CPSC 332 Web Project Deliverable 3

Recipe Finder

Jackie Ramsey

11/30/2022

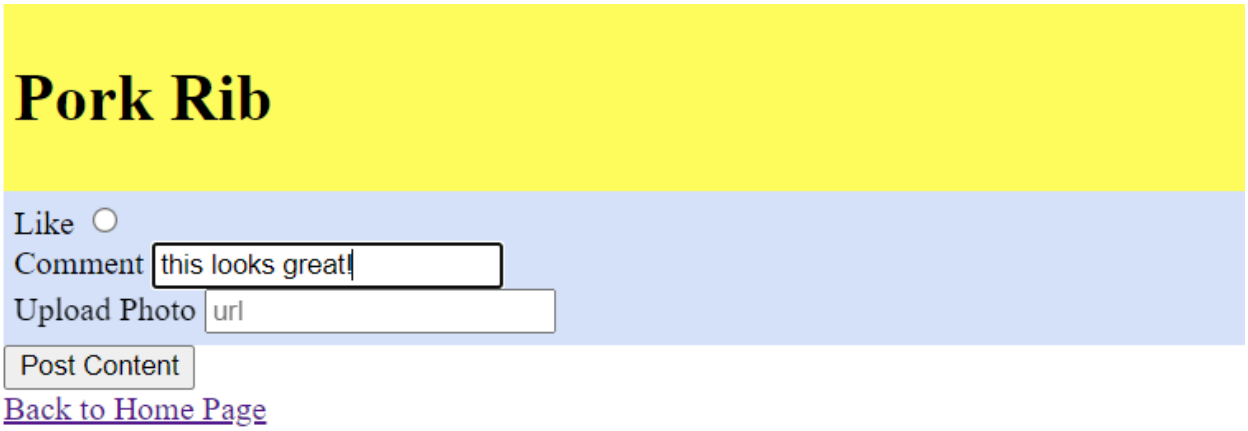
## GitHub URL

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

## Backend Functional Requirement 1

### Users must be able to Comment on a Recipe

1. Users are presented with a form to input a comment.




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

```
    .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)

        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 is then inserted into the Mongo database

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

## Backend Functional Requirement 2

### Users must be able to Like a Recipe

1. Users are presented with a form to check a radio button to like.


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

```
    .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)

        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 is then inserted into the Mongo database

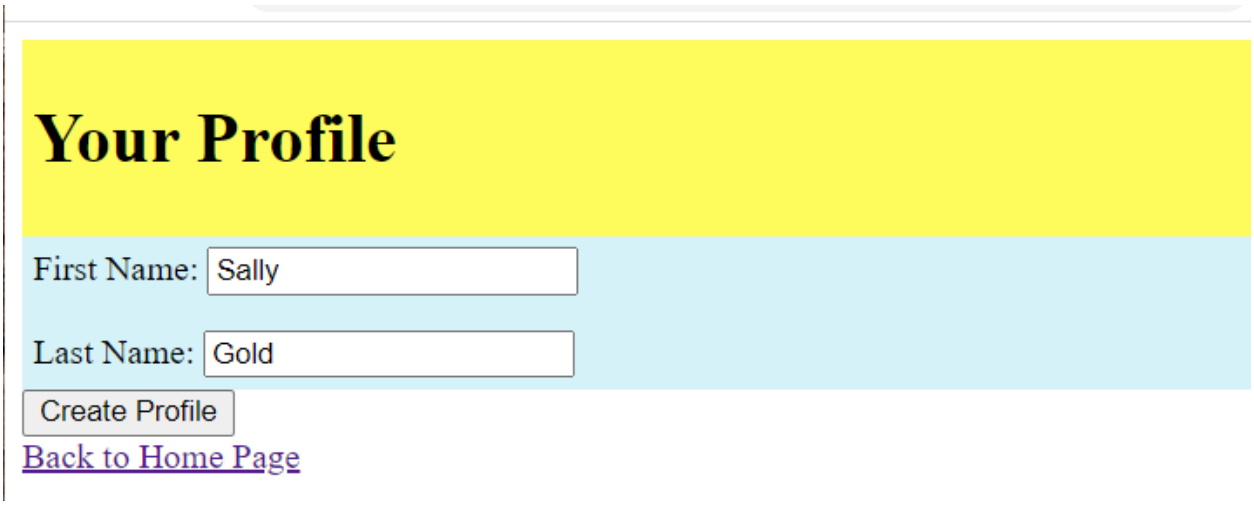
```
hicken Sandwich

{
  _id: new ObjectId("63849e7b46d2f22bb9bea4e9"),
  recipeName: 'Chicken Sandwich',
  recipeurl: 'recipe_images/biscuit_chicken.jpg',
  decription: 'this is sandwich containing gravy and chicken',
  ingredients: [ 'gravy', 'chicken', 'biscuit' ],
  steps: [ 'fry chicken', ' make gravy' ],
  likes: 1,
```

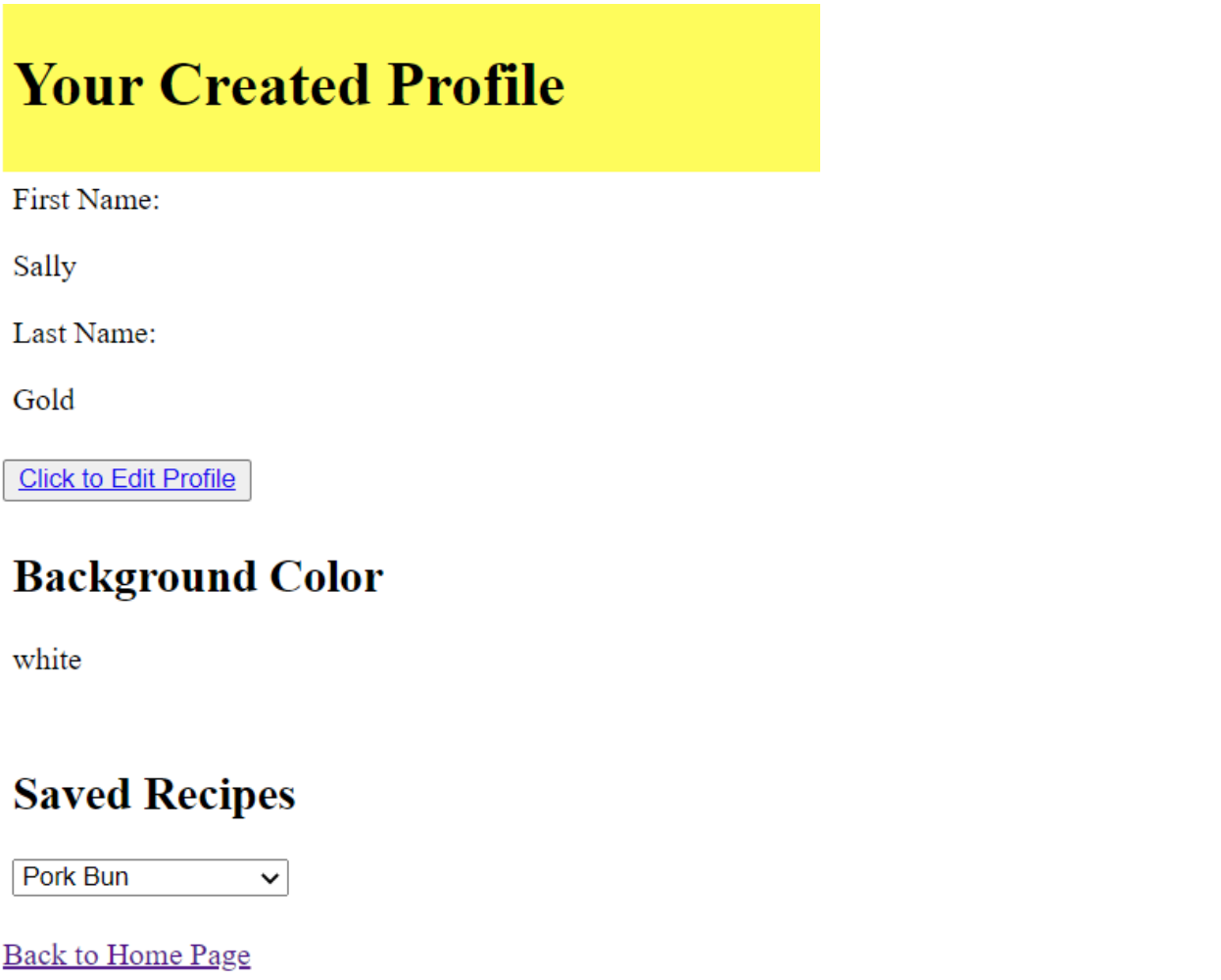
## Backend Functional Requirement 3

### Users must be able to create a profile

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




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



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

    let result = ProfileInfo(
        {
            firstName: req.body.firstName,
            lastName: req.body.lastName,
            backgroundColor: "white",
            savedRecipes: []
        });

    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");
        });
});
```

3. The form data is then inserted into the Mongo database

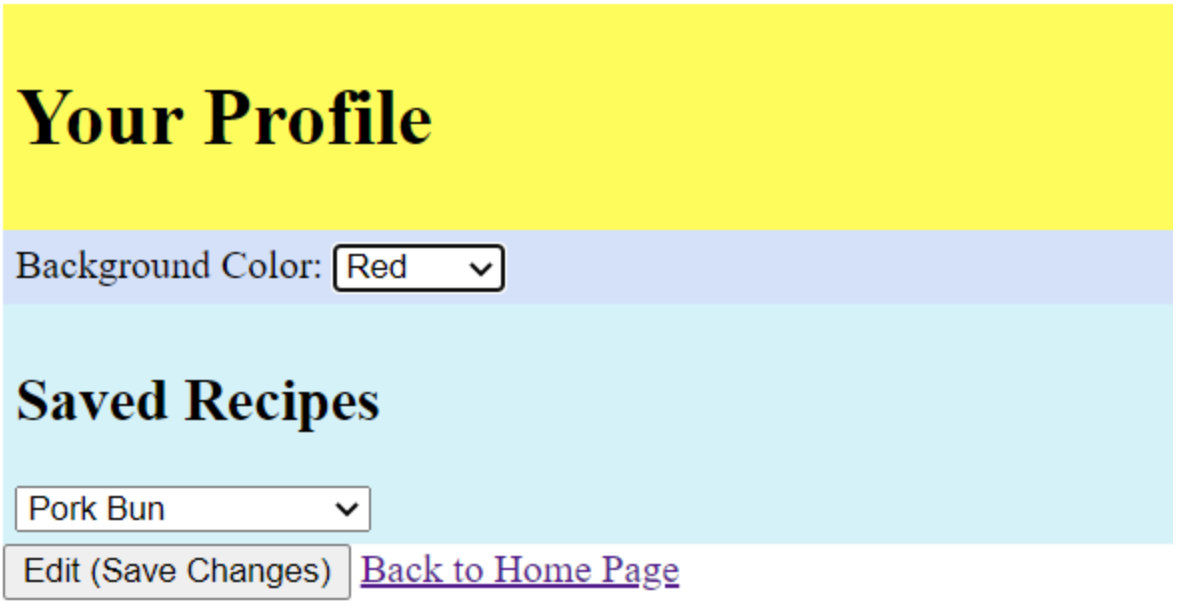
```
erver listening on port 8080
uccess! Inserted data with _id: 638810e6c5d7564c31e280ef into the database.

{
  _id: new ObjectId("638810e6c5d7564c31e280ef"),
  firstName: 'Sally',
  lastName: 'Gold',
  backgroundColor: 'white',
  savedRecipes: [],
  __v: 0
}
```

## Backend Functional Requirement 4

### Users must be able to save recipes to their favorites

1. Users are presented with a form to select a recipe option




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

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

        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`);
            });
});
```

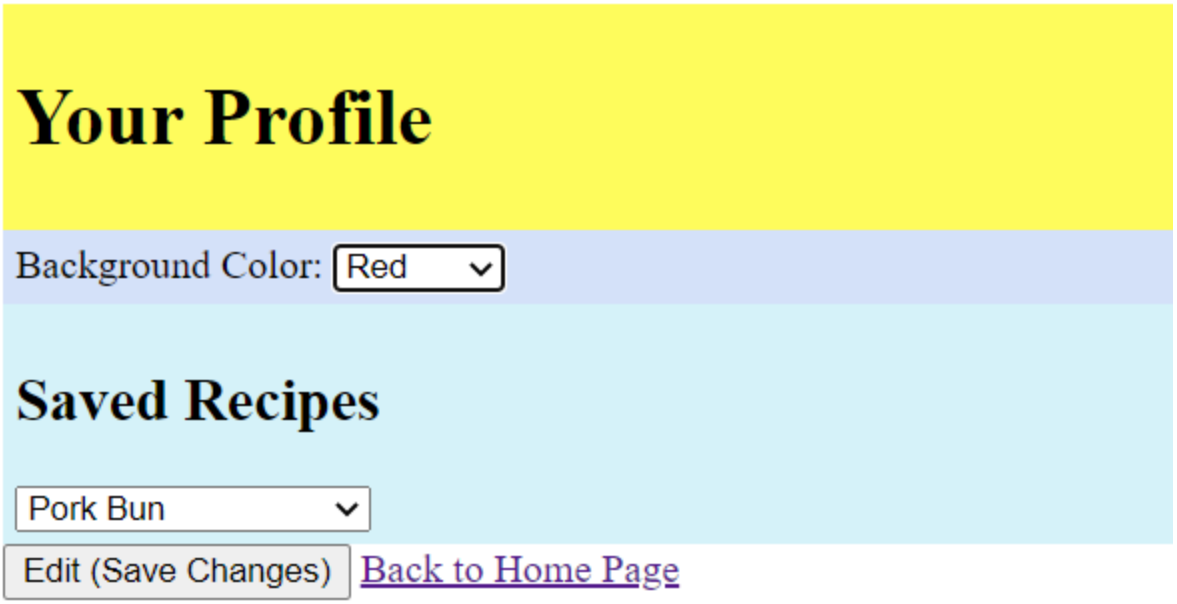
3. The form data is then inserted into the Mongo database

```
[
  {
    _id: new ObjectId("638810e6c5d7564c31e280ef"),
    firstName: 'Sally',
    lastName: 'Gold',
    backgroundColor: 'red',
    savedRecipes: [ 'Pork Bun', 'Chicken Sandwich', 'Pork Bun' ],
    __v: 0
  }
]
```

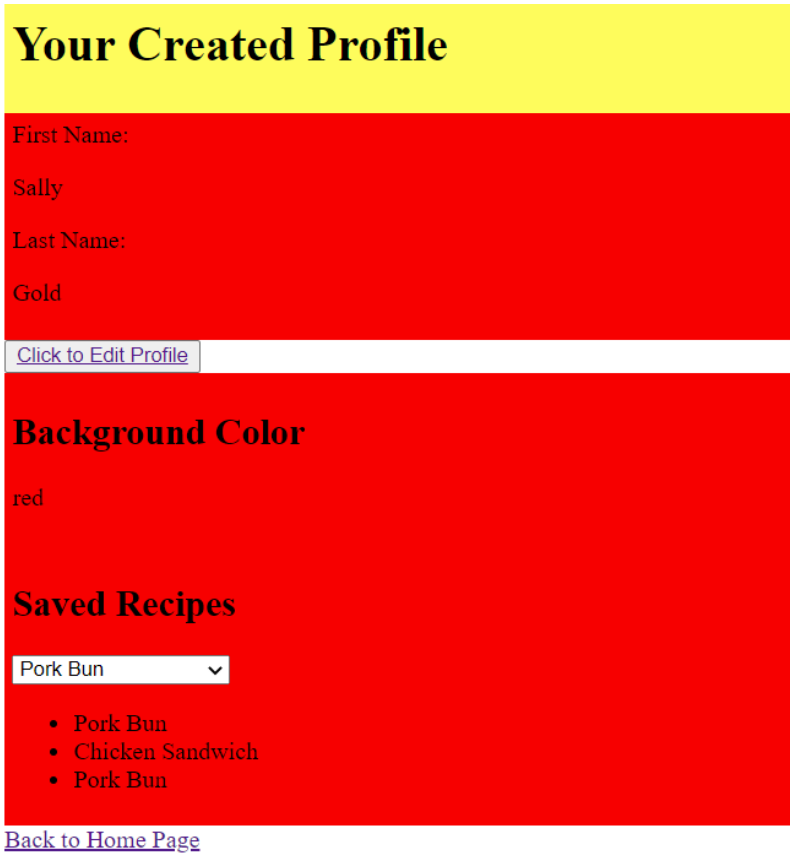
## Backend Functional Requirement 5

### Users must be able to pick their profile background color

1. Users are presented with a form to select a background color option




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



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

        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 updated into the Mongo database