

# CPSC 332 Web Project Deliverable 4

Recipe Finder

Jackie Ramsey

12/4/2022

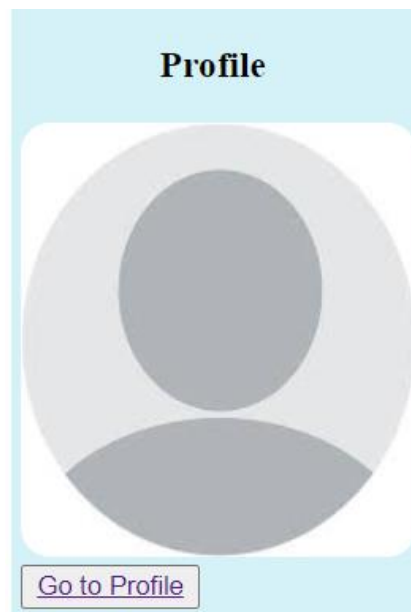
## GitHub URL

<https://github.com/GUWebDevelopment/cpsc-332-web-development-final-project-jackieaiko>

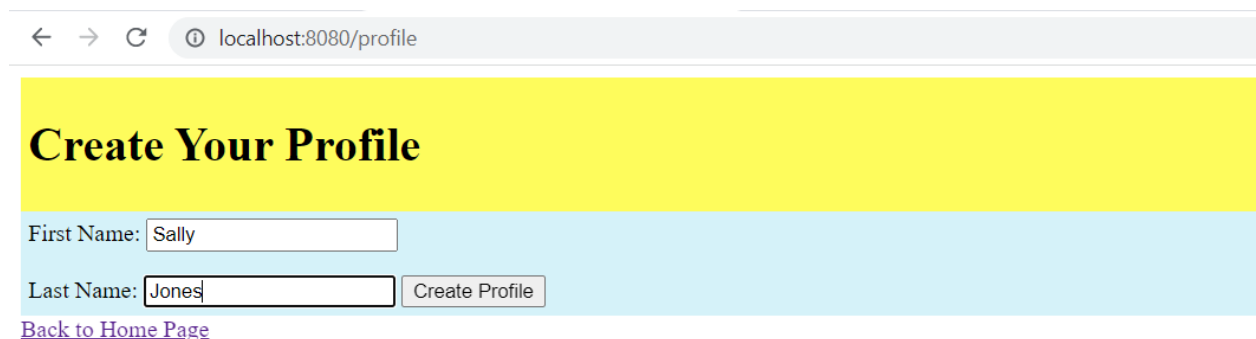
## Functional Requirement 1

### Users must be able to create a profile

1. Users are able to create a profile with their first and last name. At the home page, there is a section labelled Profile. Click the button called “Go to Profile”, and it will direct you to the create profile page.



2. Users are presented with a form to input their first and last name.

A screenshot of a web browser window. The address bar shows "localhost:8080/profile". The page has a yellow header with the text "Create Your Profile" in bold black font. Below the header is a light blue section containing a form. The form has two input fields: "First Name:" with the value "Sally" and "Last Name:" with the value "Jones". To the right of the "Last Name:" field is a button labeled "Create Profile". Below the form is a link labeled "Back to Home Page" in purple, underlined.

3. When clicking the “Create Profile” button by receiving the POST method

## Your Profile

[Click to Edit Profile](#)

First Name: Sally

Last Name: Jones

Background Color

Saved Recipes

[Back to Home Page](#)

```
app.post("/profilecreated", (req, res) => {

  let result = ProfileInfo(
    {
      firstName: req.body.firstName,
      lastName: req.body.lastName,
    });

  let validation = new Validator(result, rules);
  console.log("Validation Passes: " + validation.passes() + " Validation Fails: " + validation.fails());

  // do not want two of the same profiles, if fails, then a unique name can be inserted
  if(validation.passes()) {
    let errorsList = {
      firstName: validation.errors.first("firstName"),
      lastName: validation.errors.first("lastName")
    };

    res.render("error.ejs", {
      errors: 3,
      errorsList: errorsList
    });
  }
  else {
    result.save(
      (err, result) => {
        if (err) {
          //note that we are not handling this error! You'll want to do this yourself!
          return console.log("Error: " + err);
        }
        console.log(`Success! Inserted data with _id: ${result._id} into the database.`);
        res.redirect("/profilecreated");
      });
  }
});
```

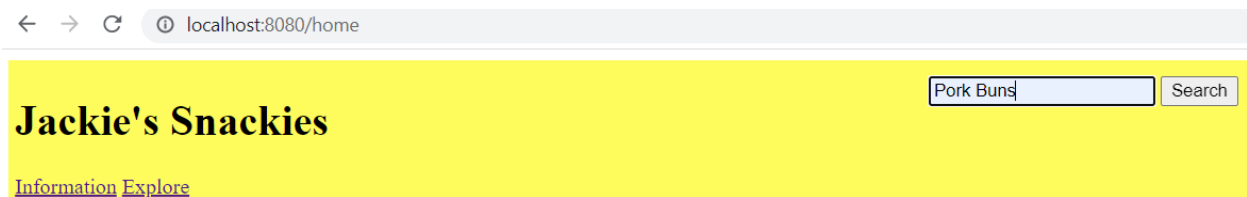
4. The form data is then inserted into the Mongo database and a new profile is created.

```
Success! Inserted data with _id: 638ec1e00049ece231bb8c45 into the database.
[
  {
    _id: new ObjectId("638ec1e00049ece231bb8c45"),
    firstName: 'Sally',
    lastName: 'Jones',
    savedRecipes: [],
    __v: 0
  }
]
```

## Functional Requirement 2

### Users must be able to search for recipes by name

1. Users are able to search for recipes by name. At the top of the home page, there is a search bar on the right. Users enter the name of the recipe and click the button called "Search"



2. When clicking the "Search" button by receiving the POST method

```
// go to recipe
app.post("/recipe", (req, res) => {

  // console.log(req.body.recipeurl)
  // console.log(req.body.recipeName)

  if(req.body.recipeName) {
    console.log(req.body.recipeName)

    RecipeInfo.find(
      {"recipeName": req.body.recipeName},
      (err, results) => {

        console.log(results)
        res.render("recipe.ejs/", {
          recipeResults: results
        });
      });
  });
}
```

3. The form data is then rendered in the recipe page.

## Pork Buns

[Recipe Description](#) [Ingredients](#) [Steps](#) [Ratings](#) [Uploaded Photos](#)



### Functional Requirement 3

Users must be able to comment on a recipe

1. Users can post comments on recipes. When clicking the “Click to rate or post” button, they are directed to a form. Users are presented with a form to input a comment.

## Pork Rib

Current Number of Likes:

Create a Comment

[Back to Home Page](#)

2. When clicking the “Post Content” button by receiving the POST method

```

.post(function (req, res) {
  console.log("posted")
  let id = req.params._id;
  let likes = req.body.likes;
  let comments = req.body.comments;

  console.log(likes)
  console.log(comments)

  RecipeInfo
    .where({ _id: id })
    .updateOne({
      $set: {
        likes: parseInt(likes)
      },
      $push: {
        comments: comments
      }
    })
    .exec(function (err, result) {
      if (err) return res.send(err);
      res.redirect("/recipe");
      console.log(`Successfully updated ${result.modifiedCount} record`);
    });
});

```

3. The form data for comments is then inserted into the Mongo database and is now on the recipe page.

```

Successfully updated 1 record
[]
recipe_images/pork_rib.jpg
[
  {
    _id: new ObjectId("63849e7b46d2f22bb9bea47e"),
    recipeName: 'Pork Rib',
    recipeurl: 'recipe_images/pork_rib.jpg',
    decription: 'this meal can be alone or on a sandwich',
    ingredients: [ 'pork', 'sauce', 'cornbread' ],
    steps: [ 'cook the meat', 'add other die dishes' ],
    likes: 2,
    comments: [
      'yeeee',
      'this is such a yummy looking meal!!! nom nom',
      'great',
      'this looks great!',
      'I am so hungry'
    ],
    __v: 0
  }
]

```

## Functional Requirement 4

### Users must be able to like a recipe

1. Users can like a recipe. When clicking the “Click to rate or post” button, they are directed to a form. Users are presented with a form to click the “Click to Like” button. If the click this button, their like will be counted when submitting the form.

## Pork Rib

Current Number of Likes:

## Pork Rib

Current Number of Likes:

2. When clicking the “Post Content” button by receiving the POST method

```
.post(function (req, res) {
  console.log("posted")
  let id = req.params._id;
  let likes = req.body.likes;
  let comments = req.body.comments;

  console.log(likes)
  console.log(comments)

  RecipeInfo
    .where({ _id: id })
    .updateOne({
      $set: {
        likes: parseInt(likes)
      },
      $push: {
        comments: comments
      }
    })
    .exec(function (err, result) {
      if (err) return res.send(err);
      res.redirect("/recipe");
      console.log(`Successfully updated ${result.modifiedCount} record`);
    });
});
```

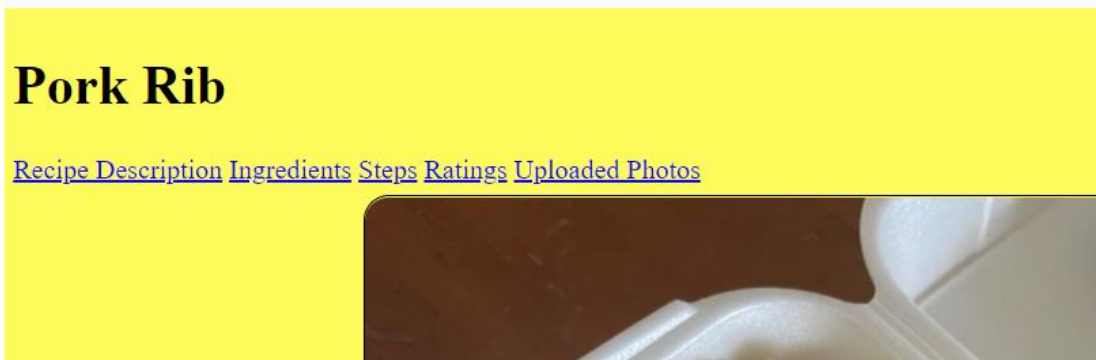
3. The form data for likes is then updated into the Mongo database and is now updated on the recipe page.

```
Successfully updated 1 record
[
  recipe_images/pork_rib.jpg
  [
    {
      _id: new ObjectId("63849e7b46d2f22bb9bea47e"),
      recipeName: 'Pork Rib',
      recipeurl: 'recipe_images/pork_rib.jpg',
      decription: 'this meal can be alone or on a sandwich',
      ingredients: [ 'pork', 'sauce', 'cornbread' ],
      steps: [ 'cook the meat', 'add other die dishes' ],
      likes: 2,
      comments: [
        'yeeee',
        'this is such a yummy looking meal!!! nom nom',
        'great',
        'this looks great!',
        'I am so hungry'
      ],
      __v: 0
    }
  ]
]
```

## Functional Requirement 5

Users must be able to jump to different section of content in a recipe

1. Towards the top of the ripe page, there are links to each section of the recipe. If a user wants to go directly to a certain area of a recipe instead of scrolling, they can click the desired link.



2. When clicking the link. They are redirected to that area of the recipe.



#### Ingredients

- pork
- sauce
- cornbread

#### Steps

1. cook the meat
2. add other die dishes

#### User Reviews and Ratings

[Click to rate or post!](#)

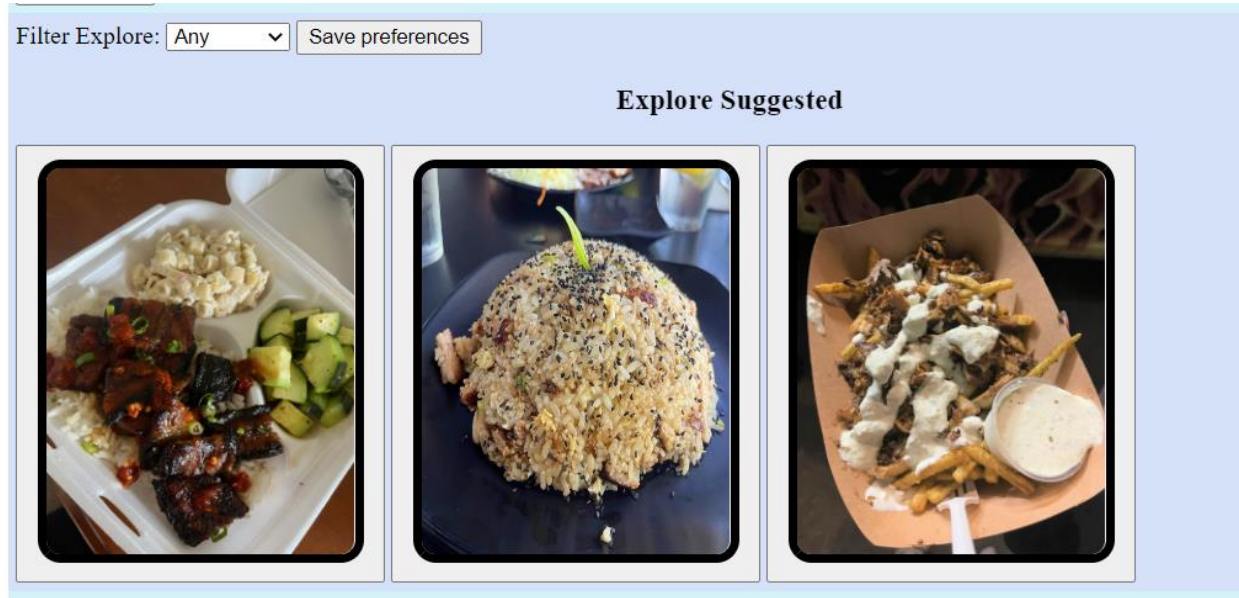
#### Likes

2

## Functional Requirement 6

### Users must be able to explore recipes by suggested

1. One-way users are able to find recipes is by using the explore section. In the middle of the home page, there are several pictures of recipes. The user is able to click any of the photos.



2. When clicking the photo by receiving the POST method

```

    }
    else if(req.body.recipeurl) {
      console.log(req.body.recipeurl)

      RecipeInfo.find(
        {"recipeurl": req.body.recipeurl},
        (err, results) => {

          console.log(results)
          res.render("recipe.ejs/", {
            recipeResults: results
          });
        });
    }
  }
}

```

3. The form data is then rendered in the recipe page and the user is directed to that recipe.

## Pork Buns

[Recipe](#) [Description](#) [Ingredients](#) [Steps](#) [Ratings](#) [Uploaded Photos](#)



## Functional Requirement 7

Users must be able tag photo recipes to favorites

1. Users are presented with a form to select a recipe to put into their favorites.

## Saved Recipes

- Crepe
- Pork Bun
- Fried Rice
- Birria Tacos
- Sushi Roll
- Shwarma
- Shwarma Fries
- Crepe
- Pork Rib
- Pho
- Saimen
- Chicken Sandwich
- Breakfast Sandwich

[Back to Home Page](#)

2. When clicking the “Edit (Save Changes)” button by receiving the POST method

```
// validate and sanitize when editing profile information
.post(function (req, res) {
  console.log("posted")
  let id = req.params._id;
  let backgroundColor = req.body.backgroundColor;
  let savedRecipes = req.body.savedRecipes;

  // validating the correct profile
  validateSession(req.params._id, res);

  id = sanitize(id);

  ProfileInfo
    .where({ _id: id })
    .updateOne({
      $set: {
        backgroundColor: backgroundColor
      },
      $push: {
        savedRecipes: savedRecipes
      }
    })
    .exec(function (err, result) {
      if (err) return res.send(err);
      res.redirect("/profilecreated");
      console.log(`Successfully updated ${result.modifiedCount} record`);
    });
});
```

3. The form data is then inserted into the mongodb database, and visible on the created profile page

```
Successfully updated 1 record
{
  _id: new ObjectId("638ec1e00049ece231bb8c45"),
  firstName: 'Sally',
  lastName: 'Jones',
  savedRecipes: [ 'recipe_images/crepe.jpg' ],
  __v: 0,
  backgroundColor: 'green'
}
```

## Functional Requirement 8

Users must be able to pick their profile background color

1. In the created profile page, users are able to edit their profile. When clicking the “Click to Edit Profile” button, users are directed to a form where they can access the background color.

**Edit Profile**

Background Color: Default Color ▾

**Saved Recipe**

[Back to Home Page](#)

2. The user is able to select any color from the background color drop down menu. When clicking the “Edit (Save Changes)” button by receiving the POST method.

```

// validate and sanitize when editing profile information
.post(function (req, res) {
  console.log("posted")
  let id = req.params._id;
  let backgroundColor = req.body.backgroundColor;
  let savedRecipes = req.body.savedRecipes;

  // validating the correct profile
  validateSession(req.params._id, res);

  id = sanitize(id);

  ProfileInfo
    .where({ _id: id })
    .updateOne({
      $set: {
        backgroundColor: backgroundColor
      },
      $push: {
        savedRecipes: savedRecipes
      }
    })
    .exec(function (err, result) {
      if (err) return res.send(err);
      res.redirect("/profilecreated");
      console.log(`Successfully updated ${result.modifiedCount} record`);
    });
});

```

3. The form data will be inserted into the Mongodb database, and the changes are now visible in the created profile page.

```

Successfully updated 1 record
[
  {
    _id: new ObjectId("638ec1e00049ece231bb8c45"),
    firstName: 'Sally',
    lastName: 'Jones',
    savedRecipes: [ 'recipe_images/crepe.jpg' ],
    __v: 0,
    backgroundColor: 'green'
  }
]

```

## Your Profile

[Click to Edit Profile](#)

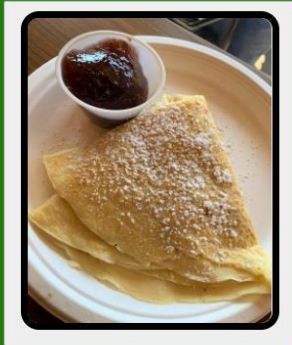
First Name: Sally

Last Name: Jones

### Background Color

green

### Saved Recipes

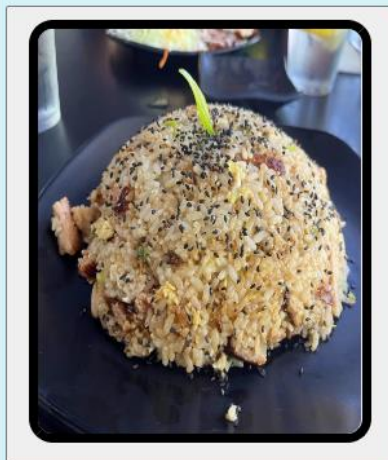


## Functional Requirement 9

Users must be able to navigate to the featured recipe

1. One-way users are able to find recipes is by using the featured recipe section. At the bottom of the home page, there is a random picture of a recipe. The user is able to click the photo.

### Featured Recipe



2. When clicking the photo by receiving the POST method

```
    }  
    else if(req.body.recipeurl) {  
      console.log(req.body.recipeurl)  
  
      RecipeInfo.find(  
        {"recipeurl": req.body.recipeurl},  
        (err, results) => {  
  
          console.log(results)  
          res.render("recipe.ejs/", {  
            recipeResults: results  
          });  
        });  
    }  
  }  
}
```

3. The form data is then rendered in the recipe page and the user is directed to that recipe.

## Fried Rice

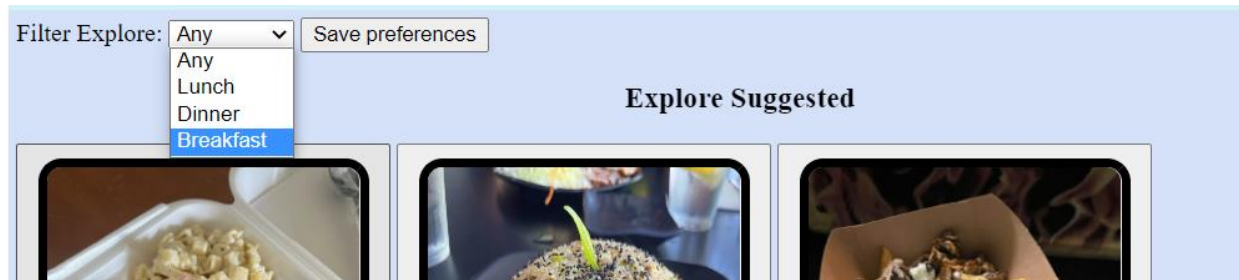
[Recipe Description](#) [Ingredients](#) [Steps](#) [Ratings](#) [Uploaded Photos](#)



## Functional Requirement 10

Users must be able to filter recipe preferences

1. Users are presented with the option to filter their recipe preferences for the explore page. In the middle of the home page, there is a dropdown menu along with a “Save preferences” button.



2. When clicking the “Save Preferences” button, the explore page recipes are configured to the selected dropdown value.

