TEAM 2: SUPERVISOR PRESENTATION

CollegeLife Recipes

Table of contents

Project Requirements & Plan

- Project Mode: Recipe Website
- Project Requirements & Plan

  (Supervisor) Presentation (Tasks 1-7)

to our supervisor

John Pasta

- *Mr. Pasta* is not aware of our progress
- Goal is to explain our technical and stylistic progress to *Mr. Pasta*

GOAL

AUDIENCE

Task Breakdown

- *TASK 1: PROJECT PLANNING AND WIREFRAMING*
- *TASK 2: SET UP THE BASIC HTML STRUCTURE*
- *TASK 3: STYLING SITE WITH CSS*
- *TASK 4: RECIPE DETAILS PAGE*

} previously completed

- TASK 5: IMPROVING USER INTERFACE WITH ANIMATION
- TASK 6: ADDING JAVASCRIPT FOR DYNAMIC RECIPE LISTINGS
- TASK 7: SEARCH FUNCTIONALITY
- EXTRA: ADDITIONAL FEATURES

} newly implemented

Task 5: Improving User Interface with Styling and Animation

1. TYPOGRAPHY
2. COLOR SCHEME
3. ANIMATIONS (EXPLAINED LATER)

CollegeLife Recipes

Recipes Favorites Timer Light Mode

what are you hungry for?

Developed by Mariano Soares, Julia Collingwood, Joshua Fouch
GitHub Link

CollegeLife Recipes

Recipes Favorites Timer Light Mode

Sweet Potato Hash with Eggs

30 min | Moderate

Ingredients

- [ ] • 1 sweet potato
- [ ] • 2 onions
- [ ] • 2 bell peppers
- [ ] • 2 eggs
- [ ] • 2 tbsp of olive oil

CollegeLife Recipes

Recipes Favorites Timer Light Mode

Method

Step 1: Peel and dice the sweet potato into small cubes.
Step 2: Heat olive oil in a large skillet over medium heat and add the sweet potato.
Step 3: Cook for 10 minutes, stirring occasionally, until the sweet potato starts to soften.
Step 4: Add diced onions and bell peppers to the skillet and cook for another 5 minutes.
Step 5: Make small wells in the hash and crack an egg into each well.
Step 6: Cover the skillet and cook until the eggs are set to your liking, about 5 minutes.
Step 7: Serve hot.

About this Dish

Sweet Potato Hash with Eggs is a hearty and wholesome breakfast that never fails to brighten my mornings. The combination of crispy, caramelized sweet potato cubes with the vibrant flavors of bell peppers and onions creates a satisfying base for perfectly cooked sunny-side-up eggs. This dish is not only packed with flavor but also offers a great balance of nutrients to keep you energized throughout the day. I first tried this recipe during a camping trip, and it has since become a weekend staple in my kitchen. Garnished with fresh parsley for a touch of brightness, this dish is as visually appealing as it is delicious. Whether served on a rustic ceramic plate or straight from the skillet, Sweet Potato Hash with Eggs is comfort food at its finest.

Task 6: Adding JavaScript for Dynamic Recipe Listings

1. DYNAMICALLY ADD RECIPES ON LIST PAGE
2. JAVASCRIPT HANDLES THE CREATION OF SECTIONS
3. DISPLAY RECIPE BY JSON DATA FETCHING

```
async function loadAllRecipes() {
    try {
        const response = await Promise.all([
            fetch('./recipe-assets/appetizer_recipes.json'),
            fetch('./recipe-assets/autumn_recipes.json'),
            fetch('./recipe-assets/beverages_recipes.json'),
            fetch('./recipe-assets/breakfast_recipes_part1.json'),
            fetch('./recipe-assets/breakfast_recipes_part2.json'),
            fetch('./recipe-assets/breakfast_recipes_part3.json'),
            fetch('./recipe-assets/lunch_recipes_part1.json'),
            fetch('./recipe-assets/lunch_recipes_part2.json'),
            fetch('./recipe-assets/dinner_recipes_part1.json'),
            fetch('./recipe-assets/dinner_recipes_part2.json'),
            fetch('./recipe-assets/winter_recipes.json'),
            fetch('./recipe-assets/summer_recipes.json'),
            fetch('./recipe-assets/spring_recipes.json'),
            fetch('./recipe-assets/snacks_recipes.json')
        ]);

        const recipeData = await Promise.all(response.map(r => r.json()));
        return recipeData.flat();
    } catch (error) {
        console.error('Error loading recipes:', error);
        return[];
    }
}

async function loadStories() {
    try {
        const response = await fetch('./recipe-assets/All_recipe_stories.json');
        const stories = await response.json();
        return stories;
    } catch (error) {
        console.error('Error loading stories: ', error);
        return[];
    }
}
```

loadAllRecipes() fetches json files and stores them in an array

loadStories() fetches story json file and stores into an array

```
[
    {
        "id": 51,
        "name": "Classic Lemonade",
        "category": "Beverages",
        "ingredients": [
            "2 lemons",
            "0.25 cup of sugar",
            "1 cup of water",
            "1 cup of ice"
        ],
        "dietary": [
            "Vegan",
            "Gluten-Free"
        ],
        "season": "Summer",
        "cuisine": "American",
        "prep_time": "10 min",
        "difficulty": "Easy",
        "images": [
            "classic_lemonade_1.webp",
            "classic_lemonade_2.webp",
            "classic_lemonade_3.webp",
            "classic_lemonade_4.webp",
            "classic_lemonade_5.webp",
            "classic_lemonade_6.webp",
            "classic_lemonade_7.webp",
            "classic_lemonade_8.webp"
        ],
        "steps": [
            "Juice the lemons and strain the juice to remove seeds.",
            "In a large pitcher, combine lemon juice and sugar, stirring until the sugar dissolves.",
            "Add water and ice, then stir well.",
            "Serve chilled with lemon slices as garnish."
        ]
    },
```


basic recipe JSON format


```
    {
        "id": 51,
        "story": "Classic Lemonade is a timeless summer drink that brings back memories of
ly barbecues, where the kids would set up a lemonade stand, and the adults couldn't resist
ns with just the right amount of sweetness from sugar. Adding a sprig of mint and serving i
he palate. Whether it's for a picnic, a party, or just a quiet afternoon, Classic Lemonade
    },
```


basic story json format

displayListingRecipes() adds a for every recipe in array, randomly.

```
function displayListingRecipes(recipes) {
  const container = document.querySelector('.widget-container');

  // get 3 random recipes from the recipe array
  const featured = recipes.sort(() => 0.5 - Math.random()).slice(0, 100);

  container.innerHTML = featured.map(recipe => `
        <div class="recipe-widget">
            <a href="recipe.html?id=${recipe.id}" style="text-decoration: none; color: inherit;">
                <div class="recipe-widget-img">
                    <img src="./recipe-assets/${recipe.images[0]}" alt="${recipe.name}">
                </div>
                <div class="recipe-widget-info">
                    <h2>${recipe.name}</h2>
                    <p>Prep Time: ${recipe.prep_time}</p>
                    <p>Difficulty: ${recipe.difficulty}</p>
                </div>
            </a>
        </div>
    `).join('');
}
```

```
async function displayRecipe() {
    try {
        // get the given recipe id from the url (given from displayListingRecipes())

        // Joshua made a regex to get the id in the form of: recipe.html?id=xx
        const pattern = /\?id=([0-9]+)/;
        const match = window.location.search.match(pattern);

        if (!match) {
            console.error('No recipe ID found');
            return;
        }

        const recipe_id = parseInt(match[1]);   // id is stored in match 1

        // load the recipes and stories into json arrays
        const [recipes, stories] = await Promise.all([
            loadAllRecipes(),
            loadStories()
        ]);

        // finds the recipe amongts the array of recipes and stories
        const recipe = recipes.find(r => r.id === recipe_id);
        const story = stories.find(s => s.id === recipe_id);

        console.log(recipe);
        console.log(story);

        document.title = `${recipe.name} - CollegeLife Recipes`;

        // load title and metadata
        document.getElementById('rtitle').textContent = recipe.name;
        document.getElementById('auth').textContent = `${recipe.prep_time} | ${recipe.difficulty}`;

        // Update recipe background image
        const recipeBg = document.querySelector('.recipe-bg');
        if (recipeBg) {
            recipeBg.style.backgroundImage = `linear-gradient(rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.7)), url('recipe-assets/${recipe.images[0]}')`;
            recipeBg.style.backgroundPosition = 'center';
            recipeBg.style.backgroundRepeat = 'no-repeat';
            recipeBg.style.backgroundSize = 'cover';
        }
```

Gets ID from URL

finds ID match from arrays

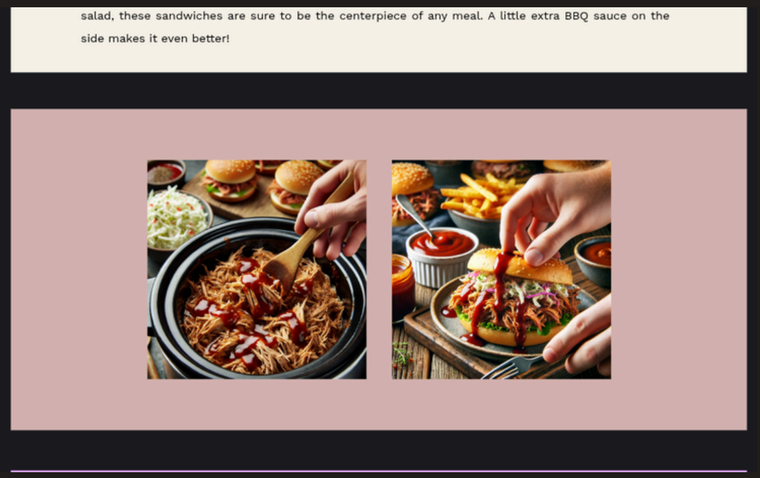
begins creating elements for each section of recipe

```
// length of recipe image array
const imgCount = recipe.images.length;

// handles layout of recipe page based on how many images there are
switch(imgCount) {
    // for one image
    case 1:

        // sets first image rec to images[0]
        img1.src = `recipe-assets/${recipe.images[0]}`;
        img1.alt = recipe.name;
        img1.onerror = () => img1.src = `assets/images/error2.webp`;

        // remove the other images
        img2?.remove();
        img3?.remove();
        img4?.remove();

        // delete additional images sections

        break;

    // for two images
    case 2:
        [img1, img2].forEach((img, i) => {
            img.src = `recipe-assets/${recipe.images[i]}`;
            img.alt = recipe.name;
            img.onerror = () => img1.src = `assets/images/error2.webp`;
        });

        // remove the other images
        img3?.remove();
        img4?.remove();

        // delete additional images sections
        addImgContainer?.remove();
        addImgs?.remove();
```

Switch statement depending on the amount of images there are in given image array

If there are more than 4 images, this will add the images in the “additional-images” section

```
// if more than 4
default:
    [img1, img2, img3, img4].forEach((img, i) => {
        img.src = `recipe-assets/${recipe.images[(i)]}`;
        img.alt = recipe.name;
        img.onerror = () => img1.src = `assets/images/error2.webp`;
    });

    // for additional images
    const otherImgs = document.getElementById('additionalImages');
    if (otherImgs) {
        // starts from index 4 since thats where it left off
        for (let i = 4; i < recipe.images.length; i++) {
            const img = document.createElement('img');
            img.src = `recipe-assets/${recipe.images[i]}`;
            img.alt = `${recipe.name}`;
            img.onerror = () => img.src = `assets/images/error2.webp`;
            otherImgs.appendChild(img);
        }
    }

    break;
```




```
<div class="additional-images-container">
    <div class="additional-images" id="additionalImages"></div>
</div>
```

Task 7: Search Functionality

1. SEARCHES ACCORDING TO TITLE OF RECIPE

```
<!--Main Content-->
<div class="listing-container">
    <h1 class="featured-header">the recipes.</h1>
    <!--search bar-->

    <form action="/search?">
        <div class="searchBox">
            <span class="material-symbols-outlined">
                search
            </span>
            <input class="searchInput" type="search" placeholder="Hungry?" id="myInput" onkeyup="searchRecipe()">
            <!-- <button class="searchButton" type="submit">Go</button> -->
        </div>
    </form>
```

searchrecipe() function called “onkeyup” i.e., when user stops typing

```
<script>

    let allRecipes = [];

    document.addEventListener('DOMContentLoaded', async () => {
        try {
            const recipes = await loadAllRecipes();
            allRecipes = recipes;
            console.log('Loaded recipes:', recipes.length); // Debug log
            displayListingRecipes(recipes);
        } catch (err) {
            console.error('Error in main:', err);
        }
    });

    function searchRecipe() {
        const input = document.getElementById("myInput").value.toLowerCase();
        const filteredRecipes = allRecipes.filter(recipe =>
            recipe.name.toLowerCase().includes(input) );
        displayListingRecipes(filteredRecipes);
    }
</script>
</body>
</html>
```

declare allRecipes as an array

stores all recipes into this array

searchRecipe() function takes user input and searches through allRecipes

the recipes.

Hungry?

Butternut Squash Risotto
Prep Time: 40 min
Difficulty: Moderate

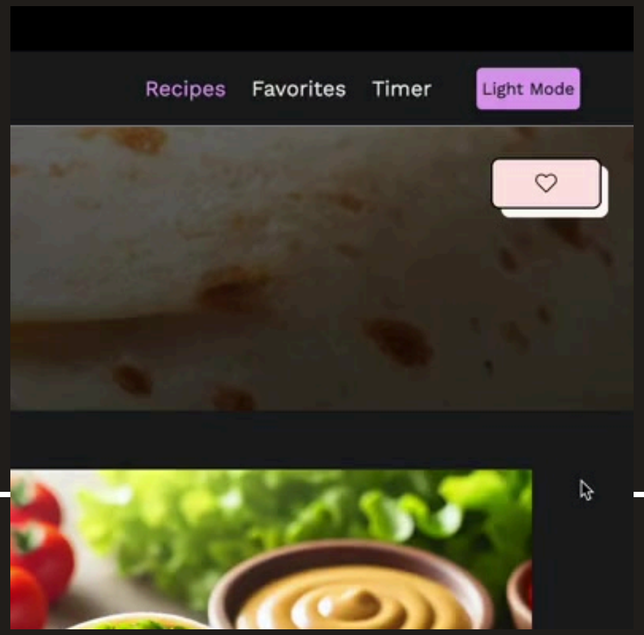
Strawberry Spinach Salad
Prep Time: 10 min
Difficulty: Easy

Roasted Herb-Crusted Chicken
Prep Time: 50 min
Difficulty: Moderate

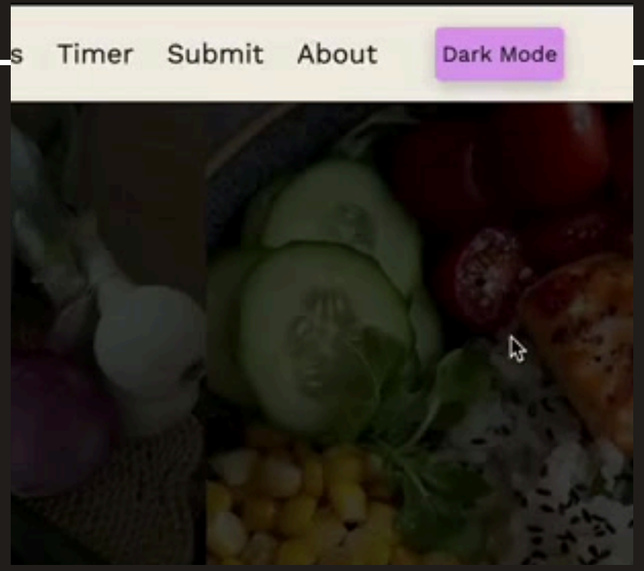
Enhancements

- Animations for Recipe Actions *(complete)*
- Cooking Timer *(complete)*
- Light and Dark Mode *(complete)*
- Interactive Ingredient Checklist *(complete)*
- Functionality on Mobile Browsers *(complete)*

Animation Examples



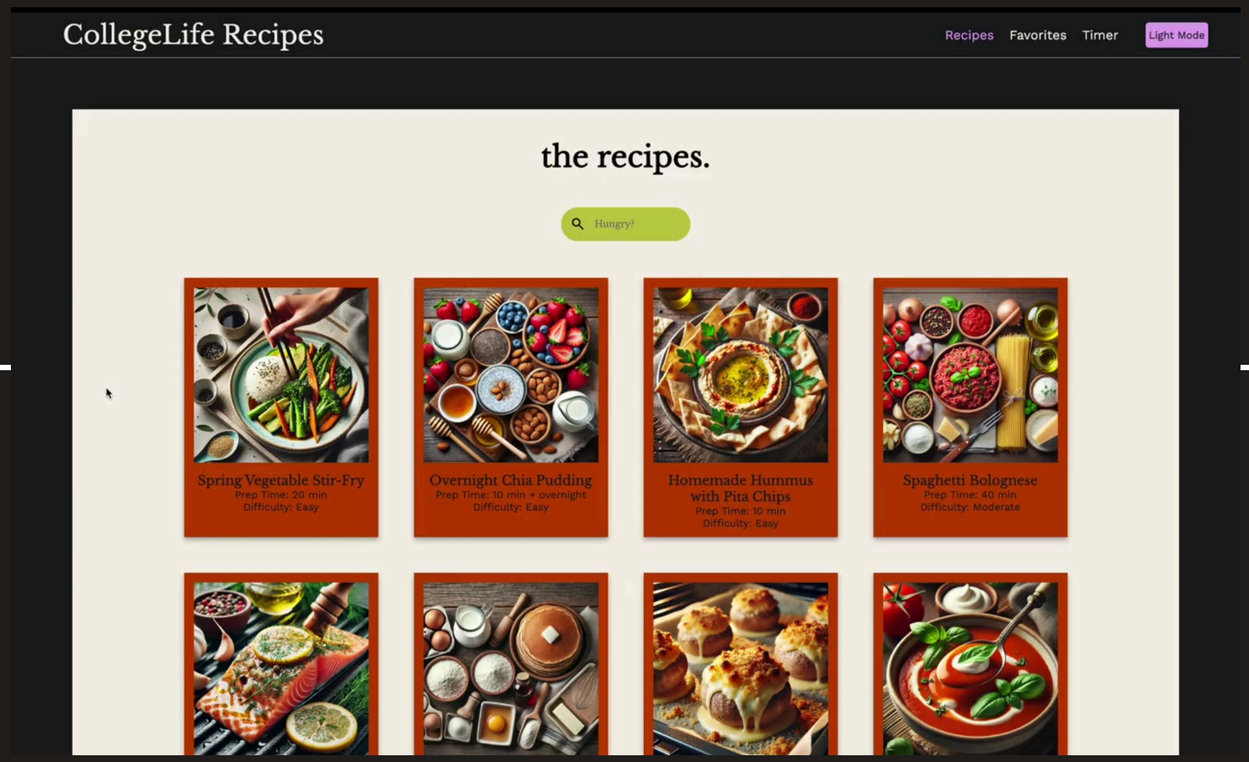
hover effects



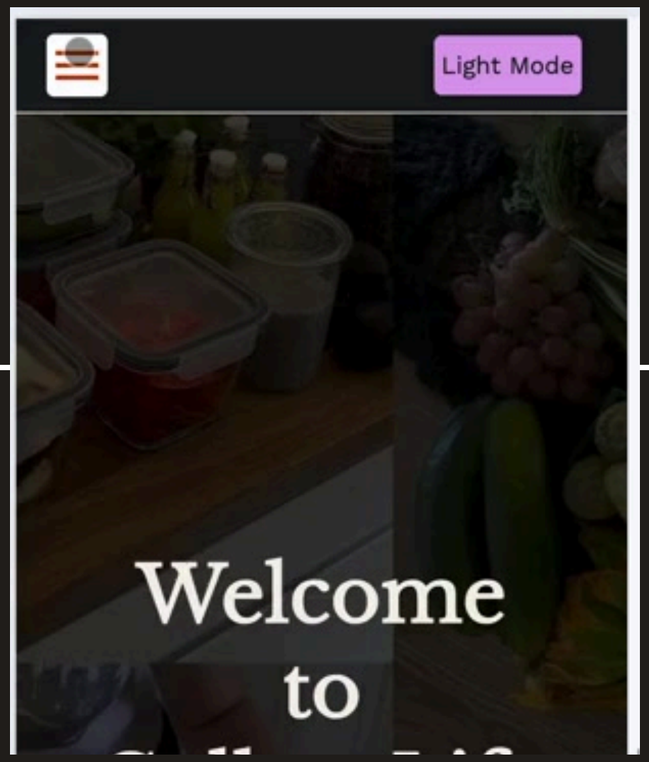
light/dark mode

checklist

Animation Examples



recipe widget



drop-down
menu ease-in

Favorite Button

```
function toggleFavorite(id, heartV) {// on/off heart, add/remove f
    let favorites = getFavorites();

    if (favorites.includes(id)) {
      favorites = favorites.filter(favId => favId !== id); // remo
      heartV.src = 'assets/images/empty-heart.svg';
    } else {
      favorites.push(id); // add
      heartV.src = 'assets/images/filled-heart.svg';
    }

    localStorage.setItem("favorites", JSON.stringify(favorites));/
}
```

Hamburger menu fade-in

```
nav ul div.items {
  display: none;
  width: 100%;
  text-align: center;
  animation: fadeIn 0.3s ease-in-out forwards;
}
```

```
/*keyframe code for fadeIn*/
/*this gives an pop up smooth animation
@keyframes fadeIn {
  from {
    opacity: 0;
    transform: translateY(-10px);
  }

  to {
    opacity: 1;
    transform: translateY(0);
  }
}
```

Hover effects

```
/* for recipe listing and favorites page */
.listing-container .widget-container .recipe-widget {
  display: flex;
  flex-direction: column;
  width: 300px;
  height: 400px;
  overflow: hidden;
  position: relative;
  cursor: pointer;
  background-color: var(--hoverLink);
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3);
  transition: all 0.2s ease-in-out;
}

.listing-container .widget-container .recipe-widget:hover {
  transform: translateY(-8px);
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3);
}
```

Light and Dark Mode

JS lightMode.js > ...

```
1   window.addEventListener("DOMContentLoaded", () => {
2       const coolButton = document.querySelector(".coolButton");
3
4       const savedTheme = localStorage.getItem("theme");
5
6       if (savedTheme === "light") {
7           document.documentElement.classList.add("light-mode");
8           coolButton.textContent = "Dark Mode";
9       } else {
10          document.documentElement.classList.remove("light-mode");
11          coolButton.textContent = "Light Mode";
12      }
13
14      coolButton.addEventListener("click", () => {
15          document.documentElement.classList.toggle("light-mode");
16
17          const isLight = document.documentElement.classList.contains("light-mode");
18          coolButton.textContent = isLight ? "Dark Mode" : "Light Mode";
19          localStorage.setItem("theme", isLight ? "light" : "dark");
20      });
21  });
22
```

uses localStorage to fetch the theme

This toggles and saves the user's preferred theme

```
    <button class="coolButton"> Switch Theme </button>
</ul>
```

HTML

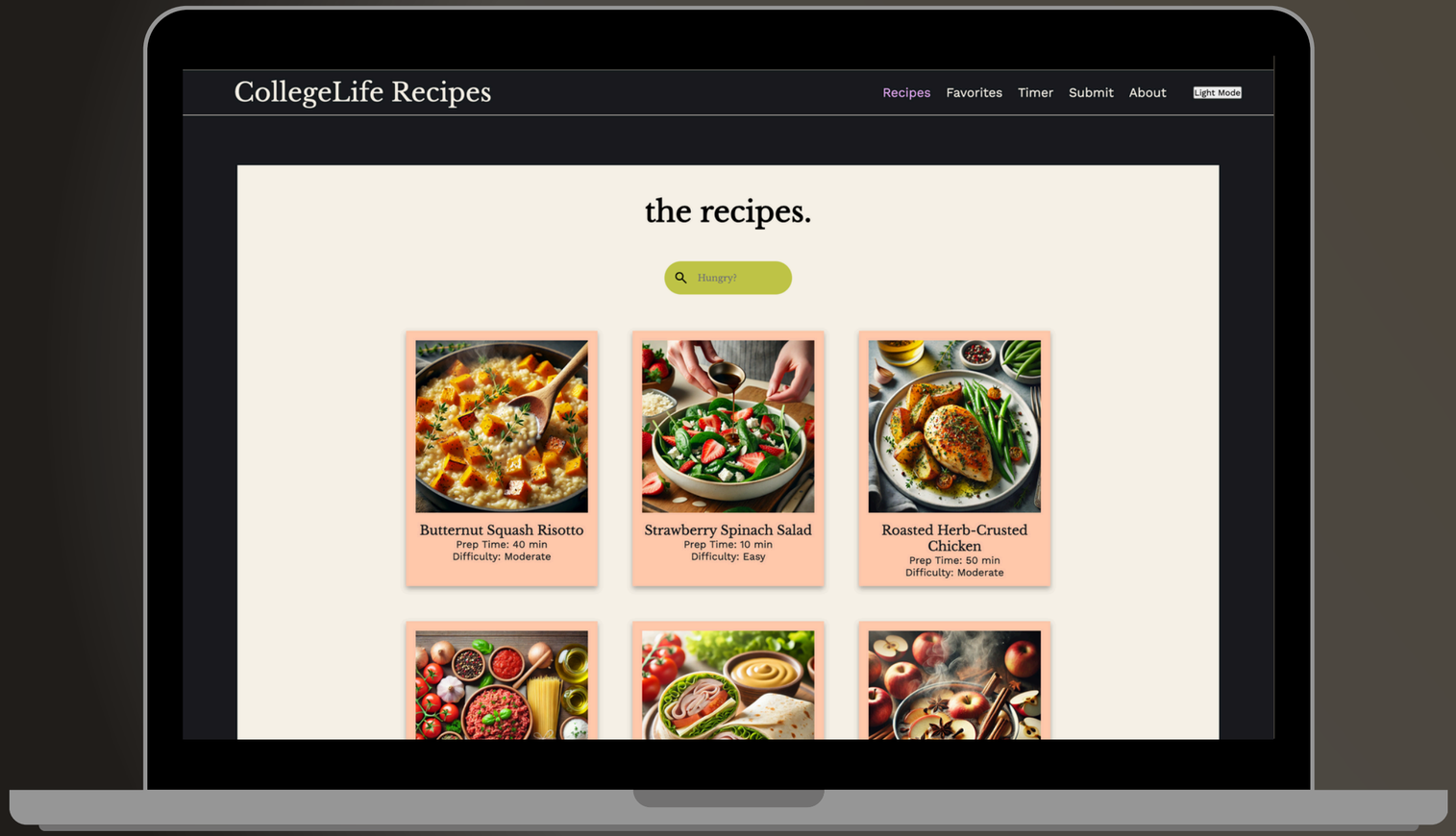
Light and Dark Mode

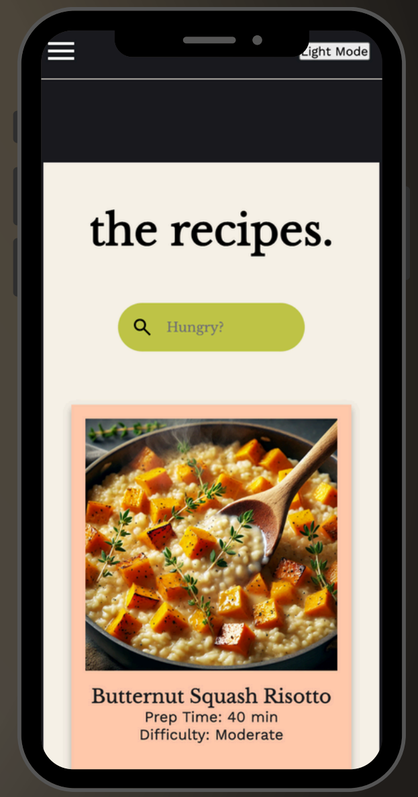
```css
/* color scheme */
:root {

  /* basic color scheme */
  --background: #1a1c20;
  --header: #f4efe5;
  --footer: #f4efe5;

  /* text */
  --h1: #f6f1e7;
  --h2: #e1a8f1;
  --recipetitle: antiquewhite;
  --body: #f6f1e7;
  --bodyRecipe: black;

  /* block color schemes */
  --ingredients: #f4efe5;
  --method: #bf4403;
  --edgy: #593f2f;

  /* widget color scheme */
  --greenWidget: #989d35;
  --purpleWidget: #e1a8f1;
  --hero: #1a1c20;
  --search: #c0c748;

  /* link color scheme */
  --hoverLink: #bf4403;
  --hoverLinkLight: #ffc8ab;
  --activeLink: #f4efe5;
  --visitedLink: #e1a8f1;

  /* other */
  --favoriteColor: #fee6e3;
  --favPress: #d4b3b0;
  --sigma: #FAFAFA;
}
```

the “root” variables are for dark mode
“light-mode” for light mode

```css
.light-mode {
  /* LIGHT MODE COLOR SCHEME */
  /* basic color scheme */
  --background: #f4efe5;
  --header: #1a1c20;
  --footer: #1a1c20;

  /* text */
  --h1: #f6f1e7;
  --h2: #e1a8f1;
  --recipetitle: rgb(226, 212, 193);
  --body: #f4efe5;
  --bodyRecipe: black;

  /* block color schemes */
  --ingredients: #f4efe5;
  --method: #d77c4f;
  --edgy: #593f2f;

  /* widget color scheme */
  --greenWidget: #989d35;
  --purpleWidget: #e1a8f1;
  --hero: #1a1c20;

  /* link color scheme */
  --hoverLink: #eca782;
  --hoverLinkLight: #b5663c;
  --activeLink: a#c54822;
  --visitedLink: #e197f6;
}
```

Mobile Sizing





Challenges

1. FINDING TIME TO WORK ON THE PROJECT
2. JSON AND IMAGE PROBLEMS
3. LEARNING JAVASCRIPT
4. LEARNING GIT BASICS

Thank you

CollegeLife Recipes