - 11/11): 35 points per sprint
- Sprint 6 (11/14 - 11/25): 5 points per sprint

Average velocity: 27.8 points per sprint

Deployment

The project is now ready to move into the deployment stage.

We have written unit tests for all API handlers using the Jest testing framework and have achieved 100% test coverage. This gives us significant confidence in the correctness of our code. Regression tests have also been implemented. We plan to integrate the tests as part of the CI/CD process as we prepare for deployment.

For deployment, we plan to use Netlify with the Astro Netlify Adapter. We will setup a MySQL database using Amazon Web Services RDS to host the persistent storage.

For CI/CD we will create a GitHub Action to first run the unit tests and then start the deployment on Netlify if all tests pass.

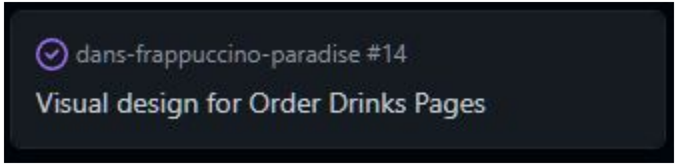
Requirements Tracking

Placing an Order

Completing an Order - Requirements

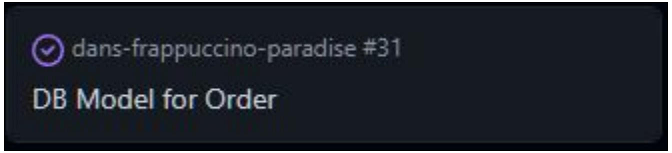
- An authenticated Account (“the Account”) is presented with a **list of all Products**, each with an image and a price
- When the Account selects a Product, they are taken to a **Product page** which lists the Visible ProductIngredients, each with a quantity, and a size selector, from which they can choose **Small (8 oz), Medium (16 oz), Large (24 oz)**
    - Changing the Size will modify the count of each ProductIngredient by a scale of 1x (Small), 2x (Medium), or 3x (Large), which will in turn **affect the item price**
    - The Account can **add new ProductIngredients** to the item, and remove ProductIngredients from the item
- The Account can click a **Place Order button** which will do the following:
    - Create an Order if there is no unpaid order for the associated Account
    - Add an OrderProduct to the Order
    - Add any applicable OrderAddOns to the OrderProducts
    - **Place the Order**
    - Redirect them to their Account page, where they can see recent Orders and their current fulfillment status
- Employees can **order on behalf** of another User on this page by searching for them by their username

Completing an Order - Scrum Tasks



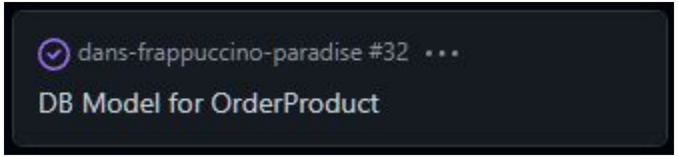
dans-frappuccino-paradise #14

Visual design for Order Drinks Pages



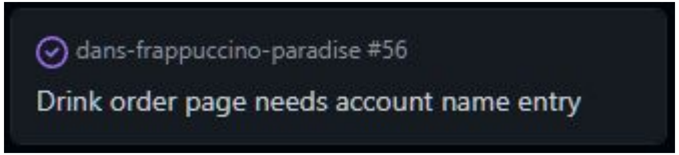
dans-frappuccino-paradise #31

DB Model for Order



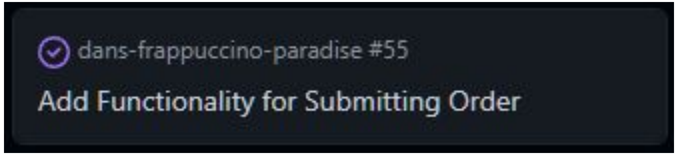
dans-frappuccino-paradise #32

DB Model for OrderProduct



dans-frappuccino-paradise #56

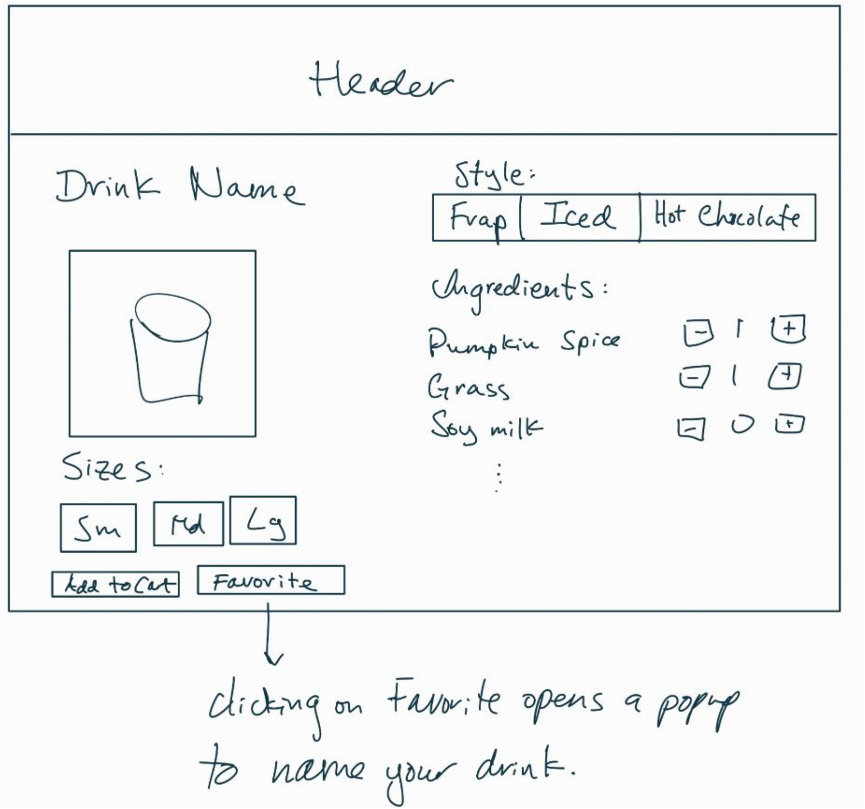
Drink order page needs account name entry



dans-frappuccino-paradise #55

Add Functionality for Submitting Order

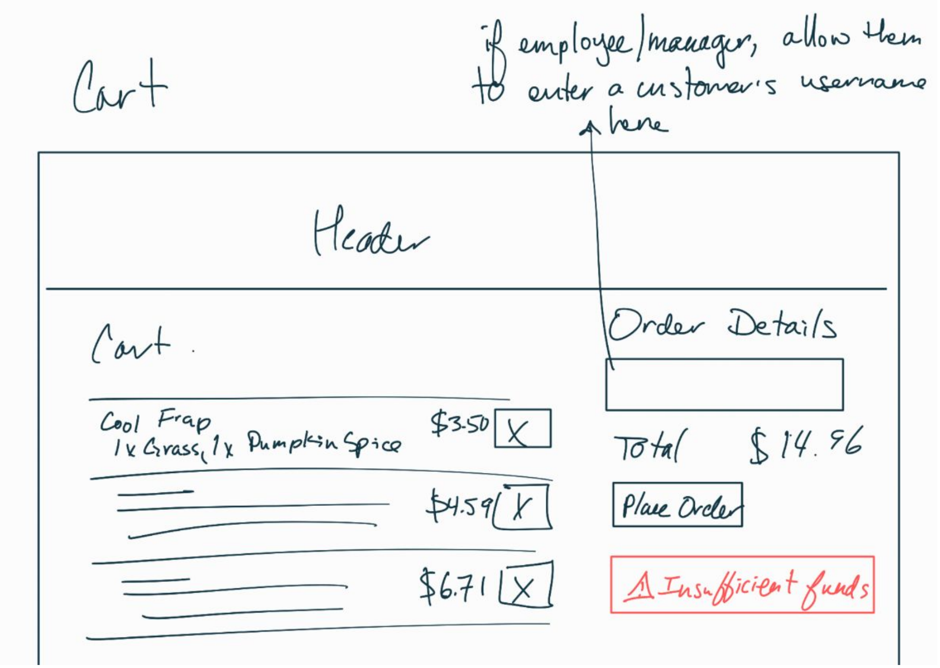
Completing an Order - Low Fidelity



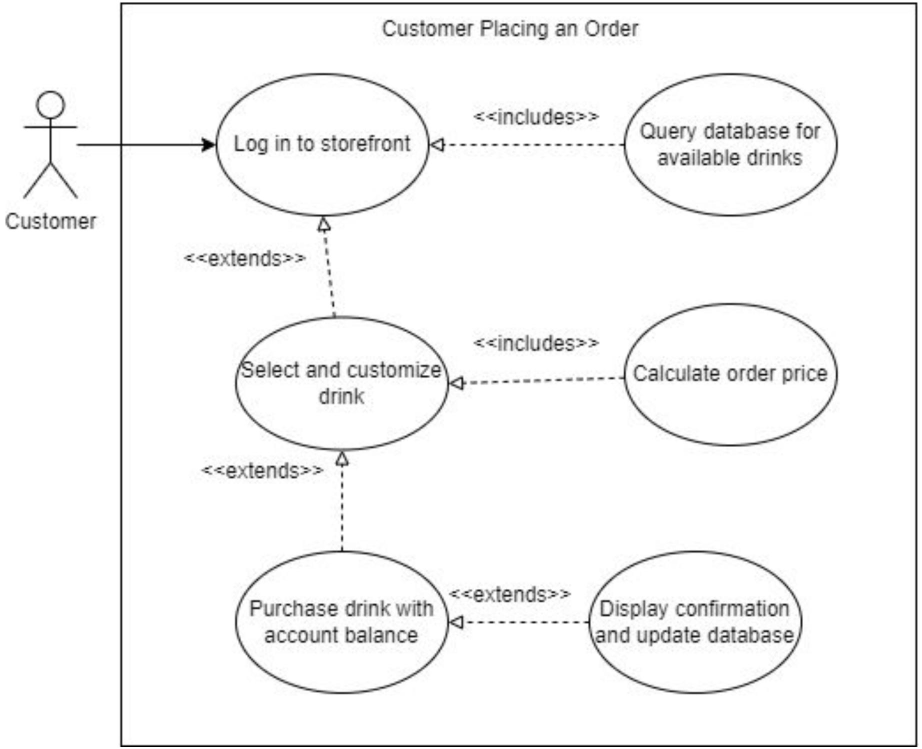
clicking on Favorite opens a popup to name your drink.

Cart

if employee/manager, allow them to enter a customer's username here



Completing an Order - Diagrams



Customer Placing an Order

Customer

Log in to storefront

<<includes>>

Query database for available drinks

<<extends>>

Select and customize drink

<<includes>>

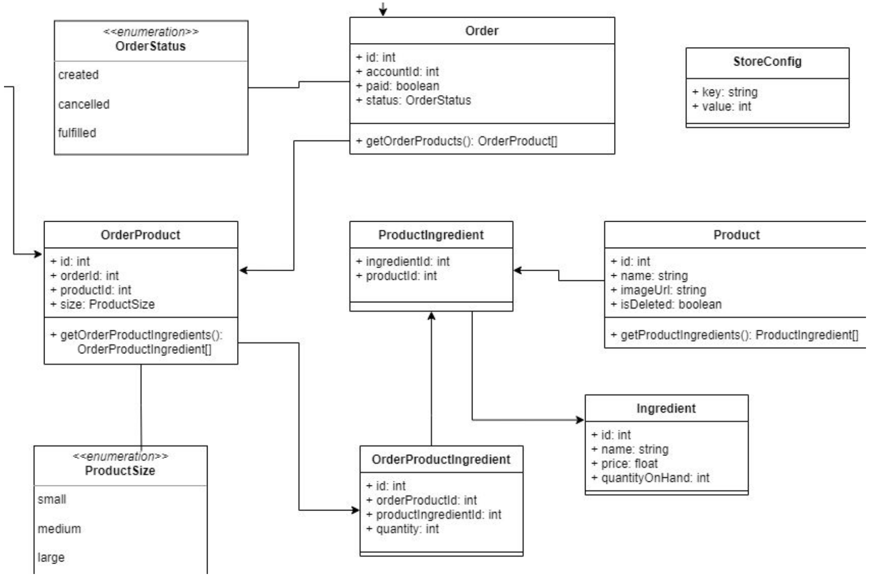
Calculate order price

<<extends>>

Purchase drink with account balance

<<extends>>

Display confirmation and update database



<<enumeration>>
OrderStatus

created

cancelled

fulfilled

Order

+ id: int
+ accountId: int
+ paid: boolean
+ status: OrderStatus

+ getOrderProducts(): OrderProduct[]

StoreConfig

+ key: string
+ value: int

OrderProduct

+ id: int
+ orderId: int
+ productId: int
+ size: ProductSize

+ getOrderProductIngredients(): OrderProductIngredient[]

ProductIngredient

+ ingredientId: int
+ productId: int

Product

+ id: int
+ name: string
+ imageUrl: string
+ isDeleted: boolean

+ getProductIngredients(): ProductIngredient[]

Ingredient

+ id: int
+ name: string
+ price: float
+ quantityOnHand: int

<<enumeration>>
ProductSize

small

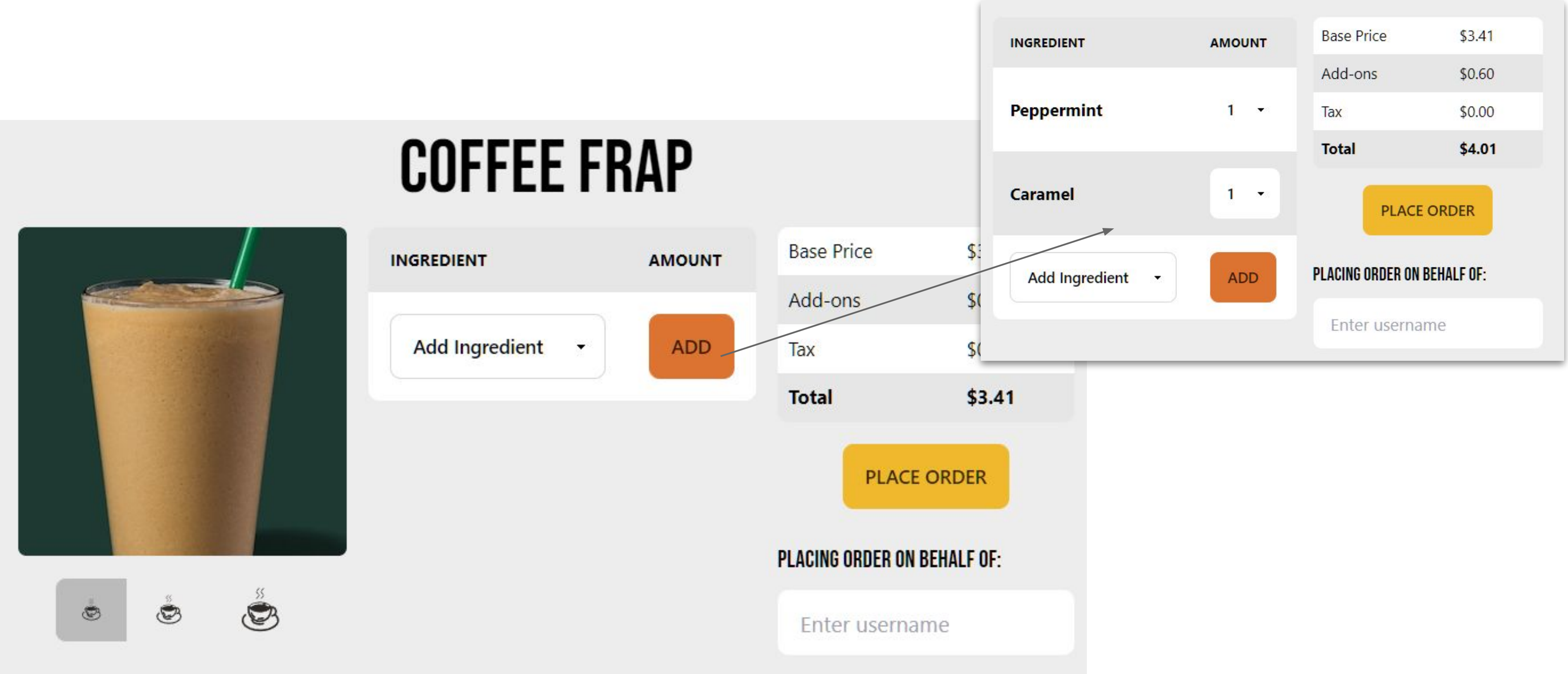
medium

large

OrderProductIngredient

+ id: int
+ orderProductId: int
+ productIngredientId: int
+ quantity: int

Completing an Order - Implementation



COFFEE FRAP

INGREDIENT	AMOUNT

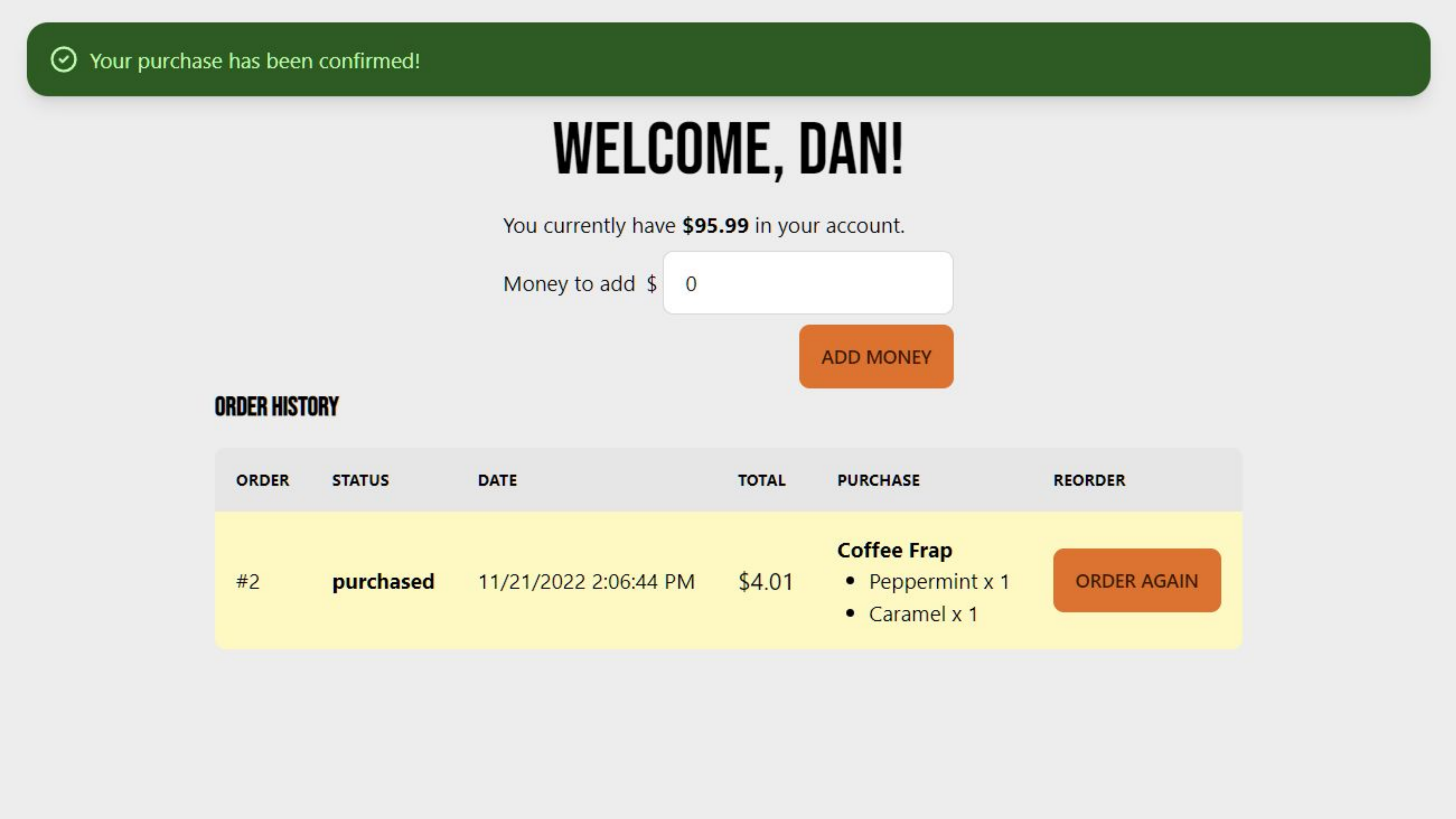
Base Price	$3
Add-ons	$0
Tax	$0
Total	**$3.41**

PLACING ORDER ON BEHALF OF:

INGREDIENT	AMOUNT
Peppermint	1
Caramel	1

Base Price	$3.41
Add-ons	$0.60
Tax	$0.00
Total	**$4.01**

PLACING ORDER ON BEHALF OF:



Your purchase has been confirmed!

WELCOME, DAN!

You currently have **$95.99** in your account.

Money to add $ 0

ORDER HISTORY

ORDER	STATUS	DATE	TOTAL	PURCHASE	REORDER
#2	**purchased**	11/21/2022 2:06:44 PM	$4.01	**Coffee Frap** • Peppermint x 1 • Caramel x 1	ORDER AGAIN

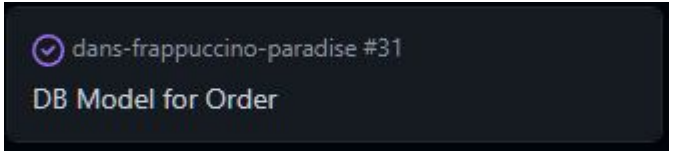
Requirements Tracking

Fulfilling an Order

Fulfilling an Order - Requirements

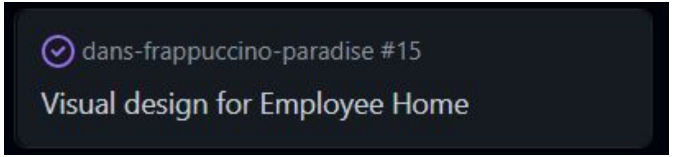
- After a “user” account places an order, they can see a list of their orders on their account page
  - This list is created by filtering from a list of all orders where the ID equals the user ID
- This page will now include a “purchased” order
- An employee or manager account is presented with a **list of all orders** that aren’t completed
  - This list is created by filtering from a list of all orders where the order status equals “purchased”
- When the employee or manager account clicks “FULFIL ORDER”, it shifts the status of the order from “purchased” to “fulfilled”

Fulfilling an Order - Scrum Tasks



dans-frappuccino-paradise #31

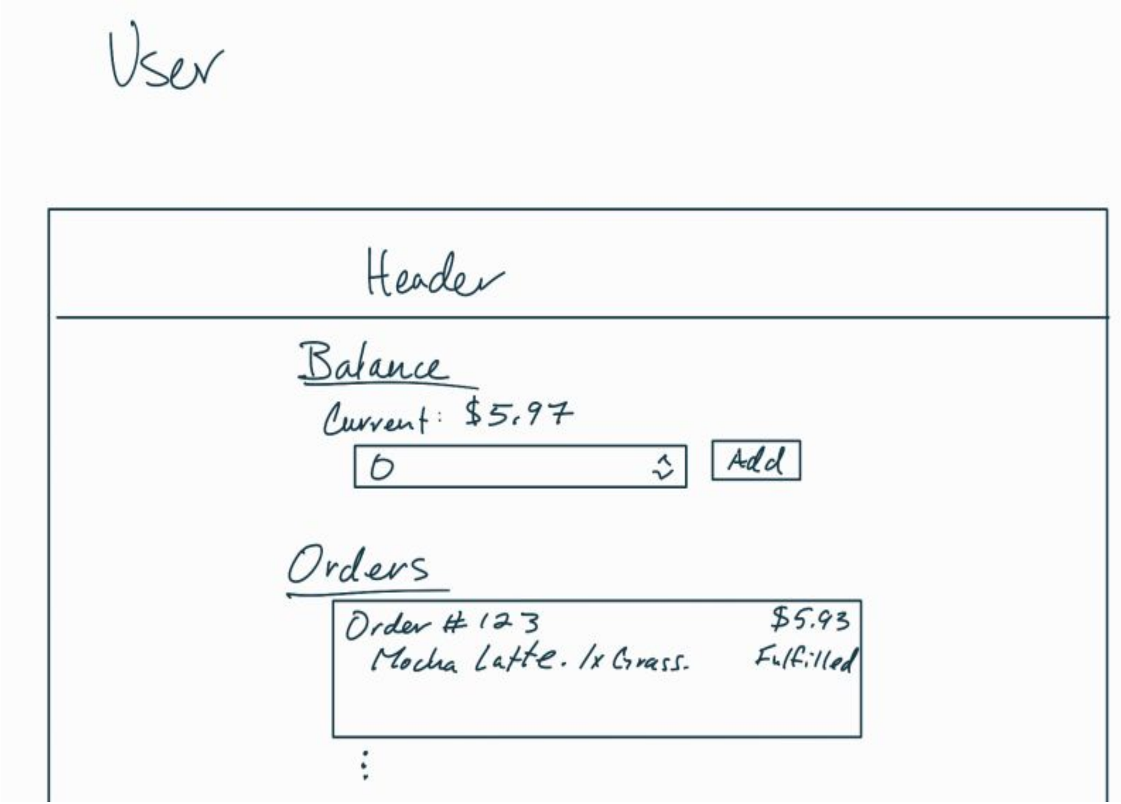
DB Model for Order

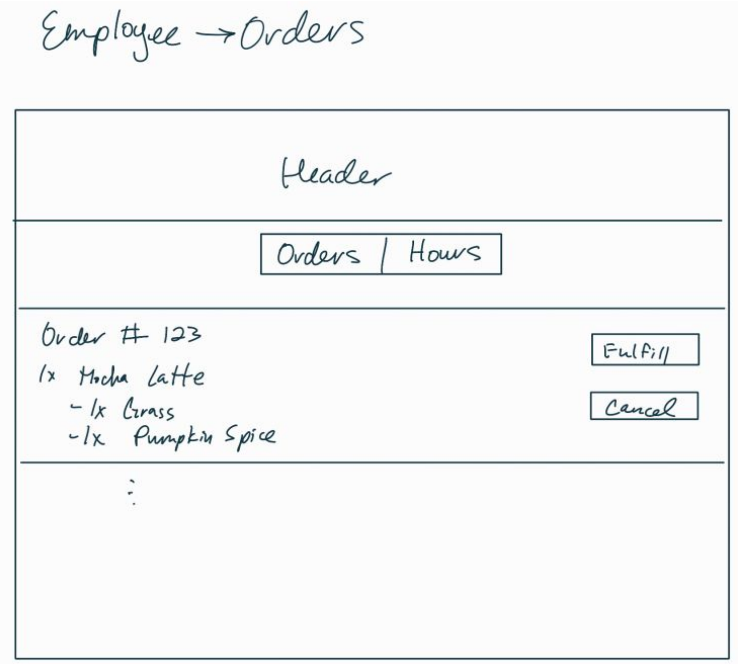


dans-frappuccino-paradise #15

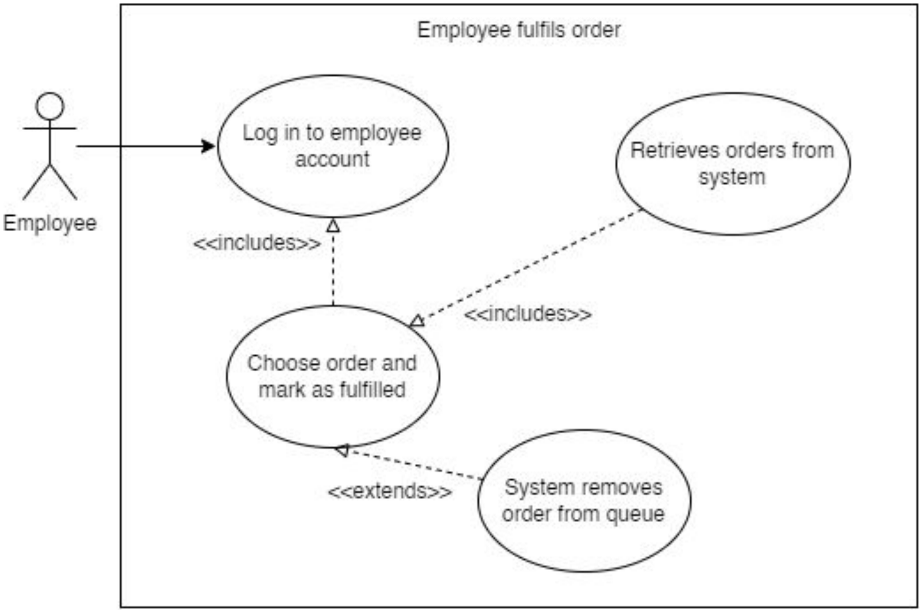
Visual design for Employee Home

Fulfilling an Order - Low Fidelity





Fulfilling an Order - Diagrams



Employee fulfils order

Employee

Log in to employee account

Retrieves orders from system

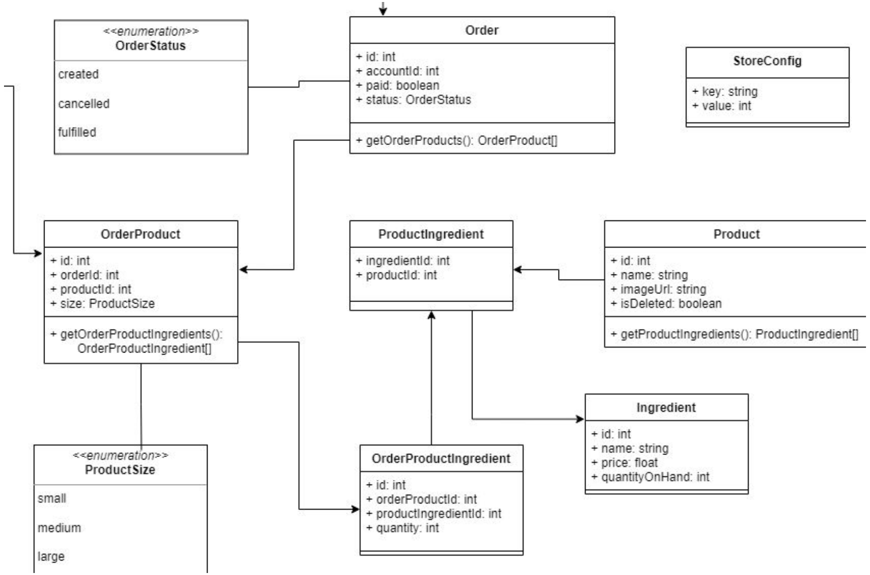
<<includes>>

<<includes>>

Choose order and mark as fulfilled

<<extends>>

System removes order from queue



<<enumeration>>
OrderStatus

created

cancelled

fulfilled

Order

+ id: int
+ accountId: int
+ paid: boolean
+ status: OrderStatus

+ getOrderProducts(): OrderProduct[]

StoreConfig

+ key: string
+ value: int

OrderProduct

+ id: int
+ orderId: int
+ productId: int
+ size: ProductSize

+ getOrderProductIngredients(): OrderProductIngredient[]

ProductIngredient

+ ingredientId: int
+ productId: int

Product

+ id: int
+ name: string
+ imageUrl: string
+ isDeleted: boolean

+ getProductIngredients(): ProductIngredient[]

Ingredient

+ id: int
+ name: string
+ price: float
+ quantityOnHand: int

<<enumeration>>
ProductSize

small

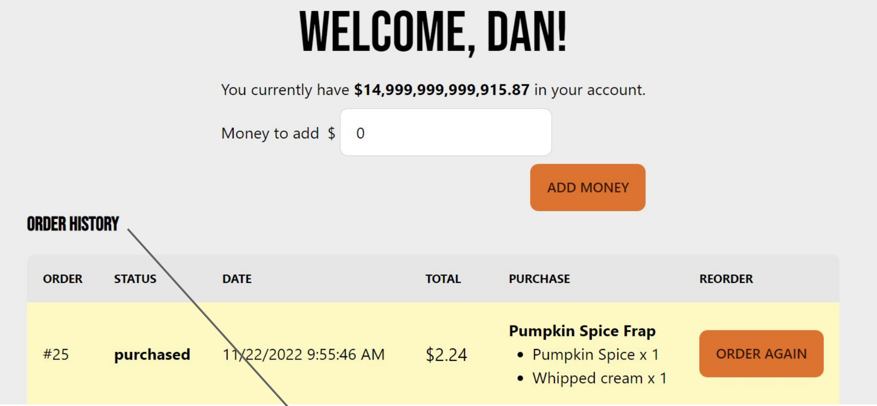
medium

large

OrderProductIngredient

+ id: int
+ orderProductId: int
+ productIngredientId: int
+ quantity: int

Fulfilling an Order - Implementation



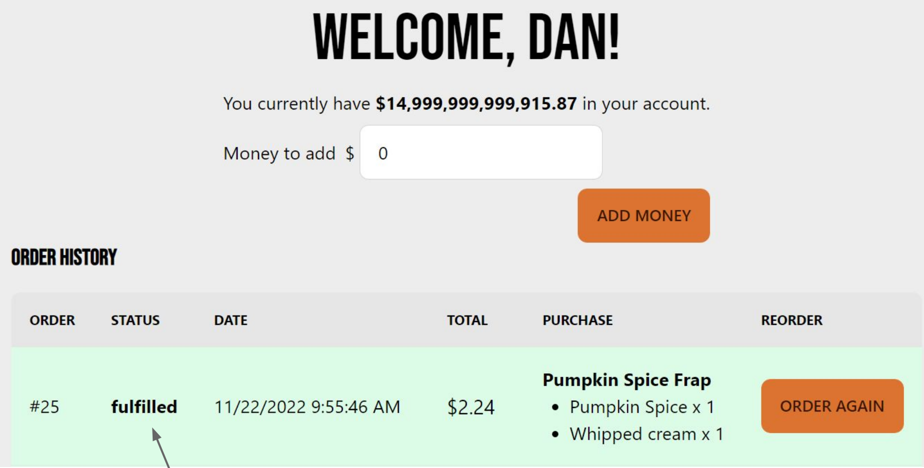
WELCOME, DAN!

You currently have **$14,999,999,999,915.87** in your account.

Money to add $ 0

ORDER HISTORY

ORDER	STATUS	DATE	TOTAL	PURCHASE	REORDER
#25	**purchased**	11/22/2022 9:55:46 AM	$2.24	**Pumpkin Spice Frap** • Pumpkin Spice x 1 • Whipped cream x 1	ORDER AGAIN



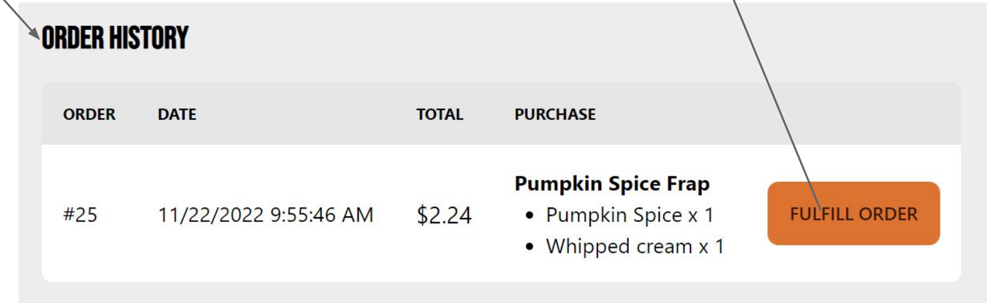
WELCOME, DAN!

You currently have **$14,999,999,999,915.87** in your account.

Money to add $ 0

ORDER HISTORY

ORDER	STATUS	DATE	TOTAL	PURCHASE	REORDER
#25	**fulfilled**	11/22/2022 9:55:46 AM	$2.24	**Pumpkin Spice Frap** • Pumpkin Spice x 1 • Whipped cream x 1	ORDER AGAIN



ORDER HISTORY

ORDER	DATE	TOTAL	PURCHASE	
#25	11/22/2022 9:55:46 AM	$2.24	**Pumpkin Spice Frap** • Pumpkin Spice x 1 • Whipped cream x 1	FULFILL ORDER

John

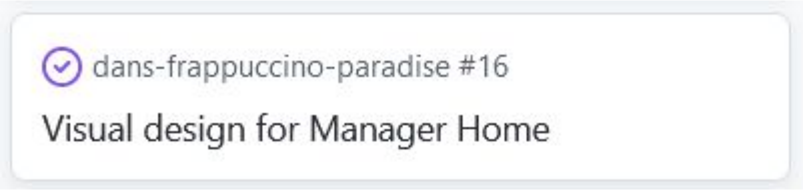
Requirements Tracking

Adding Inventory

Adding Inventory - Requirements

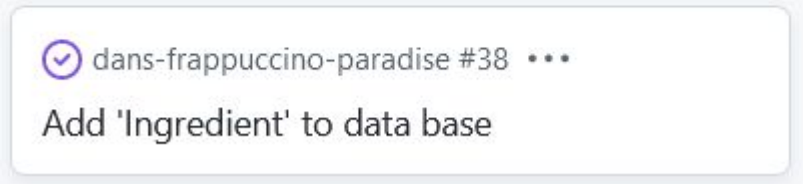
- After the manager has logged into their manager account they can see the manager page which includes the current inventory for each item
  - This list is stored in the database and updated with each order placed
- This page will now include “Place An Order”
- A manager account is presented with a **list of all ingredients** and the price for ordering more
  - This list displays quantity and price per unit, as well as order quantity and cost to place an order
- When the manager clicks on “Place An Order”, the store balance and inventory quantity will change to reflect the order quantity and cost

Adding Inventory - Scrum Tasks



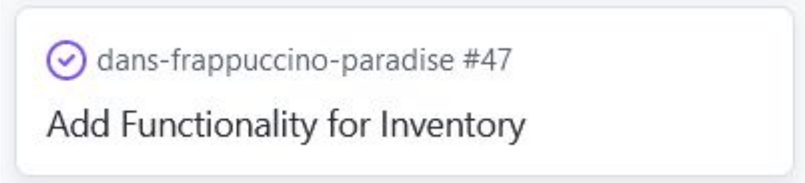
dans-frappuccino-paradise #16

Visual design for Manager Home



dans-frappuccino-paradise #38 •••

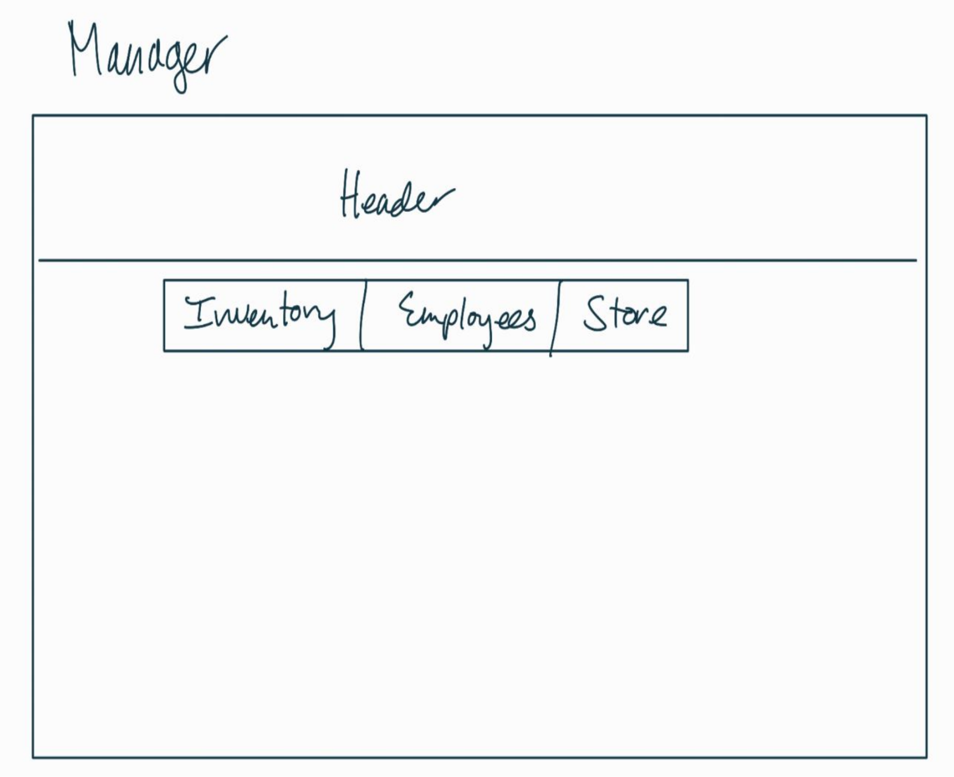
Add 'Ingredient' to data base

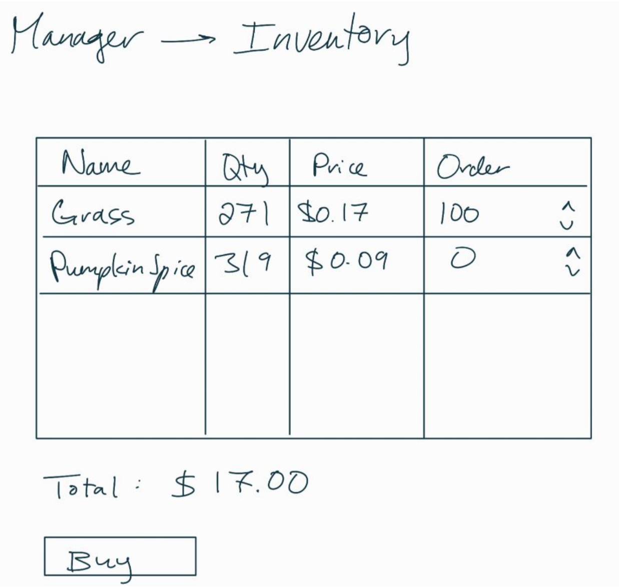


dans-frappuccino-paradise #47

Add Functionality for Inventory

Adding Inventory - Low Fidelity

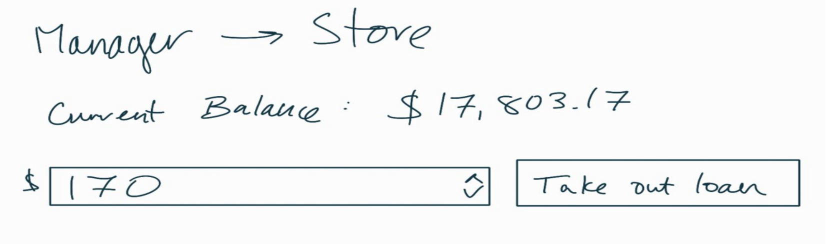




Manager → Inventory

Name	Qty	Price	Order
Grass	271	$0.17	100
Pumpkin Spice	319	$0.09	0

Total : $ 17.00

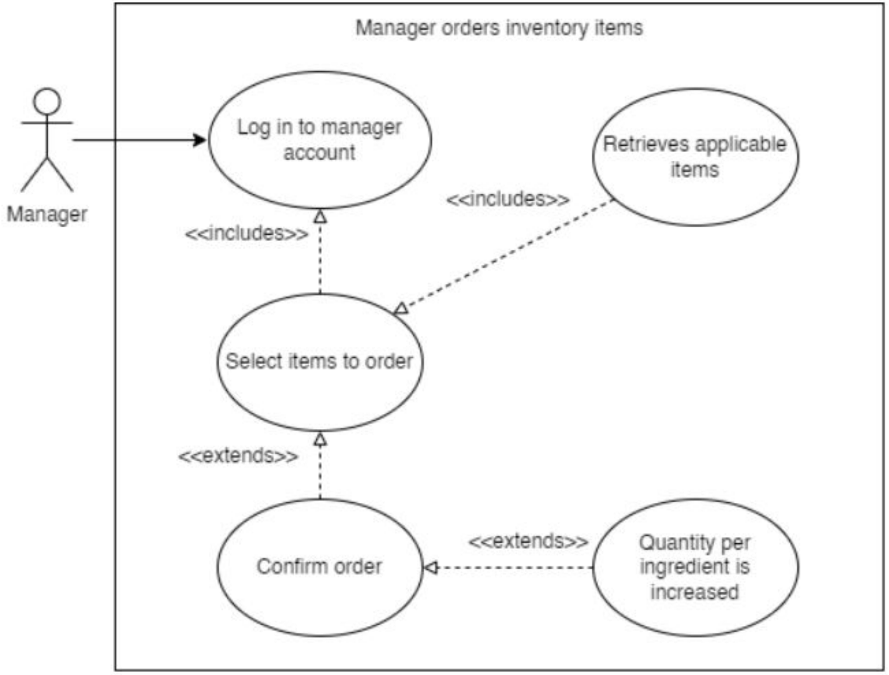


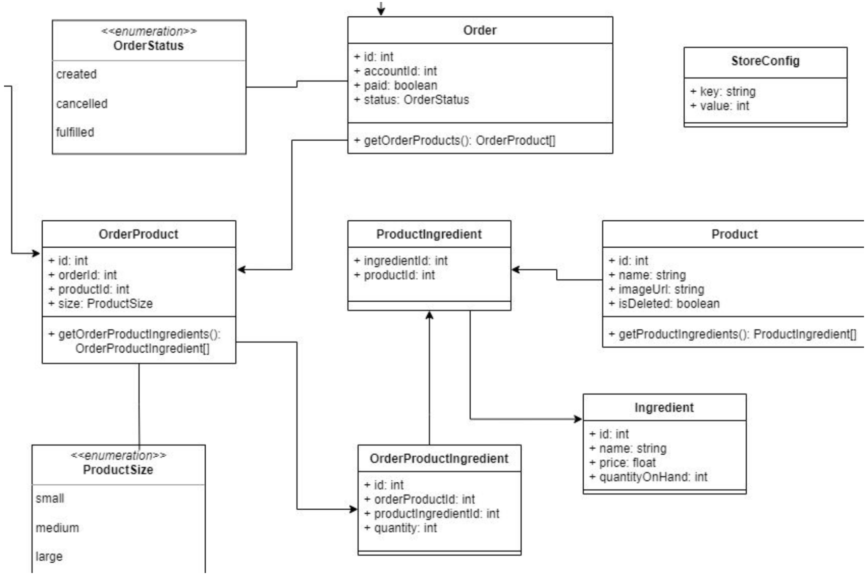
Manager → Store

Current Balance : $ 17,803.17

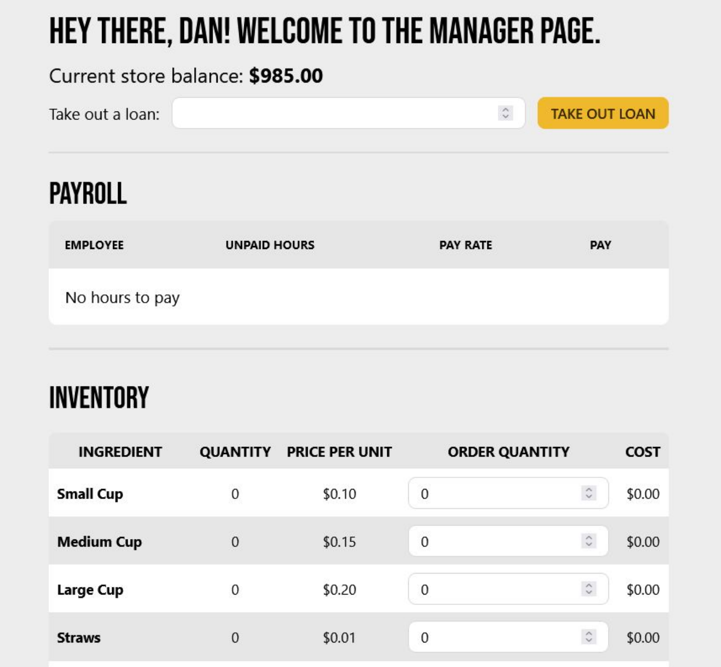
$ 170

Adding Inventory - Diagrams


Manager orders inventory items
Manager
Log in to manager account
Retrieves applicable items
<<includes>>
<<includes>>
Select items to order
<<extends>>
Confirm order
<<extends>>
Quantity per ingredient is increased



Adding Inventory - Implementation



HEY THERE, DAN! WELCOME TO THE MANAGER PAGE.

Current store balance: **$985.00**

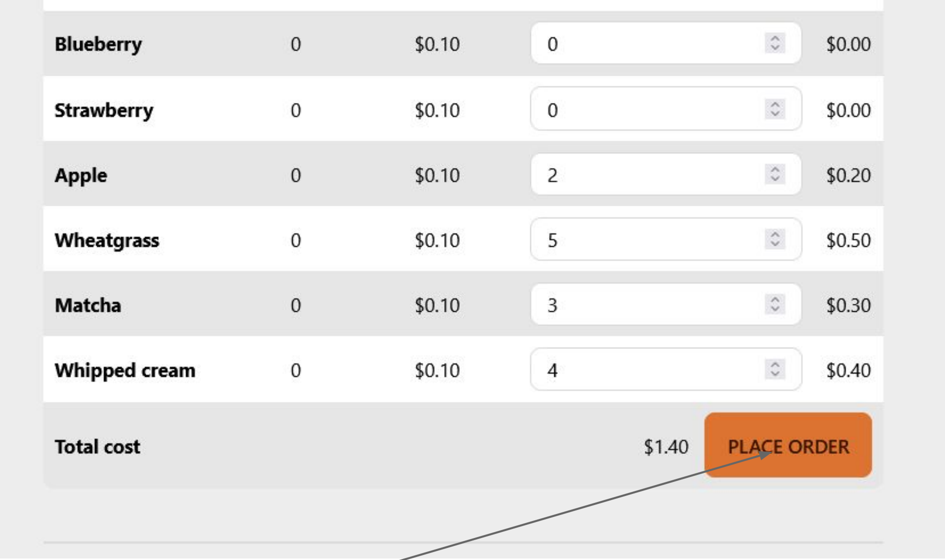
Take out a loan:

PAYROLL

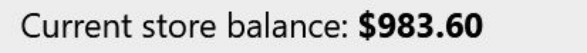
EMPLOYEE	UNPAID HOURS	PAY RATE	PAY
No hours to pay			

INVENTORY

INGREDIENT	QUANTITY	PRICE PER UNIT	ORDER QUANTITY	COST
Small Cup	0	$0.10	0	$0.00
Medium Cup	0	$0.15	0	$0.00
Large Cup	0	$0.20	0	$0.00
Straws	0	$0.01	0	$0.00

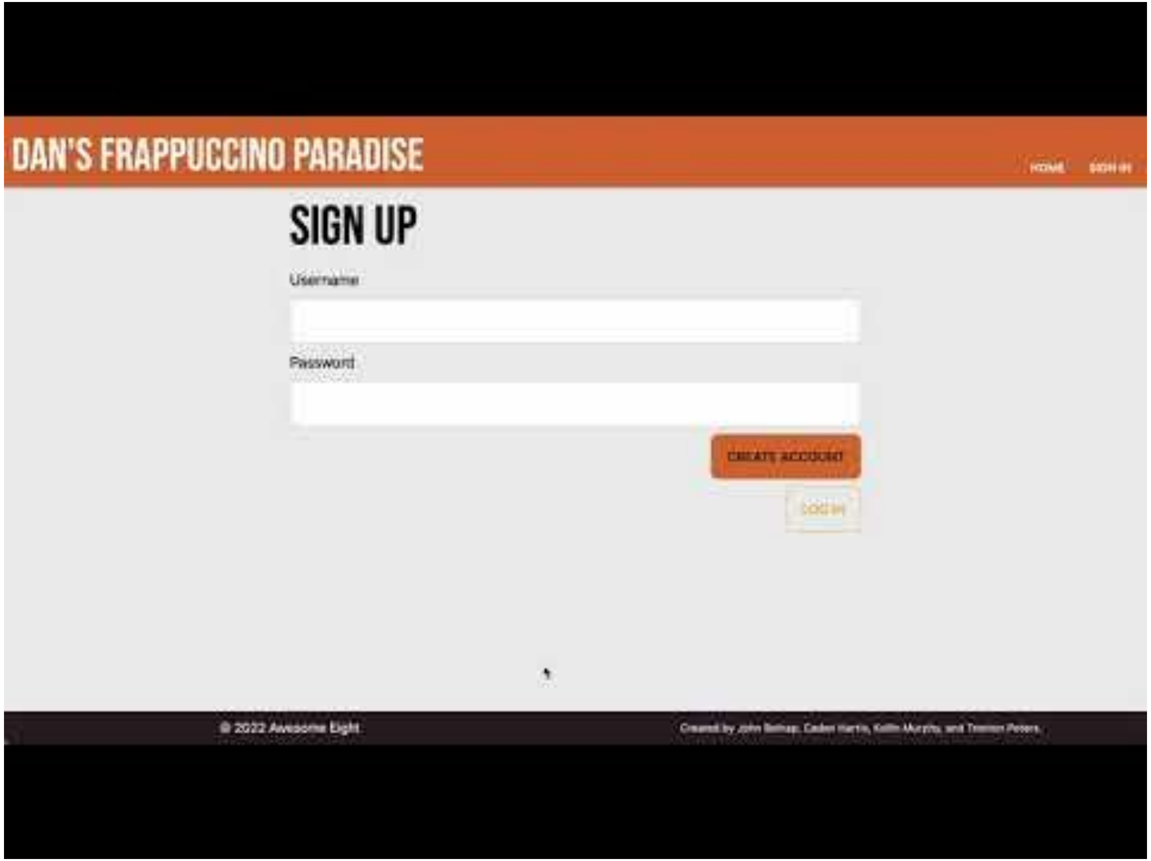


Blueberry	0	$0.10	0	$0.00
Strawberry	0	$0.10	0	$0.00
Apple	0	$0.10	2	$0.20
Wheatgrass	0	$0.10	5	$0.50
Matcha	0	$0.10	3	$0.30
Whipped cream	0	$0.10	4	$0.40
Total cost			$1.40	PLACE ORDER



Current store balance: **$983.60**

Screen Captures



SIGN UP

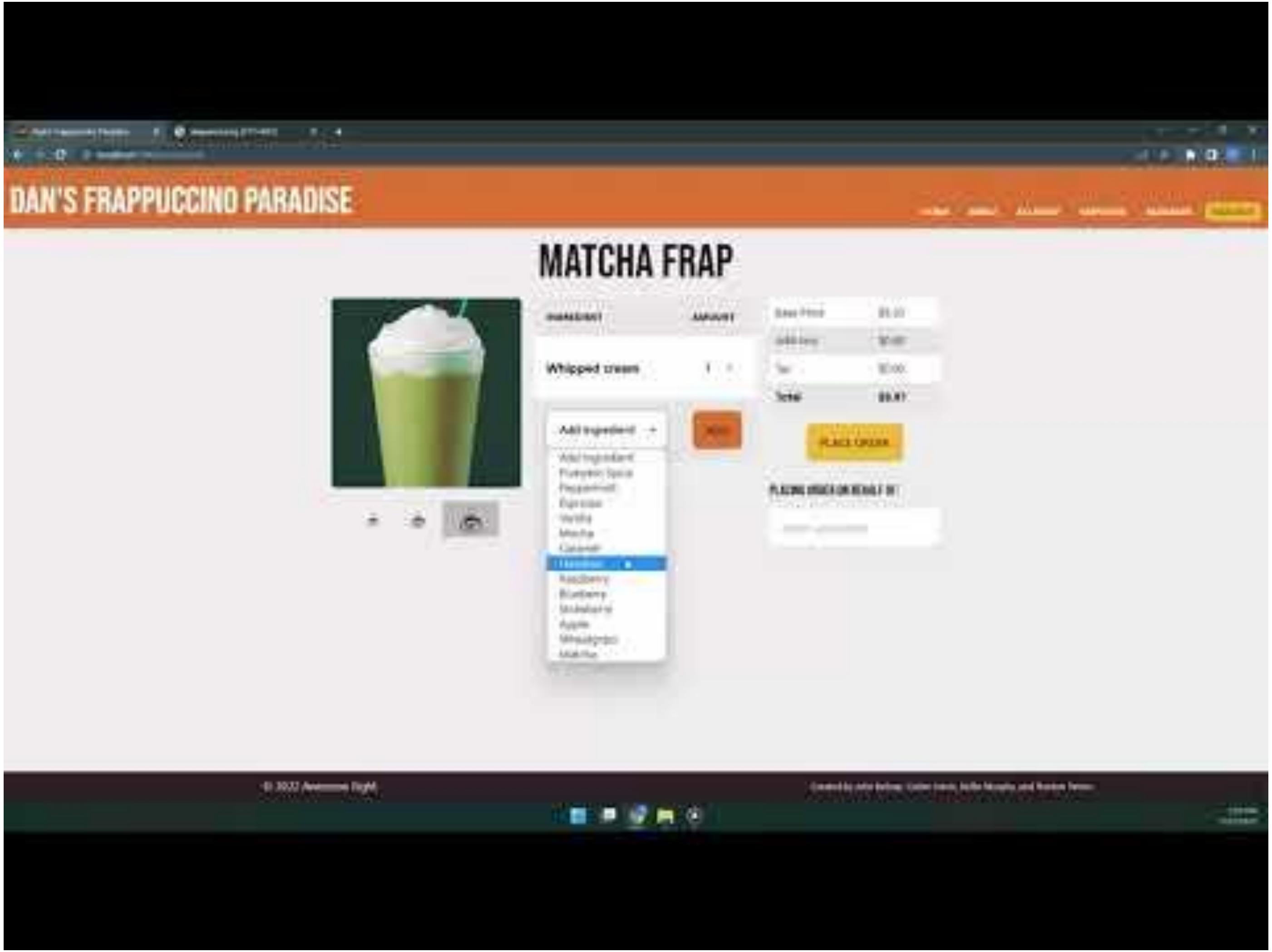
Username

Password

© 2022 Awesome Eight

Created by John Belnap, Caden Harris, Kollin Murphy, and Trenton Peters.

Creating an Account

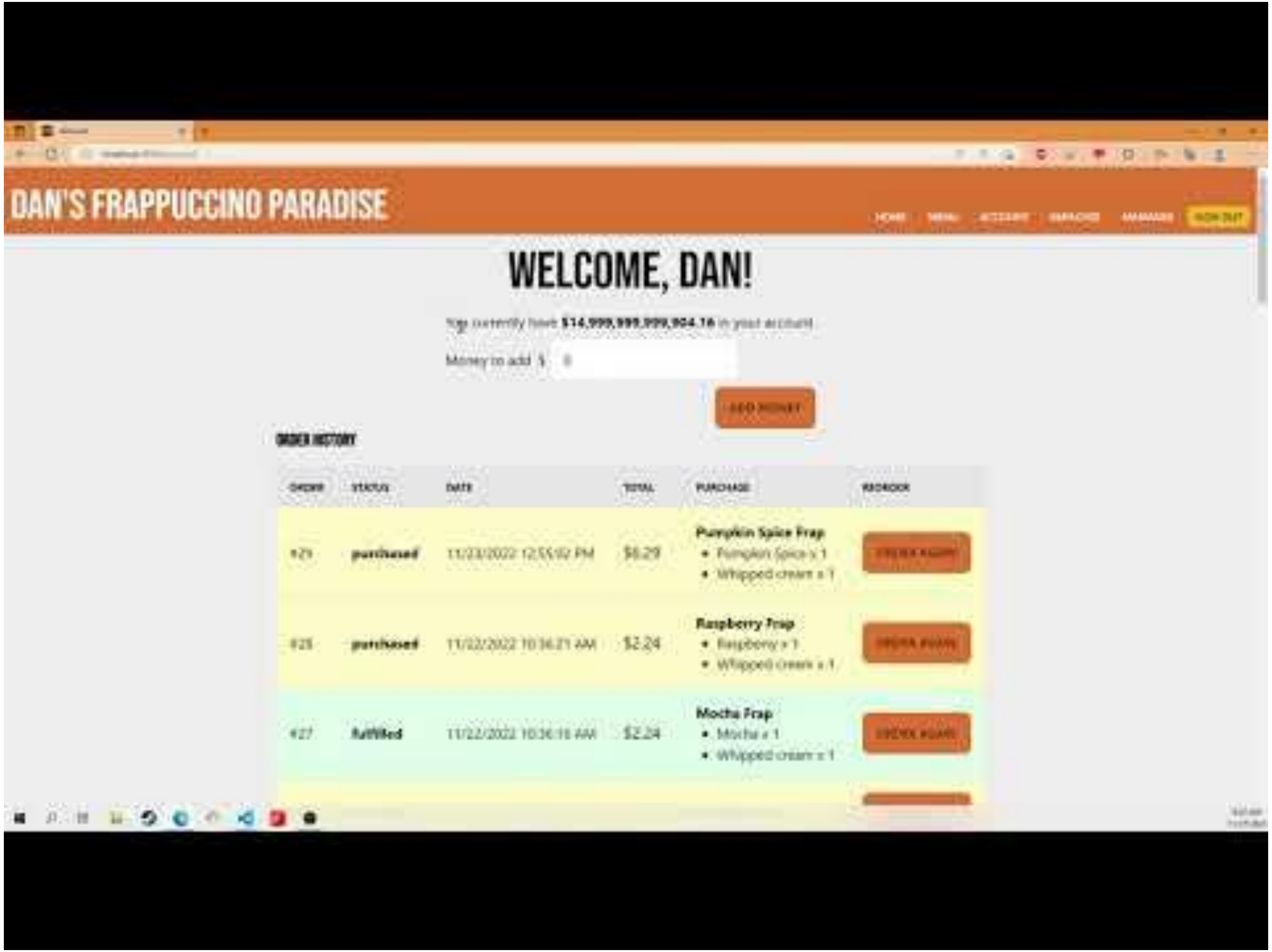


DAN'S FRAPPUCCINO PARADISE

MATCHA FRAP

INGREDIENT	AMOUNT
Whipped cream	1

Placing An Order



WELCOME, DAN!

You currently have **$14,999,999,999,904.16** in your account

Money to add $ 0

ORDER HISTORY

ORDER	STATUS	DATE	TOTAL	PURCHASE	REORDER
#29	**purchased**	11/23/2022 12:55:02 PM	$6.29	**Pumpkin Spice Frap** • Pumpkin Spice x 1 • Whipped cream x 1	ORDER AGAIN
#28	**purchased**	11/22/2022 10:36:21 AM	$2.24	**Raspberry Frap** • Raspberry x 1 • Whipped cream x 1	ORDER AGAIN
#27	**fulfilled**	11/22/2022 10:36:16 AM	$2.24	**Mocha Frap** • Mocha x 1 • Whipped cream x 1	ORDER AGAIN

Fulfilling an Order

Adding Inventory

Resources

https://docs.astro.build/en/getting-started/

https://daisyui.com/

https://www.solidjs.com/docs/latest/api

https://tailwindcss.com/docs/installation

https://sequelize.org/docs/v6/getting-started/

Questions?